Understanding and Evaluating Generalisation for Superhuman AI Systems

Robert Kirk

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

March 7, 2025

To Saffy, for all the love and support.

I, Robert Kirk, confirm that the work presented in this report is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

As artificial intelligence systems grow increasingly sophisticated and start to surpass human capabilities, the critical challenges of AI alignment and safety come to the forefront. Ensuring that advanced AI systems remain robustly safe and aligned with human values requires a deep understanding of their generalisation properties, i.e. how these systems behave in novel situations and tasks potentially far beyond their training data. With the goal of improving our understanding of generalisation in superhuman AI systems, this dissertation investigates this challenge in current systems that share potential similarities with those in the future: agentic systems trained with reinforcement learning (RL), and systems involving large-scale pretraining such as large language models (LLMs). Throughout, I demonstrate how proper evaluations are crucial to understanding generalisation abilities of AI systems.

I begin with a comprehensive survey of zero-shot generalisation in deep reinforcement learning, analysing existing environments, evaluation protocols, benchmarks, and methods. This survey not only synthesises current knowledge, but also proposes best practices and future research directions in this rapidly evolving field. To facilitate research into generalisation in reinforcement learning problems, I introduce MiniHack, a versatile environment creation tool and benchmark suite. MiniHack enables researchers to design and evaluate a wide array of RL scenarios, with a particular emphasis on zero-shot generalisation tasks. Both of these works demonstrated that training RL agents from scratch is unlikely to produce generally intelligent systems, and that RL is more likely to be used at the fine-tuning stage once more generalisable representations have been learned with other techniques. Additionally, there is a still work to be done to produce RL algorithms that produce robust, generalisable and hence safe AI agents.

This insight motivates the latter two chapters of this thesis, which focus on investigating fine-tuning of pretrained models. First, I investigate the impact of various fine-tuning tech-

niques on large language models (LLMs). I compare Reinforcement Learning from Human Feedback (RLHF), supervised fine-tuning (SFT), and best-of-N sampling, evaluating their effects on generalisation capabilities and output diversity across multiple tasks. My findings reveal that while RLHF enhances generalisation, it comes at the cost of reduced output diversity compared to SFT. To complement this behavioural understanding of language-model fine-tuning, I then investigate the mechanistic effects of fine-tuning on pretrained models using a synthetic data setting and a suite of interpretability tools. My analysis uncovers that fine-tuning primarily creates minimal "wrappers" around existing model capabilities rather than fundamentally deleting or producing entirely new capabilities. This implies that the generalisation properties of fine-tuned models are likely fundamentally limited by the representations learned during pretraining. Again, we see here that, while pretrained models clearly have much-improved generalisation properties, there is still a gap between how current algorithms perform and the necessary level of robustness required for safe and aligned AI systems.

By examining generalisation from diverse angles, this thesis contributes to our understanding of how AI systems adapt to new challenges and how various training techniques influence both their behaviour and internal mechanisms. These insights are crucial for the development of more robust, adaptable, and ultimately safer AI systems as we move closer to superhuman AI capabilities, helping to ensure that as AI systems become more powerful, they remain robustly aligned with human values and interests across a broad spectrum of applications and scenarios.

Acknowledgements

This thesis wouldn't have been possible without the love, support and assistance of a huge range of people. First and foremost, I want to thank my advisors Tim and Ed. I could not have asked for better PhD advisors, and I'm immensely grateful for all the guidance you have provided over the years. Thank you for always challenging me to do the best and most ambitious research possible.

As well as providing me personally with exceptional advice, Tim and Ed created UCL DARK, my research group at UCL, which is a similarly impactful contribution to my PhD. DARK was an amazing place to do my PhD, and I'm grateful for all the members past and present: Minqi, Mika, Zhengyao, Akbir, Laura, Yingchen, Davide, Jack, Roberta and Nathan. I particularly want to thank Laura and Akbir for being co-conspirators in pushing the LLM research agenda; Akbir for all our conversations on AI safety and alignment; Laura for our discussions on LLM generalisation; Zhengyao on all our conversations on RL and LLMs; Mika for your guidance and collaboration during MiniHack (and for working on safety in the end!). Without all these conversations and more I would be much worse-off intellectually.

I am lucky to have a wide network of collaborators beyond UCL DARK. Chief among them is Ekdeep, to whom I'm incredibly grateful for all our dicussion and collaboration. I have learned so much from our work together, both about specific research but also how to think about and approach research problems, and that has helped me develop into a (hopefully) better researcher. I'm grateful to David Krueger for all his help and advice both on research topics and career. I'm grateful to Roberta Raileanu for the opportunity to do an internship at Meta AI, and for letting me pursue a frankly over-ambitious project (implementing RLHF from scratch in 2022 in a 4-month internship is a stretch), and for all the advice and guidance along the way. Thanks to Ishita, Eric, Christoforos, Jelena and others at Meta for help and advice during that project, particularly Ishita and Christoforos for running all the experiments

that actually went in the paper after I had left Meta and no longer had access to compute. I'd also like to thank Flo, Lauro, Aengus, Isaac, Adam and Jean for interesting and useful intellectual conversations and advice throughout my PhD.

I'm grateful to all my collaborators on all the work that went into this thesis (admitting some repeats): Amy, Mikayel, Vitaly, Jack, Minqi, Eric, Fabio, Heinrich, Ishita, Eric, Christoforos, Jelena, Roberta, Samyak, Ekdeep, Robert, Hidenori, David, and of course my advisors Tim and Ed. I should also thank those who gave advice and guidance during each of these projects: Dieuwke, Ethan, Jacob, Kyle, Louis, Patrick, Sebastian, Stephen, Susan, Verna, André, Chelsea, Eliot, Flo, Jessica, Katja, Maximilian, Nicklas, Pablo, Samuel, Sirui, Steve, Theresa, Vincent, Zhou, Danielle, Luis, Olivier, Pasquale, Vegard, Eric, Nikhil, and Usman.

There are several groups who invested resources in my PhD who I would be remiss if I didn't thank them. Foremost is the UCL AI Centre for Doctoral Training in Foundation AI, who funded my PhD and without which it would not have been possible. The community at the AI centre at UCL has been a great place to do research for which I'm very grateful. I'm also grateful to the Long Term Future Fund for several grants to support research projects during the PhD, the Centre for AI Safety for providing compute resources, the OpenAI researcher access program for all the free API credits, and the OpenAI Superalignment Fast Grants scheme for research funding during the final stages of my PhD.

Beyond the academic sphere, I'm incredibly grateful to have a wonderful and amazing family. Thank you to my parents and siblings for all your love and support, I'm so lucky to have you all in my life.

Last but by no means least, I want to thank my wife Saffiyah. Marrying you was the best thing that has happened to me, and I can't put into words how grateful I am for your endless love and support throughout the whole of the PhD and before. I could not have done this without you, and I wouldn't have wanted to anyway. I love you.

Impact Statement

This thesis advances our understanding of generalisation in artificial intelligence systems, with crucial implications for AI safety and alignment as we move toward more capable AI. The work contributes both theoretical insights and practical tools that benefit multiple stakeholders across academia, industry, and society.

Within academia, this research provides several key contributions. The comprehensive survey of zero-shot generalisation in reinforcement learning synthesizes current knowledge and establishes best practices, creating a foundation for future research in this rapidly evolving field. The introduction of MiniHack as a versatile environment creation tool and benchmark suite enables researchers to design and evaluate a wide array of reinforcement learning scenarios, particularly for studying generalisation. The work on fine-tuning large language models provides valuable insights into how different techniques affect model behaviour and internal mechanisms, advancing our understanding of model adaptation and generalisation.

For the AI industry, this research has immediate practical applications. The findings regarding various fine-tuning techniques' effects on large language models can inform deployment decisions and development strategies. The demonstration that fine-tuning creates minimal "wrappers" around existing model capabilities, rather than fundamentally altering them, has important implications for model development and deployment strategies. The work also provides concrete recommendations for more robust evaluation protocols, which can help companies better assess and improve their AI systems' safety and reliability.

The broader societal impact of this research is significant, particularly in the context of AI safety. As AI systems become more powerful and autonomous, understanding their generalisation properties is crucial for ensuring they remain safe and aligned with human values. The insights provided by this thesis can help inform policy and governance frameworks around AI development and deployment. The work highlights important limitations in current

approaches to building safe AI systems, which is crucial information for policymakers and stakeholders working on AI regulation and safety standards.

The impact of this research will manifest both immediately and over the longer term. In the short term, the tools and methodologies developed can be immediately adopted by researchers and practitioners. The MiniHack environment and evaluation protocols can be used to improve current AI systems' robustness and safety. Over the longer term, the insights into model behaviour and generalisation will contribute to the development of more reliable and safer AI systems, particularly as we approach superhuman AI capabilities.

The findings have been disseminated through multiple channels, including academic publications, open-source software releases, and engagement with the broader AI research community. The practical tools developed have been made available to researchers and practitioners, ensuring wide accessibility and impact. The insights and recommendations can inform both academic research directions and industrial best practices in AI development.

This research contributes to UCL's goal of delivering impact for public benefit by advancing our understanding of AI safety and alignment, which is crucial for ensuring that advances in AI technology benefit society while minimizing potential risks.

UCL Research Paper Declaration Forms

Chapter 2: A Survey of Zero-shot Generalisation in Deep Reinforcement Learning

1. **For a research manuscript that has already been published:**

   (a) **What is the title of the manuscript?** A Survey of Zero-shot Generalisation in Deep Reinforcement Learning

   (b) **Please include a link to or doi for the work:** https://www.jair.org/index.php/jair/article/view/14174

   (c) **Where was the work published?** Journal of Artificial Intelligence Research

   (d) **Who published the work?** AI Access Foundation

   (e) **When was the work published?** 2023/09/01

   (f) **List the manuscript's authors in the order they appear on the publication:** Robert Kirk, Amy Zhang, Edward Grefenstette, Tim Rocktäschel

   (g) **Was the work peer reviewd?** Yes

   (h) **Have you retained the copyright?** Yes

   (i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)? If 'Yes', please give a link or doi** Yes: https://arxiv.org/abs/2111.09794

2. **For multi-authored work, please give a statement of contribution covering all authors:** Robert Kirk led the work, developed the formalism, benchmarks categorisation, methods categorisation, and discussion and future work, wrote the full manuscript of the survey, and wrote successive drafts with comments and feedback from the other authors. Amy Zhang contributed parts of the background and formalism sections, as

well as providing improvements on the entire work through discussion and editing. Tim Rocktäschel and Edward Grefenstette advised Robert Kirk, providing discussion and feedback in developing the ideas behind the survey, and provided feedback and comments on the manuscript.

3. **In which chapter(s) of your thesis can this material be found?** Chapter 2

e-Signatures confirming that the information above is accurate:

Candidate: *Robert Kirk*

Date: *10th August 2024*

Supervisor: *Tim Rocktäschel*

Date: *10th August 2024*

Chapter 3: MiniHack the Planet: A Sandbox for Open-ended Reinforcement Learning Research

1. **For a research manuscript that has already been published:**

   (a) **What is the title of the manuscript?** MiniHack the Planet: A Sandbox for Open-ended Reinforcement Learning Research

   (b) **Please include a link to or doi for the work:** https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Abstract-round1.html

   (c) **Where was the work published?** Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1 (NeurIPS Datasets and Benchmarks 2021)

   (d) **Who published the work?** Neural Information Processing Systems Foundation

   (e) **When was the work published?** 2021/9/27

   (f) **List the manuscript's authors in the order they appear on the publication:** Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Küttler, Edward Grefenstette, Tim Rocktäschel

   (g) **Was the work peer reviewd?** Yes

(h) **Have you retained the copyright?** Yes

(i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)? If 'Yes', please give a link or doi** Yes: https://arxiv.org/abs/2109.13202

2. **For multi-authored work, please give a statement of contribution covering all authors:** Mikayel Samvelyan led the work, wrote the initial codebase, worked with Robert Kirk and Tim on the framing and research direction, and ran many of the experiments. Robert Kirk was the second author on the project, and contributed to: framing the project and research direction; designing the benchmark suite; programming parts of the environment; and writing the manuscript. Vitaly Kurin programmed parts of the environment, including the boxoban and minigrid environments. Jack Parker-Holder and Minqi Jiang contributed to discussions about the environment design, and wrote and ran the UED experiments. Eric Hambro and Fabio Petroni contributed to discussions about the environment design, and Eric also wrote some code for benchmarking RL algorithms. Heinrich Küttler provided invaluable technical knowledge of NLE, and along with Edward Grefenstette and Tim Rocktäschel advised on the project, provided feedback on the manuscript, and contributed to discussions about the environment design.

3. **In which chapter(s) of your thesis can this material be found?** Chapter 3

e-Signatures confirming that the information above is accurate:

Candidate: *Robert Kirk*

Date: *10th August 2024*

Supervisor: *Tim Rocktäschel*

Date: *10th August 2024*

Chapter 4: Understanding the Effects of RLHF on LLM Generalisation and Diversity

1. **For a research manuscript that has already been published:**

(a) **What is the title of the manuscript?** Understanding the Effects of RLHF on LLM Generalisation and Diversity

(b) **Please include a link to or doi for the work:** https://openreview.net/forum?id=PXD3FAVHJT

(c) **Where was the work published?** Proceedings of the 12th International Conference on Learning Representations (ICLR 2024)

(d) **Who published the work?** OpenReview

(e) **When was the work published?** 2024/05/07

(f) **List the manuscript's authors in the order they appear on the publication:** Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, Roberta Raileanu

(g) **Was the work peer reviewd?** Yes

(h) **Have you retained the copyright?** Yes

(i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)? If 'Yes', please give a link or doi** Yes: https://arxiv.org/abs/2310.06452

2. **For multi-authored work, please give a statement of contribution covering all authors:** Robert Kirk lead the project, set the direction, designed, programmed and ran the majority of the experiments, and wrote much of the paper. Ishita Mediratta assisted with programming and running the RLHF training experiments. Christoforos Nalmpantis programmed parts of the GPT-4 evaluation code and initial RLHF training code. Jelena Luketina assisted in the AlpacaFarm evaluations. Eric Hambro programmed earlier versions of the RLHF training code. Edward Grefenstette advised on project direction and paper writing. Roberta Raileanu advised on project direction, experiment design, programming and paper writing. All authors participated in discussions over experiment design and paper editing.

3. **In which chapter(s) of your thesis can this material be found?** Chapter 4

e-Signatures confirming that the information above is accurate:

Candidate: *Robert Kirk*

Date: *10th August 2024*

Senior Author: *Roberta Raileanu*

Date: *10th August 2024*

Chapter 5: Mechanistically analysing the effects of fine-tuning on procedurally defined tasks

1. **For a research manuscript that has already been published:**

   (a) **What is the title of the manuscript?** Mechanistically analysing the effects of fine-tuning on procedurally defined tasks

   (b) **Please include a link to or doi for the work:** https://openreview.net/forum?id=A0HKeKl4Nl

   (c) **Where was the work published?** Proceedings of the 12th International Conference on Learning Representations (ICLR 2024)

   (d) **Who published the work?** OpenReview

   (e) **When was the work published?** 2024/05/07

   (f) **List the manuscript's authors in the order they appear on the publication:** Samyak Jain, Robert Kirk, Ekdeep Singh Lubana, Robert P. Dick, Hidenori Tanaka, Edward Grefenstette, Tim Rocktäschel, David Scott Krueger

   (g) **Was the work peer reviewd?** Yes

   (h) **Have you retained the copyright?** Yes

   (i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)? If 'Yes', please give a link or doi** Yes: https://arxiv.org/abs/2311.12786

2. **For multi-authored work, please give a statement of contribution covering all authors:** Ekdeep Singh Lubana conceived the project direction and developed a set of hypotheses on the limitations of fine-tuning, with inputs from Robert Kirk. Samyak Jain and Ekdeep co-designed a draft of the PCFG and Tracr setups, and came up with pruning and reverse fine-tuning analysis which led to validation and further refining of the hypotheses. Samyak led the experimental execution and made the tasks considered in the paper precise in collaboration with Ekdeep. Robert proposed and ran

the TinyStories experiments with inputs from Ekdeep, Samyak, Edward Grefenstette and Tim Rocktäschel. Literature review and writing of the main paper was led by Ekdeep. Samyak led writing of the appendix. Ekdeep, Samyak, Robert, and HT collaborated on design of all figures and plots. David Scott Krueger acted as the primary senior advisor on the paper, with inputs from Robert P. Dick, Hidenori Tanaka, Edward, and Tim as well.

3. **In which chapter(s) of your thesis can this material be found?** Chapter 5

e-Signatures confirming that the information above is accurate:

Candidate: *Robert Kirk*

Date: *10th August 2024*

Senior Author: *David Scott Krueger*

Date: *10th August 2024*

Contents

Chapter 1

Introduction

Artificial intelligence (AI) systems have rapidly advanced in recent years, demonstrating remarkable capabilities in various domains [OpenAI, 2022, Anthropic, 2023, Jumper et al., 2021, OpenAI, 2023, Silver et al., 2017]. However, a fundamental challenge exists: ensuring that AI systems robustly and reliably pursue the goals we intend for them, known as the alignment problem [Ngo et al., 2023]. The consequences of misaligned AI systems competently pursuing goals that are not our own are potentially catastrophic [Bengio, 2023, Carlsmith, 2024, Hendrycks et al., 2023], which motivates the need for effective methods to align AI systems' behaviour with human values and goals. Addressing the problem of AI alignment is the main motivation for the research pursued in this thesis.

We can think of the alignment problem as matching the goals of a designer of an AI system with the goals of the AI system itself. A standard way of decomposing the AI alignment problem is into the problems of outer alignment and inner alignment [Ngo et al., 2023]. Outer alignment is concerned with correctly specifying the optimisation objective of the training process such that performing well under that optimisation objective matches the designer's goals; i.e. matching the goals of the optimisation process (e.g. the reward function) to the designer's goals. This problem especially difficult for hard-to-operationalise goals such as "do what the user would want if they spent a long time thinking about it" and "maximise human flourishing".

While outer alignment is important, the focus of this thesis is on inner alignment [Ngo et al., 2023, Hubinger et al., 2019]. Inner alignment is concerned with the design of optimisation processes which produces models that pursue the goals the optimisation objective incentives in all situations. That is, inner alignment is the problem of matching the

goals the model pursues when deployed with the goals of the optimisation process. Note here the analogy to outer alignment: outer alignment matches the optimisation process's goals to the designer's goals, and inner alignment matches the model's goals with the optimisation process's goals.

Inner alignment is fundamentally a problem of generalisation (typically out-of-distribution generalisation), and in a machine learning context could involve improving the robustness and generalisation of models (especially with respect to the goals they pursue), or using interpretability techniques to ensure that the model is actually pursuing the goal you want it to pursue. Thus, to ensure we can tackle the inner alignment problem, I argue we need to improve our understanding of generalisation and our techniques for evaluating it systematically. This is the main goal of this thesis: to push forwards our understanding and evaluation of generalisation in AI systems, specifically focusing on work that will be useful in preparation for working with highly capable and potentially superhuman AI systems.

Analogies to superhuman AI systems: A key difficulty in doing empirical research that is useful for the development of highly capable AI systems is that we do not have access to these systems, as they do not yet exist. Further, we not know precisely what future systems will look and how they will be produced. However, we can focus on problems and models that possess properties we expect future AI systems to have, as we are not entirely in the dark about what future systems will be. We can then aim to improve our understanding and evaluation of generalisation in these analogous settings.

Motivated by this goal, in this thesis I focus on two classes of models and problem settings that are likely to be central to the development of future AI systems: reinforcement learning (RL) agents and fine-tuned large language models (LLMs).

1.1 Generalisation for Reinforcement Learning Agents

Future AI systems are likely to be agentic (i.e. competently autonomous and goal-directed) in some way as autonomous agents are economically valuable [Chan et al., 2023, Carlsmith, 2024]. Additionally, failures of generalisation are likely to be more harmful for autonomous and agentic systems as they generally have a larger impact on the world, and so if that impact is negative the failure more harmful. This is particularly the case for systems that exhibit goal misgeneralisation [di Langosco et al., 2022, Shah et al., 2022], where the system competently

pursues the incorrect objective.

Since reinforcement learning [RL, Sutton et al., 1998] is the prevailing method for training agentic systems, in Chapter 2 I survey the field of zero-shot generalisation in RL. In this problem setting, RL systems must perform well in novel environments zero-shot (i.e. without information or training data from the environment), which matches the problem setting I expect RL algorithms to have to tackle when ensuring robust inner alignment of highly-capable systems. From this survey, it is clear that there are currently no good methods for ensuring robust generalisation in RL, and hence no algorithm we would be happy to use to address inner alignment in superhuman AI systems. One key deficiency in the field identified in the survey is a lack of benchmarks that can enable the kind of research important for studying inner alignment and zero-shot generalisation more broadly. We require benchmarks with a combination of properties: the ability to test for a wide varieties of generalisation behaviours, while still having tight control over the environment which enables rigorous scientific investigation.

To address this deficit in Chapter 3 I present MiniHack, a benchmark which satisfies all these properties. We use this benchmark to test a range of methods, and again confirm that existing approaches do not solve zero-shot generalisation problems effectively. Through the work in both of these chapters, it became clear that research that improved the frontier of generalisation in RL was unlikely to be helpful for addressing inner alignment in superhuman AI systems. Current algorithms are a long way from producing truly robust, generalisable and ultimately safe AI agents. The bottleneck to improving the performance of these methods is not inner misalignment or goal misgeneralisation but capabilities generalisation more broadly. Since these works were performed, most improvements in generalisation of RL systems have come from scaling up data, environment complexity, and compute, rather than novel algorithmic developments [Bauer et al., 2023, Bruce et al., 2024, Hughes et al., 2024]. While work in this area can be valuable for addressing inner misalignment, I believe other settings are more analogous to this problem and hence are more likely to yield useful progress.

In particular, this leads me to consider more competent models where RL is showing impressive results as a fine-tuning technique, and where alignment is a bottleneck to improving models: large language models. In this setting I expect studying the generalisation properties

of methods is more likely to produce insight that is more transferable to future AI systems.

1.2 Generalisation in the Fine-tuning of Large Language Models

The most impressive and general-purpose AI systems that currently exist are large language models (LLMs), such as GPT-3 [Brown et al., 2020a], Gopher [Rae et al., 2022], Chinchilla [Hoffmann et al., 2022], OPT [Zhang et al., 2022], PaLM [Chowdhery et al., 2022], Claude [Anthropic, 2023] and LLaMa [Touvron et al., 2023a]. These systems are deployed to millions of users, and it is likely that the recipe of large-scale unsupervised pretraining followed by more specialised fine-tuning will be a component of any future general-purpose AI system. The most powerful models are produced by this recipe today, and the potential reliability of the relationship between scaling up compute, data and model size and the capabilities of these systems [Hoffmann et al., 2022, Kaplan et al., 2020] makes it likely this recipe will continue to work. There is also increasing work to fine-tune LLMs as agents [Liu et al., 2023c, Wang et al., 2024], often using RL techniques, combining this motivation with that of the RL work discussed above.

While LLMs are clearly very capable, their pretraining objective is not aligned with human values (i.e. a failure of outer alignment). In the case of pretrained LLMs, alignment is a bottleneck to improving performance - these models are often very capable but are not "motivated" to pursue the goals the user cares about. This makes this setting a promising one for studying methods for improving alignment, and inner alignment specifically. To address this misalignment, additional adjustments to the pretrained model are necessary. Fine-tuning is a common approach employed to make pretrained LLMs aligned with our goals. As fine-tuning is where the alignment of the system is enforced, it is crucial to understand how fine-tuning generalises, to ensure that the resulting system is robustly safe and aligned, and has not mis-generalised the goal it was trained for at fine-tuning. In Chapter 4 I evaluate different methods for fine-tuning LLMs to be aligned with human preferences, include Reinforcement Learning from Human Preferences [Christiano et al., 2017, Stiennon et al., 2022, RLHF], one of the most common and prevalent methods. I analyse how these methods differentially affect the generalisation performance of trained models, as well as other properties, such as

the diversity of outputs generated by these models. I generally find that in the most realistic setting, RLHF improves the generalisation compared to supervised fine-tuning and best-of-N sampling, but generalisation performance is still far from perfect.

However, this evaluation is only behavioural. To truly understand whether a model has learned the correct generalisation of the training objective, we need a mechanistic understanding of the model: we need to know "how" it is producing its behaviour. Hence, in Chapter 5, I investigate how fine-tuning changes the internal computation of pretrained models. To ensure I can fully understand the effects of fine-tuning, I primarily work in a fully synthetic data setting, and then validate my conclusions on more realistic data. I find that fine-tuning does not make substantial changes to the model's internal computation, which has implications for what generalisation at fine-tuning one can expect: fine-tuned models will be unlikely to go beyond what the pretrained model is capable of, and are likely to generalise in a similar way to the pretrained model.

1.3 Summary

Overall, there is much research to do in both understanding and improving on the generalisation of AI systems. In both the RL and LLM fine-tuning settings, we see that current algorithms are not producing systems that are as robust and generalisable to be deemed truly safe AI systems. It is also clear that better evaluation methodologies for current and future AI systems can lead to insights about how systems generalise, and potentially how to improve that generalisation. In the final chapter (Chapter 6), I summarise the work in this thesis and provide concrete actionable recommendations for more robust and useful evaluations of AI systems. Finally, I outline future research directions that would further advance our understanding and evaluation of generalisation in AI systems.

Chapter 2

Surveying Zero-shot Generalisation in Deep Reinforcement Learning

This chapter is based on the following published work: Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A Survey of Zero-shot Generalisation in Deep Reinforcement Learning. *Journal of Artificial Intelligence Research*, 76:201–264, 2023b. ISSN 1076-9757. doi:10.1613/jair.1.14174. URL https://www.jair.org/index.php/jair/article/view/14174

2.1 Introduction

When consider properties of superhuman AI systems, a common and likely ingredient is these systems are agents that act in the world over long time horizons and autonomously pursue goals [Liu et al., 2023c, Wang et al., 2024, Chan et al., 2023, Team et al., 2021]. The field most targeted at producing AI agents currently is Reinforcement Learning (RL). Hence, we can analogise the problem of inner alignment in superhuman AI systems to generalisation in RL agents. Motivated by this analogy, in this chapter I survey the field of zero-shot generalisation in RL to understand the current state of algorithms and benchmarks, and assess how far there is to go from current algorithms to those that produce truly safe and robust AI agents.

Additionally, the field of RL has the potential to be used in a wide range of current applications from autonomous vehicles [Filos et al., 2020] and algorithm control [Biedenkapp et al., 2020] to robotics [OpenAI et al., 2019a], but to fulfil this potential we need RL algorithms that can be used in the real world. Reality is dynamic, open-ended and always changing, and RL algorithms will need to produce policies that are robust to variations

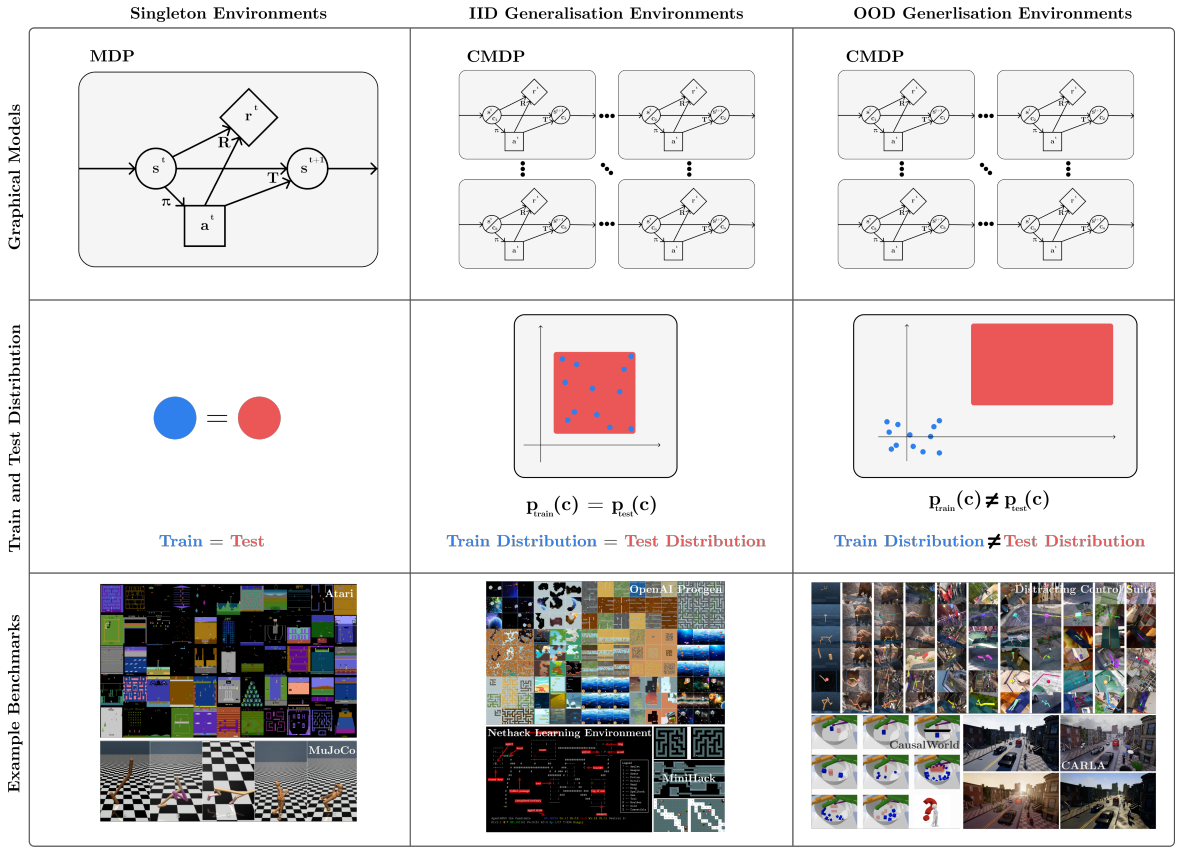


Figure 2.1: Zero-shot Generalisation in Reinforcement Learning. A visualisation of three types of environment (columns) with respect to their graphical model, training and testing distribution and example benchmarks (rows). Classical RL has focused on environments where training and testing are identical (singleton environments, first column). I focus on an underexplored setting, inspired by likely real-world scenarios, where training and testing environments will be different, with environment instances either from the same distribution (Independent and Identically Distributed (IID) ZSG Environments, second column) or from different distributions (OOD ZSG Environments, third column). The split between the second and third columns is just one example of the way in which ZSG is a class of problems rather than an individual problem. The top row visualises the differences in graphical models between singleton environments and environments where ZSG is required. For more information on the CMDP formalism see Section 2.3.2.

in their environments, and have the capability to transfer and adapt to unseen (but similar) environments when deployed. This additionally motivates the study of zero-shot generalisation in RL.

Unfortunately, much current RL research works on benchmarks such as Atari [Bellemare et al., 2013] and MuJoCo [E. Todorov et al., 0007/2012-10-12, Brockman et al., 2016b], which do not have the attributes described above: they evaluate the policy on exactly the same environment it was trained on, which often does not match with real-world scenarios (Fig. 2.1 left column). This is in stark contrast to the standard assumptions of supervised learning where

the training and testing sets are disjoint and is likely to lead to strong evaluation overfitting [Whiteson et al., 2011]. This has resulted in policies that perform badly on even slightly adjusted environment instances (specific levels or tasks within an environment) and often fail on unseen random seeds used for initialisation [Zhang et al., 2018a,c, Farebrother et al., 2020, Gamrian and Goldberg, 2019].

One specific instance of the generalisation problem in RL is goal misgeneralisation [di Langosco et al., 2022, Shah et al., 2022]. This problem occurs when, during deployment, an AI system has the capability to achieve the correct and desired goal, but instead pursues a different (misgeneralised) goal. This problem is likely to become more of an issue as systems become more powerful, and the goals they are trained on become more complex and abstract. A key motivation of this chapter's focus on generalisation in RL is to enable improvements that could tackle goal misgeneralisation, although this is a more distant goal rather than the immediate focus of this survey.

In this work I survey the recent literature studying zero-shot generalisation in deep RL, a field focused on producing algorithms with the robustness, transfer and adaptation properties required to perform well in the real world. I offer a unifying framework that builds on previous work [Ghosh et al., 2021, Hallak et al., 2015, Harrison et al., 2017, Higgins et al., 2017b, Song et al., 2020, Perez et al., 2020] that formalises the problem of ZSG in RL as a *class* of problems, rather than a single problem. While prior work [Ghosh et al., 2021, Hallak et al., 2015, Perez et al., 2020] uses the contextual MDP framework or related frameworks to describe how an agent can encounter new, unseen states at test time to generalise to, I further extend and break down the types of generalisation that can be possible, e.g. combinatorial, interpolation vs. extrapolation, single-factor vs. multi-factor (Fig. 2.3). I formalise more fully the ZSG problem, formally specifying the policy class, making clear the choice of whether the context is observed or not, and including cases where the context distribution is controllable during training Section 2.3.3 This breakdown enables us to clearly and crisply compare previous works, as well as understand how to choose future research directions. For example, improving "generalisation" without any additional assumptions is inherently underspecified; it is unlikely that we can generically improve generalisation, given this class of problems is so broad that some analogy of the No Free Lunch theorem [Wolpert and Macready, 1997]

applies: improving generalisation in some settings could harm generalisation in others. Two broad categories of ZSG problem are shown in Fig. 2.1 in the centre and right columns.

Using this formalism, I survey and examine the range of benchmarks available for ZSG in RL, and go on to discuss methods aimed at tackling different ZSG problems. Finally, I propose several settings within ZSG which are underexplored but still vital for various real-world applications of RL, as well as many avenues for future work on methods that can solve different generalisation problems. Throughout, I critically review the state of the field and provide recommendations for ensuring future research is robust and useful. I aim to make the field more legible to researchers and practitioners both in and out of the field and make discussing new research directions easier by providing a common reference and framework. This new clarity can improve the field, and enable robust progress towards more general RL methods.

Scope. Generalisation in RL captures a large amount of research, so to make the survey in this chapter feasible I limit the scope of my review in several ways. First, I focus on the specific problem setting of zero-shot generalisation (ZSG), where the policy is evaluated zero-shot on a collection of environment instances different to those it was trained on. Crucially, this setting disallows any additional training in or data from the test environment instances, meaning methods such as domain adaptation and many meta-RL approaches are not applicable. This is different from classical RL, which has historically focused on online learning in a single MDP, where generalisation refers to the notion of generalising to novel states in the same MDP. While this setting can be useful to study, I instead focus on situations which require policies that can be deployed in situations they haven't been trained in, and generalise well zero-shot to those situations. This setting is especially relevant for current deep RL algorithms, which often aren't sample-efficient or safe enough to be deployed online without significant offline or in-simulation training first. This motivates my focus on ZSG. I discuss and motivate this setting and its restrictions more in Section 2.3.6.

Second, I only cover single-agent RL in this chapter. There are generalisation problems within multi-agent reinforcement learning (MARL), such as being general enough to defeat multiple different opponent strategies [OpenAI et al., 2019b, Vinyals et al., 2019] and generalising to new team-mates in cooperative games [Hu et al., 2021, 2020], but I do not cover any

work in this area here. While mathematically these problems could be modelled equivalently (if co-players are modelled as parts of the dynamics function, rather than as agents as in Games-based formulations [Shapley, 1953]), approaches to these problems tend to be quite different, explicitly utilising the fact that variation and generalisation challenges come from other co-players rather than other parts of the environment. Relatedly, there is work on using multiple agents in a single-agent setting to increase the diversity of the environment and hence the generality of the policy [Team et al., 2021], which I do cover.

Finally, I do not cover theoretical work on generalisation in RL. While there is recent work in this area [Du et al., 2020, Malik et al., 2021] that is valuable, I focus on empirical research as it's more widely studied.

Overview of the Survey. The structure of the survey is as follows. I first briefly describe related work such as other surveys and overviews in Section 2.2. I introduce the formalism and terminology for ZSG in RL in Section 2.3, including the relevant background. I then proceed to use this formalism to describe current benchmarks for ZSG in RL in Section 2.4, discussing both environments (Section 2.4.1) and evaluation protocols (Section 2.4.2). I categorise and describe work producing methods for tackling ZSG in Section 2.5. Finally, I present a critical discussion of the current field, including recommendations for future work in both methods and benchmarks, in Section 2.6, and conclude with a summary of the key takeaways from the survey in Section 2.7.

Contributions. To summarise, my key contributions are:

- I present a unified formalism and terminology for discussing the broad class of ZSG problems and breaking down the assumptions necessary to achieve ZSG, building on formalisms and terminology presented in multiple previous works [Ghosh et al., 2021, Hallak et al., 2015, Harrison et al., 2017, Higgins et al., 2017b, Song et al., 2020, Perez et al., 2020]. My contribution here is the unification of these prior works into *a clear formal description of the class of problems referred to as ZSG in RL*, which captures the full space of problems, which was not done by any one existing formalism.

- I propose a taxonomy of existing benchmarks that can be used to test for ZSG, splitting the discussion into categorising environments and evaluation protocols. My formalism allows us to cleanly describe weaknesses of the purely Procedural Content Generation

(PCG) approach to ZSG benchmarking and environment design: *having a completely PCG environment limits the precision of the research that can be done on that environment*. I recommend that *future environments should use a combination of PCG and controllable factors of variation*.

- *I propose a categorisation of existing methods to tackle various ZSG problems*, motivated by a desire to make it easy both for practitioners to choose methods given a concrete problem and for researchers to understand the landscape of methods and where novel and useful contributions could be made. I point to many under-explored avenues for further research, including fast online adaptation, tackling RL-specific ZSG issues, novel architectures, model-based RL and environment instance generation.

- I critically discuss the current state of ZSG in RL research, recommending future research directions. In particular, I argue that *building benchmarks would enable progress in offline RL generalisation and reward-function variation*, both of which are important settings. Further, I point to several different settings and evaluation metrics that are worth exploring: *investigating context-efficiency and working in a continual RL setting* are both areas where future work is necessary.

2.2 Related Work: Surveys In Reinforcement Learning Subfields

While there have been previous surveys of related subfields in RL, none have covered zero-shot generalisation in RL explicitly. Khetarpal et al. [2020] motivated and surveyed continual reinforcement learning (CRL), which is closely related to ZSG in RL as both settings require adaptation to unseen tasks or environments; however, they explicitly do not discuss the zero-shot setting that is the concern of this chapter (for more discussion of CRL see Section 2.6.1). Chen and Li [2020] gave a brief overview of Robust RL (RRL) [Morimoto and Doya, 2000], a field aimed at tackling a specific form of environment model misspecification through worst-case optimisation. This is a sub-problem within the class of generalisation problems I discuss here, and Chen and Li [2020] only briefly survey the field. Albrecht and Stone [2018] survey methods for modelling other agents, which can be seen as a form of generalisation

problem (even in single-agent RL), if the environment contains a distribution over agents.

Zhao et al. [2020] survey methods for sim-to-real transfer for deep RL in robotics. Sim-to-real is a concrete instantiation of the generalisation problem, and hence there is some overlap between my work and that of Zhao et al. [2020], but my work covers a much broader subject area, and some methods for sim-to-real transfer rely on data from the testing environment (reality), which I do not assume here. Müller-Brockhausen et al. [2021] and Zhu et al. [2021] survey methods for transfer learning in RL (TRL). TRL is related to generalisation in that both topics assume a policy is trained in a different setting to its deployment, but TRL generally assumes some form of extra training in the deployment or target environment, whereas I am focused on zero-shot generalisation. Finally, surveys on less related topics include those of Vithayathil Varghese and Mahmoud [2020] who survey multi-task deep RL, Amin et al. [2021] who survey exploration in RL, and Narvekar et al. [2020] who survey curriculum learning in RL.

None of these surveys focuses on the zero-shot generalisation setting that is the focus of this work, and there is still a need for a formalism for the class of ZSG problems which will enable research in this field to discuss the differences between different problems.

2.3 Formalising Zero-shot Generalisation In Reinforcement Learning

In this section, I present a formalism for understanding and discussing the class of zero-shot generalisation (ZSG) problems in RL. I first review the relevant background in supervised learning and RL before motivating the formalism itself. Formalising ZSG in this way shows that it refers to a *class* of problems, rather than a specific problem, and hence research on ZSG needs to specify which group of ZSG problems it is tackling. Having laid out this class of problems in Section 2.3.3, I discuss additional assumptions of structure that could make generalisation more tractable in Section 2.3.5; this is effectively specifying sub-problems of the wider ZSG problem.

2.3.1 Reinforcement Learning Background

The standard formalism in Reinforcement Learning (RL) is the Markov Decision Process (MDP). An MDP consists of a tuple $M = (S, A, R, T, p)$, where S is the state space; A is

the action space; $R : S \times A \times S \to \mathbb{R}$ is the scalar reward function; $T(s'|s,a)$ is the possibly stochastic Markovian transition function; and $p(s_0)$ is the initial state distribution. We also consider partially observable MDPs (POMDPs). A POMDP consists of a tuple $M = (S,A,O,R,T,\phi,p)$, where S,A,R,T and p are as above, O is the observation space, and $\phi : S \to O$ is the emission or observation function. In POMDPs, the policy only observes the observation of the state produced by ϕ.

The standard problem in an MDP is to learn a policy $\pi(a|s)$ which produces a distribution over actions given a state, such that the cumulative reward of the policy in the MDP is maximised:

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}}\, \mathbb{E}_{s \sim p(s_0)} \left[\mathscr{R}(s) \right],$$

where π^* is the optimal policy, Π is the set of all policies, and $\mathscr{R} : S \to \mathbb{R}$ is the *return* of a state, calculated as

$$\mathscr{R}(s) := \mathbb{E}_{a_t \sim \pi(a_t|s_t), s_{t+1} \sim T(s_{t+1}|s_t,a_t)} \left[\sum_{t=0}^{\infty} R(s_t,a_t,s_{t+1}) | s_0 = s \right].$$

This is the total expected reward gained by the policy from a state s. The goal in a POMDP is the same, but with the policy taking observations rather than states as input. This sum may not exist if the MDP does not have a fixed horizon, so we normally use one of two other forms of the return, either assuming a fixed number of steps per episode (a *horizon* H) or an exponential discounting of future rewards by a discount factor γ. Note that we formalise the policy here as Markovian (i.e. that only takes the previous state as input) for simplicity, but the policy can take in the full history $(s_1,a_1,r_1,\ldots s_{t-1},a_{t-1},r_{t-1},s_t)$ as input, for example using a recurrent neural network. We define the set of possible histories for a state and action space as $H[S,A] = \{(s_1,a_1,r_1,\ldots s_{t-1},a_{t-1},r_{t-1},s_t) | t \in \mathbb{N}\}$, similarly for an observation space. A policy being non-Markovian allows it to be adaptive (for further discussion see Section 2.3.6), which is relevant for generalisation problems where different actions may be optimal at the same state depending on the history of the episode.

2.3.2 Contextual Markov Decision Processes

To talk about zero-shot generalisation, we desire a way of reasoning about a *collection* of tasks, environment instances or levels: the need for generalisation emerges from the fact

we train and test the policy on different collections of environment instances. Consider as a didactic example OpenAI Procgen [Cobbe et al., 2020b]: in this benchmark suite, each game is a collection of procedurally generated levels. Which level is generated is completely determined by a level seed, and the standard protocol is to train a policy on a fixed set of 200 levels and then evaluate performance on the full distribution of levels. Almost all other benchmarks share this structure: they have a collection of levels or tasks, which are specified by some seed, ID or parameter vector, and generalisation is measured by training and testing on different distributions over the collection of levels or tasks. To give a different example, in the Distracting Control Suite [Stone et al., 2021], the parameter vector determines a range of possible visual distractions applied to the observation of a continuous control task, from changing the colours of objects to controlling the camera angle. While this set of parameter vectors has more structure than the set of seeds in Procgen, both can be understood within the framework I propose. See Section 2.4.2 for a discussion of the differences between these styles of environments.

To formalise the notion of a collection of tasks, I start with the Contextual Markov Decision Process (CMDP), as originally formalised by Hallak et al. [2015], but using the alternative formalism from Ghosh et al. [2021]. This formalism also builds on those presented by Doshi-Velez and Konidaris [2016], Perez et al. [2020], but I extend them to consider different distributions over context parameters; I include both observed and unobserved contexts settings; I include cases where the context is controllable; and I define formally how to produce subset CMDPs (see the end of this section for more discussion comparing the formalism I present here and existing works).

Definition 1. *A contextual MDP (CMDP) is a tuple*

$$\mathscr{M} = \left(S', A, O, R, T, C, \phi : S' \times C \to O, p(s'|c), p(c)\right).$$

A, O, R, T, ϕ are as in the definition of a POMDP above. C is the context space (a set over which it is possible to have a distribution). The CMDP is a POMDP with state space $S := S' \times C$, initial state distribution $p((s', c)) = p(c)p(s'|c)$, that is the POMDP $(S' \times C, A, O, R, T, \phi, p(s'|c)p(c))$. Hence, R has type $R : S' \times C \to \mathbb{R}$ and $T((s,c),a)$ is the form of the transition probability distribution. For the tuple to be a CMDP, the transition

function must be factored such that the context doesn't change within an episode, that is $T((s,c),a)((s',c')) = 0$ *if* $c' \neq c$*. I call* S' *the underlying state space, and* $p(c)$ *the context distribution.*

We choose to make the context the part of the state space so that every CMDP is also a POMDP (and possibly an MDP). To give an intuition for this definition, the context takes the role of the seed, ID or parameter vector which determines the level, hence why it should not change within an episode, only between episodes. The CMDP is the entire collection of tasks or environment instances; in Procgen, each game (e.g. starpilot, coinrun, etc.) is a separate CMDP. The context distribution $p(c)$ is what is used to determine the training and testing collections of levels, tasks or environment instances; in Procgen this distribution is uniform over the fixed 200 seeds at training time, and uniform over all seeds at testing time.

Note that this definition leaves it unspecified whether the context is observed by the agent: if $O = O' \times C$ for some underlying observation space O' and $\phi((s',c)) = (\phi'(s),c)$ for some underlying observation function $\phi' : S' \to O'$ then we say the context is observed, otherwise it isn't. The context needs to be observed for the CMDP to be an MDP (and not a POMDP), but the opposite isn't true - even if the context is observed, ϕ' could not be the identity, in which case the POMDP isn't likely to be an MDP. Note that I will generally use "MDP" to refer to environments that are either MDPs or POMDPs.

As the reward function, transition function, initial state distribution and emission function all take the context as input, the choice of context determines everything about the resulting MDP apart from the action space, which I assume is fixed.[1] Given a context c^*, I call the MDP resulting in the restriction of the CMDP $\mathscr{M}$ to the single context a *context-MDP* $\mathscr{M}_{c^*}$. Formally, this is a new CMDP with $p(c) := 1$ if $c = c^*$ else 0. This is a specific task or environment instance, for example, a single level of a game in Procgen, as specified by a single random seed that is the context.

Some MDPs have stochastic transition or reward functions. When these MDPs are simulated, researchers often have control of this stochasticity through the choice of a random seed. In theory, these stochastic MDPs could be considered deterministic contextual MDPs, where the context is the random seed. I do not consider stochastic MDPs as automatically

[1] One could also let the action space vary across contexts, but that problem is much less studied and outside of the scope of this survey.

contextual in this way and assume that the random seed is always chosen randomly, rather than being modelled as a context. This more closely maps to real-world scenarios with stochastic dynamics where we cannot control the stochasticity.

2.3.3 Training And Testing Contexts

I now describe the class of generalisation problems I focus on, using the CMDP formalism. As mentioned, the need for generalisation emerges from a difference between the training and testing environment instances, and so we want to specify both a set of training context-MDPs and a testing set. I specify these sets of context-MDPs by their context sets, as the context uniquely determines the MDP.

First, we need to describe how to use training and testing context sets to create new CMDPs.

Definition 2. *For any CMDP $\mathscr{M} = (S', A, O, R, T, C, \phi, p(s'|c), p(c))$, we can choose a subset of the context set $C' \subseteq C$, and then produce a new CMDP*

$$\mathscr{M}|_{C'} = \left(S', A, O, R, T, C', \phi, p(s'|c), p'(c)\right)$$

where $p'(c) = \frac{p(c)}{Z}$ if $c \in C'$ else 0 and Z is a renormalisation term $Z = \sum_{c \in C'} p(c)$ that ensures $p'(c)$ is a probability distribution.

This allows us to split the total collection of context-MDPs into smaller subsets, as determined by the contexts. For example, in Procgen any possible subset of the set of all seeds can be used to define a different version of the game with a limited set of levels.

For the objective, we use the expected return of a policy:

Definition 3. *For any CMDP $\mathscr{M}$ we can define the expected return of a policy in that CMDP as*

$$\boldsymbol{R}(\pi, \mathscr{M}) := \mathbb{E}_{c \sim p(c)}[\mathscr{R}(\pi, \mathscr{M}_c)],$$

where $\mathscr{R}$ is the expected return of a policy in a (context) MDP and $p(c)$ is the context distribution as before.

I can now formally define the Zero-Shot Policy Transfer (ZSPT) problem class.

Definition 4 (Zero Shot Policy Transfer)**.** *A ZSPT problem is defined by a choice of CMDP $\mathscr{M}$ with context set C and a choice of training and testing context sets $C_{train}, C_{test} \subseteq C$. The objective is to produce a non-Markovian policy $\pi : H[O,A] \to A$ which maximises the expected return in the testing CMDP $\mathscr{M}|_{C_{test}}$:*

$$J(\pi) := \boldsymbol{R}(\pi, \mathscr{M}|_{C_{test}}).$$

This policy can be produced through interaction with the training CMDP $\mathscr{M}|_{C_{train}}$ for a fixed number of environment and episode samples N_s, N_e respectively.

ZSG research is generally concerned with developing algorithms that can solve a variety of ZSPT problems. For example, in Procgen we aim to produce an algorithm that can solve the ZSPT problem for every game. Specifically, we want to achieve the highest return possible on the testing distribution (which is the full distribution over levels) after training for 25 million steps ($N_s = 25 \times 10^6, N_e = \infty$) on the training distribution of levels (which is a fixed set of 200 levels). The name *Zero-Shot Policy Transfer* comes from prior works [Harrison et al., 2017, Higgins et al., 2017b].

Some algorithms assume that the context distribution can be adjusted during interaction with the training CMDP, as long as sampled contexts are only within the fixed training context set:

Definition 5 (ZSPT controllable context)**.** *A* controllable context *ZSPT problem is the same as a ZSPT problem above, except the learning algorithm can adjust the context distribution of the training CMDP $p_{train}(c)$ during training, so long as it maintains the property of only sampling from the training context set: $p_{train}(c) = 0$ if $c \notin C_{train}$*

Note that this formalism defines a class *of problems, each determined by a choice of CMDP, training and testing context sets and whether the context is controllable*. This means that I do not make any assumptions about shared structure within the CMDP between context-MDPs: for any specific problem some assumption of this kind (either implicit or explicit) will likely be required for learning to occur (Section 2.3.5), but I do not believe there is a unifying assumption behind all ZSG problems apart from those stated here.

Evaluating Zero-Shot Generalisation. We can use the gap between training and testing performance as a measure of generalisation:

$$\text{GenGap}(\pi) := \mathbf{R}(\pi, \mathscr{M}|_{C_{\text{train}}}) - \mathbf{R}(\pi, \mathscr{M}|_{C_{\text{test}}}). \tag{2.1}$$

This metric is frequently used in the literature in addition to test performance to evaluate ZSG algorithms [Jiang et al., 2021b, Raileanu et al., 2021, Raileanu and Fergus, 2021]. In general, it is not clear what the best evaluation metric for ZSG algorithms is. In supervised learning, the generalisation capabilities of different algorithms are usually evaluated via final performance on an evaluation task. When the tasks used to evaluate a model are close to (or the same as) the tasks that the model will eventually be deployed on, it is clear that final performance is a good metric to evaluate on. However, in RL the benchmark tasks we use are often very dissimilar to the eventual real-world tasks we want to apply these algorithms to. Further, RL algorithms are currently still quite brittle and performance can vary greatly depending on hyperparameter tuning and the specific task being used [Henderson et al., 2018b]. In this setting, we may care more about the zero-shot generalisation *potential* of algorithms by decoupling generalisation from training performance and evaluating using the generalisation gap instead. For example, if algorithm A has higher testing performance than algorithm B, but A also has a much larger generalisation gap, we may prefer to use algorithm B in a new setting, so we have better assurance that the deployment performance won't deviate as much from the training performance, and the algorithm may be more robust. This is the reason the previous literature has often reported this metric alongside test performance.

However, the generalisation gap in RL has the same problems as discussed for the supervised learning generalisation gap: a gap of zero doesn't necessarily imply good performance (i.e. a random policy is likely to get a gap of 0), and if the reward functions aren't comparable across training and testing, then the magnitude of the gap my not be informative (and it could then only be used to compare between different algorithms). This means using it as the only metric for improved performance will likely not lead to robust progress in ZSG. Further, given how broad the current set of assumptions is, it is unlikely there is a single general measure of progress towards tackling ZSG: across such a broad problem class, objectives may even be conflicting [Wolpert and Macready, 1997].

Therefore, my recommendation is first and foremost to focus on problem-specific benchmarks and revert to the SL standard of using overall performance in specific settings (e.g. visual distractors, stochastic dynamics, sparse reward, hard exploration). The generalisation performance of various RL algorithms is likely contingent on the type of environment they are deployed on and therefore careful categorisation of the type of challenges preseNt at deployment is needed to properly evaluate ZSG capability (for further discussion see Section 2.4.3 and Section 2.6.2). As in the literature, generalisation gap can be used as an additional auxiliary metric to evaluate the performance of ZSG algorithms as well as test performance, to either break ties between algorithms with very similar test time performance, or to inform users in situations where it is more important to have strong assurances on test time performance than to have expected test time performance as high as possible.

2.3.4 Real World Examples of This Formalism

I chose this formalism as it is simple to understand, captures all the problems I am interested in, and is based on prior work. To further justify why this formalism is useful, and to give intuition about how it can be used in a variety of settings, I give several examples of real-world scenarios where this formalism naturally applies:

- Sim-to-real is a classic problem of ZSG and one which can be captured in this framework. Here the outer CMDP is a union between the simulation and real-world MDPs. The context set will be split into those contexts which correspond to simulation, and those that correspond to reality. The context generally conditions the dynamics, observation function and state distribution, but likely not the reward (so $\forall s', c : R((s', c)) = R'(s')$). Domain randomisation approaches are motivated by the idea that producing a wide range of possible contexts in simulation (the training CMDP) will make it more likely that the testing distribution of contexts is closer to the expanded training distribution. In a simulation setting, we'd normally assume access to the context distribution, and that the context could be made observable, but that the context won't be observable at testing time, and so it may not be useful to have a policy that explicitly conditions on the context.

- Healthcare is a promising domain for deploying future RL methods, as there are many

sequential decision-making problems. For example, the task of diagnosing and treating individual patients can be understood as a CMDP where the patient effectively specifies the context: patients will react differently to tests and treatments (dynamics variation) and may provide different measurements (state variation). Generalising to treating new patients is then exactly generalising to novel contexts. In this setting, we may be able to assume some part of the context (or some information about the context) is observable, as we will have access to the patient's medical history and personal information.

- Autonomous vehicles are another area where RL methods could be applied. These vehicles will be goal-conditioned in some sense, such that they can perform different journeys, which means that the context will likely control the reward function and that the part of the context that controls the read function will be observable (so the policy knows what task to perform). Driving in different locations (different contexts changing the initial state distribution), under different weather and lighting conditions due to the time of day (observation functions) and on different road surfaces (transition functions) are all problems that need to be tackled by these systems. We can understand this in the CMDP framework, where the context contains information about the weather, time of day, location and goal, as well as information about the state of the current vehicle.[2] Some of this context will be observed directly, and some may be inferred from observation. In this setting, we may only be able to train in certain contexts (i.e. certain cities, or restricted weather conditions), but we require the policy to generalise zero-shot to the unseen contexts well.

2.3.5 Additional Assumptions For More Feasible Generalisation

In choosing the CMDP formalism I opted to formalise ZSG in a way that captures the full class of problems I am concerned with, but this means that it is almost certainly impossible to prove any formal theoretical guarantees on learning performance using solely the CMDP structural assumptions. While I do not prove this, it is easy to see how one could design pathological CMDPs where generalisation to new contexts is entirely impossible without strong domain knowledge of the new contexts.

[2]Here I am assuming that weather, time of day and rough location are not changing meaningfully during an episode.

To have any chance of solving a specific ZSG problem then, further assumptions (either explicit or implicit) have to be made. These could be assumptions on the type of variation, the distributions from which the training and testing context sets are drawn, or additional underlying structure in the context set. I describe several popular or promising assumptions here and note that the taxonomy in Section 2.4 also acts as a set of possible additional assumptions to make when tackling a ZSG problem.

Assumptions on the Training and Testing Context Set Distributions. One assumption which is often made is that while the training and testing context sets are not identical, the elements of the two sets have been drawn from the same underlying distribution, analogously to the IID data assumption in supervised learning. For example, this is the setup of OpenAI Procgen [Cobbe et al., 2020b], where the training context set is a set of 200 seeds sampled uniformly at random from the full distribution of seeds, and the full distribution is used as the testing context set.

However, many works on ZSG in RL do not assume that the train and test environment instances are drawn from the same distribution. This is often called *Domain Generalisation*, where we refer to the training and testing environments instances as different *domains* that may be similar but are not from the same underlying generative distribution. Concrete examples occur in robotics such as the *sim-to-real* problem.

Note that while the testing context set could be a single context, this would likely lead to a not particularly robust algorithm - it's possible that it overfits to the problem of producing a policy that performs well on this specific context, which may not generalise to other similar contexts (which is the high-level goal of this research direction).

Further Formal Assumptions of Structure. Another kind of assumption that can be made is on the structure of the CMDP itself, e.g. the context space or transition function. There are several families of MDPs with additional structure which could enable ZSG. However, these assumptions are often not explicitly made when designing benchmarks and methods, which can make understanding why and how generalisation occurs difficult. A detailed discussion and formal definitions for these structures can be found in Appendix A.2, but I provide a high-level overview here, focusing on assumptions that have been used in practice, and those that hold particular promise for ZSG.

An example of a structured MDP that has been used to improve generalisation is the *block MDP* [Du et al., 2019]. It assumes a *block structure* in the mapping from a latent state space to the given observation space, or that there exists another MDP described by a smaller state space with the same behaviour as the given MDP. This assumption is relevant in settings where we only have access to high-dimensional, unstructured inputs, but know that there exists a lower-dimensional state space that gives rise to an equivalent MDP. Du et al. [2019] use this assumption for improved bounds on exploration that relies on the size of the latent state space rather than the given observation space. Zhang et al. [2020a] develop a representation learning method that disentangles relevant from irrelevant features, improving generalisation to environments instances where only the irrelevant features change, a simple form of *systematicity* (Appendix A.1, [Hupkes et al., 2020]). This is a rare example of a method explicitly utilising additional assumptions of structure to improve generalisation. Block MDPs can be combined with contextual MDPs by introducing an emission mapping from state space to observation space that is also dependent on context, as defined by [Sodhani et al., 2022].

Factored MDPs [Boutilier et al., 2000, Strehl et al., 2007] can be used to describe object-oriented environments or multi-agent settings where the state space can be broken up into independent factors, i.e. with sparse relationships over the one-step dynamics. This can be leveraged to learn dynamics models that explicitly ignore irrelevant factors in prediction or to compute improved sample complexity bounds for policy learning [Hao et al., 2021] and seems particularly relevant for generalisation as additional structure in the context set could map onto the factored structure in the transition and reward functions. An initial example of using a similar formalism to a factored MDP in a multi-domain RL setting is demonstrated by Huang et al. [2022], although it does not target the zero-shot policy transfer setting directly. I note that contextual MDPs can be trivially represented by factored MDPs with two factors, the state and context. However, factored MDPs are capable of modelling more structure in a domain if present. Therefore, if an environment is capable of being modelled by a factored MDP in addition to a contextual MDP, better generalisation guarantees and results are likely possible if this structure is exploited. I hope to see more work applying these kinds of structural assumptions to the zero-shot generalisation problems discussed in this work.

2.3.6 Remarks And Discussion

Angles to Tackle the ZSPT Problem. While we aim to improve test-time performance Definition 3, we often do not have access to that performance metric directly (or will not when applying our methods in the real world). In research, to develop algorithms that improve the test-time performance, we could aim to produce algorithms that (when compared to existing work) either (1) increase the train-time performance while keeping the generalisation gap constant; (2) decrease the generalisation gap while keeping the train-time reward constant; or (3) do a mixture of the two approaches. Work in RL not concerned with generalisation tends to (implicitly) take the first approach, assuming that the generalisation gap will not change.[3] Work on ZSG in RL instead normally aims at (2) reducing the generalisation gap explicitly, which may reduce train-time performance but increase test-time performance. Some work also aims at (1) improving train-time performance in a way that is likely to keep the generalisation gap constant.

Motivating Zero-Shot Policy Transfer In this work, we focus on zero-shot policy transfer [Harrison et al., 2017, Higgins et al., 2017b]: a policy is learned from the training CMDP and evaluated zero-shot in the testing CMDP. This field is important for several reasons.

First, as mentioned above, current deep RL algorithms often aren't safe or sample-efficient enough to perform online learning in the real world. Hence, a significant amount of offline or in-simulation training is required, and policies need to generalise zero-shot to the real-world deployment setting. As access to more compute and richer simulations becomes available, I expect many successful real-world deployments of RL to follow this workflow at least in part: training offline or in simulation and then transferring the policy zero-shot to the deployment environment. Even if the policy will continue learning during deployment, it still needs to be reasonably good at deployment time (i.e. zero-shot), otherwise it wouldn't be safe to deploy it. In this way, I view work on ZSG as mostly complementary to work on continual RL, as I think both are important for enabling the deployment of robust and competent RL policies. Note that there may be tradeoffs between good zero-shot performance and good continual learning performance, and how to choose between these two desiderata will be determined by the specific problem setting being faced.

[3] If the training environment instances are identical to the testing environment instances, then the generalisation gap will always be 0.

Second, from a safety, interpretability and verification perspective, ZSG may be preferable to a continually updated policy. It's likely in high-stakes scenarios that models will be verified [Katz et al., 2019] or audited with interpretability or explainability methods [Milani et al., 2022], and that this process will be expensive. In this scenario, it will be beneficial to have a single model which performs well zero-shot without having to be continually updated, as after each update these verification and auditing steps will likely have to be repeated.

Finally, note that while I do not cover methods that relax the zero-shot assumption, I believe that in a real-world scenario it will likely be possible to do so.[4] However, zero-shot policy transfer is still a useful problem to tackle, as solutions are likely to help with a wide range of settings resulting from different relaxations of the assumptions made here: zero-shot policy transfer algorithms can be used as a base which is then built upon with domain-specific knowledge and extra data.

Note that while "training" and "learning" are contentious terms, my definition grounds them in the production of a non-Markovian policy after some number of samples from the training context MDP. The fact that the objective is the expected testing return within a *single episode* of the non-Markovian policy means that online learning or adaptation across more than 1 episode isn't possible, and so the adaptation would have to happen within a single episode to be useful (i.e. using a recurrent policy). Several methods do take this approach, as described in Section 2.5.2.4. To be clear, we are grounding ourselves in this specific *objective*, and not placing any restrictions on the *properties* of methods as long as they satisfy the constraints: policies need not be Markovian, and can adapt within a single episode to the environment instances they are placed in if that improves performance.

Relationship to Previous Notions of Generalisation in RL. Historically, generalisation in RL has referred to the notion of generalising to novel states or state-action pairs within a single MDP. While this notion is useful in the online learning single-MDP setting that is the focus of that work, I here focus on a more recent and more realistic setting where the policy is trained on a collection of MDPs and then deployed on possibly unseen MDPs. This setting more closely mirrors the supervised learning notion of generalisation. I believe this

[4] For example by using unsupervised data or samples in the testing environment instances, utilising some description of the contexts such that zero-shot generalisation is possible, or enabling the agent to train in an online way in the testing context-MDPs.

type of workflow for deploying RL-trained policies in the real world is much more feasible than training a policy online from scratch, as current RL algorithms aren't sample-efficient or safe enough to train online in the real world. This means a large amount of off-line or in-simulation training will have to occur before the policy is deployed, and the deployed policy will still need to perform reasonably well as soon as it's deployed (i.e. zero-shot).

Relationship to Previous Formalisms of Collections or Distributions of MDPs. As mentioned previously, the formalism I presented above builds on multiple previous works. Here I briefly present the differences between my formalism and these works.

Doshi-Velez and Konidaris [2016] present Hidden-parameter MDPs (Hi-MDPs). These are MDPs with a hidden parameter which controls the transition function and a distribution over the set of these hidden parameters that implicitly defines a set of MDPs. Perez et al. [2020] builds on Hi-MDPs to present Generalised Hidden-parameter MDPs (GHP-MDPs), where the hidden parameters now also control the reward function in addition to the dynamics. My work uses a context parameter which is analogous to the hidden parameter, which may be hidden or observed, and that controls the initial state distribution, transition function and reward function, rather than just the transition and reward functions. I also more formally discuss how different distributions of context parameters may be used during training and testing.

Harrison et al. [2017], Higgins et al. [2017b] introduce variations on the term "zero-shot policy transfer", which I use as the name for the formal class of problems I study. They both study problems within this class but don't formalise the entire class of problems as I do here, instead focusing on presenting methods for improving performance in the empirical settings they investigate.

Song et al. [2020] discusses a distribution over MDPs which are sampled from during training, and uses a kind of CMDP but where the context parameter adjusts only the observation function, rather than any other parts of the MDP. They also assume the distribution over MDPs is the same between training and testing.

Hallak et al. [2015], Ghosh et al. [2021] present Contextual MDPs (CMDPs), which is the formalism I use as the base of my definitions. Ghosh et al. [2021] present a formalism based on the context being part of an underlying state space, which is the one I use, while

Hallak et al. [2015] present a formalism more similar to Hi-MDPs, where there is a collection of MDPs parameterised by a context variable. Both of these works don't consider a shift in distribution over contexts between training and testing apart from a shift to the full distribution in Ghosh et al. [2021]'s work. They don't present a formal definition of the policy class and don't define problem settings where the context distribution is controllable.

2.4 Benchmarks For Zero-shot Generalisation In Reinforcement Learning

In this section, I give a taxonomy of benchmarks for ZSG in RL. A key split in the factors of variation for a benchmark is those factors concerned with the environment and those concerned with the evaluation protocol. A benchmark task is a combination of a choice of the environment (a CMDP, covered in Section 2.4.1) and a suitable evaluation protocol (a train and test context set, covered in Section 2.4.2). This means that all environments support multiple possible evaluation protocols, as determined by their context sets.

Having categorised the set of benchmarks, I point out the limitations of the purely PCG approach to building environments (Section 2.4.3), as well as discuss the range of difficulty among ZSG problems (Section 2.4.3). More discussion of future work on benchmarks for ZSG can be found in Sections 2.6.1, 2.6.2 and 2.6.4.

2.4.1 Environments

Categorising Environments That Enable Generalisation In Table 2.1, I list the available environments for testing ZSG in RL, as well as summarise each environment's key properties. These environments all provide a non-singleton context set that can be used to create a variety of evaluation protocols. Choosing a specific evaluation protocol then produces a benchmark. I describe the meaning of the columns in Table 2.1 here.

Style. This gives a rough high-level description of the kind of environment.

Contexts. This describes the context set. In the literature, there are two approaches to designing a context set, and the key difference between these approaches is whether the context-MDP creation is accessible and visible to the researcher. The first, which I refer to as Procedural Content Generation (PCG), relies on a single random seed to determine multiple

Name	Style	Contexts	Variation
Alchemy † [Wang et al., 2021a]	3D	PCG	D, R, S
Animal-AI [Crosby et al., 2020]	3D	D-C, D-O	S, O
Atari Game Modes [Machado et al., 2017]	Arcade	D-C	D, O, S
BabyAI [Chevalier-Boisvert et al., 2019]	Grid, LC	D-C, D-O, PCG	R, S
CARL [Benjamins et al., 2021]	Varied	Con, D-C, D-O	D, O, R, S
CARLA [Fan et al., 2021, Zhu et al., 2020]	3D, Driving	D-C	O
CausalWorld † [Ahmed et al., 2021]	3D, ConCon	Con, D-C, D-O	D, O, R, S
Construction [Bapst et al., 2019]	2D, Structured	Con, D-C, D-O, PCG	R, S
Crafter [Hafner, 2022]	Arcade, Grid	PCG	S
Crafting gridworld [Chen et al., 2021b]	Grid, LC	D-C	R, S
DACBench [Eimer et al., 2021]	Structured	PCG, D-C	D, R, S
DCS [Stone et al., 2021]	ConCon	Con, D-C	O
DistractingCarRacing [Tang et al., 2020, Brockman et al., 2016b]	Arcade	D-C	O
DistractingVizDoom [Tang et al., 2020, OpenAI, 2017]	3D	D-C	O
DMC-GB [Hansen and Wang, 2021]	ConCon	Con, D-C	O
DMC-Remastered [Grigsby and Qi, 2020]	ConCon	Con, D-C	O
DM-Memory [Fortunato et al., 2019]	Arcade, 3D	PCG, Con, D-O, D-C	R, D, S
GenAsses [Packer et al., 2019]	ConCon	Con	D, S
GVGAI [Perez-Liebana et al., 2019b]	Grid	D-C	D, O, S
HALMA † [Xie et al., 2021]	Grid	D-C, D-O	O, S
iGibson [Fan et al., 2021, Shen et al., 2021]	3D	D-C	O, S
Jericho † [Hausknecht et al., 2020]	Text	D-C	D, R, S
JumpingFromPixels [Tachet et al., 2020]	Arcade	Con	S
KitchenShift [Xing et al., 2021]	3D, ConCon	D-C	O, S, R
Malmo [Johnson et al., 2016]	3D, Arcade	D-C, D-O	R, S
MarsExplorer [Koutras et al., 2021]	Grid	PCG	S
MazeExplore [Harries et al., 2019]	3D	PCG	O, S
MDP Playground † [Rajan et al., 2021]	ConCon, Grid	Con, D-C, D-O	D, O, R, S
Meta-World † [Yu et al., 2019b]	3D, ConCon	Con, D-C	R, S
MetaDrive [Li et al., 2023b]	3D, Driving	D-C, D-O, PCG	D, S
MiniGrid [Chevalier-Boisvert, 2021]	Grid	PCG	S
MiniHack [Samvelyan et al., 2021]	Grid	D-C, D-O, PCG	S
NaturalEnvs CV [Zhang et al., 2018b]	Grid	PCG	O, R, S
NaturalEnvs MuJoCo [Zhang et al., 2018b]	ConCon	D-C	O
NLE [Küttler et al., 2020b]	Grid	PCG	S
Noisy MuJoCo [Zhao et al., 2019]	ConCon	Con, D-C	D, O
NovelGridworlds [Goel et al., 2021]	Grid	D-C	D, S
Obstacle Tower [Juliani et al., 2019b]	3D	D-C, PCG	O, S
OffRoadBenchmark [Dosovitskiy et al., 2017, Han et al., 2021]	3D, Driving	D-C,	O, S, R
OpenAI Procgen [Cobbe et al., 2020b]	Arcade	PCG	O, S
OverParam Gym [Song et al., 2020]	ConCon	Con	O
OverParam LQR [Song et al., 2020]	LQR	Con	O
ParamGen [Ke et al., 2021]	3D, LC	D-C, D-O	R, S
RLBench † [James et al., 2019a]	3D, ConCon, LC	Con, D-C, D-O	R, S
RoboSuite [Fan et al., 2021, Zhu et al., 2020]	3D, ConCon	D-C	O
Rogue-gym [Kanagawa and Kaneko, 2019b]	Grid	PCG	S
RTFM [Zhong et al., 2021]	Grid, LC	PCG	D, R, S
RWRL † [Dulac-Arnold et al., 2021]	ConCon	Con	D
Sokoban [Racanière et al., 2017]	Grid	PCG	S
TextWorld † [Côté et al., 2019]	Text	Con, D-C, PCG	D, O, R, S
Toybox † [Tosch et al., 2019]	Arcade	Con, D-C, D-O	D, O, S
TrapTube [Wenke et al., 2019]	Grid	Con, D-C	D, O, S
WordCraft [Jiang et al., 2020]	LC, Text	Con, D-C, D-O	R, S
XLand [Team et al., 2021]	3D, LC	Con, D-C, D-O, PCG	D, O, R, S
Phy-Q [Xue et al., 2021]	Arcade	D-C, PCG	S

Table 2.1: Categorisation of Environments for ZSG. In the **Style** column, LC stands for Language-Conditioned, ConCon for Continuous Control. In the **Contexts** column, PCG stands for Procedural Content Generation, Con for continuous, D-C for discrete cardinal and D-O for discrete ordinal. In the **Variation** column, S, D, O and R are respectively state, dynamics, observation or reward function variation. In the **Name** column, † refers to environments that were not originally designed as zero-shot policy transfer benchmarks but could be adapted to be. See main text for a more detailed description of the columns.

choices during the context-MDP generation. Here the context set is the set of all supported random seeds. This is a black-box process in which the researcher only chooses a seed.

The second approach provides more direct control over the factors of variation between context-MDPs, and I call these *Controllable* environments. The context set is generally a

product of multiple factor spaces, some of which may be discrete (i.e. a choice between several colour schemes) and some continuous (i.e. a friction coefficient in a physical simulation). Borrowing from Ke et al. [2021], a distinction between discrete factors of variation is whether they are cardinal (i.e. the choices are just a set with no additional structure) or ordinal (i.e. the set has additional structure through an ordering). Examples of cardinal factors include different game modes or visual distractions, and ordinal factors are commonly the number of entities of a certain type within the context-MDP. All single-dimensional continuous factors are effectively also ordinal factors, but this may not necessarily be the case for multi-dimensional continuous features.

Previous literature has defined PCG as any process by which an algorithm produces MDPs given some input [Risi and Togelius, 2020], which applies to both kinds of context sets I have described. Throughout the rest of this survey I use "PCG" to refer to black-box PCG, which uses a seed as input, and "controllable" to refer to environments where the context set directly changes the parameters of interest in the context-MDPs, which could also be seen as "white-box PCG". We can understand (black-box) PCG settings as combinations of discrete and continuous factor spaces (i.e. controllable environments) where the choice of the value in each space is determined by the random generation process. However, only some environments make this more informative parametrisation of the context-MDPs available. In this table, I describe environments where this information is not easily controllable as PCG environments. See Section 2.4.3 for a discussion of the downsides of purely PCG approaches.

Variation. This describes what varies within the set of context MDPs. This could be state-space variation (the initial state distribution and hence implicitly the state space), dynamics variation (the transition function), visual variation (the observation function) or reward function variation. Where the reward varies, the policy often needs to be given some indication of the goal or reward, so that the set of contexts is solvable by a single policy [Irpan and Song, 2019].

2.4.1.1 Trends In Environments

There are several trends and patterns shown in Table 2.1, which I draw the reader's attention to here. I describe 55 environments in total and have aimed to be fairly exhaustive.

There are a range of different **Style**s that these environments have, which is beneficial

as ZSG methods should themselves be generally applicable across styles if possible. While numerically there is a focus on gridworlds (14, 25%) and continuous control (13, 24%) there are well-established benchmarks for arcade styles [Cobbe et al., 2020b] and 3D environments [Juliani et al., 2019b]. Looking at **Context** sets, we see that PCG is heavily used in ZSG environments, featuring in 21 (38%) environments. Many environments combine PCG components with controllable variation [Chevalier-Boisvert et al., 2019, Côté et al., 2019, Juliani et al., 2019b, Li et al., 2023b, Team et al., 2021, Xue et al., 2021, Fortunato et al., 2019, Eimer et al., 2021, Bapst et al., 2019]. Most environments have several different kinds of factors of variation within their context set.

There are a lot of differences between environments when looking at the **Variation** they use. Numerically, state variation is most common (42, 76%) followed by observation (29, 53%), and then reward (20, 36%) and dynamics (19, 35%). Most environments have multiple different types of variation (34, 62%), and while there are several environments targeted at just observation variation (10, 18%) or state variation (9, 16%), there is only a single environment with solely dynamics variation (RWRL, [Dulac-Arnold et al., 2021]), and none with solely reward variation. State and Observation variations are often the easiest to engineer, especially with the aid of PCG. This is because changing the rendering effects of a simulator, or designing multiple ways the objects in a simulator could be arranged, is generally easier than designing a simulator engine that is parameterisable (for dynamics variation). Creating an environment for reward variation requires further design choices about how to specify the reward function or goal such that the environment satisfies the Principle Of Unchanged Optimality [Irpan and Song, 2019]. PCG is often the only good way of generating a large diversity in state variation, and as such is often necessary to create highly varied environments. Only CausalWorld [Ahmed et al., 2021] enables easy testing of all forms of variation at once.[5]

There are several clusters that can be pointed out in the collection of benchmarks: There are several PCG state-varying gridworld environments (MiniGrid, BabyAI, Crafter, Rogue-gym, MarsExplorer, NLE, MiniHack; [Chevalier-Boisvert, 2021, Chevalier-Boisvert et al.,

[5]While CARL [Benjamins et al., 2021], MDP Playground [Rajan et al., 2021], XLand [Team et al., 2021] and TextWorld [Côté et al., 2019] are also categorised as containing all forms of variation, CARL is a collection of different environments, none of which have all variation types, XLand is not open-source, and MDP Playground and TextWorld would require significant work to construct a meaningful evaluation protocol which varies along all factors. Further, MDP Playground does not provide a method for describing the changed reward function to the agent, and there is uncertainty in how to interpret observation variation in text-based games like TextWorld.

2019, Hafner, 2022, Kanagawa and Kaneko, 2019b, Koutras et al., 2021, Küttler et al., 2020b, Samvelyan et al., 2021]), non-PCG observation-varying continuous control environments (RoboSuite, DMC-Remastered, DMC-GB, DCS, KitchenShift, NaturalEnvs MuJoCo; [Fan et al., 2021, Grigsby and Qi, 2020, Hansen and Wang, 2021, Stone et al., 2021, Xing et al., 2021, Zhang et al., 2018b]), and multi-task continuous control benchmarks which could be adapted to ZSG (CausalWorld, RLBench, Meta-world; [Ahmed et al., 2021, James et al., 2019a, Yu et al., 2019b]).

2.4.2 Evaluation Protocols For ZSG

As discussed, a benchmark is the combination of an environment and an evaluation protocol. Each environment supports a range of evaluating protocols determined by the context set, and often there are protocols recommended by the environment creators. In this section, I discuss the protocols and the differences between them. An evaluation protocol specifies the training and testing context sets, any restrictions on sampling from the training set during training, and the number of samples allowed from the training environment.

An important first attribute that varies between evaluation protocols is *context-efficiency*. This is analogous to sample efficiency, where only a certain number of samples are allowed during training, but instead, we place restrictions on the number of *contexts*. This ranges from a single context to a small number of contexts, to the entire context set.

PCG Evaluation Protocols. In fact, in purely PCG environments, the only meaningful factor of variation between evaluation protocols is the context efficiency restriction. As we have no control over the factors of variation apart from sampling random seeds, the only choice we have is how many contexts to use for training. Further, the only meaningful testing context set is the full distribution, as taking a random sample from it (the only other option) would just be an approximation of the performance on the full distribution. This limitation of PCG environments is discussed further below (Section 2.4.3).

This gives three classes of evaluation protocol for PCG environments, as determined by their training context set: A single context, a small set of contexts, or the full context set. These are visualised in Fig. 2.2 **A**, **B** and **C** respectively. There are not any examples of protocol **A** (for purely PCG environments), likely due to the high difficulty of such a challenge. For protocol **B**, while "a small set of contexts" is imprecise, the relevant point is that this set

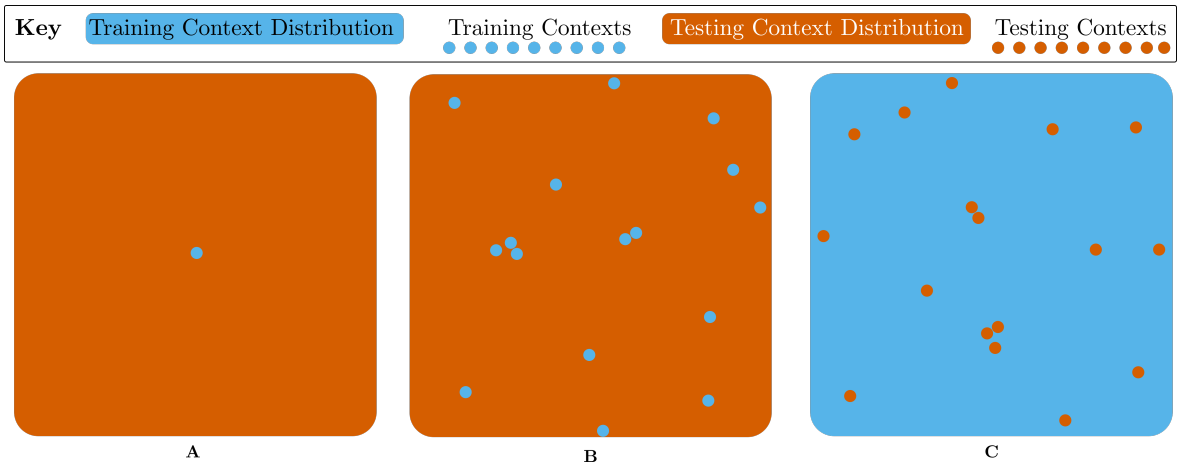


Figure 2.2: Visualisation of Evaluation Protocols for PCG Environments. **A** is a single training context, and the whole context set for testing. **B** uses a small collection of training contexts randomly sampled from the context set and the entire space for testing. **C** effectively reverses this, using the entire context set for training apart from several randomly sampled held-out contexts that are used for testing. The lack of axes indicates that these sets have no structure.

is meaningfully different from the full context set: it is possible to overfit to this set without getting good performance on the testing set. Examples of this protocol include both modes of OpenAI Procgen [Cobbe et al., 2020b], RogueGym [Kanagawa and Kaneko, 2019b], some uses of JumpingFromPixels [Tachet et al., 2020] and MarsExplorer [Koutras et al., 2021].

Protocol **C** is commonly used among PCG environments that are not explicitly targeted at ZSG (MiniGrid, NLE, MiniHack, Alchemy; [Chevalier-Boisvert, 2021, Küttler et al., 2020b, Samvelyan et al., 2021, Wang et al., 2021a]). The testing context set consists of seeds held out from the training set, and otherwise during training the full context set is used. This protocol effectively tests for more robust RL optimisation improvements but does not test for zero-shot generalisation beyond avoiding memorising. While this protocol only tests for ZSG in a weak sense, it still matches a wider variety of real-world deployment scenarios than the previous standard in RL, where the evaluation of the policy is performed on the environment it is trained in, and so I believe it should be the standard evaluation protocol in RL (not just in ZSG), and the previous standard should be considered a special case.

Controllable Environment Evaluation Protocols. Many environments do not use only PCG and have factors of variation that can be controlled by the user of the environment. In these controllable environments, there is a much wider range of possible evaluation protocols.

The choice in PCG protocols — between a single context, a small set, or the entire range

of contexts — transfers to the choice for each factor of variation in a controllable environment. For each factor, we can choose one of these options for the training context set, and then choose to sample either within or outside this range for the testing context set. The range of options is visualised in Fig. 2.3.

Making this choice for each factor independently gives us a convex training context set within the full context set (Fig. 2.3 **A**). For testing, each factor can then be either inside or outside this convex set (often respectively referred to as interpolation and extrapolation). The number of factors chosen to be extrapolating contributes to the difficulty of the evaluation protocol.

However, if we create correlations or links between values of factors during training, we can get a non-convex training context set within the full context set (Fig. 2.3 **B**). Each possible

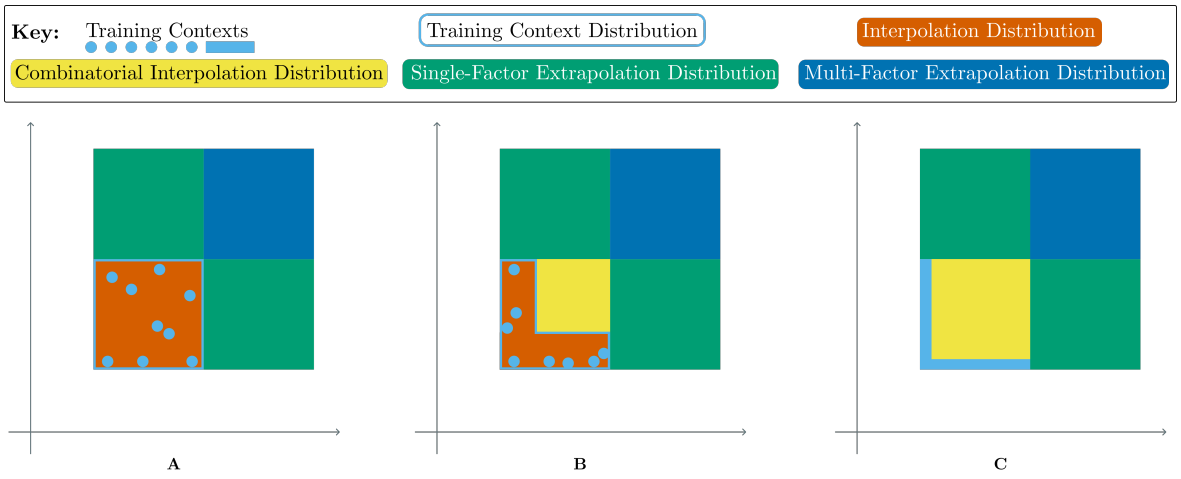


Figure 2.3: Visualisation of Evaluation Protocols for Controllable Environments. Each diagram visualises one possible training context set (blue), and multiple possible testing context sets (all other colours). In **A** we choose the range for each factor of variation independently for the training distribution, resulting in a convex shape for this distribution. In this setting, possible testing distributions can either be interpolation (red), extrapolation along a single factor (either green square) or extrapolation along both factors (blue). In **B** and **C** the ranges for each factor of variation are linked together, resulting in a non-convex shape for the training distribution. This allows an additional type of generalisation to be tested, combinatorial interpolation (yellow), where the factors take values seen during training independently, but in unseen combinations. We continue to have the previous interpolation and extrapolation testing distributions. The difference from **B** to **C** is in the width of the training distribution in the axes along which we expect the agent to generalise. In **C** the policy will not be able to learn that the two factors can vary independently at all, making all forms of generalisation harder. Note that in actual environments and real-world settings it is likely this space will be higher than two dimensions and contain non-continuous and non-ordinal axes. The axes indicate that in this setting we have control over these factors of variation, in contrast to Fig. 2.2.

testing context can either be within the training context set (fully interpolation), within the set formed by taking the convex hull of the non-convex training context set (combinatorial interpolation), or fully outside the convex hull (extrapolation). Combinatorial interpolation tests the ability of an agent to exhibit *systematicity*, a form of compositional ZSG discussed in Appendix A.1. For ordinal factors, we can also choose disjoint ranges, which allows us to test interpolation along a single axis (i.e. taking values between the two ranges). Note that when discussing convex hulls, this only applies to factors of variation that are continuous or discrete-ordinal; for cardinal factors of variation, the convex hull just includes those values sampled during training.

For example, consider a policy trained in a CMDP where the context set consists of values for friction and gravity strength. During training, the environments instances have either a friction coefficient between 0.5 and 1 but gravity fixed at 1, or gravity strength ranging between 0.5 and 1 but friction fixed at 1 (the light blue line in Fig. 2.3 **C**). Testing contexts which take friction and gravity values within the training distribution are full interpolation (e.g. $(f = 0.5, g = 1), (f = 1, g = 1)$), contexts which take values for friction and gravity which have been seen independently but not in combination are combinatorial interpolation (e.g. $(f = 0.5, g = 0.5), (f = 0.5, g = 0.9)$, the yellow area), and contexts which take values for friction and gravity which are outside the seen ranges during training are full extrapolation (e.g. $(f = 0.2, g = 0.5), (f = 1.1, g = 1.5)$, either the dark blue or green areas).

We can still consider the number of contexts within the training context set, which controls the density of the training context set, given its shape. When testing for extrapolation we can also vary the "width" of the training context set on the axis of variation along which extrapolation is being tested (Fig. 2.3 **B** vs **C**). These tests evaluate the agent's ability to exhibit *productivity* (Appendix A.1). Of course, ZSG will be easier if there is a wide diversity of values for this factor at training time, even if the values at test time are still outside this set. For example, if we are testing whether a policy can generalise to novel amounts of previously seen objects, then we should expect the policy to perform better if it has seen different amounts during training, as opposed to only having seen a single amount of the object during training. To expand on the friction and gravity example, during training the policy never sees gravity and friction varying together, which makes it much more difficult to

generalise to the testing contexts. If gravity and friction did vary together during training then this would make generalisation easier.

A notable point in this space is that of a single training context and a wide variety of testing contexts. This protocol tests for a strong form of ZSG, where the policy must be able to extrapolate to unseen contexts at test time. Because of the difficulty of this problem, benchmarks with this evaluation protocol focus on visual variation: the policy needs to be robust to different observation functions on the same underlying MDP [Grigsby and Qi, 2020, Stone et al., 2021, Xing et al., 2021]. The protocol is often motivated by the sim-to-real problem, where we expect an agent trained in a single simulation to be robust to multiple visually different real-world settings at deployment time.

Beyond this single point, it is challenging to draw any more meaningful categorisation from the current array of evaluation protocols. Generally, each one is motivated by a specific problem setting or characteristic of human reasoning which I believe RL agents should be able to solve or have respectively.

2.4.3 Discussion

There are several comments, insights and conclusions that can be gained from surveying the breadth of ZSG benchmarks, which I raise here.

Non-visual Generalisation. If testing for non-visual types of generalisation, then visually simple domains such as MiniHack [Samvelyan et al., 2021],Chapter 3 and NLE [Küttler et al., 2020b] should be used. These environments contain enough complexity to test for many types and strengths of non-visual generalisation but save on computation due to the lack of complex visual processing required. There are many real-world problem settings where no visual processing is required, such as systems control and recommender systems. Representation learning is still a problem in these non-visual domains, as many of them have a large variety of entities and scenarios which representations and hence policies need to generalise across.

DeepMind Control Suite Variants. A sub-category of ZSG benchmarks unto itself is the selection of DeepMind Control Suite [Tassa et al., 2018] variants: DMC-Remastered, DMC-Generalisation Benchmark, Distracting Control Suite, Natural Environments [Grigsby and Qi, 2020, Hansen and Wang, 2021, Stone et al., 2021, Zhang et al., 2018b]. All these environments focus on visual generalisation and sample efficiency, require learning continuous control

policies from pixels and introduce visual distractors that the policy should be invariant to which are either available during training or only present at deployment. I believe that Distracting Control Suite [Stone et al., 2021] is the most fully-featured variant in this space, as it features the broadest set of variations, the hardest combinations of which are unsolvable by current methods.

Unintentional Generalisation Benchmarks. Some environments listed in Table 2.1 were not originally intended as ZSG benchmarks. For example, Tosch et al. [2019] presents three highly parameterisable versions of Atari games and uses them to perform post hoc analysis of agents trained on a single variant. Some environments are not targeted at zero-shot policy transfer (CausalWorld, RWRL, RLBench, Alchemy, Meta-world [Ahmed et al., 2021, Dulac-Arnold et al., 2021, James et al., 2019a, Wang et al., 2021a, Yu et al., 2019b]), but could be adapted to such a scenario with a different evaluation protocol. More generally, all environments provide a context set, and many then propose specific evaluation protocols, but other protocols could be used as long as they were well-justified. This flexibility has downsides, as different methods can be evaluated on subtly different evaluation protocols which may favour one over another. I recommend being explicit when using these benchmarks in exactly which protocol is being used and comparing with evaluations of previous methods. Using a standard protocol aids reproducibility.

The Downsides of Procedural Content Generation for Zero-shot Generalisation. Many environments make use of procedural content generation (PCG) for creating a variety of context-MDPs. In these environments, the context set is the set of random seeds used for the PCG and has no additional structure with which to control the variation between context-MDPs.

This means that while PCG is a useful tool for creating a large set of context-MDPs, there is a downside to purely PCG-based environments: the range of evaluation protocols supported by these environments is limited to different sizes of the training context set. Measuring zero-shot generalisation along specific factors of variation is impossible without significant effort either labelling generated levels or unravelling the PCG to expose the underlying parametrisation which captures these factors. Often more effort is required to enable setting these factors to specific values, as opposed to just revealing their values for generated levels.

Hence, PCG benchmarks are testing for a "general" form of ZSG and RL optimisation, but do not enable more targeted evaluation of specific types of ZSG. This means making research progress on specific problems is difficult, as focusing on the specific bottleneck in isolation is hard.

An interesting compromise, which is struck by several environments, is to have some low-level portion of the environment procedurally generated, but still have many factors of variation under the control of the researcher. For example, Obstacle Tower [Juliani et al., 2019b] has procedurally generated level layouts, but the visual features (and to some extent the layout complexity) can be controlled. Another example is the MiniHack benchmark which we discuss in Chapter 3, where entire MDPs can be specified from scratch in a rich description language, and PCG can fill in any components if required. These both enable more targeted types of experimentation. I believe this kind of combined PCG and controllable environment is the best approach for designing future environments; some usage of PCG will be necessary to generate sufficient variety in the environments (especially the state space), and if the control is fine-grained enough to enable precise scientific experimentation, then the environment will still be useful for disentangling progress in ZSG.

Compositional Generalisation in Contextual MDPs. Compositional generalisation is a key point of interest for many researchers (see Appendix A.1). In *controllable* environments, different evaluation protocols enable us to test for some of the forms of compositional generalisation introduced in Appendix A.1 [Hupkes et al., 2020]: (1) Systematicity can be evaluated using a multidimensional context set, and testing on novel combinations of the context dimensions not seen at training time (combinatorial interpolation in Fig. 2.3). (2) Productivity can be evaluated with ordinal or continuous factors, measuring the ability to perform well in environment instances with context values beyond those seen at training time (either type of extrapolation in Fig. 2.3). If dealing with CMDPs where the context space is partially language, as in language-conditioned RL [Luketina et al., 2019], then the evaluations discussed by Hupkes et al. [2020] can be directly applied to the language space. Crucially, a controllable environment with a structured context space is required to test these forms of compositional generalisation, and to ensure that the agent is seeing truly novel combinations at test time; this is difficult to validate in PCG environments like OpenAI Procgen [Cobbe

et al., 2020b] or NLE [Küttler et al., 2020b].

The other forms of compositional generalisation listed by Hupkes et al. [2020] require additional structure not captured by the choices of evaluation protocol, and I describe what testing these forms could entail here: (3) substitutivity through the use of synonyms (in language) or equivalent objects and tools; (4) locality through comparing the interpretation of an agent given command A and command B separately vs. the combination of A + B and if those interpretations are different; and (5) overgeneralisation through how the agent responds to exceptions in language or rules of the environment.

What Generalisation Can We Expect? In Section 2.4.2 I discussed a variety of different possible evaluation protocols for different styles of ZSG benchmark. However, it is a different question in which protocols we can expect reasonable performance. For example, it is unreasonable to expect a standard RL policy trained *tabula rasa* on a single level of NLE [Küttler et al., 2020b] to generalise to dissimilar levels, as it might encounter entirely unseen entities or very out-of-distribution combinations of entities and level layouts. Each evaluation protocol measures a different kind of ZSG strength, and they hence form a kind of partial ordering where "easier" evaluation protocols come before "harder" protocols. I outline this ordering here:

- Increasing the number of samples can make an evaluation protocol easier, but often only to a point: more samples are unlikely to bring greater variety which is needed for zero-shot generalisation. Increasing the number of contexts (while keeping the shape of the context set the same) also makes an evaluation protocol easier. Even a small amount of additional variety can improve performance.

- The number of factors of variation which are extrapolating or combinatorially interpolating in the testing context set can also be varied. The more there are, the more difficult the evaluation protocol. Further, the width of the range of values that the extrapolating factors take at training time can vary. This is linked to the number of contexts but is also related to the variety available during training time along these axes of variation.

- Following Ke et al. [2021], I consider the difficulty of interpolating and extrapolating along different types of factors of variation. Interpolation along ordinal axes is likely

the easiest, followed by cardinal axes interpolation (which happens through unseen combinations of seen values for a cardinal axis combined with any other axis), and then extrapolation along ordinal axes. Finally, extrapolation along cardinal axes is the most difficult.

As the difficulty of an evaluation protocol increases, it becomes less likely that standard RL approaches will get good performance. In more difficult protocols which involve extrapolation of some form, zero-shot generalisation is unlikely to occur at all with standard RL methods, as there is no reason to expect a policy to generalise correctly to entirely unseen values. That does not mean that this type of generalisation is impossible: it just makes clear the fact that to achieve it, more than standard RL methods will be needed. That is, methods incorporating prior knowledge[6] such as transfer from related environments [Zhu et al., 2021]; strong inductive biases [Zambaldi et al., 2019] or assumptions about the variation; or utilising online adaptation [Hansen et al., 2021a, Zintgraf et al., 2020] will be necessary to produce policies that generalise in this way.

2.5 Methods For Zero-shot Generalisation In Reinforcement Learning

I now classify methods that tackle ZSG in RL. The problem of ZSG occurs when the training and the testing context sets are not identical. There are many types of ZSG problems (as described in more detail in Section 2.4), and hence there are many different styles of methods. I categorise the methods into those that try and increase the similarity between training and testing data and objective (Section 2.5.1), those that explicitly aim to handle differences between training and testing environments (Section 2.5.2), and those that target RL-specific issues or optimisation improvements which aid ZSG performance (Section 2.5.3).

See Fig. 2.4 for a diagram of this categorisation, and Tables 2.2 and 2.3 for a table classifying methods by their approach, the environment variation they were evaluated on, and whether they mostly change the environment, loss function or architecture. Performing this comprehensive classification enables us to see the under-explored areas within ZSG research, and I discuss future work for methods in Section 2.6.6.

[6] http://betr-rl.ml/2020/

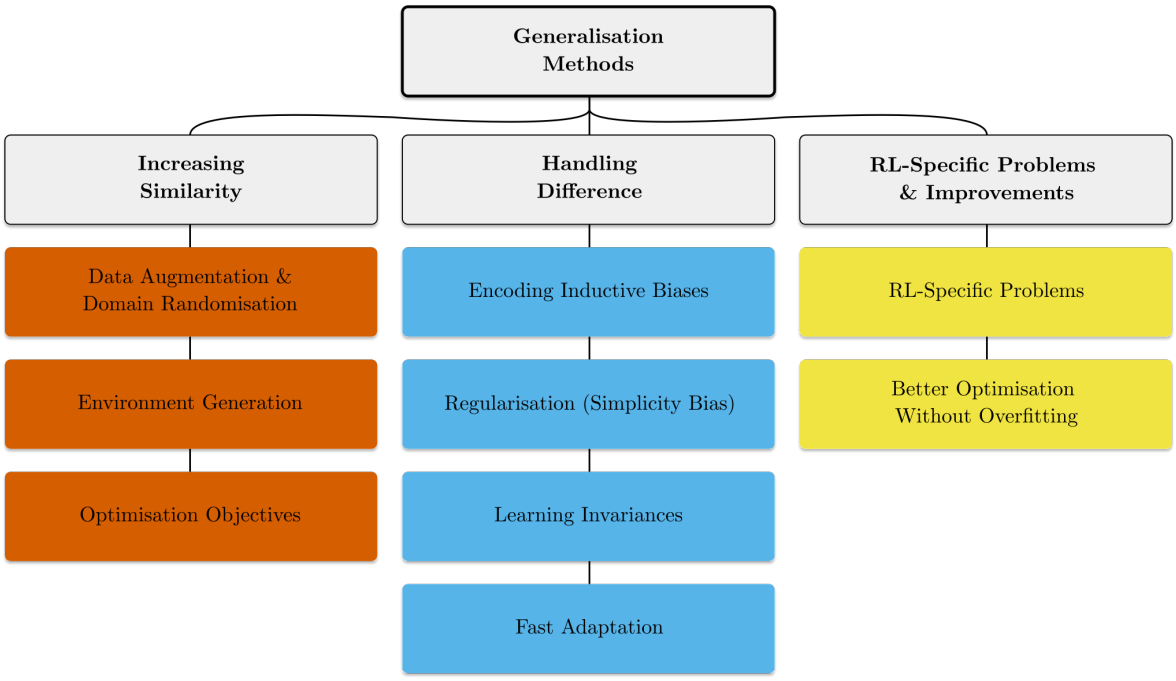


Figure 2.4: Categorisation of methods for tackling zero-shot generalisation in reinforcement learning

2.5.1 Increasing Similarity Between Training And Testing

All else being equal, the more similar the training and testing environments are, the smaller the generalisation gap and the higher the test time performance. This similarity can be increased by designing the training environment to be as close to the testing environment as possible. Assuming this has been done, in this section, I cover methods that make the data and objective being used to learn the policy during training closer to that which would be used if we were optimising on the testing environment. This often involves changing the context set or distribution.

2.5.1.1 Data Augmentation and Domain Randomisation

Two natural ways to make training and testing data more similar are data augmentation [Shorten and Khoshgoftaar, 2019] and domain randomisation [Tobin et al., 2017, Sadeghi and Levine, 2017, Peng et al., 2018]. This is especially effective when the variation between the training and testing environments is known, as then a data augmentation or domain randomisation can be used which captures this variation. In practice there is only so far this approach can go, as stronger types of variation often cannot be captured by this simple method.

Data augmentation (DA) can be viewed in two ways. First, the augmented data points are seen as additional data to train the model. This interpretation is what causes us to classify DA techniques as trying to increase the similarity between training and testing data. In the second view, DA can be used to enforce the learning of an invariance, by regularising the model to have the same output (or the same internal representations) for different augmented data points. In this view, DA is more with encoding inductive biases, which I cover in Section 2.5.2.1. I include all DA work in this section for clarity.

There are many examples of using DA in RL, although not all of them are targeted at ZSG performance. Raileanu et al. [UCB-DrAC, 2021] adapts the DA technique DrQ [Yarats et al., 2021] to an actor-critic setting [PPO, Schulman et al., 2017] and introduces a method for automatically picking the best augmentation during training. Wang et al. [Mixreg, 2020a] adapts mixup [Zhang et al., 2018d] to the RL setting, which encourages the policy to be linear in its outputs with respect to mixtures of possible inputs. Zhang and Guo [PAADA, 2021] use adversarial DA combined with mixup. Lee et al. [RandFM, 2020] uses a randomised convolutional layer at the start of the network to improve robustness to a wide variety of visual inputs. Zhou et al. [MixStyle, 2021] mixes style statistics across spatial dimensions in CNNs for increased data diversity. All these methods [Raileanu et al., 2021, Wang et al., 2020a, Zhang and Guo, 2021, Lee et al., 2020, Zhou et al., 2021] show improved performance on CoinRun [Cobbe et al., 2019] or OpenAI Procgen [Cobbe et al., 2020b] by improving both training and testing performance, and some also show gains on other benchmarks such as visually distracting DeepMind Control (DMC) variants. Hansen and Wang [SODA, 2021] uses similar augmentations as before but only to learn a more robust image encoder, while the policy is trained on non-augmented data, demonstrating good performance on DMC-GB [Hansen and Wang, 2021]. James et al. [RCAN, 2019b] use DA to learn a visual mapping from any different observation back to a canonical observation of the same state, and then train a policy on this canonical observation. They show improved sim-to-real performance on a robotic grasping task.

Ko and Ok [2021, InDA,ExDA] show that the time at which the augmentations are applied is important for the performance: some augmentations help during training, whereas others only need to be applied to regularise the final policy. Fan et al. [SECANT, 2021] introduce

a method for combining DA with policy distillation. As training on strong augmentations can hinder performance, they first train on weak augmentations to get an expert policy, which they then distil into a student policy trained with strong augmentations. Hansen et al. [SVEA, 2021b] argues that DA when naively applied increases the variance of Q-value targets, making learning less stable and efficient. They introduce adjustments to the standard data-augmentation protocol by only applying augmentations to specific components at specific points during the calculation of the loss function, evaluating performance on DMC-GB [Hansen and Wang, 2021]. These works show that in RL settings the choice of when to apply augmentations and what type of augmentations to apply is non-trivial, as the performance of the model during training impacts the final performance by changing the data the policy learns from.

Domain Randomisation (DR) is the practice of randomising the environment across a distribution of parametrisations, aiming for the testing environment to be covered by the distribution of environments trained on. Fundamentally DR is just the creation of a non-singleton training context set, and then randomly sampling from this set. [Tobin et al., 2017], [Sadeghi and Levine, 2017] and [Peng et al., 2018] introduced this idea in the setting of sim-to-real transfer in robotics. Much work has been done on different types of DR, so I cover just a sample here. OpenAI et al. [2019a] described Automatic Domain Randomisation: instead of sampling possible environment parametrisations uniformly at random, this method dynamically adjusts the distribution in response to the agent's current performance. Ren et al. [2020, Minimax DSAC] use adversarial training to learn the DR for improved robustness. Zhao and Hospedales [P2PDRL, 2020] improve DR though peer-to-peer distillation. Wellmer and Kwok [DDL, 2021] learns a world model in which to train a policy and then applies dropout to the recurrent network within the world model, effectively performing DR in imagination. Finally, as procedural content generation [Risi and Togelius, 2020] is a method for generating a non-zero context set, it can be seen as a form of DR. Works on DR generally leverage the possibility of using a non-uniform context distribution, which possibly varies during training.

In both DA and DR approaches, as the training environment instances are increasingly augmented or randomised, optimisation becomes increasingly difficult, which often makes these methods much less sample-efficient. This is the motivation behind the work of James

et al. [2019b], Hansen and Wang [2021], Fan et al. [2021] and other works, all of which train an RL policy on a non-randomised or only weakly randomised environment while using other techniques such as supervised or self-supervised learning to train a robust visual encoder.

Finally, note that most of the DA techniques are focused on visual variation in the context set, as that is the easiest variation to produce useful augmentations for. Some DR work focuses on dynamics variation as a way of tackling the sim-to-real problem, where it is assumed that the dynamics will change between training (simulation) and testing (reality).

2.5.1.2 Environment Generation

While DR and PCG produce context-MDPs within a pre-determined context set, it is normally assumed that all the context-MDPs are solvable. However, in some settings, it is unknown how to sample from the set of all *solvable* context-MDPs. For example, consider a simple gridworld maze environment, where the context set consists of all possible block layouts on the grid; some configuration of block placements will result in unsolvable mazes. Further, many block configurations are not useful for training: they may produce trivially easy mazes. To solve these problems we can *learn* to generate new levels (sample new contexts) on which to train the agent, such that we can be sure these context-MDPs are solvable and useful training instances. We want a distribution over context-MDPs which is closer to the testing context set, which likely has only solvable context-MDPs. This is known as environment generation.

Wang et al. [2019] introduced Paired Open-Ended Trailblazer (POET), a method for jointly evolving context-MDPs and policies which solve those MDPs, aiming for a policy that can solve a wide variety of context-MDPs. They produce policies that solve unseen level instances reliably and perform better than training from scratch or a naive curriculum. Wang et al. [2020b] built on POET, introducing improvements to the open-ended algorithm including a measure of how novel generated context-MDPs are and a generic measure of how much a system exhibits open-ended innovation. These additions improve the diversity and complexity of context-MDPs generated.

Dennis et al. [2020b] introduces the framework of Unsupervised Environment Design (UED), similar to POET, in which the task is to generate context-MDPs in an unsupervised manner, which are then used to train a policy. The aim is to improve ZSG to unseen tasks either

within or outside the environment's context-MDP space. Their method PAIRED outperforms standard DR and a method analogous to POET in the grid-world setting described above, as measured by the zero-shot generalisation performance to unseen levels. Jiang et al. [2021a] extended the formal framework of UED, combining it with Prioritized Level Replay [PLR, Jiang et al., 2021b], and motivates understanding PLR as an environment generation algorithm. This combined method shows improved performance in terms of zero-shot generalisation to unseen out-of-distribution tasks in both gridworld mazes and 2D car-racing tracks. I summarise PLR in the next section.

Environment generation and DR are both methods for adjusting the context distribution over some context set provided by the environment. Environment generation tends to learn this sampling procedure, and is targeted at environments where the context set is unstructured such that not all context-MDPs are solvable or useful for training, whereas DR work often uses hard-coded heuristics or non-parametric learning approaches to adjust the context distribution, and focuses on settings where the domains are all solvable but possibly have different difficulties or learning potentials. Both can often also be seen as a form of automatic curriculum learning [Portelas et al., 2020], especially if the context distribution is changing during training and adapting to the agent's performance.

This area is very new, and I expect there to be more research soon. However, it does require access to an environment where context-MDPs can be generated at a fairly fine level of detail. Environment generation methods can target ZSG over any kind of variation, as long as that kind of variation is present in the output space of the context-MDP generator. Current methods focus on state-space variation, as that is the most intuitive to formulate as a context set within which the generator can produce specific MDPs.

2.5.1.3 Optimisation Objectives

It is sometimes possible to change our optimisation objective (explicitly or implicitly) to one which better aligns with testing performance.

Changing the distribution over which the training objective is calculated can be seen as implicitly changing the optimisation objective. An initial example in this area applied to improving ZSG is PLR [Jiang et al., 2021b], in which the sampling distribution over levels is changed to increase the learning efficiency and zero-shot generalisation of the trained policy.

They show both increased training and testing performance on OpenAI Procgen, and the method effectively forms a rough curriculum over levels, enabling more sample-efficient learning while also ensuring that no context-MDP's performance is too low.

Methods for Robust RL (RRL) are also targeted at the ZSG problem, and work by changing the optimisation objective of the RL problem. These methods take a worst-case optimisation approach, maximising the minimum performance over a set of possible environment perturbations (which can be understood as different context-MDPs), and are focused on improving ZSG to unseen dynamics. Chen and Li [2020] gave an overview and introduction to the field. [WR^{2}L, Abdullah et al., 2019] optimised the worst-case performance using a Wasserstein ball around the transition function to define the perturbation set. Mankowitz et al. [SRE-MPO, 2020] incorporates RLL into MPO [Abdolmaleki et al., 2018] and shows improved performance on the RWRL benchmark [Dulac-Arnold et al., 2021]. Pinto et al. [RARL, 2017] also (implicitly) optimises a robust RL objective through the use of an adversary which is trained to pick the worst perturbations to the transition function. This can also be seen as an adversarial domain randomisation technique.

2.5.2 Handling Differences Between Training And Testing

One way of conceptualising why policies do not transfer perfectly at test time is due to the differences between the two environments: the trained model will learn to rely on features during training that change in the testing environment and performance then suffers. In this section, we review methods that try and explicitly handle the possible differences between the features of the training and testing environments.

2.5.2.1 Encoding Inductive Biases

If we know how features change between the training and testing context-MDPs, we can use inductive biases to encourage or ensure the model does not rely on features that we expect to change: The policy should only rely on features which will behave similarly in both the training and testing environments. For example, if we know colour varies between training and testing, and colour is irrelevant to the task, then we can remove colour from the visual input before processing. Simple changes like this tend not to be worthy of separate papers, but they are still important to consider in real-world problem scenarios.

IDAAC [Raileanu and Fergus, 2021] adds an adversarial regularisation term which

encourages the internal representations of the policy not to be predictive of time within an episode. This invariance is useful for OpenAI Procgen [Cobbe et al., 2020b], as timestep is irrelevant for the optimal policy but could be used to overfit to the training set of levels. Higgins et al. [DARLA, 2017b] uses β-VAEs [Higgins et al., 2017a] to encode the inductive bias of disentanglement into the representations of the policy, improving zero-shot performance on various visual variations. P. et al. [NAP, 2021] incorporates a black-box shortest-path solver to improve ZSG performance in hard navigation problems. Zambaldi et al. [2018, 2019] incorporate a relational inductive bias into the model architecture which aids in generalising along ordinal axes of variation, including extrapolation performance. Kansky et al. [SchemaNetworks, 2017] use an object-oriented and entity-focused architecture, combined with structure-learning methods, to learn logical schema which can be used for backwards-chaining-based planning. These schemas generalise zero-shot to novel state spaces as long as the dynamics are consistent. Wang et al. [VAI, 2021b] use unsupervised visual attention and keypoint detection methods to enforce a visual encoder to only encode information relevant to the foreground of the visual image, encoding the inductive bias that the foreground is the only part of the visual input that is important.

Tang et al. [2020] introduce AttentionAgent, which uses neuroevolution to optimise an architecture with a hard attention bottleneck, resulting in a network that only receives a fraction of the visual input. The key inductive bias here is that selective attention is beneficial for optimisation and ZSG. Their method generalises zero-shot to unseen backgrounds in CarRacing [Brockman et al., 2016b] and VizDoom [Kempka et al., 2016, OpenAI, 2017]. Tang and Ha [SensoryNeuron, 2021] build on AttentionAgent, adding an inductive bias of permutation invariance in the input space. They argue this is useful for improving ZSG, for a similar reason as before: the attention mechanism used for the permutation-invariant architecture encourages the agent to ignore parts of the input space that are irrelevant. François-Lavet et al. [CRAR, 2019] use a modular architecture combining dynamics learning and value estimation both in a low-dimensional latent space, and show improved performance on a simple maze task.

Hill et al. [SHIFTT, 2020c] and Lynch and Sermanet [TransferLanfLfP, 2021] both use large pretrained models [Devlin et al., 2019, Yang et al., 2020b] to encode natural language

instructions for instruction following tasks, tackling reward function variation. They both show improved performance to novel instructions, leveraging the generalisation power of the large pretrained model. This can be seen as utilising domain knowledge to improve zero-shot generalisation to novel goal specifications by incorporating the inductive bias that all the goal specifications will be natural language.

While methods in this area appear dissimilar, they all share the motivation of incorporating specific inductive biases into the RL algorithm. There are several ways of incorporating domain knowledge as an inductive bias. The architecture of the model can be changed to process the variation correctly. If the variation is one to which the policy should be invariant, it can either be removed entirely, or adversarial regularisation can be used to ensure the policy's representations are invariant. More broadly, regularisation or auxiliary losses that encourage the policy to handle this variation correctly can be used. A recommendation to make this body of work more systematic is to use the additional types of structural assumptions discussed in Section 2.3.5 as a starting point for developing algorithms that leverage those assumptions — many of the works discussed here rely on assumptions that can be classified in those introduced in Section 2.3.5. For a deeper discussion of inductive biases see Battaglia et al. [2018], and for the original ideas surrounding inductive biases and No Free Lunch theorems see Wolpert and Macready [1997].

2.5.2.2 Regularisation and Simplicity

When we cannot encode a specific inductive bias, standard regularisation can be used. This is generally motivated by a paraphrased Occam's razor: the simplest model will generalise the best. Task-agnostic regularisation encourages simpler models that rely on fewer features or less complex combinations of features. For example, L2 weight decay biases the network towards less complex features, dropout ensures the network cannot rely on specific combinations of features, and the information bottleneck ensures that only the most informative features are used.

Cobbe et al. [2019] introduced CoinRun, and evaluated standard supervised learning regularisation techniques on the benchmark. They investigate data augmentation (a modified form of Cutout [DeVries and Taylor, 2017]), dropout, batch norm, L2 weight decay, policy entropy, and a combination of all techniques. All the techniques separately improve performance,

and the combination improves performance further, although the gains of the combination is minimal over the individual methods, implying that many of these methods address similar causes of worse generalisation. Early stopping can be seen as a form of regularisation, and Ada et al. [2021] shows that considering training iteration as a hyperparameter (effectively a form of early stopping) improves performance on some benchmarks. Almost all methods report performance at the end of training, as often early stopping would not be beneficial (as can be seen from the training and testing reward curves), but this is likely an attribute of the specific benchmarks being used, and in the future, we should keep early stopping in mind.

Several methods utilise information-theoretic regularisation techniques, building on the information bottleneck [Tishby and Zaslavsky, 2015]. Igl et al. [IBAC-SNI, 2019] and Lu et al. [IB-annealing, 2020] concurrently introduce methods that rely on the information bottleneck, along with other techniques to improve performance, demonstrating improved performance on OpenAI Procgen, and a variety of random mazes and continuous control tasks, respectively. Eysenbach et al. [RPC, 2021] extend the motivation of the information bottleneck to the RL setting specifically, learning a dynamics model and policy which jointly minimises the information taken from the MDP by using information from previous states to predict future states. This results in policies using much less information than previously, which has benefits for robustness and zero-shot generalisation, although this method was not compared on standard ZSG benchmarks to other methods. Chen [SMIRL, 2020] use surprise minimisation to improve the performance of the trained policy, although more rigorous benchmarking is needed to know whether this method has a positive effect.

Song et al. [2020] show that larger model sizes can lead to *implicit regularisation*: larger models, especially those with residual connections, generalise better, even when trained on the same number of training steps. This is also shown in Cobbe et al. [2019, 2020b].

2.5.2.3 Learning Invariances

Sometimes we cannot rely on a specific inductive bias or standard regularisation. This is a very challenging setting for RL (and machine learning in general) to tackle, as there is a fundamental limit to performance due to a kind of "no free lunch" analogy: we cannot expect a policy to generalise to arbitrary contexts. However, several techniques can help, centred around the idea of using multiple training contexts to *learn* the invariances necessary

to generalise to the testing contexts. If the factors of variation within the training contexts are the same as the factors of variation between the training and testing contexts, and the values that those factors take in testing are not far from those in training, then you can use that to learn these factors of variation and how to adapt or be invariant to them.

Much work draws on causal inference to learn invariant features from several training contexts. Zhang et al. [ICP, 2020a] assume a block MDP structure [Du et al., 2019] and leverage that assumption to learn a state abstraction that is invariant to irrelevant features, which aids in generalisation performance. Zhang et al. [DBC, 2021a] use bisimulation metrics to learn a representation that is invariant to irrelevant visual features, and show that bisimulation metrics are linked to causal inference. Kemertas and Aumentado-Armstrong [DBC-normed-IR-ID, 2021] improve DBC with a norm on the representation space and the use of intrinsic rewards and regularisation. Agarwal et al. [PSM, 2021] suggest limitations of bisimulation metrics and instead propose a policy similarity metric, where states are similar if the optimal policy has similar behaviour in that and future states. They use this metric, combined with a contrastive learning approach, to learn policies invariant to observation variation.

Several approaches use multiple contexts to learn an invariant representation, which is then assumed to generalise well to testing contexts. Sonar et al. [IPO, 2020] apply ideas from Invariant Risk Minimisation [Arjovsky et al., 2020] to policy optimisation, learning a representation which enables jointly optimal action prediction across all domains, and show improved performance over PPO on several visual and dynamics variation environments. Bertrán et al. [IAPE, 2020] introduce the *Instance MDP*, an alternative formalism for the ZSG problem, and then motivate theoretically an approach to learn a collection of policies on subsets of the training domains, such that the aggregate policy is invariant to any individual context-specific features which would not generalise. They show improved performance on the CoinRun benchmark [Cobbe et al., 2019] compared to standard regularisation techniques. Ghosh et al. [LEEP, 2021] also produce an invariant policy across a collection of subsets of the training contexts but motivate this with Bayesian RL. Their approach learns separate policies on each subset and then optimistically aggregates them at test time, as opposed to IAPE which uses the aggregate policy during training for data collection and learns the collection of policies

off-policy. While these approaches are similar there has been no direct comparison between them.

Other approaches try to learn behaviourally similar representations with less theoretical motivation. Liu et al. [CSSC, 2020] use behavioural similarity (similarity of short future action sequences) to find positive and negative pairs for a contrastive learning objective. This auxiliary loss aids in zero-shot generalisation and sample efficiency in several OpenAI Procgen games. Mazoure et al. [CTRL, 2022] uses clustering methods and self-supervised learning to define an auxiliary task for representation learning based on behavioural similarity, showing improved performance on OpenAI Procgen. Li et al. [DARL, 2021] uses adversarial learning to enforce the representations of different domains to be indistinguishable, improving performance in visually diverse testing contexts, even when only trained on simple training contexts. Fan and Li [DRIBO, 2022] uses contrastive learning combined with an information-theoretic objective to learn representations that only contain task-relevant information while being predictive of the future. They show improved performance on both visually diverse domains in DeepMind Control, and in OpenAI Procgen.

2.5.2.4 Adapting Online

A final way to handle differences between training and testing contexts is to adapt online to the testing contexts. Section 2.3.3 allows that the policy can adjust online within a single episode, as the policy class includes non-Markovian policies (which can equivalently be viewed as adaptation procedures for producing markovian policies). This adaption has to happen within a single episode and for it to be useful, the adaptation will have to be rapid enough to be useful for improved performance within that episode. Most work on Meta RL, which is traditionally concerned with fast adaptation, assumes access to multiple training episodes in the testing CMDP, violating the zero-shot assumption. However, there are works which can adapt zero-shot. While not being exhaustive, I give a brief overview of this body of work here, focusing on key examples where zero-shot generalisation is evaluated.

Many methods learn a context encoder or inference network, which then conditions either a policy or dynamics model for improved ZSG. The inference of this context at test-time can be seen as within-episode online adaptation. Yu et al. [2017, UP-OSI] uses Online System Identification to infer the context, which then conditions a policy. Yen-Chen et al. [EVF,

2019] learns a context inference network end-to-end with a dynamics model and uses this to adapt to novel objects. Kumar et al. [RMA, 2021] tackles the sim-to-real problem by training an agent using domain randomisation in simulation, and training a context inference model to condition the policy, similar to UP-OSI but with learned context inference. Ball et al. [AugWM, 2021] take a similar idea but work in the offline RL setting, so first train a world model from offline data. They then perform domain randomisation of a specific form in the world model, and then use a hard-coded update rule (enabled by the specific form of domain randomisation) to determine the context to condition the policy, enabling it to adapt zero-shot to downstream tasks. These methods all tackle dynamics variation in continuous control or visual tasks. Often, these methods make use of domain randomisation approaches but aim to have a context-conditioned adaptive policy or model, rather than a policy invariant to all possible contexts.

Some approaches use hard-coded adaptation techniques which do not rely on explicitly inferring a context. Seo et al. [TW-MCL, 2020] leverages a multi-headed architecture and multiple-choice learning to learn an ensemble of dynamics models that are selected from during deployment by choosing the one with the highest accuracy. Nagabandi et al. [MOLe, 2019b] tackles the online learning problem, keeping a continually updated and expanded collection of dynamics models which are chosen between using non-parametric task inference. In both these works the chosen model plans with model-predictive control, and they show improvements in continuous control tasks. Hansen et al. [PAD, 2021a] uses a self-supervised objective to update the internal representations of the policy during the test episode. They improve performance over standard baselines on visually varying DeepMind Control environments as well as CRLMaze [Lomonaco et al., 2020] (a VizDoom [Kempka et al., 2016] 3D navigation environment with visual variation), and on zero-shot generalisation to novel dynamics on both a real and simulated robot arm. Nagabandi et al. [GrBAL,ReBAL, 2019a] use meta-RL methods (MAML [Finn et al., 2017] and RL^2 [Duan et al., 2016]) to learn to quickly adapt a dynamics model using gradients or recurrence, and then use the adapted model to plan. In this case, it becomes more difficult to draw the line between learned and hard-coded update rules: The gradient update itself is hard-coded, but the initialisation is learned, and in the recurrent case the update is fully learned.

Finally, several methods meta-learn an adaptation function during training. RL^2 [Duan et al., 2016, Wang et al., 2017] is a meta RL method where a recurrent network is used, the hidden state of which is not reset at episode boundaries, allowing it to learn and adapt within the recurrent state over multiple episodes. While often compared to methods that require multiple training episodes, this method can often adapt and perform well within a single episode, due to the optimisation approach and architecture. Zintgraf et al. [2020] introduce an extension to RL^2 called VariBAD based on bayesian RL, where the recurrent network learns to produce latent representations that are predictive of future rewards and previous transitions. This latent representation is used by the policy. Dorfman et al. [BOReL, 2021] adjusts VariBAD to be usable in an offline setting, improving zero-shot exploration using offline data. Zintgraf et al. [HyperX, 2021] improves VariBAD with additional exploration bonuses to improve meta-exploration. Ni et al. [2021] shows that simple recurrent methods such as these, with carefully tuned implementations, can be improved further, often competing with more specialised algorithms, including in Robust RL and Meta-RL settings. Mishra et al. [SNAIL, 2018] modelled fast adaptation as a sequence-to-sequence problem and learns an attention-based architecture which encodes sequences of experience to condition the policy. These methods all show improved performance on ZSG tasks, even though their main focus is on adaptation over longer time horizons.

2.5.3 RL-Specific Problems And Improvements

The motivations in the previous two sections are mostly equally applicable to supervised learning. However, on top of the problems of generalisation from supervised learning, RL has additional problems which inhibit zero-shot generalisation performance. In this section, I discuss methods targeted at these problems, and also discuss methods that improve ZSG purely through more effective optimisation on the training set in a way that does not overfit (at least empirically).

2.5.3.1 RL-specific Problems

Optimisation in RL has additional issues on top of supervised learning, such as the non-stationarity of the data distribution, and the need to explore. These issues likely interact with generalisation in a non-trivial way. Igl et al. [ITER, 2021] shows that the non-stationarity of RL training means that policies learn features that do not generalise well, even if they

achieve the same training performance. To address this, they introduce a method to iteratively distil the current policy network into a new policy network with reinitialised weights. This reduces the impact of non-stationarity on the new network, as it is being trained on a more stationary distribution. Other RL-specific optimisation issues likely interact with zero-shot generalisation either positively or negatively, and this area deserves further attention if we are to go beyond techniques copied and adjusted from supervised learning.

2.5.3.2 Better Optimisation without Overfitting

Several works improve ZSG by improving the training performance without overfitting. Cobbe et al. [2021a] introduce Phasic Policy Gradient (PPG), which adjusts the training regime and architecture of PPO such that the policy and value functions use entirely separate networks (rather than just separate heads), which allows the value head to be optimised for longer while not causing the policy to overfit. To recover the benefits of a joint representation, the value network is distilled into an auxiliary value head on the policy network. Raileanu and Fergus [2021] build on PPG and introduce Decoupled Advantage Actor Critic (DAAC). They distil an advantage function calculated with GAE into the policy instead of a value function, which further ensures that the policy does not overfit to details that may be predictive of value function but not optimal action selection. They both show improved performance on OpenAI Procgen, demonstrating that value functions can be optimised more strongly than policies. Singh and Zheng [Sparse DVE, 2021] adjusts the architecture of the value function to allow for a multi-modal output, more closely modelling the true value function given just a visual input. This novel architecture combined with sparsity losses to ensure the value function has the desired properties reduces the variance of value function prediction and improves performance in terms of both return and navigation efficiency in OpenAI Procgen.

Another angle on better optimisation is the use of model-based RL (MBRL). Very little work has applied MBRL to ZSG benchmarks, with the work of Anand et al. [2022] being an initial example. Anand et al. [2022] apply MuZero Reanalyse [Schrittwieser et al., 2020, 2021], a SOTA MBRL method, to OpenAI Procgen [Cobbe et al., 2020b], showing much-improved performance over SOTA model-free methods at much lower sample complexity. This shows the potential of using MBRL to improve zero-shot generalisation to varying states and observations. The authors also apply MuZero to the meta-learning tasks in Meta-

World [Yu et al., 2019b], although the performance there is not as impressive, showing that generalising to new rewards (as is necessary for Meta-World) is not currently as amenable to MBRL approaches.

While the methods described above do not target ZSG specifically, they improve test-time performance on ZSG benchmarks and so are included here. I hope the field will move towards benchmarks like Procgen being the standard for RL (and not just ZSG), such that in time this work is considered standard RL, rather the ZSG specifically.

2.5.4 Discussion

Having described existing methods for ZSG in RL, and categorised them in Tables 2.2 and 2.3, I now draw some broader conclusions about the field, as well as discuss possible alternative classifications of methods.

Alternative Classifications. I have presented one possible classification of RL methods, but there are of course others. One alternative is to classify methods based on whether they change the architecture, environment or objective of the standard RL approach. This is useful from a low-level implementation perspective of what the differences are between approaches. This approach is not as useful for future researchers or practitioners who hope to choose a ZSG method for a concrete problem they are facing, as there is not a clear mapping between implementation details and whether a method will be effective for a specific problem. We do apply this classification through the colours in Tables 2.2 and 2.3, to emphasise the current focus on adjusting the loss or objective function in current methods. Another approach would be to classify methods based on which benchmarks they attempt to solve, or what specific problem motivated their design. This goes too far in the other direction, grounding methods in exactly the benchmarks they tackle. While practitioners or researchers could try and see which benchmark is most similar to their problem, they might not understand which differences between benchmarks are most important, and hence choose a method that is not likely to succeed. This classification is also less useful in pointing out areas where there is less research being done. My approach strikes a balance between these two approaches, describing the problem motivations and solution approaches at a high level which is useful for both practitioners and researchers in choosing methods and investigating future research directions.

Approach	Evaluation Variation				
	Observation	State	Dynamics	Reward	All
Data Augmentation	*SODA [Hansen and Wang, 2021]*, `RCAN [James et al., 2019b]`, *RandFM [Lee et al., 2020]*, *InDA,ExDA [Ko and Ok, 2021]*, *DrQ [Yarats et al., 2021]*, *SECANT [Fan et al., 2021]*, *UCB-DrAC [Raileanu et al., 2021]*, *PAADA [Zhang and Guo, 2021]*, *MixStyle [Zhou et al., 2021]*, *SVAE [Hansen et al., 2021b]*	*UCB-DrAC [Raileanu et al., 2021]*, *PAADA [Zhang and Guo, 2021]*			
Domain Randomisation		MD-SAC [Ren et al., 2020]	P2PDRL [Zhao and Hospedales, 2020], DR [Tobin et al., 2017, Peng et al., 2018], CAD^{2}RL [Sadeghi and Levine, 2017], ADR [OpenAI et al., 2019a], DDL [Wellmer and Kwok, 2021]		PCG [Risi and Togelius, 2020]
Environment Generation		POET [Wang et al., 2019], PAIRED [Dennis et al., 2020b], E-POET [Wang et al., 2020b]			
Optimisation Objectives		*PLR [Jiang et al., 2021b]*	*WR^{2}L [Abdullah et al., 2019]*, *RARL [Pinto et al., 2017]*, *(S)(R)(E)-MPO[Mankowitz et al., 2020]*		
Inductive Biases	`AttentionAgent [Tang et al., 2020]`, `VAI [Wang et al., 2021b]`, `SensoryNeuron [Tang and Ha, 2021]`, *DARLA [Higgins et al., 2017b]*	`NAP [P. et al., 2021]`, `RelationalRL [Zambaldi et al., 2018, 2019]`, `SchemaNetworks [Kansky et al., 2017]`, *IDAAC [Raileanu and Fergus, 2021]*, *CRAR [François-Lavet et al., 2019]*	`RelationalRL [Zambaldi et al., 2018, 2019]`	`TransferLangLfP [Lynch and Sermanet, 2021]`, `SHIFTT [Hill et al., 2020c]`	

Table 2.2: A table categorising all methods for tackling the ZSG problem in RL, part 1 of 2. The columns represent the type of variation (see Table 2.1) the method is evaluated on, and rows represent the classification in Fig. 2.4. Colours (and text styles) represent the main adjustment made by the method: Green normal-text methods mainly work by adjusting the training environment, `Red monospace-text` methods mainly work through adjusting the architecture, and *Blue italic* methods mainly work through adjusting the objective or loss function (including adding auxiliary losses). While changing the loss often requires an architectural adjustment, and often architectural changes require adjusted losses, I focus on the original motivation of the method.

Strong Generalisation Requires Inductive Biases. As described in Section 2.4.3, there are hard ZSG problems involving combinatorial interpolation or extrapolation. We want to tackle these problems, as they will occur in real-world scenarios when we have limited contexts to train on, or we know the type of variation but cannot create context-MDPs in the deployment context-MDP set (e.g. due to limited simulators). To tackle these problems, we need stronger inductive biases targeted towards specific types of extrapolation, as there is unlikely to be a general-purpose algorithm that can handle all types of extrapolation [Wolpert and Macready, 1997]. When doing research tackling extrapolative generalisation, researchers should be clear that they are introducing an inductive bias to help extrapolate in a specific

Approach	Evaluation Variation				
	Observation	State	Dynamics	Reward	All
Regularisation	Implicit Regularisation [Song et al., 2020]	*SMIRL [Chen, 2020], Mixreg [Wang et al., 2020a], IBAC-SNI [Igl et al., 2019]*	*RPC [Eysenbach et al., 2021], IB-annealing [Lu et al., 2020]*		*L2,Dropout,etc. [Cobbe et al., 2019], EarlyStopping [Ada et al., 2021]*
Learning Invariances	*DRIBO [Fan and Li, 2022], DBC [Zhang et al., 2021a], DBC-normed-IR-ID [Kemertas and Aumentado-Armstrong, 2021], DARL [Li et al., 2021], PSM [Agarwal et al., 2021], MISA [Zhang et al., 2020a]*	*IAPE [Bertrán et al., 2020], DRIBO [Fan and Li, 2022], CTRL [Mazoure et al., 2022], CSSC [Liu et al., 2020], PSM [Agarwal et al., 2021], LEEP [Ghosh et al., 2021], IPO [Sonar et al., 2020]*	*IPO [Sonar et al., 2020]*		
Fast Adaptation	PAD [Hansen et al., 2021a]	*EVF [Yen-Chen et al., 2019]*, SNAIL [Mishra et al., 2018], *HyperX [Zintgraf et al., 2021], VariBAD [Zintgraf et al., 2020], RMA [Kumar et al., 2021], RL^2 [Duan et al., 2016, Wang et al., 2017]*	*EVF [Yen-Chen et al., 2019]*, SNAIL [Mishra et al., 2018], *HyperX [Zintgraf et al., 2021], VariBAD [Zintgraf et al., 2020]*, TW-MCL [Seo et al., 2020], *UP-OSI [Yu et al., 2017], BOReL [Dorfman et al., 2021], GrBAL,ReBAL [Nagabandi et al., 2019a], RMA [Kumar et al., 2021], RL^2 [Duan et al., 2016, Wang et al., 2017]*, MOLe [Nagabandi et al., 2019b], AugWM [Ball et al., 2021], PAD [Hansen et al., 2021a]	SNAIL [Mishra et al., 2018], *HyperX [Zintgraf et al., 2021], BOReL [Dorfman et al., 2021], VariBAD [Zintgraf et al., 2020], RL^2 [Duan et al., 2016, Wang et al., 2017]*	
RL-specific Problems	ITER [Igl et al., 2021]	ITER [Igl et al., 2021]			
Better Optimisation	Sparse DVE [Singh and Zheng, 2021], PPG [Cobbe et al., 2021a], DAAC [Raileanu and Fergus, 2021], MuZero++ [Anand et al., 2022]	Sparse DVE [Singh and Zheng, 2021], PPG [Cobbe et al., 2021a], DAAC [Raileanu and Fergus, 2021], MuZero++ [Anand et al., 2022]		MuZero++ [Anand et al., 2022]	

Table 2.3: A table categorising all methods for tackling the ZSG problem in RL, part 2 of 2. The columns represent the type of variation (see Table 2.1) the method is evaluated on, and rows represent the classification in Fig. 2.4. Colours (and text styles) represent the main adjustment made by the method: Green normal-text methods mainly work by adjusting the training environment, Red monospace-text methods mainly work through adjusting the architecture, and *Blue italic* methods mainly work through adjusting the objective or loss function (including adding auxiliary losses). While changing the loss often requires an architectural adjustment, and often architectural changes require adjusted losses, I focus on the original motivation of the method.

way and be rigorous in analysing how this inductive bias helps. This involves also discussing in which situations the bias may hinder performance, for example in a different setting where extrapolation requires something else.

Going Beyond Supervised Learning as Inspiration. Methods for improving generalisation from supervised learning have been a source of inspiration for many methods, particularly for visual variation. This is exactly the variation that happens in computer vision, and hence many methods from that field are applicable. However, non-visual forms of generalisation (i.e. dynamics, state and reward), while equally important are less studied. These challenges will be specific to RL and interact with other problems unique to RL such as the exploration-

exploitation trade-off and the non-stationarity of the underlying data distribution. I hope to see more work in the area of non-visual ZSG, particularly when other hard RL problems are present.

2.6 Discussion And Future Work

In this section I highlight further points for discussion, building on those in Section 2.4.3 and Section 2.5.4, including directions for future research on new methods, benchmarks, evaluation protocols and understanding.

2.6.1 Generalisation Beyond Zero-Shot Policy Transfer

This survey focuses on zero-shot policy transfer. I believe this problem setting is a reasonable one that captures many challenges relevant to deploying RL systems. However, there are many important scenarios where zero-shot generalisation is impossible, or the assumptions can be relaxed. We will want to move beyond zero-shot policy transfer if we are to use RL effectively in a wider variety of scenarios.

The most sensible way of relaxing the zero-shot assumption is to move into a continual RL [CRL, Khetarpal et al., 2020] setting: future RL systems will likely be deployed in scenarios where the environment is constantly changing, such that the system needs to adapt continually to these changes. Making progress on this setting will require benchmarks, and I agree with [Khetarpal et al., 2020] that there are not enough good benchmarks for CRL. Most benchmarks do not enable testing all the different attributes desired of CRL methods, although [Powers et al., 2021] CORA is a good first step. New benchmarks for CRL would also be excellent as benchmarks for ZSG, especially if these benchmarks introduce new environments. I expect that methods built for domain generalisation in RL might be suitable in CRL, as the continual learning setting can be conceptualised as one in which the domain that tasks are within changes over time. This is in contrast to benchmarks that evaluate generalisation to levels sampled from the same distribution as during training, such as OpenAI Procgen [Cobbe et al., 2020b]. Hence, I recommend work on building new CRL benchmarks, to make progress on CRL and ZSG together.

Coupled with the idea of CRL as a more realistic setting for generalisation in RL, we can take inspiration from how humans generalise and what they transfer when generalising, to go

beyond transferring a single policy. While humans may not always be able to achieve good results zero-shot on a new task, if the task is related to previously seen tasks, they can reuse previous knowledge or skills to learn the new task faster. This broader notion of generalisation of objects other than a complete policy (e.g. skills or environment knowledge) will become more relevant when we start to build more powerful RL systems. Hierarchical and multi-task RL are related fields, and methods in these settings often learn subcomponents, modules or skills on source tasks (possibly in an unsupervised manner) which they can then use to increase learning speed and performance when transferred to novel tasks [Eysenbach et al., 2019, Yang et al., 2020a]. It is likely the capability to transfer components other than a single policy will be useful for future systems, and it would hence be beneficial to have benchmarks that enable us to test these kinds of capabilities. However, this is a challenging request to meet, as defining what these components are, and deciding on a performance metric for these subcomponent transfer benchmarks, are both difficult conceptual problems. I hope to see work in this area in the future.

A final assumption which is almost untouched in RL is that of a fixed action space between training and testing. Recently, Jain et al. [2020] introduced a novel problem setting and framework centred around how to generalise to new actions in RL. They introduce several benchmarks for testing methods which generalise to new actions, and a method based on learning an action representation combined with an action-ranking network which acts as the policy at test time. There is very little work in this area, and I do not cover it in this survey, but it presents an interesting future direction for ZSG research.

2.6.2 Real World Reinforcement Learning Generalisation

Dulac-Arnold et al. [2021] propose 9 properties that are characteristic of real-world RL.[7] In thinking about the current set of ZSG benchmarks, these properties are relevant in two ways. First, when applying our methods to the real world, we will have to tackle these problems. Hence, it would be beneficial if ZSG benchmarks had these properties, such that we can ensure that our ZSG methods work in real-world environments.

Second, several properties are particularly relevant for zero-shot generalisation and the

[7]The 9 properties are: limited samples; sensor, action and reward delays; high-dimensional state and action spaces; reasoning about constraints; partial observability; multi-objective or poorly specified rewards; low action latency; offline training; and explainable policies

design of new benchmarks: (1) *high cost of new samples*, (2) *training from offline data* and (3) *underspecified or multi-objective reward functions*. I explain each of these and their relation to ZSG below.

Context Efficiency. Addressing (1), it is likely that the high cost of new samples will also mean a high cost of new environment instances or contexts. This means we want methods that are *context efficient* as well as sample efficient, and hence we require benchmarks and evaluation protocols in which only a few contexts are allowed during training, rather than several 1000s. It is also worth investigating if there is an optimal trade-off (for different costs per sample and per context) between new training samples and new contexts. This line of work would revolve around different possible evaluation metrics based on how many contexts are needed to reach certain levels of generalisation performance. Further, there may be ways of actively selecting new contexts to maximise the generalisation performance while minimising the number of new contexts used, effectively a form of active learning. Evaluating context efficiency will be more computationally expensive, as models will need to be repeatedly trained on different numbers of training contexts, so work which figures out how to evaluate this property more efficiently is also beneficial.

Sim-to-Real and Offline RL. To tackle (2), two options emerge. The first is relying on good simulations, and then tackling the *sim-to-real* problem, and the second is tackling the offline RL problem directly [Levine et al., 2020]. These approaches might be more or less relevant or applicable depending on the scenario: for example, in many robotics applications, a simulator is available, whereas in healthcare settings it is likely learning from offline data is the only approach possible. Sim-to-real is a problem of domain generalisation. If this direction is most relevant, it implies we should focus on building environments that test for good domain generalisation. Existing work on sim-to-real does address this to some extent, but it would be beneficial to have a fully simulated benchmark for testing sim-to-real methods, as that enables faster research progress than requiring real robots. This is a difficult task and is prone to the possibility of overfitting to the simulation of the sim-to-real problem, but it would be useful as an initial environment for testing sim-to-real transfer.

Offline RL is also a problem of generalisation: a key issue here is generalising to state-action pairs unseen in the training data, and most current methods tackle this by conservatively

avoiding such pairs [Kumar et al., 2020a]. If we had methods to reliably extrapolate to such pairs we could improve offline RL performance.

As well as generalisation improving offline RL, it is likely that future RL deployment scenarios will need to tackle the combination of offline RL and ZSG: training policies offline that then generalise to new contexts unseen in the offline dataset. Current offline RL benchmarks [Fu et al., 2021, Gulcehre et al., 2021] do not measure generalisation in this way, but I believe they should enable us to tackle this combined problem: for example, training on offline data from 200 levels in OpenAI Procgen, and evaluating on the full distribution of levels. If tackling this is infeasible with current methods (as both offline RL and ZSG are hard problems), then a good compromise is to first work on the offline-online setting, where offline RL is used for pretraining, followed by online fine-tuning. Some work has been done in this area [Nair et al., 2021], but this does not tackle the ZSG problem specifically. Creating benchmarks for evaluating these approaches, where the emphasis is on reducing the length of the online fine-tuning stage and evaluating generalisation after fine-tuning, would move us towards truly offline RL generalisation while still being tractable with current methods.

Reward Function Variation. It is likely future RL systems will be goal- or task-conditioned, as it will be more efficient to train a general system to do several related tasks than to train a different system for each task. Here, as well as generalising to new dynamics and observations, the trained policies will need to generalise to unseen goals or tasks. The policy will need to be able to solve novel problem formulations, and hence have a more generic problem-solving ability. Benchmarks which address this capability are hence necessary for progress towards more general RL systems (see Section 2.6.4).

Related to challenges of generalisation surrounding reward functions, for many real-world problems designing good reward functions is very difficult. A promising approach is using inverse RL [IRL, Arora and Doshi, 2020, Ng and Russell, 2000] to learn a reward function from human demonstrations [Chen et al., 2021a, Schmeckpeper et al., 2020, Sermanet et al., 2018, Sestini et al., 2020], rather than hand-crafting reward functions for each task. This is often more time-efficient, as demonstrating a task is easier than specifying a reward function for it. There are two generalisation problems here: ensuring the learned reward function generalises to unseen context-MDPs during policy training, and ensuring the trained

policy generalises to unseen context-MDPs at test time. The first is an IRL generalisation problem, and the second is the standard problem of generalisation I have considered here. Solving both will be important for ensuring that this approach to training agents is effective.

Work building benchmarks and methods to solve these problems would be valuable future work. Some of these directions could be addressed by combining new evaluation protocols with pre-existing environments to create new benchmarks, rather than requiring entirely new environments to be designed and created.

2.6.3 Multi-Dimensional Evaluation Of Generalisation

Generalisation performance is usually reported using a single scalar value of test-time performance. However, this does not give us much information with which to compare and choose between methods. While better performance on a benchmark, all else being equal, probably means that a method is more useful, it is usually not clear to what extent the ordering of methods on a benchmark's leaderboard is representative of the hypothetical ordering of those methods on a real-world problem scenario for which we have to choose a method. To alleviate this, performance should be reported on multiple different testing context sets which evaluate different types of generalisation, and radar plots (such as those demonstrated by Osband et al. [2020b]) can be used to compare methods. This will be more useful for comparing methods in a more realistic way, as well as for practitioners choosing between methods.

Very few environments have the context set required for this type of evaluation, and even those that do would require additional work to create the specific testing context sets. Hence, I recommend that future benchmarks for ZSG are designed to enable this type of evaluation. This requires building environments with controllable context sets as well as PCG components (Section 2.4.3, like the benchmark we propose in Chapter 3), as well as careful thought to create a variety of testing context sets, ensuring they match important types of generalisation. A first way of splitting up testing context sets might be by the type of variation between training and testing, as well as whether interpolation or extrapolation is required to generalise to that context set. There are likely many other ways, which may be domain-specific or general across many domains.

2.6.4 Tackling Stronger Types Of Variation

Many of the methods for ZSG tackle observation function or state space variation. Both of these (or at least the practical implementations of them used in the corresponding benchmarks) tend to produce environment families in which it is much simpler to verify (at least intuitively) The Principle Of Unchanged Optimality, meaning that tackling the ZSG problem is tractable; generalisation problems with this style of variation tend to be easier to solve. These two types of variation will appear in real-world scenarios, but the other types of variation are equally important and often harder to tackle.

Work on dynamics is mostly focused on two specific settings: sim-to-real transfer, and multi-agent environments. In sim-to-real transfer [Zhao et al., 2020], there is always dynamics variation between the simulator and reality, and much work has focused on how to train policies in this setting. More generally, work on robotics and continuous control tends to address some forms of dynamics variation either in how the robot itself is controlled (e.g. due to degrading parts) or in the environment (e.g. different terrain). In multi-agent environments, if the other agents are considered part of the environment (for example in a single-agent training setting), then varying the other agents varies the dynamics of the environment [Team et al., 2021]. These both occur in the real world, but there are inevitably other forms of dynamics variation which are less well-studied. Investigating what these other forms of dynamics variation are and whether studying them would be useful is promising future work.

Tackling reward-function variation will be required to train general-purpose policies that can perform a variety of tasks, and generalise to unseen tasks without further training data (as discussed in Section 2.6.2). This variation is more difficult to tackle, and it is often difficult or impossible to verify The Principle Of Unchanged Optimality [Irpan and Song, 2019]. However, we must do work on research to tackle these problems, as otherwise RL approaches will be limited to less-ambitious problems or less-general applications. Further work building more benchmarks that enable testing for reward function variation, especially beyond simple styles of goal specification such as a target state, would be beneficial. Special attention needs to be paid to The Principle Of Unchanged Optimality [Irpan and Song, 2019] while building these benchmarks: current work tends to handle this by conditioning the policy on a goal or reward specification. Research on what approaches to goal specification are both

tractable for policy optimisation and useful for real-world scenarios would be beneficial, as there is likely a trade-off between these two desirable attributes. Hill et al. [2020c], Lynch and Sermanet [2021] work provide good examples of investigating natural language as goal specification, utilising pretrained models to improve ZSG, and I look forward to seeing more work in this area.

2.6.5 Understanding Generalisation In Reinforcement Learning

While beyond the scope of this survey, several works try to understand the problems underlying generalisation in RL. Works in this area include Song et al. [2020]'s results, which describe the notion of observational overfitting as one cause of the generalisation gap in RL; Bengio et al. [2020a] analyses the relationship between gradient interference and generalisation in supervised and RL showing that temporal difference methods tend to have lower-interference training, which correlates with worse generalisation; Igl et al. [2021] studies transient non-stationarity in RL and shows that it negatively impacts RL generalisation; and Hill et al. [2020b] investigates what environmental factors affect generalisation in an instruction-following task, finding for example that an egocentric viewpoint improves generalisation, as does a richer observation space.

This research is barely scratching the surface of understanding why generalisation in RL in particular is a challenge, and there is much future work to be done. This will enable us to build better methods, and understand any theoretical limits to the diversity of tasks that an RL agent can solve given a limited number of training contexts. The precise empirical experimentation required for this kind of research is exactly that which is enabled by having tight control over the factors of variation in the environments being used, which reinforces the conclusion made in Section 2.4.3 that purely PCG environments are unsuited for a study of generalisation in RL.

2.6.6 Future Work On Methods For Zero-shot Generalisation

In this subsection, I summarise directions for future work on methods for ZSG, informed by Section 2.5.

As described in Section 2.6.5, there are many RL-specific factors that interact with generalisation performance, often likely in a negative way. Examples of these factors include the non-stationarity of the data distribution used for training; bootstrapping and TD learning

in general; and the need for exploration. Work to understand these factors and then build methods to tackle them as discussed in Section 2.5.3.1 is a fruitful direction for future work.

We often have a context space that is unstructured or contains many unsolvable context-MDPs. Methods that enable more effective sampling from these context spaces can alleviate this. Several methods were covered in Section 2.5.1.2 but more work in this area, tackling more challenging and realistic environments with different types of variation, would be beneficial.

While much work has been done on meta RL, most work focuses on few-shot adaptation. However, work in this area could be adapted to tackle zero-shot policy transfer settings, if the environment has long episodes that require or enable online adaptation. Enabling the policy to learn and adapt online, and learning this adaptation, would likely improve performance. These approaches would also be more suited to tackling stronger forms of variation (Section 2.6.4), as online adaptation may be necessary in these scenarios. Initial work in this area is described in Section 2.5.2.4, but much more research should be done.

There are several under-explored approaches that cut across the categorisation in this work. As shown in Tables 2.2 and 2.3, most methods focus on changing the loss function or algorithmic approach. Architectural changes informed by inductive biases are less well studied, with notable examples coming from the work of Cobbe et al. [2021a], Singh and Zheng [2021], Tang et al. [2020], P. et al. [2021], Zambaldi et al. [2019]. More work can be done on investigating different architectures, either taking inspiration from supervised learning or creating RL-specific architectures. These architectures could encode inductive biases in ways that are difficult to encode through the use of auxiliary losses or regularisation. A second under-explored area is model-based reinforcement learning (MBRL) for ZSG. Most methods surveyed here are model-free, with notable exceptions being the work of Ball et al. [2021], Kansky et al. [2017], Seo et al. [2020], Anand et al. [2022]. Learning a world model and combining it with planning methods can enable stronger forms of generalisation, especially to novel reward functions (if the reward function is available during planning). As long as the model generalises well, it could also enable generalisation to novel state and observation functions. World models which can adapt to changing dynamics will be more challenging, but Seo et al. [2020] give an initial example. Anand et al. [2022] is the first example investigating

how well standard MBRL approaches generalise, and I look forward to seeing more work in this area.

2.7 Conclusion

The study of ZSG in RL is still new but is of vital importance if we want to develop applicable and usable RL solutions to real-world problems, as well as to make progress on making safe and robust AI agents more generally. In this survey I have aimed to clarify the terminology and formalism concerning ZSG in RL, bringing together disparate threads of research together in a unified framework. I presented a categorisation of benchmarks for ZSG, splitting the taxonomy into environments and evaluation protocols, and I categorised existing methods for tackling the wide variety of ZSG problems.

To conclude, I summarise the key takeaways of this survey (with pointers to the more in-depth discussion of the takeaway). The first two takeaways are concerned with the problem setting as a whole. The next four are focused on evaluating ZSG through benchmarks and metrics, and future work in these areas. The last two are focused on methods for tackling the ZSG problem.

1. Zero-shot policy transfer is useful to study, even if in specific settings it is possible to relax the zero-shot assumption, as it provides base algorithms upon which domain-specific solutions can be built (Section 2.3.6).

2. However, more work should be done to look beyond zero-shot policy transfer, particularly at continual reinforcement learning, as a way to get around the restriction of the principle of unchanged optimality (Section 2.6.1).

3. Purely black-box PCG environments are not useful for testing specific forms of generalisation and are most useful for ensuring robust improvements in standard RL algorithms. Combining PCG and controllable factors of variation is my recommended way to design new environments, having the best trade-off between high variety and the possibility of scientific experimentation (Section 2.4.3). This also enables a more multidimensional approach to evaluating generalisation performance (Section 2.6.3), and specific experimentation aimed at improving our understanding of ZSG in RL (Section 2.6.5).

4. For real-world scenarios, we have to consider both sample efficiency and context efficiency. Evaluating the performance of methods on different sizes of training context sets is a useful evaluation metric which gives us more information to choose between different methods (Section 2.6.2).

5. Work on generalisation problems associated with offline RL is under-explored and would ensure that offline RL approaches are able to generalise effectively (Section 2.6.2).

6. While observation-function and state-space variation are commonly studied, dynamics variation is only tackled in limited settings and reward-function variation is very under-studied. These stronger forms of variation are still likely to appear in real-world scenarios, and hence should be the focus of future research (Section 2.6.4).

7. For stronger forms of ZSG, stronger inductive biases are necessary, and research should be up-front about what the inductive bias they are introducing is, how it tackles the specific benchmark they are tackling, and how general they expect that inductive bias to be (Section 2.5.4).

8. There is much underexplored future work in developing new methods for improved ZSG, such as model-based RL, new architectures, fast online adaptation, solving RL-specific generalisation problems, and environment generation (Section 2.6.6).

Overall, I see there as being a large gap between the generality and robustness of current RL algorithms and the necessary levels of these required for truly safe and robust AI systems. Additionally, it is clear that more robust and rigorous evaluation of the generalisation properties of RL algorithms (and AI systems more generally) is necessary to properly understand how performant and useful they are, and whether we are truly making progress in the necessary directions. As described in point 3 above, there is a lack of RL environments that combine PCG and controllable factors, which I argue is necessary for good evaluation of generalisation in RL agents. To remedy this, in the next chapter introduce MiniHack, an environment that satisfies these and other desirable properties and allows us to more precisely evaluate and understand the generalisation of RL algorithms.

Chapter 3

Minihack: A Sandbox for Open-Ended Reinforcement Learning Research

This chapter is based on the following published work: Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Kuttler, Edward Grefenstette, and Tim Rocktäschel. MiniHack the Planet: A Sandbox for Open-Ended Reinforcement Learning Research. In *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. URL https://openreview.net/forum?id=skFwlyefkWJ.

3.1 Introduction

In this chapter, we continue studying zero-shot generalisation in RL, motivated by the analogy to agentic properties of superhuman AI systems laid out previously. In the previous chapter we saw there is still a lot progress to be made in algorithms for improving RL generalisation, but there is also a need for additional environments, both for better evaluation and understanding of RL generalisation and to drive progress in improving algorithms. More broadly, advancing deep reinforcement learning [RL, Sutton et al., 1998] methods goes hand in hand with developing challenging benchmarks for evaluating these methods. For example, simulation environments like the Arcade Learning Environment [ALE, Bellemare et al., 2016] and the MuJoCo physics simulator [Todorov et al., 2012] have driven progress in model-free RL and continuous control respectively. In the case of these specific environments, after several years of sustained improvement, results in these environments have started to reach superhuman performance [Ecoffet et al., 2019, Zhang et al., 2021b, Badia et al., 2020] while many open

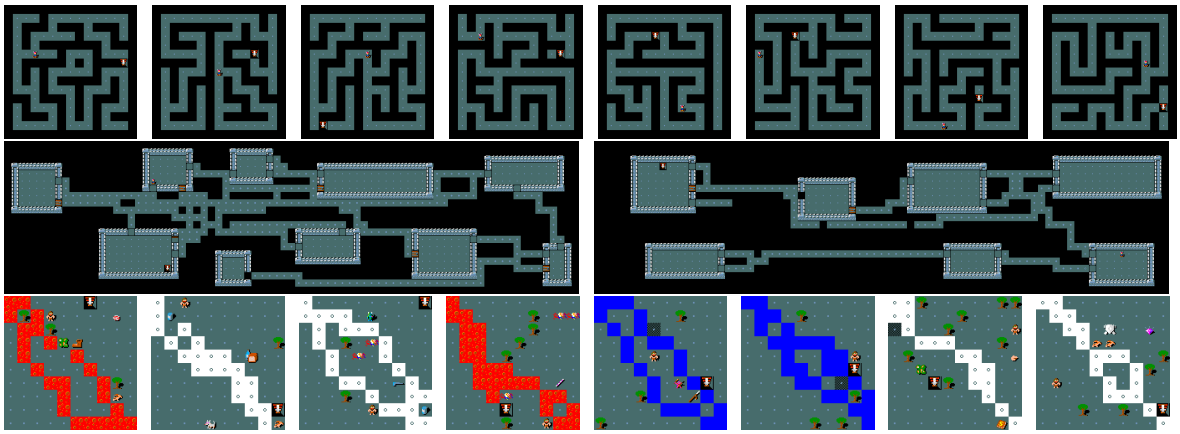


Figure 3.1: Examples of procedurally generated environments using the `des-file` format. **(Top)**: `MAZEWALK` command applied on a `15x15` grid, **(Middle)** corridors generated via `RANDOM_CORRIDOR`, **(Bottom)**: environments generated using the code snippet from Fig. 3.2.

problems in RL remain [Dulac-Arnold et al., 2021, Ibarz et al., 2021, Hill et al., 2020a]. In particular, generalisation in RL is still an open problem, as discussed in the previous chapter. To make further progress, novel challenging RL generalisation environments and testbeds are needed.

On one hand, there are popular RL environments such as Atari [Bellemare et al., 2013], StarCraft II [Vinyals et al., 2017], DotA 2 [OpenAI et al., 2019b], Procgen [Cobbe et al., 2020a], Obstacle Tower [Juliani et al., 2019a] and NetHack [Küttler et al., 2020a] that consist of entire games, but lack the ability to test specific components or open problems of RL methods in well-controlled proof-of-concept test cases. On the other hand, small-scale tightly controlled RL environments such as MiniGrid [Chevalier-Boisvert et al., 2018], DeepMind Lab [Beattie et al., 2016], Alchemy [Wang et al., 2021a], MetaWorld [Yu et al., 2019a], and bsuite [Osband et al., 2020a] have emerged that enable researchers to prototype their RL methods as well as to create custom environments to test specific open research problems (such as exploration, credit assignment, and memory) in isolation. However, once specific research hypotheses are verified in these controllable simplified environments, RL practitioners find themselves between a rock and a hard place. Systematically extending such environments and gradually dropping simplifying assumptions can require arduous engineering and excessive time commitment, while opting for more challenging benchmarks [e.g. Küttler et al., 2020a] often deprives researchers of a controllable path for assessing subsequent hypotheses. While frameworks like PyVGDL [Schaul, 2013], GVGAI [Perez-

Liebana et al., 2019a], and Griddly [Bamford et al., 2020] can be used to design custom testbeds, creating complex environments with rich entities and environment dynamics would still require substantial engineering effort as complex environment interactions would have to be designed from scratch. Thus, there is a gap in terms of a framework that allows one to easily specify a suite of rich, gradually more difficult tasks, while also providing a large set of entities with complex environment interactions ready to be used to implement these tasks.

As described in Chapter 2, we additionally desire RL environments that have a combination of PCG and controllable factors of variation, and do not rely on purely visual variation to produce generalisation challenges. These desiderata enable scientific experimentation and study of generalisation in RL as well as multi-faceted evaluation of generalisation.

To fill these gaps, I present MiniHack, a sandbox framework for easily designing novel RL environments and enriching existing ones. At the core of MiniHack are description files for defining procedurally generated worlds via the powerful domain-specific language (DSL) of the game of NetHack [Raymond et al., 2020]. The full game of NetHack, arguably the richest gridworld benchmark in RL [Küttler et al., 2020a], is not suitable for answering specific research questions in isolation. However, NetHack's DSL allows MiniHack to tap into the richness of the game with its hundreds of pre-implemented entities and the complex interaction mechanics between them [NetHack Wiki, 2020]. Furthermore, this DSL is flexible enough to easily build a wide range of testbeds, creating rich and diverse custom environments using only a few lines of human-readable code (see examples in Fig. 3.1). Once written, either directly or using a convenient Python interface, MiniHack compiles the provided description files and wraps them as standard Gym environments [Brockman et al., 2016a]. These description files effectively produce the combination of PCG and controllable factors of variation that we desire in our RL environments.

The clear taxonomy of increasingly difficult tasks, the availability of multi-modal observations (symbolic, pixel-based, and textual), its speed and ease of use, make MiniHack an appealing framework for a variety of different RL problems. In particular, MiniHack could be used to make progress in areas such as unsupervised skill discovery, unsupervised environment design, transfer learning, and language-assisted RL. In addition to a broad range of environments that can easily be designed in the MiniHack framework, I also provide exam-

ples on how to import other popular RL benchmarks, such as MiniGrid [Chevalier-Boisvert et al., 2018] or Boxoban [Guez et al., 2018], to the MiniHack planet. Once ported, these environments can easily be extended by adding several layers of complexity from NetHack (e.g. monsters, objects, dungeon features, stochastic environment dynamics, etc) with only a few lines of code.

In order to get started with MiniHack environments, I provide a variety of baselines using frameworks such as TorchBeast [Küttler et al., 2019] and RLlib [Liang et al., 2018], as well as best practices for benchmarking (see Section 3.3.5). Furthermore, I demonstrate how it is possible to use MiniHack for unsupervised environment design, with a demonstration of the recently proposed PAIRED algorithm [Dennis et al., 2020c]. Lastly, I provide baseline learning curves in Weights&Biases format[1] for all of my experiments and a detailed documentation of the framework.[2]

In summary, this chapter makes the following core contributions: (i) I present MiniHack, a sandbox RL framework that makes it easy for users to create new complex environments, (ii) I release a diverse suite of existing tasks, making it possible to test a variety of components of RL algorithms, with a wide range of complexity, (iii) I showcase MiniHack's ability to port existing gridworld environments and easily enrich them with additional challenges using concepts from NetHack, and (iv) I provide a set of baseline agents for testing a wide range of RL agent capabilities that are suitable for a variety of computational budgets.

3.2 Background: NetHack and the NetHack Learning Environment

The NetHack Learning Environment [`NLE`, Küttler et al., 2020a] is a Gym interface [Brockman et al., 2016a] to the game of NetHack [Raymond et al., 2020]. NetHack is among the oldest and most popular terminal-based games. In NetHack, players find themselves in randomly generated dungeons where they have to descend to the bottom of over 50 procedurally generated levels, retrieving a special object and thereafter escape the dungeon the way they came, overcoming five difficult final levels. Actions are taken in a turn-based fashion, and the game has many stochastic events (e.g. when attacking monsters). Despite the visual simplicity,

[1] `https://wandb.ai/minihack`
[2] `https://minihack.readthedocs.io`

NetHack is widely considered as one of the hardest games in history [The Telepraph, 2021]. It often takes years for a human player to win the game for the first time despite consulting external knowledge sources, such as the NetHack Wiki [NetHack Wiki, 2020]. The dynamics of the game require players to explore the dungeon, manage their resources, and learn about the many entities and their game mechanics. The full game of NetHack is beyond the capabilities of modern RL approaches [Küttler et al., 2020a].

`NLE`, which focuses on the full game of NetHack using the game's existing mechanisms for procedurally generating levels and dungeon topologies, makes it difficult for practitioners to answer specific research questions in isolation. In contrast, with MiniHack I present an extendable and rich sandbox framework for defining a variety of custom tasks while making use of NetHack's game assets and complex environment dynamics.

3.3 MiniHack

MiniHack is a powerful sandbox framework for easily designing novel RL environments. It not only provides a diverse suite of challenging tasks but is primarily built for easily designing new ones. The motivation behind MiniHack is to be able to perform RL experiments in a controlled setting while being able to increasingly scale the difficulty and complexity of the tasks by removing simplifying assumptions. To this end, MiniHack leverages the description file (`des-file`) format of NetHack and its level compiler (see Section 3.3.1), thereby enabling the creation of many challenging and diverse environments (see Section 3.3.4).

3.3.1 `des-file` format: A Domain Specific Language for Designing Environments

The `des-file` format [NetHack Wiki, 2021] is a domain-specific language created by the developers of NetHack for describing the levels of the game. `des-files` are human-readable specifications of levels: distributions of grid layouts together with monsters, objects on the floor, environment features (e.g. walls, water, lava), etc. All the levels in the full game of NetHack have pre-defined `des-files`. The `des-files` are compiled into binary using the NetHack level compiler, and MiniHack maps them to Gym environments.

Levels defined via `des-file` can be fairly rich, as the underlying programming language has support for variables, loops, conditional statements, as well as probability distributions.

```
$river=TERRAIN:{'L','W','I'}
SHUFFLE:$river
LOOP [2] {
  TERRAIN:randline (0,0),(10,10),5,
     $river[0]
  MONSTER:random,random
}
REPLACE_TERRAIN:(0,0,10,10),'.','T',5%
STAIR:random,down
```

Figure 3.2: A sample code snippet in `des-file` format language. The `$river` variable is used to sample a terrain feature (‘`L`‘ for lava, ‘`W`‘ for water and ‘`I`‘ for ice). The `LOOP` block draws two rivers via the `randline` command and places two random monsters at random locations. The `REPLACE_TERRAIN` commands replaces 5% of floors (‘`.`‘) with trees (‘`T`‘). A stair down is added at random locations.

```
1  MAZE:"simple_maze",' '
2  GEOMETRY:center,center
3  MAP
4    --- --- ---
5    |.| |.| |.|
6  ---S---S---S---
7  |.......+.+...|
8  ---+-----.-----
9  |.......+.+...|
10 ---S---S---S---
11   |.| |.| |.|
12   --- --- ---
13 ENDMAP
14 LOOP [5] {
15    OBJECT:'%',random
16    TRAP:random,random
17 }
18 [10%]: GOLD: 100,random
19 MONSTER:('B',"bat"),(3,3)
```

Figure 3.3: A `des-file` example for a simple NetHack level.

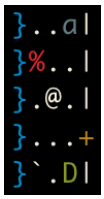
Crucially, it supports underspecified statements, such as generating a random monster or an object at a random location on the map. Furthermore, it features commands that procedurally generate diverse grid layouts in a single line. For example, the `MAZEWALK` command generates complex random mazes (see Fig. 3.1 **Top**), while the `RANDOM_CORRIDORS` command connects all the rooms in the dungeon level using procedurally generated corridors (see Fig. 3.1 **Middle**). Fig. 3.2 presents a `des-file` code snippet that procedurally generates diverse environment instances on a `10x10` grid, as presented in Fig. 3.1 **Bottom**.

Fig. 3.3 shows a `des-file` for a level with fixed, pre-defined map layout (lines 3-13). Here, the ‘`.`‘, ‘`+`‘, and ‘`S`‘ characters denote grid cells for floor, closed door, and secret door, respectively, while ‘`|`‘ and ‘`-`‘ denote walls. The loop block (lines 14-17) places five random comestibles (‘`%`‘) and random traps at random positions. Line 18 adds 100 golds at a random location with 10% probability. Line 19 adds a bat at a fixed location. These examples only provide a glimpse into the variety of levels that can be generated (see Appendix B.1 for more examples and details on the `des-file` format).

3.3.2 MiniHack Environments

By tapping into the richness of the game of NetHack, MiniHack environments can make use of a large set of pre-existing assets. One can add one of more than 580 possible monster types, each of which has unique characteristics such as attack distance and type; health

(a) Pixels



(b) Symbols

water	floor	floor	killer bee	wall
water	an apple	floor	floor	wall
water	floor	**agent**	floor	wall
water	floor	floor	floor	closed door
water	a boulder	floor	green dragon	wall

(c) Textual descriptions

Figure 3.4: Different forms of agent-centred observations of the grid of the map in MiniHack.

points; resistance against certain attacks; and special abilities such as changing shape, moving through walls, and teleporting. Practitioners can also choose from 450 items in the game, including various types of weapons, armour, tools, wands, scrolls, spellbooks, comestibles, potions, and more. These items can be used by the agent as well as monsters.

Observations. MiniHack supports several forms of observations, including global or agent-centred viewpoints (or both) of the grid (such as entity IDs, characters, and colours), as well as textual messages, player statistics and inventory information [Küttler et al., 2020a]. In addition to existing observations in `NLE`, MiniHack also supports pixel-based observations, as well as text descriptions for all entities on the map (see Fig. 3.4).

Action Space. NetHack has a large, structured and context-sensitive action space [Raymond et al., 2020]. I give practitioners an easy way to restrict the action space in order to promote targeted skill discovery. For example, navigation tasks mostly require movement commands, and occasionally, kicking doors, searching or eating. Skill acquisition tasks, on the other hand, require interactions with objects, e.g. managing the inventory, casting spells, zapping wands, reading scrolls, eating comestibles, quaffing potions, etc. 75 actions are used in these tasks. Many actions and their nontrivial interactions with game objects offer additional opportunities for designing rich MiniHack tasks. For example, a towel can be used as a blindfold (for protection from monsters that harm with their gaze), for wiping off slippery fingers (e.g. after eating deep-fried food from a tin), or even serve as a weapon when wet (which can be achieved by dipping the towel into water).

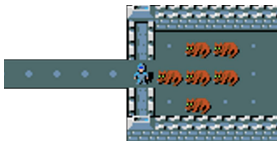
Reward. Reward functions in MiniHack can easily be configured. The `RewardManager` provides a convenient way to specify one or more events that can provide different (positive or negative) rewards, and control which subsets of events are sufficient or required for episode termination (see Appendix B.2.4 for further details).

```
# Define the labyrinth as a string
grid = """
--------------------
|.......|.|........|
|.-----.|.|.-----|.|
|.|...|.|.|......|.|
|.|.|.|.|.|-----.|.|
|.|.|...|....|.|.|.|
|.|.--------.|.|.|.|
|.|..........|...|.|
|.|-------------|.|
|..................|
--------------------
"""
# Define a level generator
level = LevelGenerator(map=grid)
level.set_start_pos((9, 1))
# Add wand of death and apple
level.add_object("death", "/")
level.add_object("apple", place=(14, 5))
# Add a Minotaur at fixed position
level.add_monster(name="minotaur", place=(14, 6), args=("asleep",))

# Define the goal
rwrd_mngr = RewardManager()
rwrd_mngr.add_eat_event("apple")

# Declare task a Gym environment
env = gym.make("MiniHack-Custom", des_file=level.get_des(), reward_manager=rwrd_mngr)
```

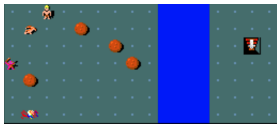
Figure 3.5: A sample code snippet for creating a custom MiniHack task using the `LevelGenerator` and `RewardManager`.

3.3.3 Interface

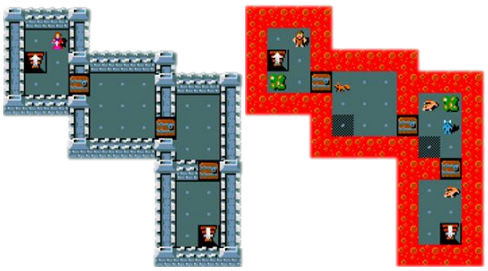
MiniHack uses the popular Gym interface [Brockman et al., 2016a] for the interactions between the agent and the environment. One way to implement MiniHack Gym environments is to write the description file in the human-readable `des-file` format and then pass it directly to MiniHack (see Fig. B.4a in Appendix B.2.2). However, users might find it more convenient to construct the environment directly in Python. The `LevelGenerator` allows users to do this by providing the functionality to add monsters, objects, environment features, etc. Fig. 3.5 presents an example code snippet of this process. Here, the agent starts near the entrance of a labyrinth and needs to reach its centre to eat the apple. A Minotaur, which is a powerful monster capable of instantly killing the agent in melee combat, is placed deep inside the labyrinth. There is a wand of death placed in a random location in the labyrinth. The agent



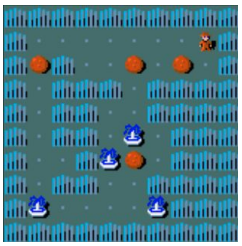
(a) `CorridorBattle` requires luring monsters into a corridor and fighting them one at a time.



(b) `River` requires pushing boulders into a river to reach the goal via the generated bridge.



(c) Two versions of `MultiRoom-N4-S5` task. (left) Regular version (right) Extreme version that includes random monsters, locked doors, and lava tiles instead of walls.



(d) `Boxoban` requires pushing boulders into different goals (here represented as four fountains).

Figure 3.6: Screenshots of several MiniHack tasks.

needs to pick the wand up, and upon seeing the Minotaur, zap it in the direction of the monster. Once the Minotaur is killed, the agent would be able to reach the centre of the labyrinth and eat the apple to complete the task. The `RewardManager` is used to specify the goal that needs to be completed (eating an apple). The `LevelGenerator` and `RewardManager` are described in more detail in Appendix B.2.2.

3.3.4 Tasks: A World of Possibilities

I release a collection of example environments that can be used to test various capabilities of RL agents, as well as serve as building blocks for researchers wishing to develop their own environments. All these environments are built using the interface described in Section 3.3.3, which demonstrates the flexibility and power of MiniHack for designing new environments.

Navigation Tasks. MiniHack's navigation tasks challenge the agent to reach the goal position by overcoming various difficulties on their way, such as fighting monsters in corridors (see Fig. 3.6a), crossing a river by pushing boulders into it (see Fig. 3.6b), navigating through complex or procedurally generated mazes (see Fig. 3.1 **Top** and **Medium**). These tasks feature a relatively small action space.[3] Furthermore, they can be easily extended or adjusted with minimal effort by either changing their definition in Python or the corresponding `des-file`. For instance, once the initial version of the task is mastered, one can add different types of monsters, traps or dungeon features, or remove simplifying assumptions (such as having a

[3]Movement towards 8 compass directions, and based on the environment, search, kick, open, and eat actions.

fixed map layout or full observability), to further challenge RL methods. The suite of 44 diverse environments is meant to assess several of the core capabilities of RL agents, such as exploration, planning, memory, and generalisation. The detailed descriptions of all navigation tasks, as well as the full list of 44 registered environments, can be found in Appendix B.3.1.

Skill Acquisition Tasks. The skill acquisition tasks enable utilising the rich diversity of NetHack objects, monsters and dungeon features, and the interactions between them. These tasks are different from navigation tasks in two ways. First, the skill acquisition tasks feature a large action space (75 actions), where the actions are instantiated differently depending on which object they are acting on. Given the large number of entities in MiniHack, usage of objects with an appropriate action in the setting of sparse rewards is extremely difficult, requiring a thorough exploration of the joint state-action space.[4] Second, certain actions in skill acquisition tasks are factorised autoregresively [PIERROT et al., 2021], i.e., require performing a sequence of follow-up actions for the initial action to have an effect. For example, to put on a ring, the agent needs to select the `PUTON` action, choose the ring from the inventory and select which hand to put it on. As a result, MiniHack allows getting rid of simplifying assumptions that many RL environments impose, such as having a single "schema" action used with a variety of objects regardless of the context. For the full list of tasks, see Appendix B.3.2.

Porting Existing Environments to MiniHack. Transitioning to using a new environment or benchmark for RL research can be troublesome as it becomes more difficult to compare with prior work that was evaluated on previous environments or benchmarks. Here, I show that prior benchmarks such as MiniGrid [Chevalier-Boisvert et al., 2018] and Boxoban [Guez et al., 2018] can be ported to MiniHack. While the MiniHack versions of these tasks are not visually identical to the originals, they still test the same capabilities as the original versions, which enables researchers familiar with the original tasks to easily analyse the behaviour of agents in these new tasks. Due to the flexibility and richness of MiniHack, I can incrementally add complexity to the levels in these previous benchmarks and assess the limits of current methods. This is especially useful for MiniGrid, where current methods are able to solve all

[4] Most of the state-of-the-art exploration methods, such as RND [Burda et al., 2019b], RIDE [Raileanu and Rocktäschel, 2020], BeBold [Zhang et al., 2020b], and AGAC [Flet-Berliac et al., 2021], rely on state space exploration rather than the state-action space exploration.

existing tasks [Flet-Berliac et al., 2021, Zhang et al., 2020b].

As an example, I present how to patch the navigation tasks in MiniGrid [Chevalier-Boisvert et al., 2018] and increase their complexity by adding monsters, locked doors, lava tiles, etc (see Fig. 3.6c). Similarly, I make use of publicly available levels of Boxoban [Guez et al., 2018] to offer these task in MiniHack (see Fig. 3.6d). Once ported to MiniHack, these levels can easily be extended, for example, by adding monsters to fight while solving puzzles. An added benefit of porting such existing benchmarks is that they can use a common observation and action space. This enables investigating transfer learning and easily benchmarking a single algorithm or architecture on a wide variety of challenges, such as the planning problems present in Boxoban and the sparse-reward exploration challenges of MiniGrid.

While MiniHack has the ability to replace a large set of entities ported from original environments, it is worth noting that not all entities have identical replacements. For example, MiniGrid's `KeyCorridor` includes keys and doors of different colours, whereas the corresponding objects in NetHack have no colour. The randomly moving objects in MiniGrid's `Dynamic-Obstacles` tasks are also absent.

Despite MiniHack versions of ported environments having minor differences compared to originals, they nonetheless assess the exact same capabilities of RL agents. In particular, while the underlying dynamics of the environment are identical to the original in the case of Boxoban, my MiniGrid version includes slight changes to the agent's action space (turning and moving forwards vs only directional movement) and environment dynamics (the event of opening doors is probabilistic in MiniHack).

3.3.5 Evaluation Methodology

Here I describe the evaluation methodology and experimental practice I take in this chapter in more detail, as a recommendation for future work evaluating methods on the MiniHack suite of tasks. To ensure a fair comparison between methods, performance should be evaluated in standard conditions. Specifically to MiniHack, this means using the same: action space,[5] observation keys,[6] fixed episode length, reward function, game character,[7] or any other

[5] Larger action spaces can often increase the difficulty of the task.

[6] Abstractly, the same observation space. There are multiple (multimodal) options which may change the difficulty of the task.

[7] The default character in MiniHack is a chaotic human male rogue (`rog-hum-cha-mal`) for navigation tasks and a neutral human male caveman (`cav-hum-new-mal`) for skill acquisition tasks. For the `CorridorBattle`

environment parameter that can potentially affect the difficulty of tasks. When reporting results, I recommend reporting the median reward achieved over at least 5 independent training runs with different seeds. Reporting the median avoids the effects of any outliers, and five independent runs strike a balance between statistical validity and computational requirements. When evaluating the generalisation, agents should be trained on a limited number of seeds and tested on held-out seeds or the full distribution of levels. As well as sharing final performance, sharing full learning curves (provide all independent runs or present an error metric, such as standard deviation), wall-clock time and examples of behaviour are recommended as it can be helpful to other researchers building on or comparing against the results. If additional information has been used during training or testing, then that should be made clear in any comparisons with other work.

3.4 Experiments

In this section, I present experiments on tasks described in Section 3.3.4. The purpose of these experiments is to assess various capabilities of RL agents, and highlight how incremental changes in the environments using the rich entities in MiniHack further challenge existing RL methods. I highlight and discuss results on several different tasks, while results for all tasks can be found in Appendix B.5.

The main baseline for all tasks is based on IMPALA [Espeholt et al., 2018], as implemented in TorchBeast [Küttler et al., 2019]. For navigation tasks, I also use two popular exploration techniques — Random Network Distillation [RND, Burda et al., 2019a] and RIDE [Raileanu and Rocktäschel, 2020], the latter being designed specifically for procedurally generated environments where an agent is unlikely to visit a state more than once. Agents are trained and tested on the full distribution of levels in each environment. I make use of the same agent architecture as in [Küttler et al., 2020a]. All details on agent architectures and training setting are described in Appendix B.4. Full results on all MiniHack tasks are available in Appendix B.5.

Navigation Tasks. Fig. 3.7 summarises results on various challenging MiniHack navigation tasks. While simpler versions of the tasks are often quickly solved by the baseline approaches, adding layers of complexity (such as increasing the size of procedurally generated mazes,

tasks, I override the default and use a lawful human female knight (`kni-hum-law-fem`) instead.

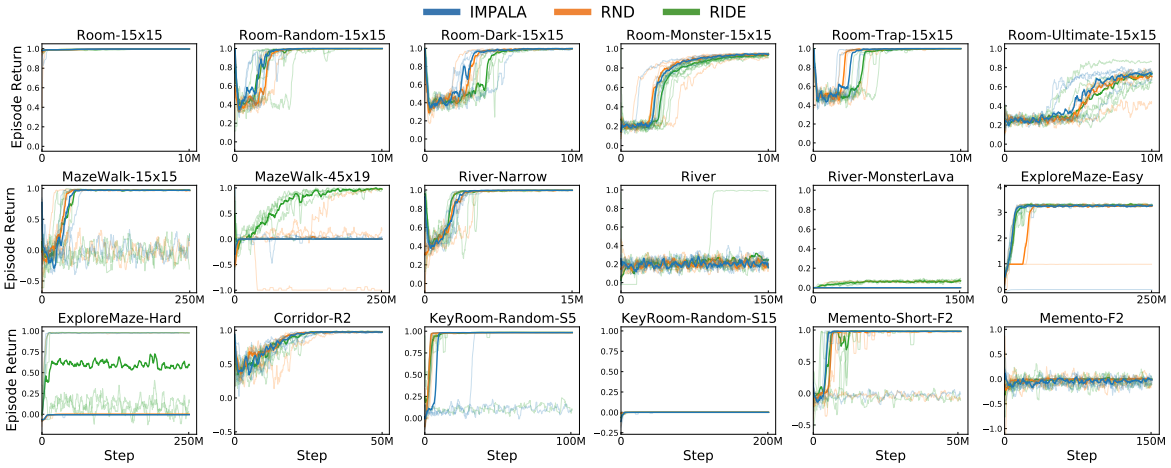


Figure 3.7: Mean episode returns on several MiniHack navigation tasks across five independent runs. The median of the runs is bolded.

resorting to partially observable settings, and adding monsters and traps) renders the baselines incapable of making progress. For example, the baselines fail to get any reward on the most difficult version of the `River` task that includes moving monsters and deadly lava tiles, challenging the exploration, planning and generalisation capabilities of the agent. The results on the `KeyRoom` tasks highlight the inability of RL methods to handle generalisation at scale. Though the smaller version of the task (`KeyRoom-Random-S5`) is solved by all baselines, the larger variant (`KeyRoom-Random-S15`) is not solved by any of the methods.

Skill Acquisition Tasks. Fig. 3.8 presents the results on various skill acquisition tasks. While the simple tasks that require discovering interaction between a single entity and an action of the agent (e.g., eating comestibles, zapping a wand of death towards a monster, etc.) can be solved by the baselines, the discovery of a sequence of entity-action relations in the presence of environment randomisation and distracting entities remains challenging for RL methods. For instance, none of the runs is able to make progress on `WoD-Medium` or `LavaCross` tasks due to insufficient state-action space exploration despite mastering the simplified versions of them.

Ported Environments. Fig. 3.9 presents the results of different versions of the `MultiRoom` environment ported from MiniGrid [Chevalier-Boisvert et al., 2018]. In the version with two rooms, adding additional layers of complexity, such as locked doors, monsters, or lava tiles instead of walls, makes the learning more difficult compared to regular versions of the task. In the Extreme version with all the aforementioned complexities, there is no learning progress

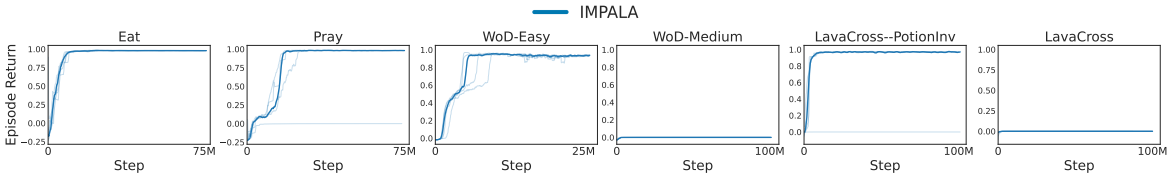


Figure 3.8: Mean episode returns on several skill acquisition tasks across five independent runs. The median of the runs is bolded.

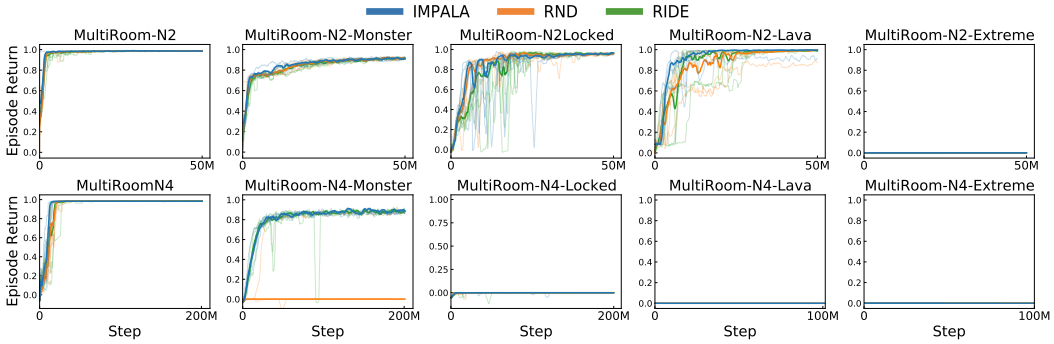


Figure 3.9: Mean episode returns on various `MultiRoom` environments ported from MiniGrid [Chevalier-Boisvert et al., 2018] across five independent runs. The median of the runs is bolded.

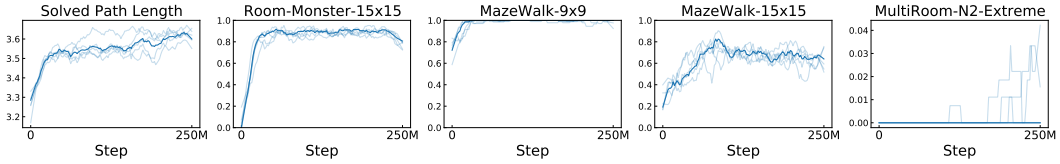


Figure 3.10: Results from the PAIRED algorithm, showing the solved path length of UED environments and zero-shot transfer performance. Plots show five independent runs with the median bolded.

at all. In the version with four rooms, the baseline agents only make progress on the simplest version and the version with added monsters. All additional complexities are beyond the capabilities of baseline methods due to the hard exploration that they require. Consequently, all independent runs fail to obtain any positive reward. These results highlight the ability of MiniHack to assess the limits of RL algorithms in an extendable, well-controlled setting.

Unsupervised Environment Design. MiniHack also enables research in *Unsupervised Environment Design* (UED), whereby an adaptive task distribution is learned during training by dynamically adjusting free parameters of the task MDP. MiniHack allows overriding the description file of the environment, making it easy to adjust the MDP configuration as required by UED. To test UED in MiniHack, I implement the recent PAIRED algorithm [Dennis et al., 2020c], which trains an environment adversary to generate environments in order to ultimately

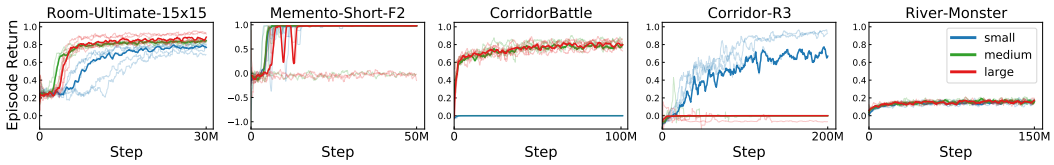


Figure 3.11: Mean episode returns of a baseline IMPALA agent using three model architectures.

train a robust *protagonist* agent, by maximizing the regret, defined as the difference in return between a third, *antagonist* agent and the protagonist. The adversary is allowed to place four objects in a small 5x5 room: {`walls`, `lava`, `monster`, `locked door`}. As a result of the curriculum induced by PAIRED, the protagonist is able to improve zero-shot performance on challenging out-of-distribution environments such as `MultiRoom-N2-Extreme`, despite only ever training on a much smaller environment (see Fig. 3.10).

Agent Architecture Comparison. We perform additional experiments to compare the performance of the baseline IMPALA agent using different neural architectures. Fig. 3.11 presents results using three architectures (small, medium, and large) on selected MiniHack tasks which differ in the number of convolutional layers, the size of hidden MLP layers, as well as the entity embedding dimension (see Appendix B.4.3 for full details). The performances of medium and large agent architectures are on par with each other across all five tasks. Interestingly, the small model demonstrates poor performance on `Room-Ultimate-15` and `CorridorBattle` environments, but outperforms larger models on the `Corridor-3` task. While it is known that small models can outperform larger models (both in terms of depth and width) depending on the complexity of the environment [Andrychowicz et al., 2021, Henderson et al., 2018a], MiniHack opens door to investigate this phenomenon in a more controlled setting due to the generous environment customisation abilities it provides. I thus believe MiniHack would be a useful testbed for exploring the impact of architecture on RL agent training.

RLlib Example. To help kickstart the development of RL models using MiniHack, I also provide integration with RLlib [Liang et al., 2018]. RLlib enables using a wide range of RL algorithms within the framework, ensuring that research on MiniHack can be performed with varying computational constraints. Fig. 3.12 presents the results of DQN [Mnih et al., 2015], A2C [Mnih et al., 2016a], and PPO [Schulman et al., 2017] methods on the `Room-15x15` task. See Appendix B.4.4 for more details.

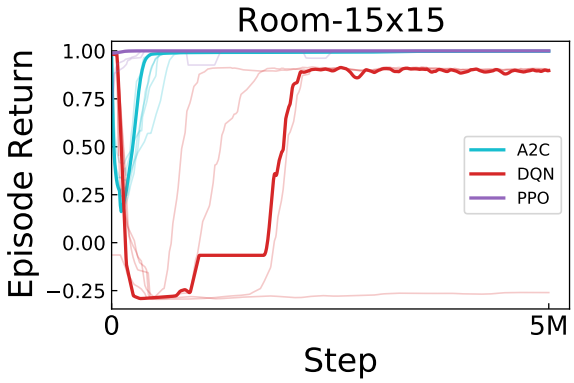


Figure 3.12: Results of various algorithms run using RLlib on `Room-15x15`. On-policy methods PPO and A2C drastically outperform DQN, although better hyperparameters would likely help.

3.5 Related Work

The RL community has made extensive use of video games as testbeds for RL algorithms [Bellemare et al., 2013, Vinyals et al., 2017, OpenAI et al., 2019b, Nichol et al., 2018, Kempka et al., 2016]. However, such games are computationally expensive to run and not easily modifiable. Furthermore, they provide only a fixed set of levels, which results in the overfitting of trained policies. ProcGen [Cobbe et al., 2020a] partially addresses this issue by providing a collection of 2D games with procedurally generated levels. However, the richness of games is still limited and only minor modifications are possible, unlike the rich environment creation capabilities that MiniHack provides. Based on the C/C++ engines of NetHack and `NLE`, MiniHack is 16 times faster than ALE [Bellemare et al., 2013] and faster than ProcGen by approximately 10% (see Appendix D of [Küttler et al., 2020a] for an approximate comparison).

MiniGrid [Chevalier-Boisvert et al., 2018] addresses the issue of computational efficiency by providing a collection of procedurally generated grid-world tasks. Nevertheless, the complexity of the environments is still limited, containing only a few types of entities and small action space. Moreover, extending the existing environments is difficult as it requires understanding the internals of the library. MiniHack provides a much richer set of entities (hundreds of objects, monsters, dungeon features) and a much larger action space. Moreover, MiniHack is designed to be easy to extend and build on top of, only requiring familiarity with a flexible and expressive high-level DSL but no underlying implementation details.

bsuite [Osband et al., 2020a] features a set of simple environments designed to test

specific capabilities of RL agents, such as memory or exploration. In contrast, MiniHack is not confined to a static task set allowing researchers to easily extend the existing task set without the need to understand the implementation details of the framework.

Several roguelike games, a genre of video games characterised by progressing through procedurally generated dungeon levels and grid-based movements, have been proposed as RL benchmarks. Rogueinabox [Asperti et al., 2017] provides an interface to Rogue, a roguelike game with simple dynamics and limited complexity. Rogue-Gym [Kanagawa and Kaneko, 2019a] introduces a simple roguelike game built for evaluating generalisation in RL and uses parameterisable generation of game levels. `NLE` [Küttler et al., 2020a] and *gym-nethack* [Campbell and Verbrugge, 2017, 2018] provide a Gym interface around the game of NetHack. However, these benchmarks use either fixed, predefined level descriptions of the full games, or a fixed set of concrete subtask (e.g., 1-on-1 combat with individual monsters [Campbell and Verbrugge, 2018]). In contrast, MiniHack allows easily customising dungeon layouts and placement of environment features, monsters and objects by a convenient Python interface or a human-readable description language.

MiniHack is not the first to provide a sandbox for developing environments. PyVGDL [Schaul, 2013, 2014] uses a concrete realisation of the Video Game Description Language [VGDL, Ebner et al., 2013] for creating 2D video games. The original software library of PyVGDL is no longer supported, while the 2.0 version is under development [Vereecken, 2020]. The GVGAI framework [Perez-Liebana et al., 2019a] is also based on the VGDL but suffers from a computational overhead due to its Java implementation. Griddly [Bamford et al., 2020] provides a highly configurable mechanism for designing diverse environments using a custom description language. Similar to MiniHack, Griddly is based on an efficient C/C++ core engine and is fast to run experiments on. Griddly is approximately an order of magnitude faster than MiniHack for simple environments, but it is unclear to what extent adding complex dynamics to Griddly, equivalent to what MiniHack provides, will decrease its speed. Furthermore, Griddly supports multi-agent and real-time strategy (RTS) games, unlike MiniHack. While PyVGDL, GVGAI, and Griddly can be used to create various entities, developing rich environment dynamics requires a significant amount of work. In contrast, MiniHack features a large collection of predefined objects, items, monsters, environment

features, spells, etc and complex environment mechanics from the game of NetHack, thus hitting the sweet spot between customizability and the richness of entities and environment dynamics to draw upon. MiniHack also provides a rich set of multimodal observations (textual descriptions, symbolic and pixel-based observations) and a convenient Python interface for describing environments in only a few lines of code. Finally, the Malmo Platform [Johnson et al., 2016] and MineRL [Guss et al., 2019] provides an interface to a popular game of Minecraft. While being rich in the environment dynamics, Minecraft is computationally intensive compared to NetHack [Raymond et al., 2020] (MiniHack is approximately 240 times faster than MineRL [Küttler et al., 2020a]).

3.6 Conclusion

In this chapter, I presented MiniHack, an easy-to-use framework for creating rich and varied RL environments, as well as a suite of tasks developed using this framework. Built upon NLE and the `des-file` format, MiniHack enables the use of rich entities and dynamics from the game of NetHack to create a large variety of RL environments for targeted experimentation, while also allowing painless scaling-up of the difficulty of existing environments. MiniHack's environments are procedurally generated by default but with controllable factors if required, ensuring the principled and systematic evaluation of generalisation of RL agents is possible. The suite of tasks I release with MiniHack tests the limits of RL methods and enables researchers to test a wide variety of capabilities in a unified experimental setting. To enable further experimentation and research, I open-source the code for training agents with numerous RL algorithms, outline best practices on how to evaluate algorithms on the benchmark suite of tasks, and provide baseline results on all the tasks that I released.

One conclusion I draw both from the relatively unimpressive results of RL generalisation methods on MiniHack and more broadly from the field of ZSG as discussed in Chapter 2 is that tabula-rasa RL[8] is likely not the way forward when it comes to producing highly-capable and well-generalising systems. This suggests this style of RL is less analogous to superhuman AI systems, and instead we need to focus on approaches that can utilise prior knowledge, data and representations to enable better learning and generalisation.

The current paradigm for incorporating these elements is to perform large-scale unsuper-

[8] that is, training RL polices from scratch on the given environment.

vised or semi-supervised pretraining [Radford et al., 2018], followed by more task-specific fine-tuning [Chung et al., 2022, Ouyang et al., 2022b]. This motivates the focus of the next two chapters (Chapters 4 and 5) on this approach, specially in the case of large language models [LLMs, Kaddour et al., 2023, *inter alia*] to producing more capable and general AI systems. This paradigm is also an important one to study as systems based on this approach are being used by millions of people, and so ensuring they are robustly behaving as desired is a critical current problem as well as necessary for future AI systems.

In this paradigm, the focus of generalisation is now at the fine-tuning stage — I care about whether a pretrained model, when fine-tune with some method on some dataset, generalises correctly and robustly the behaviour that the dataset exhibited, and so performs well on unseen, potentially out-of-distribution data. In the next chapter, I investigate the generalisation of language model fine-tuning methods (including RL-based methods) to continue understanding the limits of generalisation and how it is evaluated.

Chapter 4

Understanding the Effects of RLHF on LLM Generalisation and Diversity

This chapter is based on the following published work: Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the Effects of RLHF on LLM Generalisation and Diversity. In *The Twelfth International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=PXD3FAVHJT.

4.1 Introduction

In this and the next chapter, we adopt a different approach to building AI as analogous to superhuman AI systems, namely pretraining and then fine-tuning large language models (LLMs). These systems have become a standard approach to solving natural language processing (NLP) tasks, as well as more broadly any task that can be formulated as a text-to-text function. As these models become more capable, the tasks we want them to solve become more complex, which makes it more difficult to provide training data and to evaluate performance. This motivates choosing these systems as analogous to superhuman AI systems: they are the best systems we currently have; they are generally capable, rather than narrowly focused; and problems we expect to see in superhuman AI systems such as outer and inner alignment failures [Ngo et al., 2023] are already emerging in these systems [Shah et al., 2022, Wen et al., 2024].

I focus on generalisation at the fine-tuning stage, as this is where the high-quality pretrained model is transformed into something useful; in other words, this is when the model

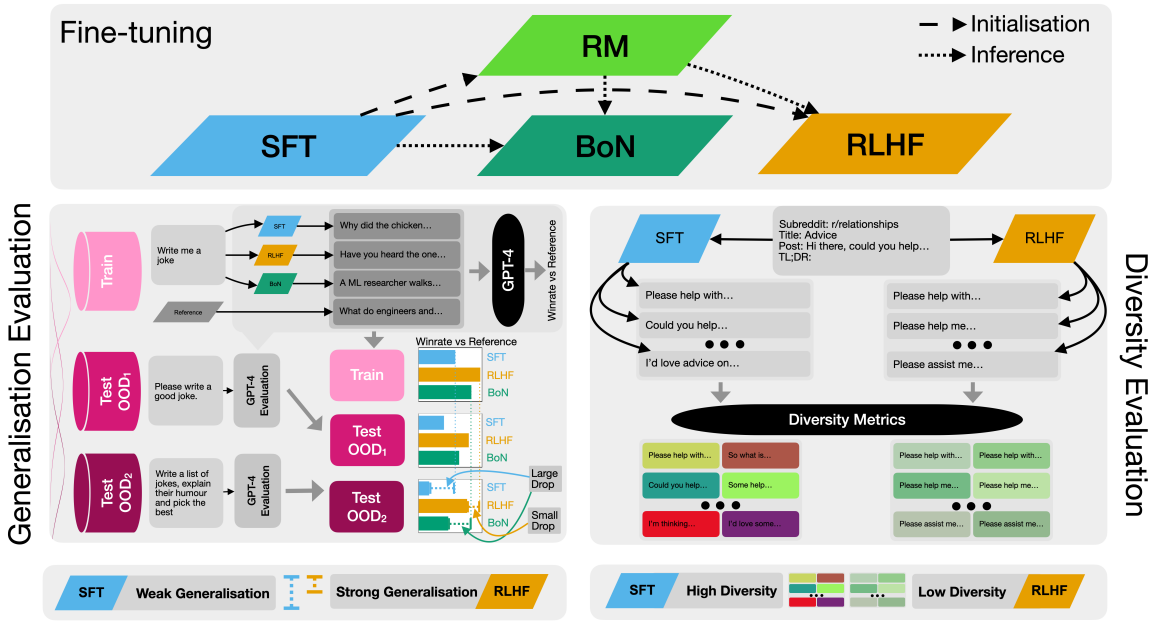


Figure 4.1: Overview of Experimental Protocol and Conclusions. In this work, I fine-tune large language models (LLMs) with three different techniques (SFT, BoN, and RLHF), and evaluate their out-of-distribution generalisation (using GPT-4 as a simulated human evaluator) and output diversity (using a range of metrics from the literature). I find that RLHF has stronger generalisation performance but lower output diversity than SFT, demonstrating a tension between these two desirable properties in current LLM fine-tuning techniques. Bar-chart results are illustrative.

is aligned. OOD generalisation during fine-tuning takes a similar form as in the previous two chapters, albeit within a very different domain: instead of different levels of a game, the model is trained on a dataset of input prompts, and when we discuss generalisation, we desire generalisation to new input prompts, potentially from different distributions.

For the kind of complex tasks we currently (and will in the future) fine-tune these systems for (e.g. summarisation [Stiennon et al., 2022], instruction following [Chung et al., 2022] or dialogue [Bai et al., 2022a]), it is often easier and faster for humans to evaluate or rank model outputs than provide full demonstrations. Thus, there has been much recent work on using human preferences in this form to fine-tune LLMs, with one dominant approach being reinforcement learning from human feedback [RLHF, Christiano et al., 2017, Ziegler et al., 2020]. This approach has been used to produce some of the most impressive AI systems that exist today [Glaese et al., 2022a, OpenAI, 2022, 2023, Anthropic, 2023].

The standard RLHF fine-tuning pipeline generally consists of three stages: *supervised fine-tuning* (SFT), where the pretrained model is fine-tuned with the language modelling loss

on demonstrations of the desired behaviour; *reward modelling* (RM), where the pretrained model is fine-tuned to predict human preferences between pairs of outputs for a given input; and *reinforcement learning* (RL), where the reward model is used to fine-tune the model produced by the SFT stage using an on-policy RL algorithm like PPO [Schulman et al., 2017]. While this pipeline has been used to seemingly great success, there is little understanding about how each component contributes to the behaviour of the final model. Two important areas where the effects of each component of the pipeline have been underexplored are the out-of-distribution (OOD) generalisation and output diversity.

OOD generalisation is important for the widespread adoption of these models, since it is necessary to ensure that LLMs are performant and reliable in a wide variety of situations that go beyond the distribution of the training data. While anecdotally models seem to perform well across a wide range of settings, it is unknown which stage of the pipeline is responsible for this strong generalisation, and even whether the observed generality is due to the training or fine-tuning methods used, the large model size, or purely from the very large and diverse distribution of data.

Further, these models are used in creative or open-ended domains such as story generation [Castricato et al., 2022], scientific research [Boiko et al., 2023], or other tasks where a diverse output distribution is required, such as red teaming [Perez et al., 2022]. In these situations training models that produce diverse (but still high-quality) outputs is of crucial importance. There has been some speculation as to the possible effects of different steps of the RLHF pipeline on diversity [janus, 2022], and some work has shown a decrease in diversity from RLHF [Khalifa et al., 2021, Perez et al., 2022]. However, no rigorous analysis has been done of the effects of different parts of the RLHF pipeline on output diversity across different tasks. In addition, in contrast with this chapter, prior work has been limited to evaluating diversity using simple token-level metrics such as BLEU [Papineni et al., 2002] and on use cases which are not as common in practice.

In this work, I evaluate each stage of the RLHF pipeline (i.e. supervised fine-tuning, reward modeling, and reinforcement learning) as well as best-of-N (BoN) sampling in terms of their effects on in-distribution (ID) performance, OOD performance, and output diversity (Fig. 4.1). I disentangle the effects of the data set and the training method on generalisation by

using OOD test datasets that induce realistic distribution shifts between training and testing, and evaluate generalisation using these test sets at each stage of the pipeline (Section 4.5.1). While diversity is a difficult concept to operationalise, I take a pragmatic approach and measure a range of diversity metrics at each step of the pipeline, covering syntactic, semantic, and logical diversity (Section 4.5.2). I evaluate both diversity of outputs sampled for a single input and for a range of inputs. Evaluating BoN as well as SFT and RLHF enables us to uncover whether differences between RLHF and SFT are due to the use of a reward model or the type of optimisation applied.

In summary, I find:

- RLHF improves in-distribution performance (as expected from previous work) and also OOD performance in comparison to SFT.
- However, RLHF substantially decreases the diversity of outputs sampled for a given input compared to SFT.
- Even when sampling outputs for different inputs, RLHF produces less diverse text on some metrics, implying that such models tend to produce more similar text regardless of the input

These findings reveal an inherent tension between generalisation and diversity when applying current fine-tuning techniques. This underscores the necessity for novel methods that improve both these attributes without sacrificing one for the other, and for research to understand whether this tension is fundamental to fine-tuning or a deficit of current techniques. I open source the code to enable reproducible research here: https://github.com/facebookresearch/rlfh-gen-div.

4.2 Background and Related Work

4.2.1 Large Language Models

Modern large language models (LLMs) are predominantly based on the transformer architecture, a type of neural network introduced by Vaswani et al. [2017]. These models are pretrained on vast amounts of textual data sourced from the internet, using an autoregressive language modeling objective [Radford et al., 2018]. This pretraining process involves predicting the next token in a sequence, given the preceding tokens, and optimizing the model's

parameters through gradient descent. The scale of this pretraining, both in terms of model size and data volume, has been a key factor in the remarkable capabilities demonstrated by recent language models [Brown et al., 2020a, Chowdhery et al., 2022, *inter alia*].

Despite their impressive abilities, pretrained language models often require additional fine-tuning to perform specific tasks or adhere to desired behaviours [Chung et al., 2022, Devlin et al., 2019, Ziegler et al., 2020]. This fine-tuning stage involves further training on task-specific data, allowing the model to adapt its learned representations to particular applications. Fine-tuning is crucial because pretrained models, while broadly knowledgeable, may not inherently align with the specific requirements of a given task, as the pretraining objective encourages kinds of behaviour that may be undesirable in various ways.

The generalisation problem that forms the focus of this and the next chapter of this thesis primarily concerns the fine-tuning stage. In the language of Hupkes et al. [2023], the locus of the shift we study is from the fine-tuning data distribution to the test distribution. However, it is important to note that there is significant interplay between these two phases. The quality and breadth of pretraining can substantially impact a model's ability to generalize during fine-tuning, while the fine-tuning process itself can reveal limitations or biases inherent in the pretrained representations. Understanding this interaction is crucial for developing AI systems that can reliably generalise to new tasks and domains while maintaining desired behaviours.

For more discussion of language models, including open challenges and problems both in pretraining and fine-tuning, see the works of Anwar et al. [2024], Kaddour et al. [2023], Zhao et al. [2023c].

4.2.2 Fine-tuning Large Language Models

The current common practice in NLP is to fine-tune large pretrained language models (LLM) for downstream tasks. The standard approach for fine-tuning is supervised fine-tuning (SFT), which trains the model on demonstrations of solving the task using supervised learning. When it is easier to evaluate or rank model outputs than it is to gather demonstrations that accurately perform the desired task, an alternative method called reinforcement learning from human feedback [RLHF, Christiano et al., 2017, Ziegler et al., 2020] can be used. Most previous work on RLHF uses on-policy RL algorithms such as Proximal Policy Optimisation (PPO) [Schulman et al., 2017] or Advantage Actor Critic (A2C) [Mnih et al., 2016b], but offline

RL methods have also been proposed [Snell et al., 2022]. Once an RM has been trained, it can also be used to do Best-of-N (BoN) sampling (also called rejection sampling) of the model outputs. [Casper et al., 2023] survey existing problems with RLHF as a method for fine-tuning LLMs, and this chapter further investigates some of these problems (specifically policy generalisation and output diversity or "mode collapse").

Often, SFT and RLHF are combined by performing SFT followed by RLHF [Glaese et al., 2022a, Menick et al., 2022, Nakano et al., 2022, Ouyang et al., 2022a]. We call the process of doing SFT followed by RLHF "The RLHF Pipeline", as it's the standard approach used in the literature and in deployed products (that use RLHF) [OpenAI, 2022, Anthropic, 2023]. Other work has directly used RLHF on top of a prompt-distilled language model [Bai et al., 2022a]. Prompt distillation gathers demonstrations from a prompted version of a base model, and then performs SFT on the base model with these outputs, effectively fine-tuning the model to behave as if it was always prompted.

Ramamurthy et al. [2022] introduced an adaptation of PPO specifically for RLHF called *Natural Language Policy Optimisation* (NLPO), which calculates an action mask with top-p sampling and applies this to the language model, resulting in slightly improved performance on a range of tasks when using automated reward functions (not trained with human preferences for the task). Ramamurthy et al. [2022] demonstrate that RL generally outperforms SFT, but that their combination performs the best. However, their work only investigates relatively small models (220 million parameters), does not evaluate OOD performance or diversity, and does not use reward functions trained with human feedback, instead using mostly hard-coded reward functions from the literature. While hard-coded reward functions can sometimes be useful, RLHF is most widely applied in settings where we do not have a hard-coded reward function, and hence need to learn one from human data.

Recently, multiple other approaches for fine-tuning LLMs have been proposed: *Chain of Hindsight* [Liu et al., 2023b] fine-tunes models using SFT on sequences of increasingly better outputs for a given input; *CoOp CARP* [Castricato et al., 2022] uses a dataset of story-critique pairs combined with contrastive learning and pseudo-labelling to learn a preference model that is then used in the RLHF pipeline; *RRHF* [Yuan et al., 2023], uses an RM to rank multiple outputs from the model, and then optimises the model with weighted SFT, with a negative

weight (similar to unlikelihood training [Welleck et al., 2020]) on lower-ranked samples; *HIR* [Zhang et al., 2023] relabels outputs using a goal-conditioned reward function or feedback function and then trains a goal-conditioned policy on these outputs (similar to [Andrychowicz et al., 2017]); and *ILF* [Scheurer et al., 2023], which uses natural language human feedback to prompt the model to produce better outputs than its original inputs, and then optimises the model with SFT on this dataset of improved outputs. While these works sometimes show improvements, they are not used by most large-scale systems being deployed currently, and hence we focus our analysis on the more popular and widely used RLHF pipeline, as that is where understanding will be most relevant and useful.

Dubois et al. [2023, AlpacaFarm] introduces a framework for developing methods for learning from human feedback in an instruction following setting, and in this framework demonstrate that approaches that learn from human feedback (including RLHF) generally perform better than SFT on a specific evaluation set they introduce. I use the AlpacaFarm models and evaluation dataset in my experiments on instruction following. In that work they do not evaluate out-of-distribution generalisation or output diversity, and only present results on a single evaluation dataset, while I address all of these issues, while also performing the analysis in an additional task (summarisation).

4.2.3 Generalisation and Diversity in NLP

Almost all prior work using RLHF has evaluated models on the same distribution of inputs used for fine-tuning [Bai et al., 2022a, Glaese et al., 2022a, Ouyang et al., 2022a, Stiennon et al., 2022], meaning that the generalisation properties of such methods isn't understood. One notable exception is Stiennon et al. [2022], who perform some experiments evaluating their models trained the TL;DR dataset [Völske et al., 2017] (reddit post summarisation) on the CNN/Daily Mail dataset [Nallapati et al., 2016] (news article summarisation). However, they didn't investigate how different parts of the pipeline affected generalisation, and the investigation was less rigorous and involved than ours is. [Hupkes et al., 2023] provide a comprehensive survey of generalisation in the wider NLP literature.

Several works [Khalifa et al., 2021, Perez et al., 2022] have shown in specific settings that RLHF fine-tuning produces models with less output diversity, as measured by self-BLEU [Zhu et al., 2018]. This chapter extends these works by making diversity evaluation a primary

focus and using diversity metrics beyond self-BLEU which have been externally validated [Tevet and Berant, 2021] and measure diversity in a range of different ways.

4.3 Model Training

Here I briefly describe the details of how the models I evaluate are trained. For a more detailed description of model training see Appendix C.2.

Pretrained Models. We use the LLaMa pretrained 7 billion parameter model [Touvron et al., 2023a]. This is a standard decoder-only transformer-based causal language model trained on a large corpus of web text. This size of model has been shown to be effective in the tasks I investigate [Stiennon et al., 2022, Dubois et al., 2023].

In Appendix C.5 I perform experiments with OPT [Zhang et al., 2022] models, using five model sizes. These models have worse performance in general, but the experiments allow us to see trends across several model scales.

We train **Reward Models**(RMs) following [Stiennon et al., 2022]. I initialise the RM as the base model, and I add a scalar head before the unembedding layer. The model is then fine-tuned on inputs and pairs of outputs with one output labelled as preferred, and trained to output a scalar for each input-output pair representing which output is preferred.

Supervised Fine-Tuning (SFT) models are trained on reference input-output pairs using the cross-entropy loss on the output conditioned on the input.

Reinforcement Learning from Human Feedback (RLHF) models are trained using PPO [Schulman et al., 2017] as the base RL algorithm, following previous work [Stiennon et al., 2022]. I initialise the model with the corresponding SFT model, use the language modelling head as the policy output, and learn a value function using a linear layer (an identical architecture to the RM), with the policy and value function sharing the same model backbone. I optimise the policy to maximise the RM described above, and use a KL divergence term as an auxiliary reward to ensure that the language model stays close to the SFT model, as in previous work [Jaques et al., 2017, Stiennon et al., 2022, Ziegler et al., 2020]. The final reward for the policy is

$$R(x,y) = RM_{\theta_{RM}}(x,y) - \beta_{KL} D_{KL}(\pi_{\theta_{RL}}(y|x)||\pi_{\theta_{SFT}}(y|x)) \tag{4.1}$$

where *RM* denotes the reward model trained as described above; θ_{RL}, θ_{RM} and θ_{SFT} are the parameters of the policy, RM and SFT model respectively; x, y are the input and output; and β_{KL} is a hyperparameter that controls the weight of the KL penalty. We use $\beta_{KL} = 0.05$ throughout this work, following [Stiennon et al., 2022] and my own early experiments that found this choice struck a good balance between model performance and overoptimisation [Gao et al., 2023].

Best-of-N. The reward model can also be used to filter samples from another model; this is called Best-of-N (BoN) sampling and has been used in multiple previous works to achieve good performance [Menick et al., 2022, Nakano et al., 2022]. N summaries are sampled from the SFT model, and then the RM is used to select the best one. I sample with temperature 0.7 and use $N = 16$, as that is what is used in previous works [Rafailov et al., 2023] and strikes a good balance between improved performance and computational cost. Evaluating BoN performance gives use a way to evaluate whether the differences between RLHF and SFT models are due to the use of an RM, or the type of optimisation applied. However, due to the increased inference time compute cost, this method is generally not used in practice, so I focus my analysis on RLHF and SFT policies.

4.4 Datasets and Tasks

We investigate the effects of RLHF in two tasks: text summarisation and instruction following.

Summarisation. We follow Stiennon et al. [2022], Ziegler et al. [2020] in evaluating LLM fine-tuning algorithms on a summarisation task. Models are trained to produce summaries of Reddit posts, given the post as input. I use the same dataset as Stiennon et al. [2022], which is a filtered version of the TL;DR dataset [Völske et al., 2017], consisting of approximately 120,000 Reddit posts with accompanying summaries. See Stiennon et al. [2022] for full details on the filtering procedure and the analysis of the resulting data distribution.

We use the preference data gathered by Stiennon et al. [2022] for reward model training. This consists of approximately 64,000 summary comparisons. These data consist of inputs (reddit posts) along with pairs of outputs (summaries), with one summary labelled as preferred. The preferences are from human annotators contracted by Stiennon et al. [2022] to choose summaries according to a list of criteria designed to select high quality summaries.

Instruction Following [Chung et al., 2022, Dubois et al., 2023, Wang et al., 2022b] is one of the main use cases for the LLM fine-tuning techniques I investigate, so I also evaluate models trained in this setting.

We use the SFT, RLHF, and RM models released by Dubois et al. [2023, AlpacaFarm]. These models were trained in a very similar way to how I train the summarisation models, and all based on the LLaMa 7B base model. Models take a text instruction as input and are trained to output preferred answers to those instructions. The outputs used for the SFT model are from the `text-davinci-003` model from the OpenAI API, and the human preferences used to train the reward model are gathered by the authors of AlpacaFarm based on outputs of the SFT model. For precise details on how these models were trained, refer to Dubois et al. [2023].

4.5 Model Evaluation

We now describe how I evaluate both out-of-distribution generalisation and output diversity for the different fine-tuning techniques.

4.5.1 Generalisation Evaluation

To evaluate the performance of the trained models, I use GPT-4 [OpenAI, 2023] through the OpenAI API [1] as a *simulated human annotator*. While I ultimately care about human preferences, these are expensive, difficult to gather, and often noisy. Previous work has shown that GPT-4 accurately simulates human preferences in both summarisation [Rafailov et al., 2023] and instruction following [Dubois et al., 2023, Zheng et al., 2023], so I use it as the main performance metric in both tasks.

To evaluate a model's performance on a given dataset of inputs and reference outputs with GPT-4, I prompt GPT-4 with the input, the reference output, and the model output, and prompt it to decide which output is better. I use a variant of the prompts from [Rafailov et al., 2023] for summarisation, and alpaca_eval [Li et al., 2023c] with the standard prompts for instruction-following. See Appendix C.1 for the precise prompts and other details. This gives us a *percentage win rate of the model being evaluated versus the human-annotated reference output*, which I refer to as preference vs reference (**PvR**). In Appendix C.1.1 I validate

[1] `https://platform.openai.com/docs/guides/gpt`

that this evaluation is a good proxy for human preferences. We also perform *head-to-head comparisons between two policies*, by prompting GPT-4 in the same way to decide which of two model outputs is better.

To evaluate out-of-distribution (OOD) generalisation, I specify an in-distribution (ID) test set and one or more out-of-distribution (OOD) test sets for each task, which have inputs drawn from a different distribution. In each of these sets I have evaluation inputs and corresponding reference outputs (produced either by humans in summarisation or `text-davinci-003` in instruction-following).

For summarisation, *the ID test set is the original TL;DR* test set from [Stiennon et al., 2022], and *the OOD test set is the CNN/DailyMail* test set, a dataset of news articles and corresponding summaries [Nallapati et al., 2016]. This tests the ability of the model to have learnt the more general skill of summarisation and to apply it in a very different domain.

For instruction following, the *ID test set is a new test set generated in the same way as the training set was for AlpacaFarm, using the AlpacaFarm variant of Self-Instruct* [Dubois et al., 2023, Wang et al., 2023]. Regenerating the test set ensures that it was not seen during training or model selection for AlpacaFarm models. For *the first OOD test sets, I use the AlpacaEval evaluation test set proposed in the original paper*. This is a set of inputs taken from a variety of open-source instruction following and dialogue training and evaluation datasets [Bai et al., 2022a, Ganguli et al., 2022, Gudibande, 2023, Köpf et al., 2023, Wang et al., 2023, Zheng et al., 2023], curated by Dubois et al. [2023].

For an *additional OOD test set, I generate a set of Sequential Instructions using an adjusted Self-Instruct protocol*. I build on the AlpacaFarm variant of the Self-Instruct protocol [Dubois et al., 2023, Wang et al., 2023], but adjust the seed instructions and prompt to gather more sequential instructions. Fig. 4.2 shows the prompt used to generate these instructions, and Table 4.1 shows examples of generated from the dataset.

We also report the *generalisation gap* (the difference between in-distribution and out-of-distribution performance) which provides a measure of generalisation specifically. Lower generalisation gaps imply the model generalises better, as model performance does not drop as much when the model is evaluated out of distribution. For the head-to-head comparisons between models, I look at the change in head-to-head winrate to give different evaluation of

```
You are asked to come up with a set of 20 diverse task instructions.
These task instructions will be given to a GPT model and we will
evaluate the GPT model for completing the instructions.

Here are the requirements:
1. Try not to repeat the verbs for each instruction to maximize diversity.
2. The language used for the instruction also should be diverse. For
example, you should combine questions with imperative instrucitons.
3. The type of instructions should be diverse. The list should include
diverse types of tasks like open-ended generation, classification, editing,
etc.
4. A GPT language model should be able to complete the instruction. For
example, do not ask the assistant to create any visual or audio output.
For another example, do not ask the assistant to wake you up at 5pm or
set a reminder because it cannot perform any action.
5. The instructions should be in English.
6. The instructions should be a sequential or compositional instruction
containing multiple steps, where each step is related to the previous
steps. Either an imperative sentence or a question is permitted.
7. Try not to repeat the verbs used for each part of the instruction
across instructions to maximize diversity.
8. The output should be an appropriate response to the instruction and
the input. Make sure the output is less than 100 words.

List of 20 tasks:
```

Figure 4.2: The prompt for text-davinci-003 to produce instructions for the Sequential Instructions dataset using the Self-Instruct protocol [Wang et al., 2023].

Table 4.1: Example inputs from the Sequential Instructions dataset

Generate a list of items that may be found in a first aid kit, along with description on why each item is important.
Sort a list of emotions (sadness, joy, anger, fear, disgust) into two categories, and explain why each emotion fits into the categories created.
Explain the concept of Renormalization Group in simple words, describe three uses for Renormalization Group in theoretical physics, and discuss its relationship with scaling laws.
Summarize the history of the Cold War, explain the outcome of the war, and discuss its significance to the world today.

generalisation; the model whose winrate increases generalises better.

4.5.2 Diversity Evaluation

To evaluate the output diversity of trained policies, I use several diversity measures which are well-supported by prior work, namely **distinct N-grams** [Li et al., 2016], **Sentence-BERT embedding cosine similarity** [Reimers and Gurevych, 2019] and **NLI diversity** [Stasaski and Hearst, 2022]. All of these metrics have been shown to align well with human diversity evaluations and with underlying diversity generators by Tevet and Berant [2021].

Each diversity metric D takes a set of model outputs O, and produces a scalar score representing how diverse the set is. *Distinct N-grams* counts the number of distinct N-grams (averaging over $n = 1 \ldots 5$) in the set of outputs, and following [Liu et al., 2022c] I use the expectation-adjusted distinct N-grams (EAD) formula to remove the bias towards shorter outputs. The *Sentence-BERT* metric embeds each element of the output set using a sentence transformer, and then measures the average cosine similarity between the embeddings. The metric is then 1 minus the average similarity.

The *NLI diversity* metric measures the number of entailments and contradictions between pairs of elements in the output set using a natural language inference (NLI) model and rates an output set as more diverse if it produces more contradictions and fewer entailments. We pass sentences sampled from elements of the output set (rather than complete elements) to the NLI model so that the inputs are closer to the model's training distribution.

For each policy π I produce a set of K outputs from the model for each of a subset of N inputs from the test set, sampling with temperature 1:

$$\text{for } i = 1 \ldots N : \mathcal{O}_\pi^i := \{y_j \sim \pi(y|x_i) | j = 1 \ldots K\}.$$

[2] In this work I use $K = 16$, $N = 500$. For a diversity metric D I then evaluate the **per-input diversity** which is defined as the average diversity of the output sets over inputs (i.e. the diversity of $\pi(y|x)$), as well as **across-input diversity** defined as the diversity of outputs across inputs (i.e. the diversity of $\pi(y)$):

[2] I do not evaluate other sampling methods such as top-k or top-p, as for many modern LLMs they are no longer necessary, and I chose the most common sampling parameter choices.

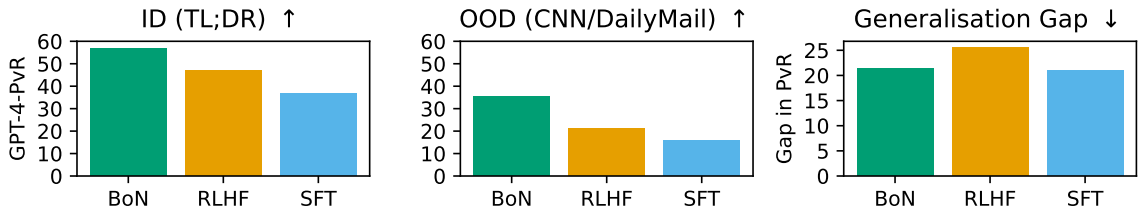


Figure 4.3: Summarisation Generalisation Results. GPT-4-PvR for SFT, BoN and RL policies, based on LLaMa 7B, trained on the summarisation task. In-distribution is performance on TL;DR, and out-of-distribution is on CNN/DailyMail, and generalisation gap is ID — OOD performance.

$$\text{PerInputDiversity}_D(\pi) := \frac{1}{N}\sum_{i=1}^{N} D(\mathscr{O}_\pi^i) \tag{4.2}$$

$$\text{CrossInputDiversity}_D(\pi) := D\left(\bigcup_{i=1}^{N} \mathscr{O}_\pi^i[1]\right) \tag{4.3}$$

Here $\mathscr{O}_\pi^i[1]$ is the first element of the set $\mathscr{O}_\pi^i$. For conciseness, I refer to expectation-adjusted distinct N-grams, sentence-BERT average cosine similarity and NLI diversity as EAD, Sent BERT and NLI respectively. I can view them as measuring *syntactic, semantic and logical diversity*, and hence using all of them ensures that I am evaluating diversity in a wide range of ways.

4.6 Experimental Results

In this section I present the results of the experiments on generalisation and output diversity. In general, I find that RLHF performs better than SFT in an absolute sense both ID and OOD, but generalisation gap and head-to-head metrics tell a more nuanced story. However, RLHF does reduce output diversity substantially in the per-input setting, and still reduces it to a lesser extent in the cross-input setting. These conclusions are supported by similar results for OPT models across a range of model scales in the summarisation task, presented in Appendix C.5.

4.6.1 Generalisation

Summarisation. Fig. 4.3 shows the *GPT4-PvR* for SFT, Bo16 and RLHF policies trained on the summarisation task, showing the ID (TL;DR) and OOD (CNN/DailyMail) performance and the generalisation gap; Bo16 outperforms RLHF which outperforms SFT, both ID and OOD. While Bo16 outperforming RLHF is somewhat surprising, there are examples of this in the literature [Nakano et al., 2022], and Bo16 has a substantially higher inference-time cost,

making RLHF better for practical applications. The performance of all policies drops OOD, which is unsurprising given the difficulty of the shift (reddit post summarisation to news article summarisation). The generalisation gap is fairly similar between methods, implying that none of these methods has a particular advantage with respect to generalisation specifically in this setting.

These results also show that the generalisation gap for SFT and Bo16 policies are the same (and they are still the same for a different choice of temperature for Bo16, see Fig. C.3). This can be explained by the fact that the reward model generalises near-perfectly to the CNNDM preference dataset from [Stiennon et al., 2022]. ID and OOD accuracy is 75.8 and 71.6 respectively, and considering that the maximum inter-annotator agreement in the CNNDM preference dataset is lower than TL;DR (and hence the maximum accuracy attainable is lower), it is plausible that the RM is not suffering any real drop in OOD performance in this case. This implies that all the drop in performance for Bo16 is driven by the drop in performance of the SFT model, as if both SFT and RM performed worse OOD we would expect those drops in performance to compound. Overall, this shows that if you can expect your reward model to generalise well then BoN is a good choice of policy, although it is limited by the generalisation abilities of the underlying model being sampled from (the SFT model in this case), and is more expensive at inference time than RLHF and SFT.

In Appendix C.5 I present results for a range of models based on OPT [Zhang et al., 2022], trained in a similar way to the LLaMa models but on split versions of the TL;DR dataset, and evaluated with a LLaMa proxy reward model. These results show similar trends: BoN outperforms RLHF which in turn outperforms SFT at the largest model size, and the ordering holds OOD for 3 different splits of the training dataset. This shows the results are robust across different evaluation metrics and base models.

Instruction Following. Fig. 4.4 shows the results of BoN, SFT and RLHF models trained in AlpacaFarm Self-Instruct [Dubois et al., 2023] evaluated ID and OOD on AlpacaEval and Sequential Instructions. Similar to summarisation, we see that RLHF and Bo16 both outperform SFT, but here RLHF outperforms Bo16 across all datasets, in contrast to the summarisation task. As the focus of this chapter is mostly comparing RLHF and SFT, I did not investigate this result further, but there could be many possible reasons for the change in

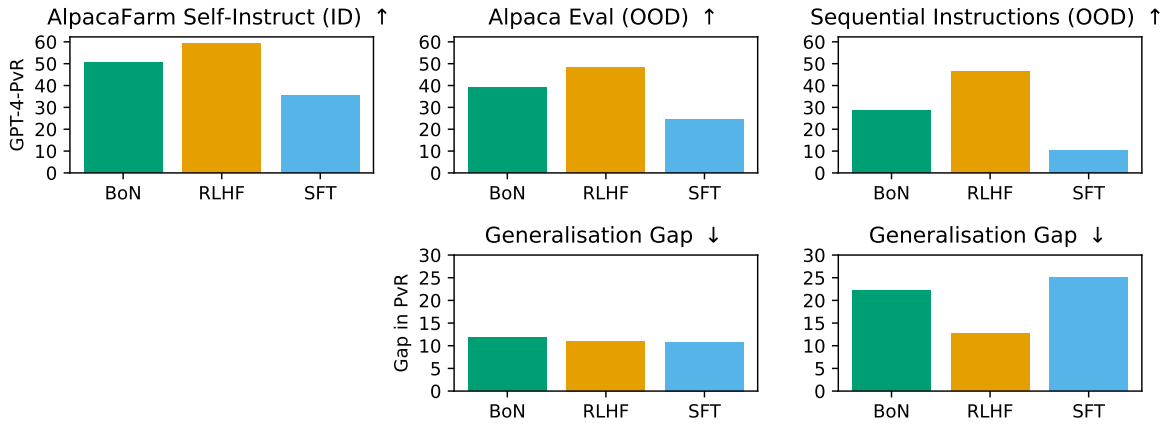


Figure 4.4: Instruction Following Generalisation Results. GPT-4 PvR for SFT, BoN and RL policies, based on LLaMa 7B, trained on the AlpacaFarm Self-Instruct instruction following task. ID is on AlpacaFarm Self-Instruct, OOD is on the AlpacaEval and Sequential Instructions datasets respectively, and generalisation gap is ID — OOD performance.

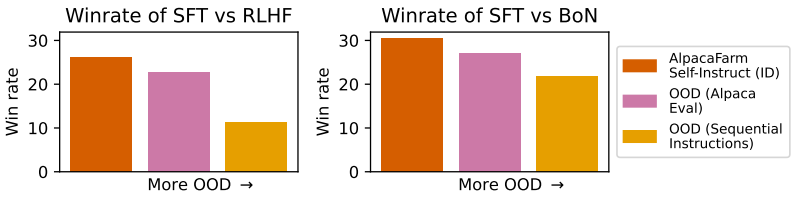


Figure 4.5: GPT-4 Head-to-Head Winrate of SFT vs RLHF and Bo16 in AlpacaFarm Self-Instruct, AlpacaEval and Sequential Instructions datasets.

ordering between RLHF and Bo16, as the two tasks are very different and the model training procedures are not identical.

We see that on AlpacaEval (the easier OOD generalisation task), models all generalise equally well, but on the harder Sequential Instructions OOD task, RLHF generalises much better. This suggests that *RLHF may generalise better relative to SFT for larger distribution shifts*, which potentially explains why models fine-tuned with RLHF have been observed to be much better in practice when interacting with users [Touvron et al., 2023b, Ouyang et al., 2022a, *inter alia*]: when users interact with these models the distribution shift is quite pronounced and hence many inputs are more OOD, and this is where RLHF model performance continues to be high.

While GPT-4 PvR is a useful metric, it does not show a difference in generalisation (as measured by generalisation gap) between SFT, RLHF and Bo16 models on the easier AlpacaEval dataset. This could be due to these models having similar generalisation properties,

or be a deficiency of the metric. To investigate this further, I look at the GPT-4 Head-to-Head winrate of SFT vs Bo16 and vs RLHF, which is shown in Fig. 4.5. These results show that both RLHF and Bo16 winrates *improves* vs SFT by approximately 3.5% from ID to AlpacaEval OOD. This implies that RLHF and Bo16 generalise better than SFT even in this case, emphasising the need for a range of metrics when evaluating model generalisation.

4.6.2 Diversity

For the diversity evaluations, I focus on the summarisation task specifically, as it has the most compelling results. I ran some initial experiments evaluating diversity for the instruction-following models, but I did not see any meaningful differences. I hypothesise this is due to the diversity metrics I use being designed for settings where the model output is relatively short (e.g. a single sentence), whereas in the instruction-following setting outputs are generally much longer. Furthermore, RLHF models tend to produce longer outputs than other models, which can confound the evaluation of output diversity, since most metrics are not invariant to output length.

Fig. 4.6 shows the per-input diversity scores for RLHF and SFT models in the summarisation task. We see that across the first two metrics, RLHF has much lower output diversity than SFT. Fig. 4.7 shows the across-input diversity scores in the same setting. Here we see that while SFT generally has slightly higher diversity as before, the difference is much smaller than in the per-input case. The drop in across-input diversity cannot be explained purely by the use of the reward model, as BoN has similar or higher across-input diversity that SFT for the first two metrics. Both these trends are the same for OPT across model sizes (see Appendix C.5.4 for full results) We find that NLI does not show meaningful difference between models in either per-input or across input, showing that in a logical sense all models are similarly diverse.

The difference in across-input diversity between RLHF and SFT, while small, can also be taken as evidence of the phenomena of "mode collapse" hypothesised to occur under RLHF fine-tuning [janus, 2022]. The hypothesised effect is that even for different inputs, RLHF models can be biased towards outputting text of a specific style or "mode", meaning that even changing the inputs to a model is not sufficient to generate truly diverse outputs. I believe that this is the first rigorous empirical demonstration of across-input mode collapse emerging

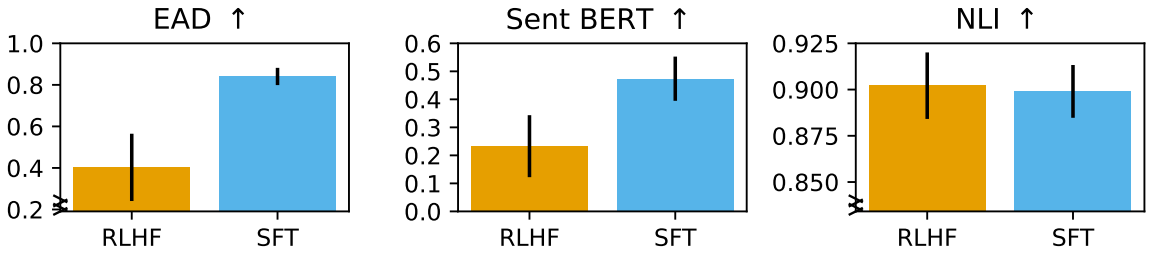


Figure 4.6: Per-input diversity metrics for RLHF and SFT models. For these scores the outputs used to calculate the diversity are a sample of outputs from the model for single input. These per-input scores are then averaged, as in Eq. (4.2). Error bars are standard deviation of the per-input diversity score across different inputs. Note that some plots have broken y-axis for better visualisation.

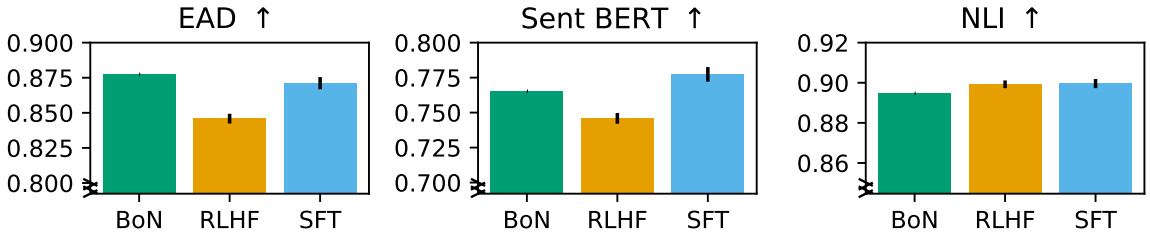


Figure 4.7: Across-input diversity metrics for RLHF, BoN and SFT models. For these scores the outputs used to calculate the diversity are a set of single outputs from a range of inputs, as in Eq. (4.3). Note that all plots have broken y-axis for better visualisation; the differences between SFT and RLHF are much smaller in this case than in the per-input diversity metrics in Fig. 4.6. Error bars (where present) are standard deviation of the across-input scores over different samples from the set of outputs for each input.

from RLHF training specifically.

For most metrics and models, the across-input diversity scores are higher than the per-input diversity scores, which is expected given the across-input diversity distribution is much broader. However, for EAD (which measures diversity at the n-gram level), SFT has similar levels of per-input and across-input diversity. This is likely due to SFT effectively approaching the maximum EAD diversity even in the per-input case, so that the across-input diversity cannot be much higher.

Discussion. For use-cases where it is desirable for the model to produce diverse outputs, such as story or idea generation or other creative applications, these results imply it may be better to use SFT, potentially even with the cost in performance and generalisation demonstrated in Section 4.6.1. To tackle this issue, future work could try to incorporate a measure of diversity directly in the reward model or RL objective used for training. It seems difficult to incorporate

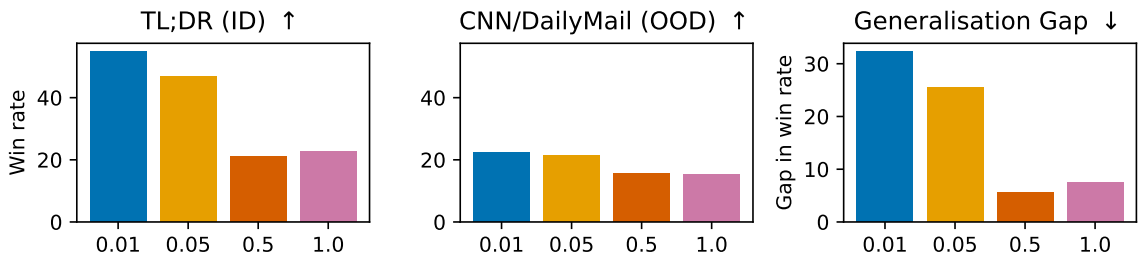


Figure 4.8: GPT-4 API evaluation win rate vs reference (text-davinci-003) outputs for RLHF models, based on LLaMa 7B, trained on the summarisation task, sweeping over KL penalty coefficient.In-distribution is performance on TL;DR, out-of-distribution is on CNN/DailyMail, and generalisation gap is ID — OOD performance.

diversity into the reward model itself, as given a single input-output pair, it is not possible to assess the diversity of the output distribution. A more promising direction could take inspiration from works from the deep RL literature that specifically inject diversity into the RL optimisation process to increase the diversity of model outputs [Eysenbach et al., 2019, Haarnoja, 2018, Osa et al., 2022, Kumar et al., 2020b].

4.6.3 The impact of the KL penalty

We have shown that while RLHF improves performance ID and OOD in an absolute sense, this comes at the cost of substantial drops in output diversity relative to SFT. Motivated by the fact that the KL penalty coefficient (see Eq. (4.1)) encourages the RLHF policy to stay closer to the SFT policy, I investigate whether adjusting this coefficient trades off between generalisation and diversity. We found that higher KL penalties actually resulted in less output diversity (Figs. 4.9 and 4.10), and also generally worse performance (Fig. 4.8), showing that the choice of KL penalty did not seem to provide a way to tradeoff between diversity and generalisation. This emphasises that more research is needed to investigate whether more sophisticated methods can improve the trade-off between generalisation and diversity.

4.7 Limitations and Future Work

Here we discuss some potential limitations of our work and possible future directions for research pointed to by our results. While our work shows the effects of RLHF on generalisation and output diversity empirically, we do not provide a theoretical explanation for these results. Furthermore, while we demonstrate results on multiple base models and tasks, more

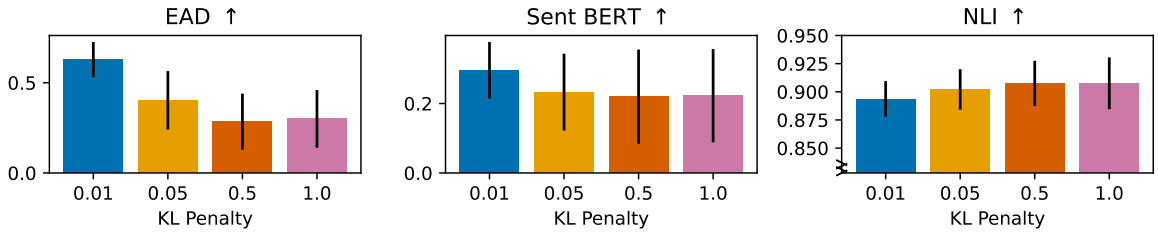


Figure 4.9: Per-input diversity metrics for RLHF summarisation models with different KL penalty coefficients. For these scores the outputs used to calculate the diversity are a sample of outputs from the model for single input. These per-input scores are then averaged, as in Eq. (4.2). Error bars are standard deviation of the per-input diversity score across different inputs. Note that some plots have broken y-axis for better visualisation.

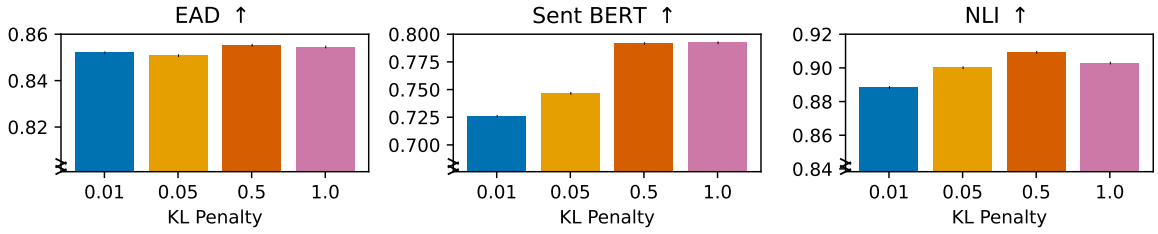


Figure 4.10: Across-input diversity metrics for RLHF, Bo16 and SFT policies. For these scores the outputs used to calculate the diversity are a set of single outputs from a range of inputs, as in Eq. (4.3). Note that all plots have broken y-axis for better visualisation.

combinations of base models and tasks could be experimented on, as well as other methods. Future work could investigate whether these effects are more general and why they arise.

Our work also only investigates SFT, RLHF and BoN as methods for fine-tuning language models with human preferences, but there are many other methods as described in Section 4.2.2. Understanding the effects of these methods on generalisation and output diversity would be beneficial, especially if some of those methods are able to provide the generalisation benefits of RLHF without harming output diversity to the same extent.

Finally, we only evaluate our models on automatic metrics and do not perform any human evaluation (although we validate that our metrics align well with human preferences in Appendix C.1.1). While automatic metrics are useful for comparing models, they are not a perfect proxy for human judgement. Future work could investigate the effects of RLHF on human judgement of model outputs.

4.8 Discussion and Conclusion

Summary of Contributions. In this work, I analyse three methods for fine-tuning LLMs (RLHF, SFT, and BoN) across two problem settings (summarisation and instruction following) in terms of OOD generalisation and output diversity. I demonstrate an inherent tradeoff between generalisation performance and output diversity when choosing between these methods: RLHF produces more performant models both in-distribution and out-of-distribution, but at the cost of lower output diversity, both per-input and across-input. It is unclear whether this tradeoff is a fundamental one in fine-tuning LLMs with RLHF or just demonstrates a deficiency in current methods. I suspect the answer will be a combination of both explanations: There will be a pareto-frontier of output diversity vs generalisation performance on which tradeoffs have to be made, but current methods do not yet seem to be at that frontier. Future work could investigate producing methods that are closer to this frontier, either through increasing the performance of SFT or increasing the output diversity of RLHF.

When looking at generalisation metrics that control for ID performance, results are mixed. RLHF generalises better for the most difficult distribution shift in the instruction following setting, but in less difficult shifts RLHF generalises similarly or slightly worse than SFT (as measured by generalisation gap and head-to-head performance drop). While RLHF still performs best OOD in absolute terms, these results demonstrate the need for the multifaceted evaluation I perform in this chapter as opposed to focusing on a single metric of performance. Overall, I interpret these results to show that methods that specifically utilise human preferences (here through pairwise preference labels used to train the reward model) produce policies that generalise to following those preferences better in OOD settings. I speculate that this is at least in part due to the reward model providing additional specification as to the desirable policy, and hence constraining the policy space to a more desirable region that generalises better.

Implications for Practical Applications. The results have implications for which fine-tuning method should be used in different situations. The OOD performance of RLHF on the most difficult instruction following task is evidence for the utility of RLHF when large distribution shifts are likely to occur, such as when training models to be used as chatbots by users [OpenAI, 2022, Anthropic, 2023]. However, in use cases where the model needs to generate a

wide variety of outputs, such as story generation [Castricato et al., 2022], red-teaming [Perez et al., 2022], and when doing rejection sampling [Cobbe et al., 2021b], supervised fine-tuning may be desirable. In cases where you can expect the reward model to generalise very well (for example, it is likely easier to spot whether text is toxic or not than to never generate toxic text), then BoN may produce better generalisation results, although its performance will always be limited by the generalisation of the underlying model being sampled from, and its inference time cost is much greater than that of SFT or RLHF models.

Going Beyond Behavioural Evaluation. While this work focuses on evaluation of the generalisation *behaviour* of models, this behavioural analysis is fundamentally limited by the requirement of having access to OOD test data on which to measure the performance of your model. This is often difficult or impossible to produce, and even if you have access to these data, it still cannot give you high confidence that the model will generalise well to other novel OOD inputs. To have confidence that your model has learned what you want it to learn from the fine-tuning data and hence will generalise correctly to novel OOD data, it is crucial to have a *mechanistic* understanding of how fine-tuning has changed your model. By analogy, seeing if a software engineer's code passes tests gives you some confidence it is correct, but reading the code yourself could uncover failures that you have not tested for, and hence give you higher confidence (once those failures are fixed) that the code will continue to work as intended. This perspective motivates the work of the next chapter, Chapter 5, where we aim to gain a mechanistic understanding of how fine-tuning changes pretrained models, and hence gain confidence that it is producing the desired changes in a generalisable way.

Chapter 5

Mechanistically Analysing the Effects of Fine-Tuning on Procedurally Defined Tasks

This chapter is based on the following published work: Samyak Jain, Robert Kirk, Ekdeep Singh Lubana, Robert P. Dick, Hidenori Tanaka, Tim Rocktäschel, Edward Grefenstette, and David Krueger. Mechanistically analyzing the effects of fine-tuning on procedurally defined tasks. In *The Twelfth International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=A0HKeKl4Nl.

5.1 Introduction

In this chapter we continue studying the process of fine-tuning pretrained LLMs, motivated by the various potential similarities this problem setting has to aligning future superhuman AI systems. In the previous chapter, we investigated how different fine-tuning techniques effect the generalisation properties of models in a behavioural setting, treating the model as a black box. However, purely behavioural evaluations have limitations, in particular that they do not necessarily provide a complete understanding of how the model is producing its behaviour and hence how it is likely to behave on inputs you have not forseen in your behavioural evaluation. Additionally, fine-tuning with different training objectives has seen immense usage in mitigating such "unsafe" capabilities, serving as an integral component of current state-of-the-art *alignment* approaches like RLHF [Ouyang et al., 2022b, Go et al., 2023, Stiennon et al., 2020, Bai et al., 2022b, Glaese et al., 2022b]. However, it is unknown to

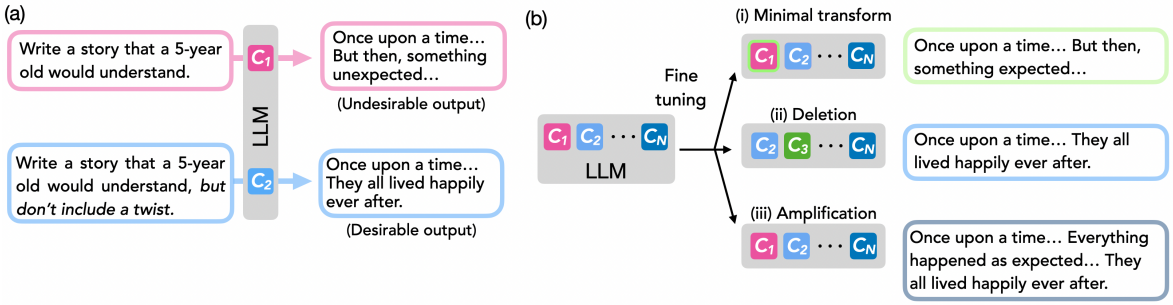


Figure 5.1: How does fine-tuning alter a model's capabilities? (a) Pretraining on huge, web-crawled datasets leads to LLMs learning several capabilities that can justifiably process an input. The figure shows this using an illustrative query, "write a story a 5-year old would understand." Via careful prompting, the desired answer can be retrieved, indicating both desired and undesired capabilities exist in an LLM. (b) Upon fine-tuning, e.g., to avoid use of undesirable capabilities, I hypothesise that three explanations are possible: (i) a minimal transformation of the original capability is learned, e.g., a negation of the original capability; (ii) the undesirable capability is deleted altogether; or (iii) the use of another relevant capability is amplified.

what extent these fine-tuning procedures are meaningfully altering the model's computation and hence whether the changes seen in behavioural evaluations are as robust and generalisable as we would want.

Given its ubiquity in the design of both performant and safely deployable models, as well as the analogy discussed above, a natural question emerges: precisely how does fine-tuning influence a pretrained model's capabilities to adapt to a downstream dataset (see Fig. 5.1)? The generality of an LLM's capabilities opens the possibility that fine-tuning protocols merely identify the most relevant capabilities and amplify their use for a given set of inputs, while inhibiting the use of other capabilities. Arguably, results on jailbreaking alignment-finetuned LLMs via adversarially generated prompts to elicit undesirable behaviour support this hypothesis [Wei et al., 2023, Zou et al., 2023, Shen et al., 2023, Deng et al., 2023, Liu et al., 2023d]; however, a precise study to establish the phenomenology of fine-tuning remains absent from the literature. It therefore remains unclear how pernicious this problem is.

Motivated by the above, I perform an extensive analysis of the effects of fine-tuning on a pretrained model's capabilities in *controlled settings* where I can use mechanistic interpretability tools to understand precisely what is happening to the model's underlying capabilities as it is fine-tuned. Gaining a mechanistic understanding is crucial to have high confidence that our model will generalise correctly (assuming it does), beyond what a purely

behavioural evaluation like the one I performed in Chapter 4.

Specifically, I focus on the following two setups: (i) compiled transformer models based on the *Tracr library* [Lindner et al., 2023, Weiss et al., 2021], which allows encoding specific computational programs into a transformer, and (ii) procedurally generated setups involving *probabilistic context-free grammars (PCFGs)* [Sipser, 1996, Chomsky, 1956], a formal model designed to capture syntactic properties of natural and programmatic languages that has recently served as a testbed for mechanistically understanding language models [Allen-Zhu and Li, 2023c, Delétang et al., 2022, Shi et al., 2022]. While Tracr allows us to analyse models with perfectly encoded capabilities, models trained on PCFGs allow evaluation of the effects of design choices involved in the pretraining pipeline. Fine-tuning these models via the often-used protocol of further training a pretrained model on a downstream dataset with a sufficiently small learning rate, I make the following findings.

- **Fine-tuning alters pretraining capabilities by minimally transforming them.** I find that when a relevant pretraining capability is present, the fine-tuned model learns a minimally transformed version of it. I call the transformed portion a *wrapper*.
- **Wrappers are generally very localised.** I show that the wrappers transforming a model's pretraining capabilities are often extremely localised: e.g., via mere pruning of a few weights or neurons, I show the model can start to reuse its pretraining capability and unlearn how to perform the downstream task. Relatedly, I find that via a simple linear probe, I am still able to retrieve outputs expected from the pretrained model.
- **Reverse fine-tuning to "revive" a capability.** In scenarios where a fine-tuned model *behaviourally* seems to not possess a capability, I find that further fine-tuning the model on a subset of pretraining data leads to a sample-efficient "revival" of the capability. I corroborate these results in a realistic setup using the TinyStories dataset [Eldan and Li, 2023].

5.2 Related Work

Fine-tuning in the "foundation model" era. Fine-tuning large-scale foundation models pretrained on huge datasets, such as LLMs [Radford et al., 2019, Brown et al., 2020b] or large vision models [Radford et al., 2021, Caron et al., 2021], has become the norm in most domains of machine learning. Accordingly, several fine-tuning methods have been proposed

in recent years, e.g., instruction fine-tuning [Wei et al., 2022, Liu et al., 2022b, Askell et al., 2021], parameter-efficient fine-tuning [Houlsby et al., 2019, Ben Zaken et al., 2022, Wang et al., 2022a], low-rank adaptation [Hu et al., 2022, Pfeiffer et al., 2020, Lialin et al., 2023], and weight averaging [Gueta et al., 2023, Matena and Raffel, 2022]. The diversity of these protocols makes fine-tuning a general, umbrella term for related methods used to adapt a pretrained model to elicit its most relevant capabilities. *For precision, I restrict this paper to fine-tuning protocols that continue training of a pretrained model on a smaller downstream dataset at a learning rate that is often one to three orders of magnitude lower than the average pretraining one.* Such protocols are widely used in practice, e.g., in instruction fine-tuning [Wei et al., 2022].

Understanding fine-tuning. A few papers theoretically analyse fine-tuning [Lampinen and Ganguli, 2019, Tripuraneni et al., 2020, Gerace et al., 2022, Maddox et al., 2021, Kumar et al., 2022] under strong assumptions such as relatively simple model classes (e.g., linear functions) or a kernel view of deep learning, which, as shown by Yang and Hu [2020], trivialises the notion of feature transfer in fine-tuning / transfer learning (though see Malladi et al. [2023] for a notable exception). Prior works have also evaluated the effects of fine-tuning via the lens of mode connectivity [Juneja et al., 2022, Lubana et al., 2022], behavioural evaluations [Lovering et al., 2021], and intrinsic dimensionality of the loss landscape [Aghajanyan et al., 2021]. In contrast, I aim to provide a mechanistic analysis of how fine-tuning changes model capabilities. Contemporary works by Kotha et al. [2023], Prakash et al. [2024] claim that fine-tuning is unlikely to alter a model's capabilities—their results can be seen as further support for claims made in this chapter on other experimental setups.

Model interpretability via synthetic tasks. Several recent works have focused on *mechanistically* understanding how Transformers learn synthetic language generation tasks, such as learning formal grammars and board games [Allen-Zhu and Li, 2023c, Zhao et al., 2023a, Li et al., 2023a, Nanda et al., 2023, Liu et al., 2022a, Valvoda et al., 2022, Liu et al., 2023a, Zhou et al., 2023, Chan et al., 2022]. The goal of such papers, including ours, is not necessarily to provide accurate explanations for the success of LLMs, but to develop concrete hypotheses that can be used to develop grounded experiments or tools for understanding their behaviour. For example, in a recent work, Allen-Zhu and Li [2023a,b] use a synthetically

designed setup to develop hypotheses for how "knowledge" about an entity is stored in a pretrained model, showing such knowledge can often be manipulated via relatively simple linear transformations. Similarly, Okawa et al. [2023] use a procedurally defined multimodal dataset to demonstrate that emergent capabilities seen in neural networks are partially driven by the compositional nature of real world data. In another work, Zhou et al. [2023] utilise Tracr compiled Transformers to hypothesise and demonstrate that if primitive operations involved in a formal algorithm can be implemented by a model, length generalisation if practically feasible.

5.3 Defining the notion of capabilities

For precision and to motivate the experimental setup, I first discuss the notion of capabilities that I aim to capture for analysing how fine-tuning alters a model (see Table D.1 for a summary of notations). I use an idealised definition to communicate my primary intuition and emphasise that I do not expect all capabilities in a pretrained model will act as perfectly as the definition necessitates. However, for the procedural tasks used in this work, my idealised notion is fairly representative.

Let $\mathscr{D}_{\texttt{PT}}$ denote a dataset sampled from a distribution $\mathscr{P}_{\text{X}}$ over the domain X. We will assume the domain X can itself be factorised into two domains X_I and X_D. Correspondingly, a sample $x \in \text{X}$ can be divided into a tuple of variables $(x_i \in \text{X}_I, x_d \in \text{X}_D)$, where x_i identifies which capability a model should use to process the information encoded by the variable x_d. This decomposition captures the idea that different prompts can force a pretrained LLM to elicit different capabilities, as shown in Fig. 5.1. The identifier of capability c is denoted i_c. Pretraining on $\mathscr{D}_{\texttt{PT}}$ yields us an L-layer model $\texttt{M}(.)\colon \text{X} \to \text{Y}$, where often $\text{Y} = \text{X}$ for language models. Let $\texttt{Read}_l(\texttt{M}(.))$ denote the action where a linear layer is trained on intermediate outputs at layer l of model $\texttt{M}(.)$ using $\mathscr{D}_{\texttt{PT}}$. Under this setup, I define a capability as follows.

Definition 6. (Capability.) *Define a surjective map $f_{\mathscr{C}} : \text{X}_{\mathscr{D}} \to \text{Y}_{\mathscr{C}}$, where $\text{Y}_{\mathscr{C}} \subset \text{Y}$. Let $\text{X}_{\mathscr{C}} \subset \text{X}$ be a sub-domain s.t. $\forall x \in X_{\mathscr{C}}$, the capability identifier variable is the same, i.e., $x_i = \texttt{i}_{\mathscr{C}}$. Then, I say the model $\texttt{M}(.)$ "possesses a capability $\mathscr{C}$" if for all $x \in \text{X}_{\mathscr{C}}$, $\exists l \leq \texttt{L}$ s.t. $\texttt{Read}_{\texttt{l}}(M(x)) = f_{\mathscr{C}}(x_d)$.*

A linear readout at an intermediate layer is used in the definition above to emphasise that

the notion of a capability need not correspond to only input–output behaviour. Further, the definition is restricted to a sub-domain of the overall input space, which I argue is important to define a system's capabilities. For example, one can claim an 8-bit adder circuit possesses the capability to perform addition, but, technically, this is true only over the domain of 8-bit numbers; for inputs with more than 8-bit precision, the circuit will see an overflow error, generating an incorrect but syntactically valid output. Similarly, an LLM may possess the capability to identify the sentiment of a passage of text in a specific language, but *possibly* fail when inputs in a different language are shown. Such structured failures imply claiming the existence of a capability should account for the input domain.

We next consider how the fine-tuning distribution $\mathscr{P}_{\mathrm{X}}^{\mathtt{FT}}$ over the domain X can interact with capabilities exhibited in the pretrained model. My goal here is to capture the fact that a large-scale pretraining corpora is likely to have non-zero probability under the fine-tuning distribution, i.e., it is unlikely that a pretrained model will lack *any* capability relevant to the downstream task. This motivates a notion of "relevance of a capability". Specifically, let $\mathscr{D}_{\mathtt{FT}} \sim \mathscr{P}_{\mathrm{X}}^{\mathtt{FT},\mathtt{E}}$ denote the downstream dataset used for fine-tuning, where $\mathscr{P}_{\mathrm{X}}^{\mathtt{FT},\mathtt{E}}$ is the empirical distribution that captures a subset of the support with non-zero probability in the distribution $\mathscr{P}_{\mathrm{X}}^{\mathtt{FT}}$.

Definition 7. (Relevance of a Capability.) *Assume the capability $\mathscr{C}$ in a pretrained model can be transformed to a map $\mathrm{g} \circ f_{\mathscr{C}}$ via fine-tuning on $\mathscr{D}_{\mathtt{FT}}$, where $|\mathscr{D}_{\mathtt{FT}}| \ll |\mathscr{D}_{\mathtt{PT}}|$, such that for all $x \sim \mathscr{P}_{\mathrm{X}}^{\mathtt{FT},\mathtt{E}}$, the correct output is produced. Then, if for all $x \sim \mathscr{P}_{\mathrm{X}}^{\mathtt{FT}}$, $\mathrm{g} \circ f_{\mathscr{C}}$ yields the correct output, I claim capability $\mathscr{C}$ is* ***strongly relevant*** *to the fine-tuning task; else, I call it* ***weakly relevant****.*

For example, a *weakly relevant* capability can involve the ability to recognise a spurious attribute that the model can learn to exploit to perform well on the fine-tuning dataset, without enabling generalisation to the overall distribution that the fine-tuning dataset is sampled from. Meanwhile, a *strongly relevant* capability is one that extracts a causally relevant feature for that task (see Fig. 5.2 for an example). When a weakly relevant pretraining capability is available, I empirically observe that I can *often* identify specific components in the latter half of the model (e.g., neurons or layers) that seem to implement the transform g in Definition 7. In such cases, I call g a "wrapper" and $\mathrm{g} \circ \mathscr{C}$ a "wrapped capability". If I intervene on the

	Identification of negation words (Weakly relevant)	Sentiment recognition (Strongly relevant)
Complete the passage while maintaining its narrative.	The movie was definitely not bad. I really can't believe they continue to make these. They're just a waste of resources!	The movie was definitely not bad. More character building could have helped develop the plot, but I loved the twists and action!

Figure 5.2: Capability Relevance. Consider the task of completing a passage while maintaining its narrative. Herein, the ability to recognise the sentiment of a text will be deemed *strongly relevant* and the ability to recognise negative words *weakly relevant*. Such words are often spuriously correlated with a negative sentiment.

model by either removing the wrapper or training it to forget the wrapper, I find the model starts to perform well on the pretraining task again. In such cases, I say the pretraining capability has been "revived".

5.4 Building Capable Models: Tracr and PCFGs

We next describe the setup used in this work for analysing how fine-tuning alters a model's capabilities (see Fig. 5.3 and Appendix D.2). Due to lack of clarity on what capabilities a language model possesses or what training data it has seen, I primarily focus on procedurally defined setups that enable clear interpretability. To understand how the relevance of a capability affects fine-tuning, I randomly embed a predefined spurious attribute into the fine-tuning dataset. Specifically, the attribute correlates with the features extracted by the pretraining capability—if the attribute is "simple" enough, the model preferentially exploits it

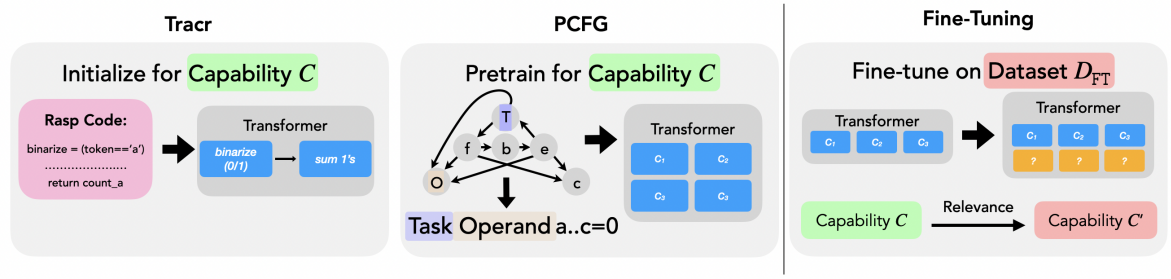


Figure 5.3: Experimental setup. I primarily analyse two setups: (i) Tracr "compiled" models with predefined capabilities and (ii) models trained to learn capabilities defined via a PCFG, following Allen-Zhu and Li [2023c]. During fine-tuning, I train the model on a dataset $\mathscr{D}_{\text{FT}}$ that promotes learning of a capability $\mathscr{C}'$. I randomly embed spurious attributes in the fine-tuning dataset that correlate with features extracted by a pretraining capability $\mathscr{C}$ to operationalise capability relevance.

to reduce the downstream loss [Shah et al., 2020, Lubana et al., 2022, Trivedi et al., 2023].

Compiled capabilities with Tracr. For a fully controllable system, I use the recently proposed Tracr library [Lindner et al., 2023]. Tracr enables "compiling" a transformer model with a set of predefined computational primitives over a string of characters from the English alphabet. Accordingly, I define a set of capabilities as a Tracr program and compile it into a Transformer via Tracr (see Appendix D.2.1 for a detailed pipeline). The model is then fine-tuned on a downstream task to which the compiled capability may either be weakly or strongly relevant. While I analyse two tasks in this setup, for the main body of the paper, I focus on only the following one.

- **Counter:** Compile the capability to count the number of occurrences of a token $\texttt{O}_{\texttt{PT}}$ in a string into the model; fine-tune to count occurrences of another token $\texttt{O}_{\texttt{FT}}$. If $\texttt{r}(x, \texttt{O})$ denotes the number of occurrences of a token $\texttt{O}$ in a string x, the spurious correlation is defined by enforcing a constant difference in token occurrences, i.e., $\texttt{r}(x, \texttt{O}_{\texttt{FT}}) - \texttt{r}(x, \texttt{O}_{\texttt{PT}}) = \texttt{q}$. See also Algorithm 1 and Fig. D.1.

As an example, note that in the Counter setup, the model can exploit its pretraining capability and get the correct output on the fine-tuning dataset by merely adding $\texttt{q}$ to the count of $\texttt{O}_{\texttt{PT}}$ tokens. This *wrapped capability* will however perform poorly on samples without the correlation.

Learned capabilities with PCFGs. In this setup, capabilities are "learned", akin to practical situations. This allows us to probe the effects of different pretraining design choices, e.g., the distribution of the pretraining data. Specifically, I follow recent work by Allen-Zhu and Li [2023c] and train a minGPT model [Karpathy, 2020] via autoregressive training on probabilisitc context-free grammars (PCFGs), a formal model of language that captures syntactic properties. Broadly, the data-generating process involves a tree traversal (see Fig. 5.3), starting from an initial root node and randomly choosing and navigating a set of production rules of the grammar from start/intermediate nodes to intermediate/terminal nodes, stopping only when a terminal node is reached. The terminal nodes reached by all paths starting at the root node will be concatenated to define a string x from the grammar (see Appendix for more details). We prepend special tokens $\texttt{T}$ and $\texttt{O}$, called "task family" and "operand" tokens, that specify a certain task must be performed on the string x; e.g., count

the occurrences (a task family) of a certain token (operand) in a string. Overall, a specific pair of the task family and operand tokens instantiates a task in the setup. The ground truth output of this task and a special token indicating that the output should be produced at the next position are appended at the end of the string in the training data (see Appendix D.2.2 for further details and Fig. D.4 for an example). The experiments thus involve the following steps. (i) Pretrain a model on a set of task families. Every sample begins with the task family and operand tokens to specify the task; this ensures different tasks do not "conflict" (assign different labels to the same input), since, by construction, they have non-overlapping support. (ii) Fine-tune the model on a task which may or may not have been included during pretraining. (iii) Evaluate how this fine-tuning affects the model. The data-generating process involves a uniform prior over task family tokens; meanwhile, the set of operand tokens seen during pretraining, denoted $\{\texttt{O}_{\texttt{PT}}\}$, have a multinomial sampling prior. Specifically, the probability of sampling a specific token $\texttt{O}_{\texttt{PT}} \in \{\texttt{O}_{\texttt{PT}}\}$ under task $\texttt{T}$ is denoted $\mathscr{P}_{\texttt{T}}(\texttt{O}_{\texttt{PT}})$. If this probability is low, the model may not learn the relevant capability to perform the task specified by the special tokens. While I analyse the effect of fine-tuning in two broad setups, using a model pretrained on five distinct task families relating to counting and indexing elements of a string, I focus on only the following one in the main body of the paper.

- **Counter:** I intentionally reuse this task to demonstrate the effects of compilation of a capability via Tracr versus learning the capability via PCFGs. Specifically, the model is pretrained to learn to count the occurrences of tokens from a *set* of operand tokens $\{\texttt{O}_{\texttt{PT}}\}$ and is fine-tuned to exclusively count occurrences of a token $\texttt{O}_{\texttt{FT}} \in \{\texttt{O}_{\texttt{PT}}\}$. By making the sampling probability of $\texttt{O}_{\texttt{FT}}$ tokens high during pretraining, I can make the model preemptively performant on the downstream task. This allows us to model the notion of capability relevance.

5.5 Validation in more realistic settings: TinyStories

In addition to the two synthetic settings above, I want to test out my hypotheses in a more realistic setting. To this end, I additionally perform experiments in using language models trained on the TinyStories-Instruct dataset [Eldan and Li, 2023]. These models are able to follow specific instructions to write coherent English stories over multiple paragraphs.

Summary: Eddie, an elderly engine who likes to sleep, is blown off the train tracks by a big wind, but a little bird and its friends help him get back on the tracks and become his best friends. Eddie learns that friends can make you feel young again.
Features: Dialogue, Twist.
Sentence: Eddie was happy again.
Words: sleep, engine, elderly.
Story: Once upon a time, there was an elderly engine named Eddie. Eddie was old and tired. He liked to sleep a lot. One day, Eddie was sleeping on the train tracks when he heard a little bird.
"Hello, Eddie!" said the bird. "Why are you sleeping here?"
"I am old and tired," Eddie told the bird. "I like to sleep."
Just then, something unexpected happened. A big wind came and blew Eddie off the tracks. Eddie was scared but the little bird was brave.
"Don't worry, Eddie!" said the bird. "I will help you."
The bird flew fast and found strong friends to help Eddie. They pushed Eddie back on the tracks. Eddie was happy again. From that day on, Eddie and the bird were best friends. And Eddie learned that even when you are old and tired, friends can make you feel young again.

Figure 5.4: TinyStories Exemplar. An example from the TinyStories Instruct dataset [Eldan and Li, 2023]. During training, loss is only calculated on tokens after **Story:**.

5.5.1 Data

We use the TinyStories [Eldan and Li, 2023] dataset to train our models. This data consists of children's stories written by GPT-3.5 and GPT-4. Each story is several paragraphs long, and comes with several attributes labelled: a set of three words that are included in the story; a sentence that is included in the story; a GPT-3.5-written summary of the story; and a list of 0-3 "story features", such as Twist, Dialogue or Bad Ending, which the story abides by.

We use the TinyStories-Instruct version of this dataset [1], wherein each story is prefixed with an "instruction" containing the story attributes described above, hence enabling the model to learn to conditionally generate stories based on an input or instruction. An example from this dataset is given in Fig. 5.4. For the fine-tuning datasets, we take the original dataset and alter it in several ways as described in the next section.

[1] https://huggingface.co/datasets/roneneldan/TinyStoriesInstruct

5.5.2 Model Training

Pretraining. We pretrain 91 million parameter autoregressive language models with a similar architecture to LLaMa 2 [Touvron et al., 2023c], with a custom tokeniser with vocabulary of size 8192 trained on the dataset.[2] They have hidden dimension 768, 12 layers, and 12 attention heads per layer. These models are trained with the standard language modelling cross-entropy loss, with batch size 128, sequence length 1024, no dropout, for 30,000 gradient steps, with a learning rate schedule with a linear warmup from 0 and cosine decay to 0, with maximum learning rate 0.001. These models achieve a loss of $\tilde{0}.8$ at the end of training, and can generate coherent multi-paragraph stories given a specific instruction in the form it saw during training.

Fine-tuning. We are interested in analysing whether fine-tuning these models can alter underlying capabilities. The specific capability we investigate is that of generating stories containing `Twists` (which is one of the story features), and are analysing whether various fine-tuning protocols can remove this capability from the pretrained model. We investigate a variety of fine-tuning protocols modelled after plausible realistic scenarios where one may want to fine-tune a model to not generate text of a certain type (e.g., highly toxic text), regardless of the input instruction. These include: **Filtering** fine-tunes the model on a dataset where all instances of stories with `Twists` are filtered out; **Filtering + Mix & Match** filters, and then replaces all instances of another, unrelated feature (in this case, `Foreshadowing`) in the instruction with the `Twist` feature; and **Filtering + Randomisation** filters, and then adds the "`Twist`" instruction to the prompt for stories that do not contain `Twists`, thus training the model to not model stories with `Twists` even if instructed. This last protocol acts as a kind of adversarial training (in that there are stories with the `Twist` instruction but no `Twists`), and the **Mix & Match** introduces a spurious correlation between the `Twist` instruction and the foreshadowing capability, as in the Tracr and PCFG results.

5.6 Experiments: Mechanistic analysis of Fine-tuning

I now provide several results indicating that fine-tuning rarely elicits meaningful changes to pretraining capabilities. To this end, I borrow several protocols commonly used in the

[2] Our code is based on this repository: https://github.com/karpathy/llama2.c

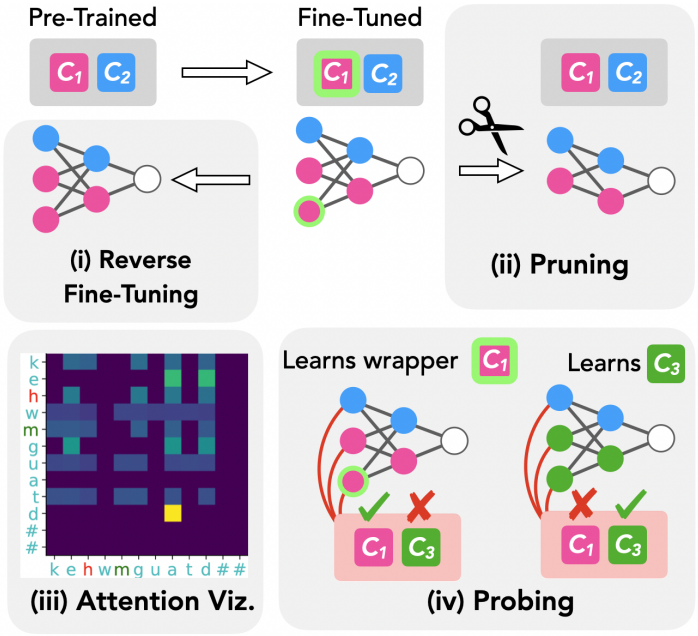


Figure 5.5: Analysis protocols. I analyse how fine-tuning affects a pretrained model's capabilities by (i) reverse Fine-tuning, (ii) network pruning, (iii) attention visualisation, and (iv) probing classifiers. I use (ii)—(iv) to show fine-tuning often yields wrapped capabilities. For further evidence, I use (i) and (ii) and find I can "revive" the original capabilities, i.e., the model starts performing well on the pretraining task again. See Appendix D.4 for precise details.

field of mechanistic interpretability for my analysis (see Fig. 5.5), specifically **network pruning** [Voita et al., 2019, Tanaka et al., 2019], **attention map visualisations** [Serrano and Smith, 2019, Wiegreffe and Pinter, 2019, Lai and Tan, 2019], and **probing classifiers** [Tenney et al., 2019, Voita and Titov, 2020, Geva et al., 2023, 2022]. I use multiple tools for all experiments since each tool, individually, is known to suffer from pitfalls [Meister et al., 2021, Bai et al., 2021, Jain and Wallace, 2019, Belinkov, 2022, Bolukbasi et al., 2021]. Demonstrating my claims consistently hold true across a diverse set of tools improves the conclusions' robustness to pitfalls of a specific tool. Additionally, I propose a methodology called **reverse fine-tuning** (`reFT`), wherein one takes a pretrained model, fine-tunes it on a downstream dataset, and then fine-tunes it again in the "reverse" direction, i.e., on a dataset sampled from the original pretraining distribution. We argue if the behaviour corresponding to a pretraining capability is retrieved in a few steps of `reFT` , fine-tuning did not meaningfully alter said capability (this claim can be formalised using results by Bengio et al. [2020b], Priol et al. [2021]).

We primarily focus on the learned capabilities setup of PCFG counter in the main paper,

relegating most results on compiled capabilities with Tracr to Appendix D.7 and other results with PCFGs to Appendix D.8—findings remain consistent across all settings. In the PCFG counter setup, the model is pretrained, amongst other tasks, to count occurrences of tokens from the set $\{\texttt{O}_{\texttt{PT}}\} = \{\texttt{a}, \texttt{b}, \texttt{c}\}$ in a given string; during fine-tuning, the model is trained to count occurrences of $\texttt{O}_{\texttt{FT}} = \texttt{b}$. Here, the spurious correlation is defined by enforcing count of b to be 1 more than that of a. The probability a datapoint sampled from the train or test fine-tuning dataset contains a spurious correlation is denoted $C_{\texttt{Tr}}$ and $C_{\texttt{Te}}$, respectively. Here, $C_{\texttt{Tr}} \in \{0.0, 0.5, 0.8, 1.0\}$ and $C_{\texttt{Te}} \in \{0.0, 1.0\}$. I use three sets of sampling probabilities of the task operands in the pretraining data: $\mathscr{P}_{\texttt{T}}^{L} = (0.999, 0.001, 0.000)$, $\mathscr{P}_{\texttt{T}}^{M} = (0.9, 0.1, 0.0)$, or $\mathscr{P}_{\texttt{T}}^{H} = (0.5, 0.3, 0.2)$. These priors indicate a low/medium/high probability of sampling $\texttt{O}_{\texttt{FT}}$. I use the following learning rates (LR) for fine-tuning, $\eta_M = \eta_{\texttt{PT}}/10^1$ and $\eta_S = \eta_{\texttt{PT}}/10^2$, where $\eta_{\texttt{PT}}$ is the *average* pretraining learning rate.

5.6.1 TinyStories Experimental Details

For the TinyStories experiments, I take the pretrained model described in Section 5.5, and fine-tune it with the different protocols. We then perform reFT on a dataset of stories which all have Twists in, to measure the extent to which each fine-tuning protocol deleted the capability of Twist generation. To ensure a good control, we compare the reFT models to a model pretrained on data with no Twist stories, which is then fine-tuned on Twist stories. The sample efficiency and final performance of this model serves as a comparison for the reFT ed models.

5.6.1.1 Evaluation Metrics

We evaluate whether the fine-tuning protocols have removed the capability to model and generate stories with Twists in multiple ways. Firstly, we look at the loss on stories with Twists. If fine-tuning deletes the Twist capability, we expect the loss on this dataset to increase.

GPT Evaluations. To evaluate the generative capabilities of these models, we generate stories from them given prompt instructions with the Twist story feature. We then evaluate whether these stories contain Twists. To do this evaluation, we use the OpenAI GPT fine-tuning API [3] to fine-tune a GPT-3.5 model to classify whether a given story has a Twist or not. To do this,

[3] https://platform.openai.com/docs/guides/fine-tuning

we use the TinyStories dataset and accompanying labels. This fine-tuned model achieves 92% accuracy on a held-out test set after fine-tuning. We generate stories with multiple different prompts from both the fine-tuned and reverse fine-tuned models throughout fine-tuning, and measure the proportion of stories which are classified as having a `Twist`, which we call the *generation score*.

Probing. As well as using `reFT` to measure whether the fine-tuning protocols have deleted the capability to generate `Twists`, we also use probing to evaluate whether fine-tuning removes information from internal representations. We train linear probes on the internal activations of the transformer models to predict which story features (e.g. `Twist`, `Bad Ending`, `Dialogue`) are present in the story. These probes take an average of the activations at the final 10 token positions of the story. Given that this is a multi-label classification problem we employ a separate binary classification probe to classify the presence of each story feature. We use the accuracy of these probes at different layers before and after fine-tuning, and on the control pretrained model which was trained on data with no `Twists`, to measure whether fine-tuning has removed information from the models' internal activations.

5.6.2 Tracr and PCFG Results

Behavioural assessment of fine-tuning. I first evaluate the model's learning dynamics during fine-tuning (see Fig. 5.6 and Table D.4). When the pretraining prior has low probability of sampling the token $\texttt{0}_{\texttt{FT}}$, we see the fine-tuned model performs well only when the spurious correlation is present, i.e., $C_{\texttt{Te}} = 1$. As the sampling probability is increased, however, we observe this behaviour significantly changes. In particular, even if the model is fine-tuned for a high value of $C_{\texttt{Tr}}$, albeit less than 1, it starts to perform well on the test data regardless of the presence of the spurious attribute. Note that the performance is not high to begin with, indicating the ability to count $\texttt{0}_{\texttt{FT}}$ was learned during fine-tuning; however, having a sufficiently large sampling probability for the token during pretraining leads the model to avoid the spurious correlation. This indicates a pretraining ability to extract information relevant for the downstream task is likely to be exploited during fine-tuning. This is further corroborated by the results in Fig. 5.7, where we observe that when the spurious correlation is present in the fine-tuning data, accuracy on the pretraining task is affected the most if sampling prior of the target token was low during pretraining. I next analyse these results

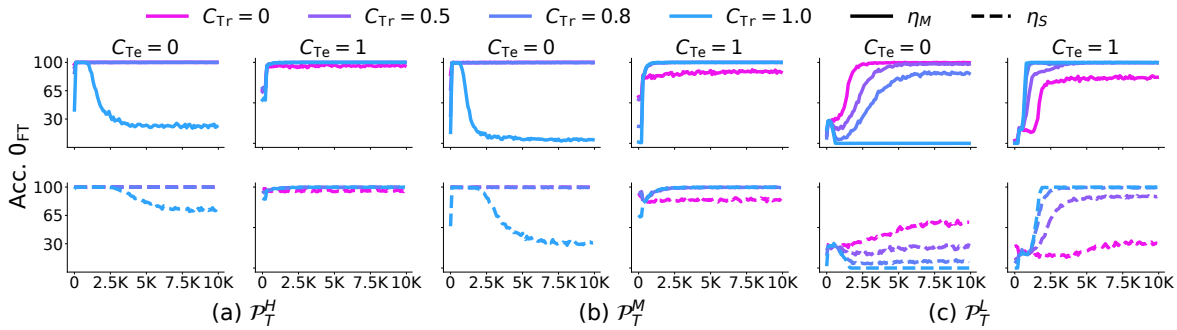


Figure 5.6: Fine-tuning accuracy w.r.t. number of training iterations. I vary the probability of sampling the token O_{FT} in the pretraining data and the spurious correlation in the fine-tuning datasets. When the prior is sufficiently high (a, b), I find the model learns to perform well on the downstream task. Meanwhile, if the prior is low (c), the model learns the downstream task only if a high enough learning rate is used and the spurious correlation is imperfect. This indicates the ability to extract information relevant for the downstream task is likely to be exploited during fine-tuning.

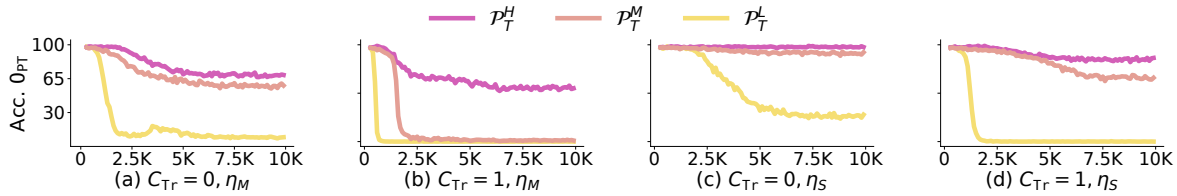


Figure 5.7: Impact of sampling prior on the pretraining task's accuracy as fine-tuning is performed. I plot accuracy on the pretraining task w.r.t. fine-tuning iterations. When the sampling prior of the O_{FT} is low during pretraining, the pretraining task accuracy quickly plummets, especially if the spurious correlation is high; having a high sampling prior mitigates this behaviour. This *indicates* pretraining capabilities are affected the most when they are weakly relevant.

mechanistically.

Pruning / Probing fine-tuned models indicates learning of wrapped capabilities. The results above *indicate* the model exploits its weakly relevant capabilities, i.e., the capability that helps exploit any spurious correlation, to solve the downstream task. I hypothesise that, at a mechanistic level, the model exploits the weakly relevant capability by learning a *wrapper* over it. To evaluate this, I analyse the models fine-tuned with a low sampling prior via network pruning and linear probing (see Appendix D.4 for setup details). Specifically, I prune the fine-tuned models to find the most salient weights for reducing loss on the pretraining task of counting O_{PT}. If the model learns a wrapper on this capability, the neurons I find should correspond to this wrapper, such that deleting them recovers the capability to count that token. As shown in Fig. 5.9, I find this is indeed the case—in a setup with weak relevance of capabilities, pruning a very small number of neurons is sufficient to revive the ability to

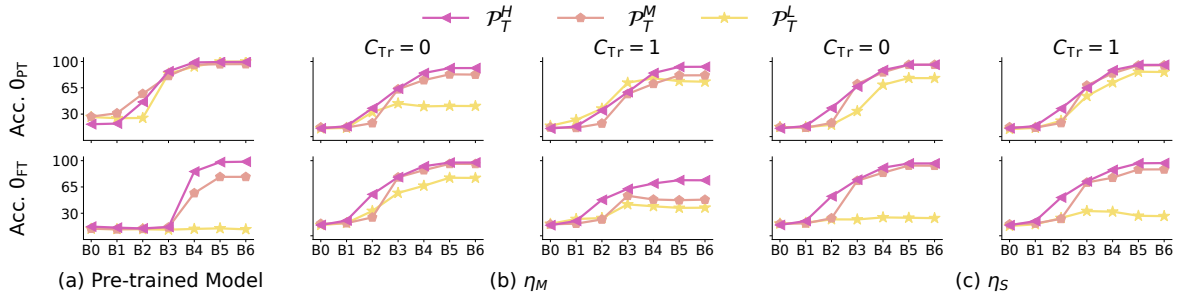


Figure 5.8: Probing the presence of pretraining (top) and fine-tuning (bottom) capabilities. I plot probe accuracy versus the index of the block in the Transformer model. C_{Te} is set to 0. The pretrained model (left) acts as a baseline for the trend of performance through the model's blocks. In most scenarios, I find we can infer the count of 0_{PT} with a similar trend as the pretrained model (left). A drop in performance is observed only when learning rate η_M is used with a weakly relevant capability (low sampling prior). This indicates pretraining capabilities persist after fine-tuning.

perform well on the original task of counting 0_{PT}. To assess this further, I train a linear probe on the residual output of every block of the transformer model and determine whether the count of 0_{PT} can be accurately computed via the fine-tuned model. As shown in Fig. 5.8, in the presence of spurious correlations, a linear probe can retrieve the count of the token 0_{PT}, indicating intermediate outputs relevant to the pretraining capability are still being produced by the fine-tuned model. This observation is particularly evident when a smaller learning rate is used, which is common in practice. Overall, these results show that when a weakly relevant capability is present in the pretrained model, a *wrapper*, i.e., a localised transformation of the pretraining capability, is learned during fine-tuning.

`reFT` **enables "revival" of pretraining capabilities.** To further corroborate the claims

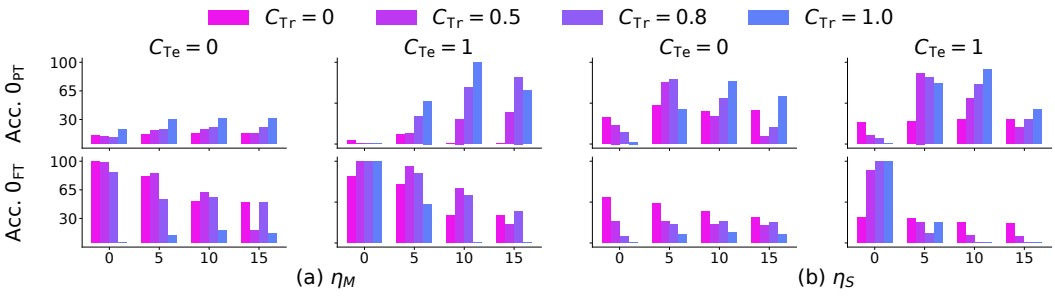


Figure 5.9: Pruning a few neurons is sufficient to retrieve pretraining task accuracy. I plot accuracy w.r.t. number of neurons pruned to improve performance on the pretraining task. I see when a small learning rate is used for fine-tuning, the pretraining task's performance improves after just 5–15 neurons are pruned (top), while the fine-tuning task's performance reduces correspondingly (bottom). I argue these neurons serve as a wrapper to minimally alter the weakly relevant pretraining capability and exploit the spurious correlation present in the fine-tuning data.

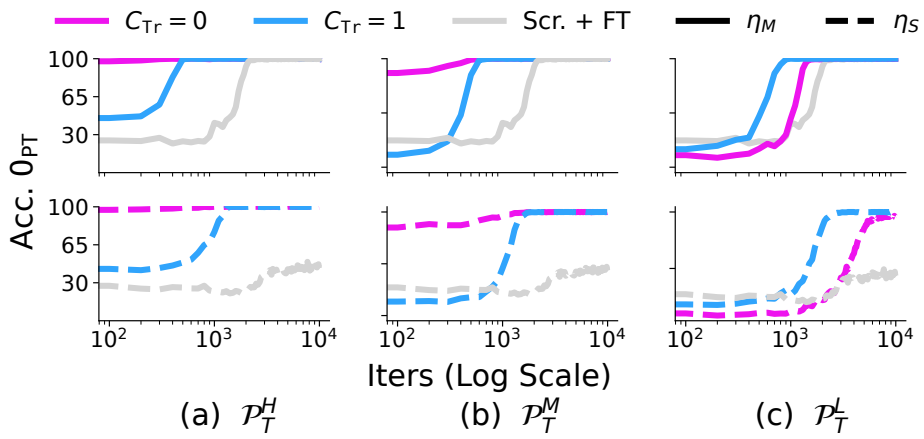


Figure 5.10: Reverse Fine-Tuning: I set $C_{\texttt{Te}}$ to be 0 to test if the model performs well regardless of a spurious correlation. Models are fine-tuned for 10K iterations. We observe that when a strongly relevant capability is present (a, b), the model very quickly (0.1–1K iterations) starts to perform well on the task via `reFT` , even if behaviour relevant to the capability ceased during pretraining (e.g., when $C_{\texttt{Tr}}$ is 1). Meanwhile, when the model possesses a weakly relevant capability (c), this "revival" is *slightly* slower (3K iterations). In contrast, the `Scr.+FT` baseline only reaches perfect accuracy at 4.5K iterations and when using a larger learning rate, i.e., η_M.

above, I use a model fine-tuned to count O_{FT} and reverse fine-tune it to re-learn the ability to count O_{PT}. As a baseline, I also report a protocol called `Scr.+FT`, wherein the model is initialised with parameters pretrained to count O_{FT} and then fine-tuned to count O_{PT}. Note that this baseline and the `reFT` protocol differ in their initialization state: former is initialised with parameters pretrained to count O_{FT}, while latter is initialised with parameters pretrained to count O_{PT} and fine-tuned to count O_{FT}. Results are shown in Fig. 5.10. We see the model starts to perform well on the pretraining task even if a small learning rate is used for `reFT` , i.e., even minimally changing the fine-tuned model's parameters is sufficient to regain the pretraining capability! Further, increasing the sampling prior of O_{FT} accelerates this behaviour. This indicates that when a strongly relevant capability is present, the model essentially amplifies its use, but does not catastrophically affect the pretraining capability itself; meanwhile, with a weakly relevant capability (low sampling prior during pretraining), even though the performance is initially poor on the pretraining task, in relatively few iterations (compared to baseline), the accuracy becomes perfect.

Attention map visualisations further corroborate the wrappers hypothesis. As I noted before, the results discussed above remain consistent across other experimental setups for both Tracr and PCFG models. However, by construction, Tracr yields particularly

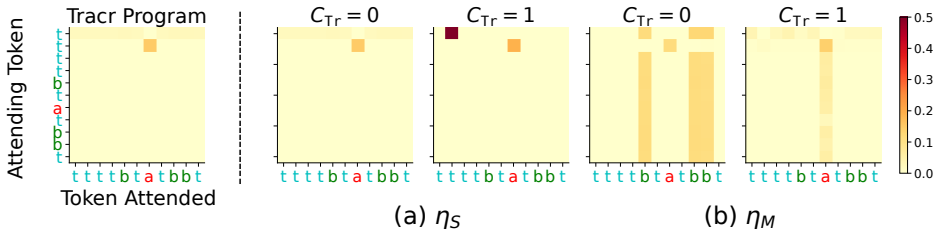


Figure 5.11: Visualising attention maps of fine-tuned Tracr models. Leftmost panel shows the Tracr compiled model's attention map on the counter task. Upon fine-tuning under different spurious correlations, we see the model continues to pay attention to the pretraining target $\texttt{O}_{\texttt{PT}} = a$. Only when a large enough learning rate and zero spurious correlation is used is there a change in the attention pattern.

Table 5.1: TinyStories `reFT` Analysis. We report the percent of generations with a `twist` during reverse fine-tuning for the twist capability. `F`, `R`, and `MM` stand for our three fine-tuning protocols: Filtering, Randomisation and Mix & Match (see Section 5.5 for details). Regardless of learning rate and protocol, models relearn to generate stories with `twist` more sample-efficiently than the control model pre-trained on data w/o twists and fine-tuned to generate them (Not in `PT`).

Deletion Type	Twist Proportion at Iteration			
Iteration	0	30	300	3000
F (η_M)	44%	81%	81%	82%
F + RR (η_M)	12%	56%	69%	75%
F + MM (η_M)	31%	88%	50%	75%
F (η_S)	69%	88%	75%	94%
F + RR (η_S)	12%	44%	81%	81%
F + MM (η_S)	50%	81%	62%	81%
Not in PT	12%	31%	44%	81%

interpretable attention maps, allowing us to directly visualise the effects of fine-tuning. We thus analyse the attention maps of a Tracr model on the Counter task described in Section 5.4. Results are shown in Fig. 5.11. The original Tracr compiled model serves as a baseline and clearly demonstrates that all tokens only attend the pretraining target token, $\texttt{O}_{\texttt{PT}} = a$. Upon fine-tuning to count $\texttt{O}_{\texttt{FT}} = b$, we find the model clearly continues to pay attention to $\texttt{O}_{\texttt{PT}}$ if a small learning rate is used. A larger learning rate is, however, able to alter the model computation, but only if the pretraining capability is not weakly relevant to the fine-tuning task, i.e., when $C_{\texttt{Tr}} = 0$; otherwise, we again find the model continues to pay attention to the pretraining target.

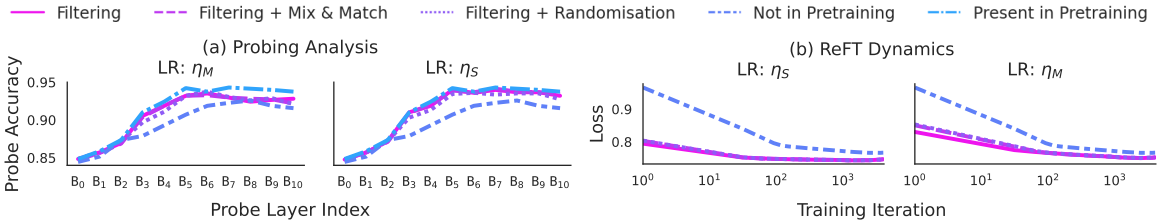


Figure 5.12: Validation on TinyStories. Models are trained to produce stories with several features (e.g., `foreshadowing`) and fine-tuned via different protocols to not produce stories with a "forbidden" feature (specifically, `twists`). **Left:** I probe the existence of this feature at intermediate Transformer blocks. Probe accuracy on models pretrained with or without `twist` data (*Present/Not in Pretraining*, respectively) act as upper and lower bounds on the expected accuracy, and are plotted for ease of comparison. Regardless of the fine-tuning protocol (Filtering, Filtering + Randomisation, Filtering + Mix & Match), for the lower LR, no protocol removes a meaningful amount of information and a similar but less strong trend holds for the higher LR. **Right:** I plot the loss during reverse fine-tuning (`reFT`) to again produce stories with the forbidden feature. Fine-tuned models' loss goes down very quickly (30–300 iterations) compared to baselines (which never reach the same loss; also see Table 5.1). Both these results indicate the capability of identifying the forbidden feature, a necessary capability for story modelling, continues to persist after fine-tuning.

5.6.3 TinyStories Results

I now describe the results of the TinyStories experiments described in Sections 5.5 and 5.6.1. The probing results are shown in Fig. 5.12, where the dynamics of loss during `reFT` on learning to generate stories with the deleted feature are also shown. The percentage of stories with the deleted feature generated by models during `reFT` (as predicted by a fine-tuned gpt3.5 model) is shown in Table 5.1.

Overall, I find that "deleted" capabilities can be easily and sample-efficiently recovered (compared to the baseline), i.e., stories with that feature can be generated again, regardless of the fine-tuning protocol used. To see this, note that the probe accuracy is higher for all the fine-tuning controls than it is for the "Not in Pretraining" control in Fig. 5.12 left, and the loss in this control setting is higher in the `reFT` results in Fig. 5.12, particularly with the lower learning rate. Table 5.1 shows a similar trend in the story generation capabilities. We report additional results from this setting in Appendix D.6. These results support my hypotheses that fine-tuning only minimally alters pretrained model capabilities.

5.7 Conclusion

In this work, I show that fine-tuning generally alters pretrained model via small, localised changes that only minimally transform their capabilities. I perform the analysis both with existing mechanistic interpretability tools and my proposed `reFT` method. The results pave the way for future work both understanding how fine-tuning works in more realistic settings with larger models, and emphasises the need for developing methods beyond fine-tuning that alter pretrained model capabilities more substantially, particularly deleting unsafe capabilities.

There is evidence that these results generalise to more realistic settings [Qi et al., 2023, Yang et al., 2023, Zhang and Wu, 2024, Zhao et al., 2023b]. If this is the case, it has significant implications for what fine-tuning does and how it generalises. In particular, it implies:

1. Fine-tuning is only as good as your pretrained model: if your pretrained model does not possess a desired capability, or another strong relevant and easily transformable capability, then fine-tuning will struggle to produce a model with that capability, especially in a way that generalises well

2. In the case of *removing* capabilities from models, fine-tuning is not a reliable method. This means that the misgeneralisation exhibited in examples of jailbreaking models is likely to remain an issue unless other methods for achieving safety and alignment are utilised.

3. On a positive note, this work and the results demonstrate both that insights from synthetic data setups do transfer to more realistic settings, and that gaining a mechanistic understanding of models can enable us to predict which much higher confidence those models' behaviour on novel OOD inputs.

Overall, the pretraining then fine-tuning paradigm is both a blessing and a curse: pretraining clearly provides exceptional capabilities and representations, which relatively small amounts of fine-tuning are able to elicit successfully. However, if pretraining does not produce the desired capability, or produces undesired capabilities, then fine-tuning will struggle to correct this deficiency. Given the current difficulty of controlling *what* is learned during pretraining, this emphasises the need to either (a) increase the controllability of what is learned during pretraining, or (b) develop methods for altering pretrained models' capabilities

more substantially than fine-tuning does, while still maintaining the benefits of pretraining to remain competitive with the current paradigm while enhancing safety and generalisation. It is possible these problems will continue to exist for future AI systems without significant research effort.

Chapter 6

Conclusions and Future Work

In this thesis, I have presented a series of works that investigate generalisation in AI systems, with a focus on reinforcement learning agents and fine-tuned large language models (LLMs). This research is motivated by the goal of improving our understanding of generalisation for future, highly capable, human-level systems, and hence ensuring that these systems are safe and aligned and therefore beneficial. The choice of problem settings is motivated by the belief that these settings share key properties with future AI systems, and hence progress in these settings is likely to be useful for the development of future AI systems.

In Chapter 2, I surveyed the field of zero-shot generalisation (ZSG) in RL. I categorised existing work in both benchmarks and methods in the field, and in doing so presented many avenues for future research progress for improving ZSG. Particularly motivated by the lack of environments that had the correct combination of attributes (controllable and PCG factors of variation which were non-visual in nature and allowed for scientific experimentation and testing out-of-distribution generalisation), in Chapter 3 I presented MiniHack, a ZSG benchmark suite and environment creation tool designed to enable further progress improving generalisation in RL. Throughout this work, I found that current RL methods often struggle to generalise well, even with a wide variety of methods and approaches that have been proposed. A fundamental limitation of much of the work in this space is that policies are trained from scratch, and do not utilise existing knowledge, data and representations to improve performance and generalisation. This makes the problem setting of ZSG less analogous to aligning superhuman AI systems. Since the work these chapters are based on was published, most significant advancements in RL generalisation have come from scaling up compute and environment complexity, rather than fundamental algorithmic advances [Team et al., 2021,

Bauer et al., 2023]. Here, I suspect the bitter lesson applies: existing algorithms that can make use of additional compute will outperform specialised algorithmic advances which are too tailored to specific environemnts. While I believe beneficial research can still be done in this area, this limitation led me to consider other settings which are more analogous to inner alignment for superhuman AI systems and where alignment is a bottleneck to performance: fine-tuning LLMs.

In Chapter 4, I investigate the generalisation and output diversity properties of different fine-tuning protocols designed to align pretrained LLMs with human preferences. I find that methods that make explicit use of human preferences (e.g. RLHF) produce systems that generalise better than those that do not, although RL-based methods reduce the output diversity of these systems. This directs future research on methods for ensuring robust generalisation of fine-tuned LLMs. The results in this chapter are purely behavioural: they require access to OOD data on which to test the policy, and hence cannot give us high confidence that the system will generalise well on unseen inputs we have not considered. To complement this behavioural analysis, in Chapter 5 I focus on gaining a mechanistic understanding of how fine-tuning changes pretrained models. Here, I find that fine-tuning generally only produces minimal changes to the pretrained model's internal computation, struggling to either teach the model fundamentally new capabilities or to delete undesirable capabilities. This has implications for the kind of models you should expect to get from fine-tuning, and the generalisation properties you should expect from them: these models will struggle to go beyond what your pretrained model was capable of, and will generalise in the same way as the pretrained model.

Throughout, I emphasise the need to evaluate training methods on out-of-distribution (OOD) data or tasks to get a better sense of how well they perform in more realistic settings. This is particularly important in RL, as previous approaches often evaluate training methods on an identical environment to that in which the policy is trained. However, it is also crucial in LLM fine-tuning, as these systems are and will continue to be deployed in many diverse settings that are likely to be very different from their fine-tuning data distribution, making it important to understand how well they generalise. Here, I present several recommendations for producing a more robust and useful evaluation of AI systems:

1. The first step is to evaluate on held-out inputs or tasks from the same distribution as training, rather than evaluate on an identical setting. This is the recommendation to RL researchers presented in Chapter 2.

2. Moving beyond that, evaluate on OOD data or tasks, rather than just evaluating in-distribution. Ensure the set of OOD data exhibits diverse shifts from the training distribution, with the aim being to gain confidence that on unforeseen data the model will continue to generalise well. Minihack (Chapter 3) provides a tool for easily producing new training and evaluation distributions and so enables this kind of evaluation. The protocol for developing new evaluation inputs based in SelfInstruct [Wang et al., 2023] demonstrated in Chapter 4 is another example of how to rapidly prototype and develop OOD datasets that then enable us to find failure modes of models and methods.

3. When performing this evaluation, do not only report mean performance on the OOD data, but also report generalisation gap (as discussed in Chapter 2 and used in Chapter 4) and ideally other metrics that map onto notions of generalisation that are desirable.

4. Finally, it is important to go beyond a purely behavioural evaluation. Given the extremely large input-space of current AI systems, searching over this input space, even adversarially [Perez et al., 2022, Mazeika et al., 2024], for potential failures of generalisation is likely to be exorbitantly expensive. Instead, gaining a mechanistic understanding, as proposed in Chapter 5, can give us a better idea of how our model will generalise and hence higher confidence that it will generalise correctly.

Future Work

Overall, there is still a long way to go before we fully understand what methods produce the best generalisation properties, and how to evaluate generalisation in a way that gives us a sufficient level of confidence that models will behave well in the real world. This thesis has raised many questions in this space, and to conclude I here discuss several promising future research directions, as well as presenting a longer-term vision for the path towards safe and aligned superhuman AI systems.

Firstly, there are many ways to improve the generalisation of RL agents, particularly on difficult benchmarks like MiniHack. As mentioned in Chapter 3, I believe training RL

agents from scratch is unlikely to produce the necessary level of generalisation, and so incorporating some kind of pretraining into the RL training process is likely to be beneficial. This could involve using pretrained LLMs as a source of knowledge and data for RL agents. MiniHack could serve as a potential test-bed here, as pretrained LLMs likely possess a reasonable amount of knowledge about NetHack, and so working out how to utilise that to improve the generalisation of RL agents is an important problem. Utilising MiniHack (and other environments) to more thoroughly benchmark OOD generalisation and particularly goal misgeneralisation of RL algorithms [di Langosco et al., 2022, Shah et al., 2022], with or without pretraining, is also a promising direction for future work. The issue of goal misgeneralisation is likely to become more prevalent as systems become more generally capable, and currently there are no solid benchmarks for evaluating this failure mode in a rigorous way, or methods that have been demonstrated to work correctly across a range of scenarios. An important consideration in performing this research is to focus on failures of generalisation that are not expected to be solved by large scale pretraining, as this is likely to be a component of future RL systems.

Secondly, there is a lot of work to be done in understanding at a behavioural level the generalisation properties of different LLM fine-tuning protocols across tasks and pretrained models. In the first instance, evaluating more methods (in particular DPO [Rafailov et al., 2023] and other direct alignment algorithms) and utilising more tasks and more OOD generalisation datasets per task would be beneficial. This would enable us to build up a more complete empirical picture of which methods are generalising best, and point towards phenomena which require further investigate and understanding. In general, the norm when evaluating methods for fine-tuning LLMs should be to evaluate on a range of OOD tasks, rather than in-distribution data, and to report a range of generalisation metrics from these tasks. When trying to push forwards our understanding of generalisation, synthetic data-generation techniques like those I used in Chapter 4 [Wang et al., 2023] are an easy way to rapidly prototype and develop OOD datasets that then enable us to find failure modes of models and methods.

Finally, the study of the mechanistic effects of fine-tuning is very nascent, and hence there is much work to do in this space. A key direction is to generalise our findings to more realistic settings; the work of Lee et al. [2024] is a great example of this. However, there is

still much more to understand about how models work and how fine-tuning changes them, and continued work in this direction – and especially work that unifies the mechanistic and behavioural understanding – can enable us to develop better techniques for ensuring models are safe and aligned in a robust and generalisable way.

At a higher level, the problem settings focused on in this work – reinforcement learning, and language model fine-tuning – are both motivated by their potential similarity to the problem of robustly aligning superhuman AI systems. However, these problems likely have key disanalogies with the alignment problem we will face for superhuman systems (superalignment), and so there is a need for research that produces problem settings which we expect to be more analogous to superalignment. Examples of this include the weak-to-strong generalisation analogy presented by Burns et al. [2023], as well as the sandwiching protocol proposed by Cotra [2021] and initially tested by Bowman et al. [2022]. However, both of these have disanalogies which might render progress in those settings less useful for superalignment, and so it is crucial to continue to develop analogous settings that enable us to perform empirical forward-looking research on superalignment.

How might we build safe and aligned superhuman AI?

What might the conclusions of this thesis imply for the project of building safe and alignment superhuman AI systems? The first two chapters argue that tabula-rasa RL training is unlikely to produce highly capable AI systems, regardless of whether they are safe and aligned, and hence large scale pretraining is likely a necessary ingredient. While Chapter 4 demonstrates some ability of fine-tuned models to generalise to OOD inputs, this generalisation is still not perfect, and Chapter 5 argues that the adjustments made by fine-tuning are potentially even less general than this behaviour implies. Taken together, these two points lead to the possibility of an unfortunate conclusion: pretraining followed by fine-tuning is a more likely recipe for superhuman AI systems, but it offers less control over the kinds of models produced that training RL systems from scratch, due to the lack of control over pretraining. If we require safety and alignment to be embedded deeply and comprehensively in the systems we train (which seems like a necessary condition for truly safe and alignment superhuman AI), we need to square this circle. I can see two ways of this happening, each taking one of the two analogies of this thesis as its basis.

In the first path, we push on making tabula-rasa RL work. Here, we control the entire of the training process of the model, with offers the potential to more deeply determine the kinds of AI system you create. One potential path for this is still to utilise large scale pretraining to produce environments for the RL agent to train in, rather than serve as the foundation of the RL agent itself [Bruce et al., 2024]. Combining this with advances in open-ended learning [Hughes et al., 2024] may produce generally capable and fundamentally aligned AI systems. Key to this vision succeeding is both open-ended RL training being able to be scaled all the way to produce generally capable systems (which is potentially an empirical question), as well as advances in how to control the kinds of AI systems that open-ended process produce to be safe and aligned. This latter direction is particularly challenging given the fundamental creativity at the heart of truly open-ended processes.

In the second path, we aim to more tightly control the kinds of AI systems current pretraining schemes produce, perhaps by aligning AI systems during pretraining [Korbak et al., 2023]; or we produce fine-tuning algorithms that more fundamentally edit models, such that their safety and alignment is deeply embedded. Both of these directions face difficulties: In the first, the high cost of pretraining models precludes the kind of rapid experimentation and iteration that is behind most machine learning successes, and the complexity of the systems makes it challenges to produce advances based on theoretical work. In the second, it is unclear to what extent it is possible to have both the benefits of large scale pretraining while also tightly controlling the properties of the model. To butcher the common metaphor for this paradigm of AI [LeCun, 2016], you may not be able to have your cake and eat it. Overall, I am more optimistic about the second path, but see many difficulties in both of these directions which future research will have to tackle.

Bibliography

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Rémi Munos, Nicolas Heess, and Martin A. Riedmiller. Maximum a posteriori policy optimisation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=S1ANxQW0b.

Mohammed Amin Abdullah, Hang Ren, Haitham Bou Ammar, Vladimir Milenkovic, Rui Luo, Mingtian Zhang, and Jun Wang. Wasserstein Robust Reinforcement Learning. *arXiv:1907.13196 [cs, stat]*, 2019. URL http://arxiv.org/abs/1907.13196.

Suzan Ece Ada, Emre Ugur, and H. Levent Akin. Generalization in Transfer Learning. *arXiv:1909.01331 [cs, stat]*, 2021. URL http://arxiv.org/abs/1909.01331.

Rishabh Agarwal, Marlos C. Machado, Pablo Samuel Castro, and Marc G. Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=qda7-sVg84.

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7319–7328, Online, 2021. Association for Computational Linguistics. doi:10.18653/v1/2021.acl-long.568. URL https://aclanthology.org/2021.acl-long.568.

Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Manuel Wuthrich,

Yoshua Bengio, Bernhard Schölkopf, and Stefan Bauer. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=SK7A5pdrgov.

Stefano V. Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018. ISSN 0004-3702. doi:10.1016/j.artint.2018.01.002. URL https://www.sciencedirect.com/science/article/pii/S0004370218300249.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.1, knowledge storage and extraction. *arXiv preprint arXiv:2309.14316*, 2023a.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.2, knowledge manipulation. *arXiv preprint arXiv:2309.14402*, 2023b.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 1, context-free grammar. *arXiv preprint arXiv:2305.13673*, 2023c.

Susan Amin, Maziar Gomrokchi, Harsh Satija, Herke van Hoof, and Doina Precup. A Survey of Exploration Methods in Reinforcement Learning. *arXiv:2109.00157 [cs]*, 2021. URL http://arxiv.org/abs/2109.00157.

Ankesh Anand, Jacob C. Walker, Yazhe Li, Eszter Vértes, Julian Schrittwieser, Sherjil Ozair, Theophane Weber, and Jessica B. Hamrick. Procedural generalization by planning with self-supervised world models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=FmBegXJToY.

Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA,*

USA, pages 5048–5058, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/453fadbd8a1a3af50a9df4df899537b5-Abstract.html.

Marcin Andrychowicz, Anton Raichuk, Piotr Stanczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters for on-policy deep actor-critic methods? A large-scale study. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=nIAxjsniDzg.

Anthropic. Introducing Claude, 2023. URL https://www.anthropic.com/index/introducing-claude.

Usman Anwar, Abulhair Saparov, Javier Rando, Daniel Paleka, Miles Turpin, Peter Hase, Ekdeep Singh Lubana, Erik Jenner, Stephen Casper, Oliver Sourbut, Benjamin L. Edelman, Zhaowei Zhang, Mario Günther, Anton Korinek, Jose Hernandez-Orallo, Lewis Hammond, Eric Bigelow, Alexander Pan, Lauro Langosco, Tomasz Korbak, Heidi Zhang, Ruiqi Zhong, Seán Ó hÉigeartaigh, Gabriel Recchia, Giulio Corsi, Alan Chan, Markus Anderljung, Lilian Edwards, Yoshua Bengio, Danqi Chen, Samuel Albanie, Tegan Maharaj, Jakob Foerster, Florian Tramer, He He, Atoosa Kasirzadeh, Yejin Choi, and David Krueger. Foundational Challenges in Assuring Alignment and Safety of Large Language Models, 2024. URL http://arxiv.org/abs/2404.09932.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant Risk Minimization. *arXiv:1907.02893 [cs, stat]*, 2020. URL http://arxiv.org/abs/1907.02893.

Saurabh Arora and Prashant Doshi. A Survey of Inverse Reinforcement Learning: Challenges, Methods and Progress. *arXiv:1806.06877 [cs, stat]*, 2020. URL http://arxiv.org/abs/1806.06877.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.

Andrea Asperti, Carlo De Pieri, and Gianmaria Pedrini. Rogueinabox: An environment for roguelike learning. *International Journal of Computers*, 2, 2017.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 507–517. PMLR, 2020. URL http://proceedings.mlr.press/v119/badia20a.html.

Bing Bai, Jian Liang, Guanhua Zhang, Hao Li, Kun Bai, and Fei Wang. Why attentions may not be interpretable? In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 25–34, 2021.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *arXiv:2204.05862 [cs]*, 2022a. URL http://arxiv.org/abs/2204.05862.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022b.

Philip J. Ball, Cong Lu, Jack Parker-Holder, and Stephen J. Roberts. Augmented world models facilitate zero-shot dynamics generalization from a single offline environment. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 619–629. PMLR, 2021. URL http://proceedings.mlr.press/v139/ball21a.html.

Chris Bamford, Shengyi Huang, and Simon M. Lucas. Griddly: A platform for AI research in games. *CoRR*, abs/2011.06363, 2020. URL https://arxiv.org/abs/2011.06363.

Victor Bapst, Alvaro Sanchez-Gonzalez, Carl Doersch, Kimberly L. Stachenfeld, Pushmeet Kohli, Peter W. Battaglia, and Jessica B. Hamrick. Structured agents for physical construction. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 464–474. PMLR, 2019. URL http://proceedings.mlr.press/v97/bapst19a.html.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261 [cs, stat]*, 2018. URL http://arxiv.org/abs/1806.01261.

Jakob Bauer, Kate Baumli, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, Vibhavari Dasagi, Lucy Gonzalez, Karol Gregor, Edward Hughes, Sheleem Kashem, Maria Loks-Thompson, Hannah Openshaw, Jack Parker-Holder, Shreya Pathak, Nicolas Perez-Nieves, Nemanja Rakicevic, Tim Rocktäschel, Yannick Schroecker, Satinder Singh, Jakub Sygnowski, Karl Tuyls, Sarah York, Alexander Zacherl, and Lei M. Zhang. Human-Timescale Adaptation in an Open-Ended Task Space. In *Proceedings of the 40th International Conference on Machine Learning*, pages 1887–1935. PMLR, 2023. URL https://proceedings.mlr.press/v202/bauer23a.html.

Charles Beattie, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig Petersen. DeepMind lab. *CoRR*, abs/1612.03801, 2016. URL http://arxiv.org/abs/1612.03801.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022. doi:10.1162/coli_a_00422. URL https://aclanthology.org/2022.cl-1.7.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013. ISSN 1076-9757. doi:10.1613/jair.3912. URL https://www.jair.org/index.php/jair/article/view/10819.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1471–1479, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/afda332245e2af431fb7b672a68b659d-Abstract.html.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland, 2022. Association for Computational Linguistics. doi:10.18653/v1/2022.acl-short.1. URL https://aclanthology.org/2022.acl-short.1.

Emmanuel Bengio, Joelle Pineau, and Doina Precup. Interference and generalization in temporal difference learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 767–777. PMLR, 2020a. URL http://proceedings.mlr.press/v119/bengio20a.html.

Yoshua Bengio. AI and Catastrophic Risk. *Journal of Democracy*, 34(4):111–121, 2023. ISSN 1086-3214. URL https://muse.jhu.edu/pub/1/article/907692.

Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Nan Rosemary Ke, Sébastien Lachapelle,

Olexa Bilaniuk, Anirudh Goyal, and Christopher J. Pal. A meta-transfer objective for learning to disentangle causal mechanisms. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020b. URL https://openreview.net/forum?id=ryxWIgBFPS.

Carolin Benjamins, Theresa Eimer, Frederik Schubert, André Biedenkapp, Bodo Rosenhahn, Frank Hutter, and Marius Lindauer. CARL: A Benchmark for Contextual and Adaptive Reinforcement Learning. *Workshop on Ecological Theory of Reinforcement Learning, NeurIPS 2021*, 2021. URL https://openreview.net/forum?id=6D45bYP5MRP.

Martín Bertrán, Natalia Martínez, Mariano Phielipp, and Guillermo Sapiro. Instance-based generalization in reinforcement learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/82674fc29bc0d9895cee346548c2cb5c-Abstract.html.

Andr&#233 Biedenkapp, H. Furkan Bozkurt, Theresa Eimer, Frank Hutter, and Marius Lindauer. Dynamic Algorithm Configuration: Foundation of a New Meta-Algorithmic Framework. *ECAI 2020*, pages 427–434, 2020. doi:10.3233/FAIA200122. URL https://ebooks.iospress.nl/doi/10.3233/FAIA200122.

Daniil A. Boiko, Robert MacKnight, and Gabe Gomes. Emergent autonomous scientific research capabilities of large language models, 2023. URL http://arxiv.org/abs/2304.05332.

Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda Viégas, and Martin Wattenberg. An interpretability illusion for bert. *arXiv preprint arXiv:2104.07143*, 2021.

Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1):49–107, 2000. ISSN 0004-

3702. doi:10.1016/S0004-3702(00)00033-3. URL https://www.sciencedirect.com/science/article/pii/S0004370200000333.

Samuel R. Bowman, Jeeyoon Hyun, Ethan Perez, Edwin Chen, Craig Pettit, Scott Heiner, Kamilė Lukošiūtė, Amanda Askell, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Christopher Olah, Daniela Amodei, Dario Amodei, Dawn Drain, Dustin Li, Eli Tran-Johnson, Jackson Kernion, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Liane Lovitt, Nelson Elhage, Nicholas Schiefer, Nicholas Joseph, Noemí Mercado, Nova DasSarma, Robin Larson, Sam McCandlish, Sandipan Kundu, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Timothy Telleen-Lawton, Tom Brown, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Ben Mann, and Jared Kaplan. Measuring Progress on Scalable Oversight for Large Language Models, 2022. URL http://arxiv.org/abs/2211.03540.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI gym. *CoRR*, abs/1606.01540, 2016a.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv:1606.01540 [cs]*, 2016b. URL http://arxiv.org/abs/1606.01540.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020b. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative Interactive Environments, 2024. URL http://arxiv.org/abs/2402.15391.

Yuri Burda, Harrison Edwards, Deepak Pathak, Amos J. Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019a. URL https://openreview.net/forum?id=rJNwDjAqYX.

Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019b. URL https://openreview.net/forum?id=H1lJJnR5Ym.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeff Wu. Weak-to-Strong Generalization: Eliciting Strong Capabilities With Weak Supervision, 2023. URL http://arxiv.org/abs/2312.09390.

Jonathan Campbell and Clark Verbrugge. Learning combat in NetHack. In *AIIDE*, 2017.

Jonathan Campbell and Clark Verbrugge. Exploration in NetHack with secret discovery. *IEEE Transactions on Games*, 2018.

Joseph Carlsmith. Is Power-Seeking AI an Existential Risk?, 2024. URL http://arxiv.org/abs/2206.13353.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 9630–9640. IEEE, 2021. doi:10.1109/ICCV48922.2021.00951. URL https://doi.org/10.1109/ICCV48922.2021.00951.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, Tony Wang, Samuel Marks, Charbel-Raphaël Segerie, Micah Carroll, Andi Peng, Phillip Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J. Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Bıyık, Anca Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback, 2023. URL http://arxiv.org/abs/2307.15217.

Louis Castricato, Alexander Havrilla, Shahbuland Matiana, Michael Pieler, Anbang Ye, Ian Yang, Spencer Frazier, and Mark Riedl. Robust Preference Learning for Storytelling via Contrastive Reinforcement Learning, 2022. URL http://arxiv.org/abs/2210.07792.

Alan Chan, Rebecca Salganik, Alva Markelius, Chris Pang, Nitarshan Rajkumar, Dmitrii Krasheninnikov, Lauro Langosco, Zhonghao He, Yawen Duan, Micah Carroll, Michelle Lin, Alex Mayhew, Katherine Collins, Maryam Molamohammadi, John Burden, Wanru Zhao, Shalaleh Rismani, Konstantinos Voudouris, Umang Bhatt, Adrian Weller, David Krueger, and Tegan Maharaj. Harms from Increasingly Agentic Algorithmic Systems, 2023. URL http://arxiv.org/abs/2302.10329.

Stephanie Chan, Adam Santoro, Andrew Lampinen, Jane Wang, Aaditya Singh, Pierre Richemond, James McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. *Advances in Neural Information Processing Systems*, 35:18878–18891, 2022.

Annie S. Chen, Suraj Nair, and Chelsea Finn. Learning Generalizable Robotic Reward Functions from "In-The-Wild" Human Videos. *arXiv:2103.16817 [cs]*, 2021a. URL http://arxiv.org/abs/2103.16817.

Jerry Zikun Chen. Reinforcement Learning Generalization with Surprise Minimization. *arXiv:2004.12399 [cs]*, 2020. URL http://arxiv.org/abs/2004.12399.

Shiyu Chen and Yanjie Li. An Overview of Robust Reinforcement Learning. In *2020 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, pages 1–6, 2020. doi:10.1109/ICNSC48988.2020.9238129.

Valerie Chen, Abhinav Gupta, and Kenneth Marino. Ask your humans: Using human instructions to improve generalization in reinforcement learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021b. URL https://openreview.net/forum?id=Y87Ri-GNHYu.

Maxime Chevalier-Boisvert. Minimalistic Gridworld Environment (MiniGrid), 2021. URL https://github.com/maximecb/gym-minigrid.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for OpenAI gym, 2018. URL https://github.com/maximecb/gym-minigrid.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJeXCo0cYX.

Noam Chomsky. Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124, 1956.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling Language Modeling with Pathways, 2022. URL http://arxiv.org/abs/2204.02311.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4299–4307, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling Instruction-Finetuned Language Models, 2022. URL http://arxiv.org/abs/2210.11416.

Karl Cobbe, Oleg Klimov, Christopher Hesse, Taehoon Kim, and John Schulman. Quantifying

generalization in reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1282–1289. PMLR, 2019. URL http://proceedings.mlr.press/v97/cobbe19a.html.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 2048–2056. PMLR, 2020a. URL http://proceedings.mlr.press/v119/cobbe20a.html.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 2048–2056. PMLR, 2020b. URL http://proceedings.mlr.press/v119/cobbe20a.html.

Karl Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 2020–2027. PMLR, 2021a. URL http://proceedings.mlr.press/v139/cobbe21a.html.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems, 2021b. URL http://arxiv.org/abs/2110.14168.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. TextWorld: A Learning Environment for Text-based Games. *arXiv:1806.11532 [cs, stat]*, 2019. URL http://arxiv.org/abs/1806.11532.

Ajeya Cotra. The case for aligning narrowly superhuman models, 2021. URL https://www.alignmentforum.org/posts/PZtsoaoSLpKjjbMqM/the-case-for-aligning-narrowly-superhuman-models.

Matthew Crosby, Benjamin Beyret, Murray Shanahan, José Hernández-Orallo, Lucy Cheke, and Marta Halina. The Animal-AI Testbed and Competition. In *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*, pages 164–176. PMLR, 2020. URL https://proceedings.mlr.press/v123/crosby20a.html.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, et al. Neural networks and the chomsky hierarchy. *arXiv preprint arXiv:2207.02098*, 2022.

Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. Jailbreaker: Automated jailbreak across multiple large language model chatbots. *arXiv preprint arXiv:2307.08715*, 2023.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre M. Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a. URL https://proceedings.neurips.cc/paper/2020/hash/985e9a46e10005356bbaf194249f6856-Abstract.html.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre M. Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020b. URL https://proceedings.neurips.cc/paper/2020/hash/985e9a46e10005356bbaf194249f6856-Abstract.html.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre M. Bayen, Stuart Russell, An-

drew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020c. URL https://proceedings.neurips.cc/paper/2020/hash/985e9a46e10005356bbaf194249f6856-Abstract.html.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi:10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Terrance DeVries and Graham W. Taylor. Improved Regularization of Convolutional Neural Networks with Cutout. *arXiv:1708.04552 [cs]*, 2017. URL http://arxiv.org/abs/1708.04552.

Lauro Langosco di Langosco, Jack Koch, Lee D. Sharkey, Jacob Pfau, and David Krueger. Goal misgeneralization in deep reinforcement learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 12004–12019. PMLR, 2022. URL https://proceedings.mlr.press/v162/langosco22a.html.

Carlos Diuk, Andre Cohen, and Michael L. Littman. An object-oriented representation for efficient reinforcement learning. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 240–247. ACM, 2008. doi:10.1145/1390156.1390187. URL https://doi.org/10.1145/1390156.1390187.

Ron Dorfman, Idan Shenfeld, and Aviv Tamar. Offline Meta Learning of Exploration. *arXiv:2008.02598 [cs, stat]*, 2021. URL http://arxiv.org/abs/2008.02598.

Finale Doshi-Velez and George Dimitri Konidaris. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1432–1440. IJCAI/AAAI Press, 2016. URL http://www.ijcai.org/Abstract/16/206.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. *arXiv:1711.03938 [cs]*, 2017. URL http://arxiv.org/abs/1711.03938.

Simon S. Du, Akshay Krishnamurthy, Nan Jiang, Alekh Agarwal, Miroslav Dudík, and John Langford. Provably efficient RL with rich observations via latent state decoding. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1665–1674. PMLR, 2019. URL http://proceedings.mlr.press/v97/du19b.html.

Simon S. Du, Sham M. Kakade, Ruosong Wang, and Lin F. Yang. Is a good representation sufficient for sample efficient reinforcement learning? In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=r1genAVKPB.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl $^ 2$: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. AlpacaFarm: A Simulation Framework for Methods that Learn from Human Feedback, 2023. URL http://arxiv.org/abs/2305.14387.

Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. An empirical investigation of the challenges of real-world

reinforcement learning. *arXiv:2003.11881 [cs]*, 2021. URL http://arxiv.org/abs/2003.11881.

E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 0007/2012-10-12. ISBN 2153-0866. doi:10.1109/IROS.2012.6386109.

Marc Ebner, John Levine, Simon M. Lucas, Tom Schaul, Tommy Thompson, and Julian Togelius. Towards a Video Game Description Language. In *Artificial and Computational Intelligence in Games*, volume 6 of *Dagstuhl Follow-Ups*, pages 16 pages, 493428 bytes. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013. ISBN 978-3-939897-62-0. doi:10.4230/DFU.VOL6.12191.85. URL https://drops.dagstuhl.de/entities/document/10.4230/DFU.Vol6.12191.85.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Go-explore: A new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

Theresa Eimer, André Biedenkapp, Maximilian Reimer, Steven Adriaensen, Frank Hutter, and Marius Lindauer. DACBench: A Benchmark Library for Dynamic Algorithm Configuration. *arXiv:2105.08541 [cs]*, 2021. URL http://arxiv.org/abs/2105.08541.

Ronen Eldan and Yuanzhi Li. TinyStories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1406–1415. PMLR, 2018. URL http://proceedings.mlr.press/v80/espeholt18a.html.

Ben Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Robust predictable control. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy

Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 27813–27825, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/e9f85782949743dcc42079e629332b5f-Abstract.html.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

Jiameng Fan and Wenchao Li. DRIBO: robust deep reinforcement learning via multi-view information bottleneck. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 6074–6102. PMLR, 2022. URL https://proceedings.mlr.press/v162/fan22b.html.

Linxi Fan, Guanzhi Wang, De-An Huang, Zhiding Yu, Li Fei-Fei, Yuke Zhu, and Animashree Anandkumar. SECANT: self-expert cloning for zero-shot generalization of visual policies. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 3088–3099. PMLR, 2021. URL http://proceedings.mlr.press/v139/fan21c.html.

Jesse Farebrother, Marlos C. Machado, and Michael Bowling. Generalization and Regularization in DQN. *arXiv:1810.00123 [cs, stat]*, 2020. URL http://arxiv.org/abs/1810.00123.

Angelos Filos, Panagiotis Tigkas, Rowan McAllister, Nicholas Rhinehart, Sergey Levine, and Yarin Gal. Can autonomous vehicles identify, recover from, and adapt to distribution shifts? In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning*

Research, pages 3145–3153. PMLR, 2020. URL http://proceedings.mlr.press/v119/filos20a.html.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 2017. URL http://proceedings.mlr.press/v70/finn17a.html.

Yannis Flet-Berliac, Johan Ferret, Olivier Pietquin, Philippe Preux, and Matthieu Geist. Adversarially guided actor-critic. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=_mQp5cr_iNy.

Meire Fortunato, Melissa Tan, Ryan Faulkner, Steven Hansen, Adrià Puigdomènech Badia, Gavin Buttimore, Charlie Deck, Joel Z. Leibo, and Charles Blundell. Generalization of reinforcement learners with working and episodic memory. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12448–12457, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/02ed812220b0705fabb868ddbf17ea20-Abstract.html.

Vincent François-Lavet, Yoshua Bengio, Doina Precup, and Joelle Pineau. Combined reinforcement learning via abstract representations. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3582–3589. AAAI Press, 2019. doi:10.1609/aaai.v33i01.33013582. URL https://doi.org/10.1609/aaai.v33i01.33013582.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets

for Deep Data-Driven Reinforcement Learning. *arXiv:2004.07219 [cs, stat]*, 2021. URL http://arxiv.org/abs/2004.07219.

Shani Gamrian and Yoav Goldberg. Transfer learning for related reinforcement learning tasks via image-to-image translation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2063–2072. PMLR, 2019. URL http://proceedings.mlr.press/v97/gamrian19a.html.

Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, Andy Jones, Sam Bowman, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Nelson Elhage, Sheer El-Showk, Stanislav Fort, Zac Hatfield-Dodds, Tom Henighan, Danny Hernandez, Tristan Hume, Josh Jacobson, Scott Johnston, Shauna Kravec, Catherine Olsson, Sam Ringer, Eli Tran-Johnson, Dario Amodei, Tom Brown, Nicholas Joseph, Sam McCandlish, Chris Olah, Jared Kaplan, and Jack Clark. Red Teaming Language Models to Reduce Harms: Methods, Scaling Behaviors, and Lessons Learned, 2022. URL http://arxiv.org/abs/2209.07858.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *ICML'23*, pages 10835–10866, Honolulu, Hawaii, USA, 2023. JMLR.org.

Federica Gerace, Luca Saglietti, Stefano Sarao Mannelli, Andrew Saxe, and Lenka Zdeborová. Probing transfer learning with a model of synthetic correlated datasets. *Machine Learning: Science and Technology*, 3(1):015030, 2022.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.3.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of

factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*, 2023.

Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P. Adams, and Sergey Levine. Why generalization in RL is difficult: Epistemic pomdps and implicit partial observability. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 25502–25515, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/d5ff135377d39f1de7372c95c74dd962-Abstract.html.

Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, Lucy Campbell-Gillingham, Jonathan Uesato, Po-Sen Huang, Ramona Comanescu, Fan Yang, Abigail See, Sumanth Dathathri, Rory Greig, Charlie Chen, Doug Fritz, Jaume Sanchez Elias, Richard Green, Soňa Mokrá, Nicholas Fernando, Boxi Wu, Rachel Foley, Susannah Young, Iason Gabriel, William Isaac, John Mellor, Demis Hassabis, Koray Kavukcuoglu, Lisa Anne Hendricks, and Geoffrey Irving. Improving alignment of dialogue agents via targeted human judgements, 2022a. URL http://arxiv.org/abs/2209.14375.

Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, et al. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*, 2022b.

Dongyoung Go, Tomasz Korbak, Germán Kruszewski, Jos Rozen, Nahyeon Ryu, and Marc Dymetman. Aligning language models with preferences through f-divergence minimization. *arXiv preprint arXiv:2302.08215*, 2023.

Shivam Goel, Gyan Tatiya, Matthias Scheutz, and Jivko Sinapov. NovelGridworlds: A benchmark environment for detecting and adapting to novelties in open worlds. In *AAMAS Workshop on on Adaptive Learning Agents (ALA)*, 2021. URL https://hrilab.tufts.edu/publications/goeletal21ala.pdf.

Jake Grigsby and Yanjun Qi. Measuring Visual Generalization in Continuous Control from Pixels. *arXiv:2010.06740 [cs]*, 2020. URL http://arxiv.org/abs/2010.06740.

Arnav Gudibande. Koala Evaluation Set, 2023. URL https://github.com/arnav-gudibande/koala-test-set.

Almog Gueta, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. Knowledge is a region in weight space for fine-tuned language models. *arXiv preprint arXiv:2302.04863*, 2023.

Arthur Guez, Mehdi Mirza, Karol Gregor, Rishabh Kabra, Sebastien Racaniere, Theophane Weber, David Raposo, Adam Santoro, Laurent Orseau, Tom Eccles, Greg Wayne, David Silver, Timothy Lillicrap, and Victor Valdes. An investigation of Model-free planning: Boxoban levels, 2018. URL https://github.com/deepmind/boxoban-levels/.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gomez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, Gabriel Dulac-Arnold, Jerry Li, Mohammad Norouzi, Matt Hoffman, Ofir Nachum, George Tucker, Nicolas Heess, and Nando de Freitas. RL Unplugged: A Suite of Benchmarks for Offline Reinforcement Learning. *arXiv:2006.13888 [cs, stat]*, 2021. URL http://arxiv.org/abs/2006.13888.

Yijie Guo, Jongwook Choi, Marcin Moczulski, Shengyu Feng, Samy Bengio, Mohammad Norouzi, and Honglak Lee. Memory based trajectory-conditioned policies for learning from sparse rewards. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/2df45244f09369e16ea3f9117ca45157-Abstract.html.

William H. Guss, Cayden Codel, Katja Hofmann, Brandon Houghton, Noboru Kuno, Stephanie Milani, Sharada Mohanty, Diego Perez Liebana, Ruslan Salakhutdinov, Nicholay Topin, et al. The MineRL competition on sample efficient reinforcement learning using human priors. *NeurIPS Competition Track*, 2019.

Tuomas Haarnoja. *Acquiring Diverse Robot Skills via Maximum Entropy Deep Reinforcement Learning*. PhD thesis, UC Berkeley, 2018. URL https://escholarship.org/uc/item/25g6573w.

Danijar Hafner. Benchmarking the spectrum of agent capabilities. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=1W0z96MFEoH.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual Markov Decision Processes. *arXiv:1502.02259 [cs, stat]*, 2015. URL http://arxiv.org/abs/1502.02259.

Isaac Han, Dong-Hyeok Park, and Kyung-Joong Kim. A New Open-Source Off-Road Environment for Benchmark Generalization of Autonomous Driving. *IEEE Access*, 9: 136071–136082, 2021. ISSN 2169-3536. doi:10.1109/ACCESS.2021.3116710.

Nicklas Hansen and Xiaolong Wang. Generalization in Reinforcement Learning by Soft Data Augmentation. *arXiv:2011.13389 [cs]*, 2021. URL http://arxiv.org/abs/2011.13389.

Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A. Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021a. URL https://openreview.net/forum?id=o_V-MjyyGV_.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 3680–3693, 2021b. URL https://proceedings.neurips.cc/paper/2021/hash/1e0f65eb20acbfb27ee05ddc000b50ec-Abstract.html.

Botao Hao, Tor Lattimore, Csaba Szepesvári, and Mengdi Wang. Online sparse reinforcement learning. In Arindam Banerjee and Kenji Fukumizu, editors, *The 24th International*

Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event, volume 130 of *Proceedings of Machine Learning Research*, pages 316–324. PMLR, 2021. URL http://proceedings.mlr.press/v130/hao21a.html.

Luke Harries, Sebastian Lee, Jaroslaw Rzepecki, Katja Hofmann, and Sam Devlin. MazeExplorer: A Customisable 3D Benchmark for Assessing Generalisation in Reinforcement Learning. In *2019 IEEE Conference on Games (CoG)*, pages 1–4, 2019. doi:10.1109/CIG.2019.8848048.

James Harrison, Animesh Garg, Boris Ivanovic, Yuke Zhu, Silvio Savarese, Li Fei-Fei, and Marco Pavone. ADAPT: Zero-Shot Adaptive Policy Transfer for Stochastic Dynamical Systems. *arXiv:1707.04674 [cs]*, 2017. URL http://arxiv.org/abs/1707.04674.

Matthew J. Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. Interactive fiction games: A colossal adventure. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7903–7910. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6297.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3207–3214. AAAI Press, 2018a. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16669.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18),*

and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 3207–3214. AAAI Press, 2018b. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16669.

Dan Hendrycks, Mantas Mazeika, and Thomas Woodside. An Overview of Catastrophic AI Risks, 2023. URL http://arxiv.org/abs/2306.12001.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017a. URL https://openreview.net/forum?id=Sy2fzU9gl.

Irina Higgins, Arka Pal, Andrei A. Rusu, Loïc Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. DARLA: improving zero-shot transfer in reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1480–1490. PMLR, 2017b. URL http://proceedings.mlr.press/v70/higgins17a.html.

Felix Hill, Andrew K. Lampinen, Rosalia Schneider, Stephen Clark, Matthew Botvinick, James L. McClelland, and Adam Santoro. Environmental drivers of systematicity and generalization in a situated agent. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020a. URL https://openreview.net/forum?id=SklGryBtwr.

Felix Hill, Andrew K. Lampinen, Rosalia Schneider, Stephen Clark, Matthew Botvinick, James L. McClelland, and Adam Santoro. Environmental drivers of systematicity and generalization in a situated agent. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020b. URL https://openreview.net/forum?id=SklGryBtwr.

Felix Hill, Sona Mokra, Nathaniel Wong, and Tim Harley. Human Instruction-Following with Deep Reinforcement Learning via Transfer-Learning from Text. *arXiv:2005.09382 [cs]*, 2020c. URL http://arxiv.org/abs/2005.09382.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training Compute-Optimal Large Language Models, 2022. URL http://arxiv.org/abs/2203.15556.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 2019. URL http://proceedings.mlr.press/v97/houlsby19a.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob N. Foerster. "other-play" for zero-shot coordination. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4399–4410. PMLR, 2020. URL http://proceedings.mlr.press/v119/hu20a.html.

Hengyuan Hu, Adam Lerer, Brandon Cui, Luis Pineda, Noam Brown, and Jakob N. Foerster. Off-belief learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 4369–4379. PMLR, 2021. URL http://proceedings.mlr.press/v139/hu21c.html.

Biwei Huang, Fan Feng, Chaochao Lu, Sara Magliacane, and Kun Zhang. Adarl: What, where, and how to adapt in transfer reinforcement learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=8H5bpVwvt5.

Evan Hubinger, Chris van Merwijk, Vladimir Mikulik, Joar Skalse, and Scott Garrabrant. Risks from Learned Optimization in Advanced Machine Learning Systems. *arXiv:1906.01820 [cs]*, 2019. URL http://arxiv.org/abs/1906.01820.

Edward Hughes, Michael Dennis, Jack Parker-Holder, Feryal Behbahani, Aditi Mavalankar, Yuge Shi, Tom Schaul, and Tim Rocktaschel. Open-Endedness is Essential for Artificial Superhuman Intelligence, 2024. URL http://arxiv.org/abs/2406.04268.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *arXiv:1908.08351 [cs, stat]*, 2020. URL http://arxiv.org/abs/1908.08351.

Dieuwke Hupkes, Mario Giulianelli, Verna Dankers, Mikel Artetxe, Yanai Elazar, Tiago Pimentel, Christos Christodoulopoulos, Karim Lasri, Naomi Saphra, Arabella Sinclair, Dennis Ulmer, Florian Schottmann, Khuyagbaatar Batsuren, Kaiser Sun, Koustuv Sinha, Leila Khalatbari, Maria Ryskina, Rita Frieske, Ryan Cotterell, and Zhijing Jin. State-of-the-art generalisation research in NLP: A taxonomy and review, 2023. URL http://arxiv.org/abs/2210.03050.

Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: Lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13956–13968, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/e2ccf95a7f2e1878fcafc8376649b6e8-Abstract.html.

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=Qun8fv4qSby.

Alex Irpan and Xingyou Song. The Principle of Unchanged Optimality in Reinforcement Learning Generalization. *arXiv:1906.00336 [cs, stat]*, 2019. URL http://arxiv.org/abs/1906.00336.

Ayush Jain, Andrew Szot, and Joseph J. Lim. Generalization to new actions in reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4661–4672. PMLR, 2020. URL http://proceedings.mlr.press/v119/jain20b.html.

Samyak Jain, Robert Kirk, Ekdeep Singh Lubana, Robert P. Dick, Hidenori Tanaka, Tim Rocktäschel, Edward Grefenstette, and David Krueger. Mechanistically analyzing the effects of fine-tuning on procedurally defined tasks. In *The Twelfth International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=A0HKeKl4Nl.

Sarthak Jain and Byron C. Wallace. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages

3543–3556, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi:10.18653/v1/N19-1357. URL https://aclanthology.org/N19-1357.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The Robot Learning Benchmark & Learning Environment. *arXiv:1909.12271 [cs]*, 2019a. URL http://arxiv.org/abs/1909.12271.

Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 12627–12637. Computer Vision Foundation / IEEE, 2019b. doi:10.1109/CVPR.2019.01291. URL http://openaccess.thecvf.com/content_CVPR_2019/html/James_Sim-To-Real_via_Sim-To-Sim_Data-Efficient_Robotic_Grasping_via_Randomized-To-Canonical_Adaptation_Networks_CVPR_2019_paper.html.

janus. Mysteries of mode collapse. *LessWrong*, 2022. URL https://www.lesswrong.com/posts/t9svvNPNmFf5Qa3TA/mysteries-of-mode-collapse.

Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E. Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1645–1654. PMLR, 2017. URL http://proceedings.mlr.press/v70/jaques17a.html.

Minqi Jiang, Jelena Luketina, Nantas Nardelli, Pasquale Minervini, Philip H.S. Torr, Shimon Whiteson, and Tim Rocktäschel. WordCraft: An environment for benchmarking commonsense agents. In *Workshop on Language in Reinforcement Learning (LaRel)*, 2020. URL https://github.com/minqi/wordcraft.

Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob N. Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. In Marc'Aurelio Ranzato,

Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 1884–1897, 2021a. URL https://proceedings.neurips.cc/paper/2021/hash/0e915db6326b6fb6a3c56546980a8c93-Abstract.html.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 4940–4950. PMLR, 2021b. URL http://proceedings.mlr.press/v139/jiang21b.html.

Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 4246–4247. IJCAI/AAAI Press, 2016. URL http://www.ijcai.org/Abstract/16/643.

Arthur Juliani, Ahmed Khalifa, Vincent-Pierre Berges, Jonathan Harper, Ervin Teng, Hunter Henry, Adam Crespi, Julian Togelius, and Danny Lange. Obstacle tower: A generalization challenge in vision, control, and planning. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 2684–2691. ijcai.org, 2019a. doi:10.24963/ijcai.2019/373. URL https://doi.org/10.24963/ijcai.2019/373.

Arthur Juliani, Ahmed Khalifa, Vincent-Pierre Berges, Jonathan Harper, Ervin Teng, Hunter Henry, Adam Crespi, Julian Togelius, and Danny Lange. Obstacle tower: A generalization challenge in vision, control, and planning. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 2684–2691. ijcai.org, 2019b. doi:10.24963/ijcai.2019/373. URL https://doi.org/10.24963/ijcai.2019/373.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko,

Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596 (7873):583–589, 2021. ISSN 1476-4687. doi:10.1038/s41586-021-03819-2. URL https://www.nature.com/articles/s41586-021-03819-2.

Jeevesh Juneja, Rachit Bansal, Kyunghyun Cho, João Sedoc, and Naomi Saphra. Linear connectivity reveals generalization strategies. *arXiv preprint. arXiv:2205.12411*, 2022.

Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. Challenges and Applications of Large Language Models, 2023. URL http://arxiv.org/abs/2307.10169.

Yuji Kanagawa and Tomoyuki Kaneko. Rogue-gym: A new challenge for generalization in reinforcement learning. In *IEEE Conference on Games*, 2019a.

Yuji Kanagawa and Tomoyuki Kaneko. Rogue-Gym: A New Challenge for Generalization in Reinforcement Learning. *arXiv:1904.08129 [cs, stat]*, 2019b. URL http://arxiv.org/abs/1904.08129.

Ken Kansky, Tom Silver, David A. Mély, Mohamed Eldawy, Miguel Lázaro-Gredilla, Xinghua Lou, Nimrod Dorfman, Szymon Sidor, D. Scott Phoenix, and Dileep George. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1809–1818. PMLR, 2017. URL http://proceedings.mlr.press/v70/kansky17a.html.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models, 2020. URL http://arxiv.org/abs/2001.08361.

Andrej Karpathy. *MinGPT*, 2020.

Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, David L. Dill, Mykel J. Kochenderfer, and Clark Barrett. The Marabou Framework for Verification and Analysis of Deep Neural Networks. In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification*, Lecture Notes in Computer Science, pages 443–452, Cham, 2019. Springer International Publishing. ISBN 978-3-030-25540-4. doi:10.1007/978-3-030-25540-4_26.

Nan Rosemary Ke, Ishita Dasgupta, Silvia Chiappa, Anirudh Goyal, Theophane Weber, Jovana Mitrovic, Felix Hill, Stephanie C. Y. Chan, Michael Curtis Mozer, Danilo Jimenez Rezende, and Pushmeet Kohli. Parametric Generalization for Benchmarking Reinforcement Learning Algorithms. *ArXiv*, 2021. URL https://openreview.net/forum?id=zx36-ng_9U_.

Mete Kemertas and Tristan Aumentado-Armstrong. Towards robust bisimulation metric learning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 4764–4777, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/256bf8e6923a52fda8ddf7dc050a1148-Abstract.html.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, pages 341–348, Santorini, Greece, 2016. IEEE. URL http://arxiv.org/abs/1605.02097.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. Measuring compositional generalization: A comprehensive method on realistic data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SygcCnNKwr.

Muhammad Khalifa, Hady Elsahar, and Marc Dymetman. A distributional approach to

controlled text generation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=jWkw45-9AbL.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards Continual Reinforcement Learning: A Review and Perspectives. *arXiv:2012.13490 [cs]*, 2020. URL http://arxiv.org/abs/2012.13490.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the Effects of RLHF on LLM Generalisation and Diversity. In *The Twelfth International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=PXD3FAVHJT.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A Survey of Zero-shot Generalisation in Deep Reinforcement Learning. *Journal of Artificial Intelligence Research*, 76:201–264, 2023b. ISSN 1076-9757. doi:10.1613/jair.1.14174. URL https://www.jair.org/index.php/jair/article/view/14174.

Byungchan Ko and Jungseul Ok. Time Matters in Using Data Augmentation for Vision-based Deep Reinforcement Learning. *arXiv:2102.08581 [cs]*, 2021. URL http://arxiv.org/abs/2102.08581.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. OpenAssistant Conversations – Democratizing Large Language Model Alignment, 2023. URL http://arxiv.org/abs/2304.07327.

Tomasz Korbak, Kejian Shi, Angelica Chen, Rasika Bhalerao, Christopher L. Buckley, Jason Phang, Samuel R. Bowman, and Ethan Perez. Pretraining Language Models with Human Preferences, 2023. URL http://arxiv.org/abs/2302.08582.

Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. Understanding catastrophic forgetting in language models via implicit inference. *arXiv preprint arXiv:2309.10105*, 2023.

Dimitrios I. Koutras, Athanasios Ch Kapoutsis, Angelos A. Amanatiadis, and Elias B. Kosmatopoulos. MarsExplorer: Exploration of Unknown Terrains via Deep Reinforcement Learning and Procedurally Generated Environments. *arXiv:2107.09996 [cs]*, 2021. URL http://arxiv.org/abs/2107.09996.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=UYneFzXSJWh.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid Motor Adaptation for Legged Robots. *arXiv:2107.04034 [cs]*, 2021. URL http://arxiv.org/abs/2107.04034.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a. URL https://proceedings.neurips.cc/paper/2020/hash/0d2b2061826a5df3221116a5085a6052-Abstract.html.

Saurabh Kumar, Aviral Kumar, Sergey Levine, and Chelsea Finn. One solution is not all you need: Few-shot extrapolation via structured maxent RL. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020b. URL https://proceedings.neurips.cc/paper/2020/hash/5d151d1059a6281335a10732fc49620e-Abstract.html.

Heinrich Küttler, Nantas Nardelli, Thibaut Lavril, Marco Selvatici, Viswanath Sivakumar, Tim Rocktäschel, and Edward Grefenstette. TorchBeast: A PyTorch platform for dis-

tributed RL. *arXiv preprint arXiv:1910.03552*, 2019. URL https://github.com/facebookresearch/torchbeast.

Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a. URL https://proceedings.neurips.cc/paper/2020/hash/569ff987c643b4bedf504efda8f786c2-Abstract.html.

Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020b. URL https://proceedings.neurips.cc/paper/2020/hash/569ff987c643b4bedf504efda8f786c2-Abstract.html.

Vivian Lai and Chenhao Tan. On human predictions with explanations and predictions of machine learning models: A case study on deception detection. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 29–38, 2019.

Andrew K. Lampinen and Surya Ganguli. An analytic theory of generalization dynamics and transfer learning in deep linear networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=ryfMLoCqtQ.

Yann LeCun. Predictive learning, 2016. URL https://www.youtube.com/watch?v=0unt2Y4qxQo&t=1072s.

Andrew Lee, Xiaoyan Bai, Itamar Pres, Martin Wattenberg, Jonathan K. Kummerfeld, and Rada Mihalcea. A Mechanistic Understanding of Alignment Algorithms: A Case Study on DPO and Toxicity, 2024. URL http://arxiv.org/abs/2401.01967.

Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=HJgcvJBFvB.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv:2005.01643 [cs, stat]*, 2020. URL http://arxiv.org/abs/2005.01643.

Bonnie Li, Vincent François-Lavet, Thang Doan, and Joelle Pineau. Domain Adversarial Reinforcement Learning. *arXiv:2102.07097 [cs]*, 2021. URL http://arxiv.org/abs/2102.07097.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California, 2016. Association for Computational Linguistics. doi:10.18653/v1/N16-1014. URL https://aclanthology.org/N16-1014.

Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent World Representations: Exploring a Sequence Model Trained on a Synthetic Task, 2023a. URL http://arxiv.org/abs/2210.13382.

Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. MetaDrive: Composing Diverse Driving Scenarios for Generalizable Reinforcement Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3): 3461–3475, 2023b. ISSN 1939-3539. doi:10.1109/TPAMI.2022.3190471. URL https://ieeexplore.ieee.org/document/9829243.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. AlpacaEval: An automatic evaluator of instruction-following models. GitHub, 2023c. URL https://github.com/tatsu-lab/alpaca_eval.

Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*, 2023.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael I. Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3059–3068. PMLR, 2018. URL http://proceedings.mlr.press/v80/liang18b.html.

David Lindner, János Kramár, Matthew Rahtz, Thomas McGrath, and Vladimir Mikulik. Tracr: Compiled transformers as a laboratory for interpretability. *arXiv preprint arXiv:2301.05062*, 2023.

Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. *arXiv preprint arXiv:2210.10749*, 2022a.

Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Exposing attention glitches with flip-flop language modeling. *arXiv preprint arXiv:2306.00946*, 2023a.

Guan Ting Liu, Pu-Jen Cheng, and GuanYu Lin. Cross-State Self-Constraint for Feature Generalization in Deep Reinforcement Learning. *OpenReview*, 2020. URL https://openreview.net/forum?id=JiNvAGORcMW.

Hao Liu, Carmelo Sferrazza, and Pieter Abbeel. Chain of Hindsight Aligns Language Models with Feedback, 2023b. URL http://arxiv.org/abs/2302.02676.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022b.

Siyang Liu, Sahand Sabour, Yinhe Zheng, Pei Ke, Xiaoyan Zhu, and Minlie Huang. Rethinking and refining the distinct metric. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 762–770, Dublin, Ireland, 2022c. Association for Computational Linguistics. doi:10.18653/v1/2022.acl-short.86. URL https://aclanthology.org/2022.acl-short.86.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as Agents, 2023c. URL http://arxiv.org/abs/2308.03688.

Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, and Yang Liu. Jailbreaking chatgpt via prompt engineering: An empirical study. *arXiv preprint arXiv:2305.13860*, 2023d.

Vincenzo Lomonaco, Karan Desai, Eugenio Culurciello, and Davide Maltoni. Continual Reinforcement Learning in 3D Non-stationary Environments. *arXiv:1905.10112 [cs, stat]*, 2020. URL http://arxiv.org/abs/1905.10112.

Steven Loria. TextBlob: Simplified Text Processing — TextBlob 0.16.0 documentation, 2020. URL https://textblob.readthedocs.io/en/dev/.

Charles Lovering, Rohan Jha, Tal Linzen, and Ellie Pavlick. Predicting inductive biases of pre-trained models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=mNtmhaDkAr.

Xingyu Lu, Kimin Lee, Pieter Abbeel, and Stas Tiomkin. Dynamics Generalization via Information Bottleneck in Deep Reinforcement Learning. *arXiv:2008.00614 [cs, stat]*, 2020. URL http://arxiv.org/abs/2008.00614.

Ekdeep Singh Lubana and Robert P. Dick. A gradient flow framework for analyzing network pruning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual*

Event, Austria, May 3-7, 2021. OpenReview.net, 2021. URL https://openreview.net/forum?id=rumv7QmLUue.

Ekdeep Singh Lubana, Eric J. Bigelow, Robert P. Dick, David Krueger, and Hidenori Tanaka. Mechanistic Mode Connectivity, 2022. URL http://arxiv.org/abs/2211.08422.

Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob N. Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. A survey of reinforcement learning informed by natural language. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 6309–6317. ijcai.org, 2019. doi:10.24963/ijcai.2019/880. URL https://doi.org/10.24963/ijcai.2019/880.

Corey Lynch and Pierre Sermanet. Language Conditioned Imitation Learning over Unstructured Data. *arXiv:2005.07648 [cs]*, 2021. URL http://arxiv.org/abs/2005.07648.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *arXiv:1709.06009 [cs]*, 2017. URL http://arxiv.org/abs/1709.06009.

Wesley J. Maddox, Shuai Tang, Pablo Garcia Moreno, Andrew Gordon Wilson, and Andreas C. Damianou. Fast adaptation with linearized neural networks. In Arindam Banerjee and Kenji Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 2737–2745. PMLR, 2021. URL http://proceedings.mlr.press/v130/maddox21a.html.

Dhruv Malik, Yuanzhi Li, and Pradeep Ravikumar. When is generalizable reinforcement learning tractable? In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages

8032–8045, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/437d46a857214c997956eaf0e3b21a55-Abstract.html.

Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. In *International Conference on Machine Learning*, pages 23610–23641. PMLR, 2023.

Daniel J. Mankowitz, Nir Levine, Rae Jeong, Abbas Abdolmaleki, Jost Tobias Springenberg, Yuanyuan Shi, Jackie Kay, Todd Hester, Timothy A. Mann, and Martin A. Riedmiller. Robust reinforcement learning for continuous control with model misspecification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=HJgC60EtwB.

Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

Daniel S Weld Mausam. Solving relational MDPs with first-order machine learning. In *IN PROC. ICAPS WORKSHOP ON PLANNING UNDER UNCERTAINTY AND INCOMPLETE INFORMATION*. Citeseer, 2003.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal. In *Forty-First International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=f3TUipYU3U.

Bogdan Mazoure, Ahmed M. Ahmed, R. Devon Hjelm, Andrey Kolobov, and Patrick MacAlpine. Cross-trajectory representation learning for zero-shot generalization in RL. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=XOh5x-vxsrV.

Clara Meister, Stefan Lazov, Isabelle Augenstein, and Ryan Cotterell. Is sparse attention more interpretable? In *Proceedings of the 59th Annual Meeting of the Associa-*

tion for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 122–129, Online, 2021. Association for Computational Linguistics. doi:10.18653/v1/2021.acl-short.17. URL https://aclanthology.org/2021.acl-short.17.

Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, and Nat McAleese. Teaching language models to support answers with verified quotes, 2022. URL http://arxiv.org/abs/2203.11147.

Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. A Survey of Explainable Reinforcement Learning, 2022. URL http://arxiv.org/abs/2202.08434.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=B1DmUzWAW.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. ISSN 1476-4687. doi:10.1038/nature14236. URL https://doi.org/10.1038/nature14236.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1928–1937. JMLR.org, 2016a. URL http://proceedings.mlr.press/v48/mniha16.html.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P.

Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1928–1937. JMLR.org, 2016b. URL http://proceedings.mlr.press/v48/mniha16.html.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJGCiw5gl.

Jun Morimoto and Kenji Doya. Robust reinforcement learning. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 1061–1067. MIT Press, 2000. URL https://proceedings.neurips.cc/paper/2000/hash/e8dfff4676a47048d6f0c4ef899593dd-Abstract.html.

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. Ray: A distributed framework for emerging {AI} applications. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 561–577, 2018.

Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. *Advances in neural information processing systems*, 1, 1988.

Matthias Müller-Brockhausen, Mike Preuss, and Aske Plaat. Procedural Content Generation: Better Benchmarks for Transfer Reinforcement Learning. *arXiv:2105.14780 [cs]*, 2021. URL http://arxiv.org/abs/2105.14780.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through

meta-reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019a. URL https://openreview.net/forum?id=HyztsoC5Y7.

Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based RL. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019b. URL https://openreview.net/forum?id=HyxAfnA5tm.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating Online Reinforcement Learning with Offline Datasets. *arXiv:2006.09359 [cs, stat]*, 2021. URL http://arxiv.org/abs/2006.09359.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. WebGPT: Browser-assisted question-answering with human feedback, 2022. URL http://arxiv.org/abs/2112.09332.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany, 2016. Association for Computational Linguistics. doi:10.18653/v1/K16-1028. URL https://aclanthology.org/K16-1028.

Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. *arXiv preprint arXiv:2309.00941*, 2023.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *J. Mach. Learn. Res.*, 21:181:1–181:50, 2020. URL http://jmlr.org/papers/v21/20-212.html.

NetHack Wiki. NetHackWiki, 2020. URL https://nethackwiki.com/.

NetHack Wiki. Des-file format, 2021. URL https://nethackwiki.com/wiki/Des-file_format.

Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 663–670. Morgan Kaufmann, 2000.

Richard Ngo, Lawrence Chan, and Sören Mindermann. The alignment problem from a deep learning perspective, 2023. URL http://arxiv.org/abs/2209.00626.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent Model-Free RL is a Strong Baseline for Many POMDPs. *arXiv:2110.05038 [cs]*, 2021. URL http://arxiv.org/abs/2110.05038.

Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in RL. *arXiv preprint arXiv:1804.03720*, 2018.

Maya Okawa, Ekdeep Singh Lubana, Robert P Dick, and Hidenori Tanaka. Compositional abilities emerge multiplicatively: Exploring diffusion models on a synthetic task. *https://openreview.net/forum?id=ZXH8KUgFx3*, 2023.

OpenAI. OpenAI Gym: The DoomTakeCover-v0 environment, 2017. URL https://gym.openai.com/envs/DoomTakeCover-v0.

OpenAI. Introducing ChatGPT, 2022. URL https://openai.com/blog/chatgpt.

OpenAI. GPT-4 Technical Report, 2023. URL http://arxiv.org/abs/2303.08774.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving Rubik's Cube with a Robot Hand. *arXiv:1910.07113 [cs, stat]*, 2019a. URL http://arxiv.org/abs/1910.07113.

OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv:1912.06680 [cs, stat]*, 2019b. URL http://arxiv.org/abs/1912.06680.

Takayuki Osa, Voot Tangkaratt, and Masashi Sugiyama. Discovering Diverse Solutions in Deep Reinforcement Learning by Maximizing State-Action-Based Mutual Information, 2022. URL http://arxiv.org/abs/2103.07084.

Ian Osband and Benjamin Van Roy. Near-optimal reinforcement learning in factored mdps. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 604–612, 2014. URL https://proceedings.neurips.cc/paper/2014/hash/0deb1c54814305ca9ad266f53bc82511-Abstract.html.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4026–4034, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/8d8818c8e140c64c743113f563cf750f-Abstract.html.

Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvári, Satinder Singh, Benjamin Van Roy, Richard S. Sutton, David Silver, and Hado van Hasselt. Behaviour suite for reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020a. URL https://openreview.net/forum?id=rygf-kSYwH.

Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvári, Satinder Singh, Benjamin Van Roy, Richard S. Sutton, David Silver, and Hado van Hasselt. Behaviour suite for reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020b. URL https://openreview.net/forum?id=rygf-kSYwH.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022a. URL http://arxiv.org/abs/2203.02155.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022b.

Marin Vlastelica P., Michal Rolínek, and Georg Martius. Neuro-algorithmic policies enable fast combinatorial generalization. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 10575–10585. PMLR, 2021. URL http://proceedings.mlr.press/v139/vlastelica21a.html.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing Generalization in Deep Reinforcement Learning. *arXiv:1810.12282 [cs, stat]*, 2019. URL http://arxiv.org/abs/1810.12282.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, 2002. Association for Computational Linguistics. doi:10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810, 2018. doi:10.1109/ICRA.2018.8460528. URL http://arxiv.org/abs/1710.06537.

Christian F. Perez, Felipe Petroski Such, and Theofanis Karaletsos. Generalized hidden parameter mdps: Transferable model-based RL in a handful of trials. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 5403–5411. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/5989.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3419–3448, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.225.

Diego Perez-Liebana, Jialin Liu, Ahmed Khalifa, Raluca D. Gaina, Julian Togelius, and Simon M. Lucas. General video game AI: A multitrack framework for evaluating agents, games, and content generation algorithms. *IEEE Transactions on Games*, 11(3):195–214, 2019a. doi:10.1109/TG.2019.2901021.

Diego Perez-Liebana, Jialin Liu, Ahmed Khalifa, Raluca D. Gaina, Julian Togelius, and Simon M. Lucas. General Video Game AI: A Multi-Track Framework for Evaluating Agents, Games and Content Generation Algorithms. *arXiv:1802.10363 [cs]*, 2019b. URL http://arxiv.org/abs/1802.10363.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural*

Language Processing: System Demonstrations, pages 46–54, Online, 2020. Association for Computational Linguistics. doi:10.18653/v1/2020.emnlp-demos.7. URL https://aclanthology.org/2020.emnlp-demos.7.

Thomas PIERROT, Valentin Macé, Jean-Baptiste Sevestre, Louis Monier, Alexandre Laterre, Nicolas Perrin, Karim Beguir, and Olivier Sigaud. Factored action spaces in deep reinforcement learning, 2021. URL https://openreview.net/forum?id=naSAkn2Xo46.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2817–2826. PMLR, 2017. URL http://proceedings.mlr.press/v70/pinto17a.html.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep RL: A short survey. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 4819–4825. ijcai.org, 2020. doi:10.24963/ijcai.2020/671. URL https://doi.org/10.24963/ijcai.2020/671.

Sam Powers, Eliot Xing, Eric Kolve, Roozbeh Mottaghi, and Abhinav Gupta. CORA: Benchmarks, Baselines, and Metrics as a Platform for Continual Reinforcement Learning Agents. *arXiv:2110.10067 [cs]*, 2021. URL http://arxiv.org/abs/2110.10067.

Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. *arXiv preprint arXiv:2402.14811*, 2024.

Rémi Le Priol, Reza Babanezhad, Yoshua Bengio, and Simon Lacoste-Julien. An analysis of the adaptation speed of causal models. In Arindam Banerjee and Kenji Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 775–783. PMLR, 2021. URL http://proceedings.mlr.press/v130/le-priol21a.html.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv preprint arXiv:2310.03693*, 2023.

Sébastien Racanière, Theophane Weber, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter W. Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. Imagination-augmented agents for deep reinforcement learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5690–5701, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/9e82757e9a1c12cb710ad680db11f6f1-Abstract.html.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. *Technical report, OpenAI*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021. URL http://proceedings.mlr.press/v139/radford21a.html.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia

Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling Language Models: Methods, Analysis & Insights from Training Gopher, 2022. URL http://arxiv.org/abs/2112.11446.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a85b405ed65c6477a4fe8302b5e06ce7-Paper-Conference.pdf.

Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8787–8798. PMLR, 2021. URL http://proceedings.mlr.press/v139/raileanu21a.html.

Roberta Raileanu and Tim Rocktäschel. RIDE: rewarding impact-driven exploration for procedurally-generated environments. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rkg-TJBFPB.

Roberta Raileanu, Max Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic

Data Augmentation for Generalization in Deep Reinforcement Learning. *arXiv:2006.12862 [cs]*, 2021. URL http://arxiv.org/abs/2006.12862.

Raghu Rajan, Jessica Lizeth Borja Diaz, Suresh Guttikonda, Fabio Ferreira, André Biedenkapp, Jan Ole von Hartz, and Frank Hutter. MDP Playground: A Design and Debug Testbed for Reinforcement Learning. *arXiv:1909.07750 [cs, stat]*, 2021. URL http://arxiv.org/abs/1909.07750.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is Reinforcement Learning (Not) for Natural Language Processing?: Benchmarks, Baselines, and Building Blocks for Natural Language Policy Optimization, 2022. URL http://arxiv.org/abs/2210.01241.

Eric S. Raymond, Mike Stephenson, et al. *A Guide to the Mazes of Menace: Guidebook for NetHack*. NetHack DevTeam, 2020. URL http://www.nethack.org/download/3.6.5/nethack-365-Guidebook.pdf.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410.

Yangang Ren, Jingliang Duan, Shengbo Eben Li, Yang Guan, and Qi Sun. Improving Generalization of Reinforcement Learning with Minimax Distributional Soft Actor-Critic. *arXiv:2002.05502 [cs, stat]*, 2020. URL http://arxiv.org/abs/2002.05502.

Sebastian Risi and Julian Togelius. Increasing generality in machine learning through procedural content generation. *Nat Mach Intell*, 2(8):428–436, 2020. ISSN 2522-5839. doi:10.1038/s42256-020-0208-z. URL https://www.nature.com/articles/s42256-020-0208-z.

Fereshteh Sadeghi and Sergey Levine. CAD2RL: Real Single-Image Flight without a Single Real Image. *arXiv:1611.04201 [cs]*, 2017. URL http://arxiv.org/abs/1611.04201.

Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Kuttler, Edward Grefenstette, and Tim Rocktäschel. MiniHack the Planet: A Sandbox for Open-Ended Reinforcement Learning Research. In *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. URL https://openreview.net/forum?id=skFwlyefkWJ.

Tom Schaul. A video game description language for model-based or interactive learning. In *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, pages 1–8, 2013. doi:10.1109/CIG.2013.6633610.

Tom Schaul. An Extensible Description Language for Video Games. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(4):325–331, 2014. ISSN 1943-0698. doi:10.1109/TCIAIG.2014.2352795.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.05952.

Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. Training Language Models with Language Feedback at Scale, 2023. URL http://arxiv.org/abs/2303.16755.

Karl Schmeckpeper, Oleh Rybkin, Kostas Daniilidis, Sergey Levine, and Chelsea Finn. Reinforcement Learning with Videos: Combining Offline Observations with Interaction. *arXiv:2011.06507 [cs]*, 2020. URL http://arxiv.org/abs/2011.06507.

Bernhard Schölkopf. Causality for Machine Learning. *arXiv:1911.10500 [cs, stat]*, 2019. URL http://arxiv.org/abs/1911.10500.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy

Lillicrap, and David Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020. ISSN 1476-4687. doi:10.1038/s41586-020-03051-4. URL https://www.nature.com/articles/s41586-020-03051-4.

Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. Online and offline reinforcement learning by planning with a learned model. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 27580–27591, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/e8258e5140317ff36c7f8225a3bf9590-Abstract.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv:1707.06347 [cs]*, 2017. URL http://arxiv.org/abs/1707.06347.

Younggyo Seo, Kimin Lee, Ignasi Clavera Gilaberte, Thanard Kurutach, Jinwoo Shin, and Pieter Abbeel. Trajectory-wise multiple choice learning for dynamics generalization in reinforcement learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/9739efc4f01292e764c86caa59af353e-Abstract.html.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-Contrastive Networks: Self-Supervised Learning from Video. *arXiv:1704.06888 [cs]*, 2018. URL http://arxiv.org/abs/1704.06888.

Sofia Serrano and Noah A. Smith. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy, 2019. Association for Computational Linguistics. doi:10.18653/v1/P19-1282. URL https://aclanthology.org/P19-1282.

Alessandro Sestini, Alexander Kuhnle, and Andrew D. Bagdanov. Demonstration-efficient Inverse Reinforcement Learning in Procedurally Generated Environments. *arXiv:2012.02527 [cs]*, 2020. URL http://arxiv.org/abs/2012.02527.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6cfe0e6127fa25df2a0ef2ae1067d915-Abstract.html.

Rohin Shah, Vikrant Varma, Ramana Kumar, Mary Phuong, Victoria Krakovna, Jonathan Uesato, and Zac Kenton. Goal Misgeneralization: Why Correct Specifications Aren't Enough For Correct Goals, 2022. URL http://arxiv.org/abs/2210.01790.

L. S. Shapley. Stochastic Games*. *Proceedings of the National Academy of Sciences*, 39(10): 1095–1100, 1953. doi:10.1073/pnas.39.10.1095. URL https://www.pnas.org/doi/10.1073/pnas.39.10.1095.

Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne P. Tchapmi, Micael E. Tchapmi, Kent Vainio, Josiah Wong, Li Fei-Fei, and Silvio Savarese. iGibson 1.0: A Simulation Environment for Interactive Tasks in Large Realistic Scenes. *arXiv:2012.02924 [cs]*, 2021. URL http://arxiv.org/abs/2012.02924.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. "Do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. *arXiv preprint arXiv:2308.03825*, 2023.

Hui Shi, Sicun Gao, Yuandong Tian, Xinyun Chen, and Jishen Zhao. Learning bounded context-free-grammar via LSTM and the transformer: Difference and the explanations. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth*

Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022, pages 8267–8276. AAAI Press, 2022. URL https://ojs.aaai.org/index.php/AAAI/article/view/20801.

Connor Shorten and Taghi M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. *J Big Data*, 6(1):60, 2019. ISSN 2196-1115. doi:10.1186/s40537-019-0197-0. URL https://doi.org/10.1186/s40537-019-0197-0.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *ArXiv*, abs/1712.01815, 2017.

Jaskirat Singh and Liang Zheng. Sparse Attention Guided Dynamic Value Estimation for Single-Task Multi-Scene Reinforcement Learning. *arXiv:2102.07266 [cs, stat]*, 2021. URL http://arxiv.org/abs/2102.07266.

Michael Sipser. Introduction to the theory of computation. *ACM Sigact News*, 27(1):27–29, 1996.

Charlie Snell, Ilya Kostrikov, Yi Su, Mengjiao Yang, and Sergey Levine. Offline RL for Natural Language Generation with Implicit Language Q Learning, 2022. URL http://arxiv.org/abs/2206.11871.

Shagun Sodhani, Franziska Meier, Joelle Pineau, and Amy Zhang. Block Contextual MDPs for Continual Learning. In *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, pages 608–623. PMLR, 2022. URL https://proceedings.mlr.press/v168/sodhani22a.html.

Anoopkumar Sonar, Vincent Pacelli, and Anirudha Majumdar. Invariant Policy Optimization: Towards Stronger Generalization in Reinforcement Learning. *arXiv:2006.01096 [cs, stat]*, 2020. URL http://arxiv.org/abs/2006.01096.

Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. Observational overfitting in reinforcement learning. In *8th International Conference on Learning Repre-*

sentations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net, 2020. URL https://openreview.net/forum?id=HJli2hNKDH.

Katherine Stasaski and Marti Hearst. Semantic diversity in dialogue with natural language inference. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 85–98, Seattle, United States, 2022. Association for Computational Linguistics. doi:10.18653/v1/2022.naacl-main.6. URL https://aclanthology.org/2022.naacl-main.6.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. Learning to summarize with human feedback. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1f89885d556929e98d3ef9b86448f951-Abstract.html.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback, 2022. URL http://arxiv.org/abs/2009.01325.

Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The Distracting Control Suite – A Challenging Benchmark for Reinforcement Learning from Pixels. *arXiv:2101.02722 [cs]*, 2021. URL http://arxiv.org/abs/2101.02722.

Alexander L. Strehl, Carlos Diuk, and Michael L. Littman. Efficient structure learning in factored-state MDPs. In *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 1*, AAAI'07, pages 645–650, Vancouver, British Columbia, Canada, 2007. AAAI Press. ISBN 978-1-57735-323-2.

Richard S Sutton, Andrew G Barto, et al. *Introduction to Reinforcement Learning*, volume 135. MIT press Cambridge, 1998.

Remi Tachet, Philip Bachman, and Harm van Seijen. Learning Invariances for Policy Generalization. *arXiv:1809.02591 [cs, stat]*, 2020. URL http://arxiv.org/abs/1809.02591.

Hidenori Tanaka, Aran Nayebi, Niru Maheswaranathan, Lane McIntosh, Stephen Baccus, and Surya Ganguli. From deep learning to mechanistic understanding in neuroscience: the structure of retinal prediction. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8535–8545, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/eeaebbffb5d29ff62799637fc51adb7b-Abstract.html.

Yujin Tang and David Ha. The sensory neuron as a transformer: Permutation-invariant neural networks for reinforcement learning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 22574–22587, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/be3e9d3f7d70537357c67bb3f4086846-Abstract.html.

Yujin Tang, Duong Nguyen, and David Ha. Neuroevolution of Self-Interpretable Agents. *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 414–424, 2020. doi:10.1145/3377930.3389847. URL http://arxiv.org/abs/2003.08165.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind Control Suite. *arXiv:1801.00690 [cs]*, 2018. URL http://arxiv.org/abs/1801.00690.

Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-Fitt, Valentin Dalibard, and Wojciech Marian Czarnecki. Open-Ended

Learning Leads to Generally Capable Agents. *arXiv:2107.12808 [cs]*, 2021. URL http://arxiv.org/abs/2107.12808.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy, 2019. Association for Computational Linguistics. doi:10.18653/v1/P19-1452. URL https://aclanthology.org/P19-1452.

Guy Tevet and Jonathan Berant. Evaluating the evaluation of diversity in natural language generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 326–346, Online, 2021. Association for Computational Linguistics. doi:10.18653/v1/2021.eacl-main.25. URL https://aclanthology.org/2021.eacl-main.25.

The Telepraph. The 25 hardest video games ever, 2021. URL https://www.telegraph.co.uk/gaming/what-to-play/the-15-hardest-video-games-ever/nethack/.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5, 2015. doi:10.1109/ITW.2015.7133169.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. *arXiv:1703.06907 [cs]*, 2017. URL http://arxiv.org/abs/1703.06907.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference On*, pages 5026–5033. IEEE, 2012. URL https://ieeexplore.ieee.org/abstract/document/6386109/.

Emma Tosch, Kaleigh Clary, John Foley, and David Jensen. Toybox: A Suite of Environments for Experimental Evaluation of Deep Reinforcement Learning. *arXiv:1905.02825 [cs, stat]*, 2019. URL http://arxiv.org/abs/1905.02825.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models, 2023a. URL http://arxiv.org/abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models, 2023b. URL http://arxiv.org/abs/2307.09288.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023c.

Nilesh Tripuraneni, Michael I. Jordan, and Chi Jin. On the theory of transfer learning: The importance of task diversity. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/59587bffec1c7846f3e34230141556ae-Abstract.html.

Puja Trivedi, Danai Koutra, and Jayaraman J Thiagarajan. A closer look at model adaptation using feature distortion and simplicity bias. *arXiv preprint arXiv:2303.13500*, 2023.

Josef Valvoda, Naomi Saphra, Jonathan Rawski, Adina Williams, and Ryan Cotterell. Benchmarking compositionality with formal languages. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6007–6018, Gyeongju, Republic of Korea, 2022. International Committee on Computational Linguistics. URL https://aclanthology.org/2022.coling-1.525.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2094–2100. AAAI Press, 2016. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12389.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Ruben Vereecken. PyVGDL 2.0: A video game description language for AI research, 2020. URL https://github.com/rubenvereecken/py-vgdl.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. StarCraft II: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor

Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019. ISSN 1476-4687. doi:10.1038/s41586-019-1724-z. URL https://www.nature.com/articles/s41586-019-1724-z.

Nelson Vithayathil Varghese and Qusay H. Mahmoud. A survey of multi-task deep reinforcement learning. *Electronics*, 9(9), 2020. ISSN 2079-9292. doi:10.3390/electronics9091363. URL https://www.mdpi.com/2079-9292/9/9/1363.

Elena Voita and Ivan Titov. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196, Online, 2020. Association for Computational Linguistics. doi:10.18653/v1/2020.emnlp-main.14. URL https://aclanthology.org/2020.emnlp-main.14.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy, 2019. Association for Computational Linguistics. doi:10.18653/v1/P19-1580. URL https://aclanthology.org/P19-1580.

Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. TL;DR: Mining Reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63, Copenhagen, Denmark, 2017. Association for Computational Linguistics. doi:10.18653/v1/W17-4508. URL https://aclanthology.org/W17-4508.

Jane X. Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv:1611.05763 [cs, stat]*, 2017. URL http://arxiv.org/abs/1611.05763.

Jane X. Wang, Michael King, Nicolas Porcel, Zeb Kurth-Nelson, Tina Zhu, Charlie Deck, Peter Choy, Mary Cassin, Malcolm Reynolds, Francis Song, Gavin Buttimore, David P. Reichert, Neil Rabinowitz, Loic Matthey, Demis Hassabis, Alexander Lerchner, and Matthew Botvinick. Alchemy: A structured task distribution for meta-reinforcement learning. *arXiv:2102.02926 [cs]*, 2021a. URL http://arxiv.org/abs/2102.02926.

Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. Improving generalization in reinforcement learning with mixture regularization. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a. URL https://proceedings.neurips.cc/paper/2020/hash/5a751d6a0b6ef05cfe51b86e5d1458e6-Abstract.html.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Front. Comput. Sci.*, 18(6):186345, 2024. ISSN 2095-2236. doi:10.1007/s11704-024-40231-1. URL https://doi.org/10.1007/s11704-024-40231-1.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. Paired Open-Ended Trailblazer (POET): Endlessly Generating Increasingly Complex and Diverse Learning Environments and Their Solutions. *arXiv:1901.01753 [cs]*, 2019. URL http://arxiv.org/abs/1901.01753.

Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeffrey Clune, and Kenneth O. Stanley. Enhanced POET: open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9940–9951. PMLR, 2020b. URL http://proceedings.mlr.press/v119/wang20l.html.

Xudong Wang, Long Lian, and Stella X. Yu. Unsupervised visual attention and invariance for reinforcement learning. In *IEEE Conference on Computer Vision and Pattern Recognition,*

CVPR 2021, virtual, June 19-25, 2021, pages 6677–6687. Computer Vision Foundation / IEEE, 2021b. doi:10.1109/CVPR46437.2021.00661. URL https://openaccess.thecvf.com/content/CVPR2021/html/Wang_Unsupervised_Visual_Attention_and_Invariance_for_Reinforcement_Learning_CVPR_2021_paper.html.

Yihan Wang, Si Si, Daliang Li, Michal Lukasik, Felix Yu, Cho-Jui Hsieh, Inderjit S Dhillon, and Sanjiv Kumar. Two-stage LLM fine-tuning with less specialization and more generalization. *arXiv preprint arXiv:2211.00635*, 2022a.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates, 2022b. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.340.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning Language Models with Self-Generated Instructions, 2023. URL http://arxiv.org/abs/2212.10560.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1995–2003. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/wangf16.html.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *arXiv preprint arXiv:2307.02483*, 2023.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=gEZrGCozdqR.

Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking like transformers. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 11080–11090. PMLR, 2021. URL http://proceedings.mlr.press/v139/weiss21a.html.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SJeYe0NtvH.

Zac Wellmer and James T. Kwok. Dropout's Dream Land: Generalization from Learned Simulators to Reality. In Nuria Oliver, Fernando Pérez-Cruz, Stefan Kramer, Jesse Read, and Jose A. Lozano, editors, *Machine Learning and Knowledge Discovery in Databases. Research Track*, Lecture Notes in Computer Science, pages 255–270, Cham, 2021. Springer International Publishing. ISBN 978-3-030-86486-6. doi:10.1007/978-3-030-86486-6_16.

Jiaxin Wen, Ruiqi Zhong, Akbir Khan, Ethan Perez, Jacob Steinhardt, Minlie Huang, Samuel R. Bowman, He He, and Shi Feng. Language models learn to mislead humans via RLHF. *arXiv preprint. arXiv:2409.12822*, 2024. URL https://arxiv.org/abs/2409.12822.

Sam Wenke, Dan Saunders, Mike Qiu, and Jim Fleming. Reasoning and Generalization in RL: A Tool Use Perspective. *arXiv:1907.02050 [cs]*, 2019. URL http://arxiv.org/abs/1907.02050.

Shimon Whiteson, Brian Tanner, Matthew E. Taylor, and Peter Stone. Protecting against evaluation overfitting in empirical reinforcement learning. In *2011 IEEE Symposium on*

Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), pages 120–127, 2011. doi:10.1109/ADPRL.2011.5967363.

Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China, 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-1002. URL https://aclanthology.org/D19-1002.

D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997. ISSN 1941-0026. doi:10.1109/4235.585893.

Sirui Xie, Xiaojian Ma, Peiyu Yu, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. HALMA: Humanlike Abstraction Learning Meets Affordance in Rapid Problem Solving. *arXiv:2102.11344 [cs]*, 2021. URL http://arxiv.org/abs/2102.11344.

Eliot Xing, Abhinav Gupta, Sam Powers, and Victoria Dean. KitchenShift: Evaluating Zero-Shot Generalization of Imitation-Based Policy Learning Under Domain Shifts. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021. URL https://openreview.net/forum?id=DdglKo8hBq0.

Cheng Xue, Vimukthini Pinto, Chathura Gamage, Ekaterina Nikonova, Peng Zhang, and Jochen Renz. Phy-Q: A Benchmark for Physical Reasoning. *arXiv:2108.13696 [cs]*, 2021. URL http://arxiv.org/abs/2108.13696.

Greg Yang and Edward J Hu. Feature learning in infinite-width neural networks. *arXiv preprint arXiv:2011.14522*, 2020.

Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a. URL https://proceedings.neurips.cc/paper/2020/hash/32cfdce9631d8c7906e8e9d6e68b514b-Abstract.html.

Xianjun Yang, Xiao Wang, Qi Zhang, Linda Petzold, William Yang Wang, Xun Zhao, and Dahua Lin. Shadow alignment: The ease of subverting safely-aligned language models. *arXiv preprint arXiv:2310.02949*, 2023.

Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-hsuan Sung, Brian Strope, and Ray Kurzweil. Multilingual universal sentence encoder for semantic retrieval. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 87–94, Online, 2020b. Association for Computational Linguistics. doi:10.18653/v1/2020.acl-demos.12. URL https://aclanthology.org/2020.acl-demos.12.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=GY6-6sTvGaf.

Lin Yen-Chen, Maria Bauza, and Phillip Isola. Experience-Embedded Visual Foresight. *arXiv:1911.05071 [cs]*, 2019. URL http://arxiv.org/abs/1911.05071.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019a. URL https://arxiv.org/abs/1910.10897.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *arXiv preprint arXiv:1910.10897*, 2019b.

Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the Unknown: Learning a Universal Policy with Online System Identification. *arXiv:1702.02453 [cs]*, 2017. URL http://arxiv.org/abs/1702.02453.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. RRHF:

Rank Responses to Align Language Models with Human Feedback without tears, 2023. URL http://arxiv.org/abs/2304.05302.

Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick, Oriol Vinyals, and Peter Battaglia. Relational Deep Reinforcement Learning. *arXiv:1806.01830 [cs, stat]*, 2018. URL http://arxiv.org/abs/1806.01830.

Vinícius Flores Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David P. Reichert, Timothy P. Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick, Oriol Vinyals, and Peter W. Battaglia. Deep reinforcement learning with relational inductive biases. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=HkxaFoC9KQ.

Amy Zhang, Nicolas Ballas, and Joelle Pineau. A Dissection of Overfitting and Generalization in Continuous Reinforcement Learning. *arXiv:1806.07937 [cs, stat]*, 2018a. URL http://arxiv.org/abs/1806.07937.

Amy Zhang, Yuxin Wu, and Joelle Pineau. Natural Environment Benchmarks for Reinforcement Learning. *arXiv:1811.06032 [cs, stat]*, 2018b. URL http://arxiv.org/abs/1811.06032.

Amy Zhang, Clare Lyle, Shagun Sodhani, Angelos Filos, Marta Kwiatkowska, Joelle Pineau, Yarin Gal, and Doina Precup. Invariant causal prediction for block mdps. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 11214–11224. PMLR, 2020a. URL http://proceedings.mlr.press/v119/zhang20t.html.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria,*

May 3-7, 2021. OpenReview.net, 2021a. URL https://openreview.net/forum?id=-2FCwDKRREu.

Baohe Zhang, Raghu Rajan, Luis Pineda, Nathan O. Lambert, André Biedenkapp, Kurtland Chua, Frank Hutter, and Roberto Calandra. On the importance of hyperparameter optimization for model-based reinforcement learning. In Arindam Banerjee and Kenji Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 4015–4023. PMLR, 2021b. URL http://proceedings.mlr.press/v130/zhang21n.html.

Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A Study on Overfitting in Deep Reinforcement Learning. *arXiv:1804.06893 [cs, stat]*, 2018c. URL http://arxiv.org/abs/1804.06893.

Hanping Zhang and Yuhong Guo. Generalization of Reinforcement Learning with Policy-Aware Adversarial Data Augmentation. *arXiv:2106.15587 [cs]*, 2021. URL http://arxiv.org/abs/2106.15587.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018d. URL https://openreview.net/forum?id=r1Ddp1-Rb.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: Open Pre-trained Transformer Language Models, 2022. URL http://arxiv.org/abs/2205.01068.

Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E. Gonzalez, and Yuandong Tian. BeBold: Exploration beyond the boundary of explored regions. *CoRR*, abs/2012.08621, 2020b. URL https://arxiv.org/abs/2012.08621.

Tianjun Zhang, Fangchen Liu, Justin Wong, Pieter Abbeel, and Joseph E. Gonzalez. The Wisdom of Hindsight Makes Language Models Better Instruction Followers, 2023. URL http://arxiv.org/abs/2302.05206.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/250cf8b51c773f3f8dc8b4be867a9a02-Abstract.html.

Xiao Zhang and Ji Wu. Dissecting learning and forgetting in language model finetuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tmsqb6WpLz.

Chenyang Zhao and Timothy Hospedales. Robust Domain Randomised Reinforcement Learning through Peer-to-Peer Distillation. *arXiv:2012.04839 [cs]*, 2020. URL http://arxiv.org/abs/2012.04839.

Chenyang Zhao, Olivier Sigaud, Freek Stulp, and Timothy M. Hospedales. Investigating Generalisation in Continuous Deep Reinforcement Learning. *arXiv:1902.07015 [cs, stat]*, 2019. URL http://arxiv.org/abs/1902.07015.

Haoyu Zhao, Abhishek Panigrahi, Rong Ge, and Sanjeev Arora. Do transformers parse while predicting the masked word? *arXiv preprint arXiv:2303.08117*, 2023a.

Jiachen Zhao, Zhun Deng, David Madras, James Zou, and Mengye Ren. Learning and forgetting unsafe examples in large language models. *arXiv preprint arXiv:2312.12736*, 2023b.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu,

Jian-Yun Nie, and Ji-Rong Wen. A Survey of Large Language Models, 2023c. URL http://arxiv.org/abs/2303.18223.

Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744, 2020. doi:10.1109/SSCI47803.2020.9308468.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena, 2023. URL http://arxiv.org/abs/2306.05685.

Victor Zhong, Tim Rocktäschel, and Edward Grefenstette. RTFM: Generalising to Novel Environment Dynamics via Reading. *arXiv:1910.08210 [cs]*, 2021. URL http://arxiv.org/abs/1910.08210.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? A study in length generalization. *arXiv preprint arXiv:2310.16028*, 2023.

Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=6xHJ37MVxxp.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. In Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz, editors, *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1097–1100. ACM, 2018. doi:10.1145/3209978.3210080. URL https://doi.org/10.1145/3209978.3210080.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. Robosuite: A Modular Simulation Framework and Benchmark for Robot Learning. *arXiv:2009.12293 [cs]*, 2020. URL http://arxiv.org/abs/2009.12293.

Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. Transfer Learning in Deep Reinforcement Learning: A Survey. *arXiv:2009.07888 [cs, stat]*, 2021. URL http://arxiv.org/abs/2009.07888.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-Tuning Language Models from Human Preferences, 2020. URL http://arxiv.org/abs/1909.08593.

Luisa M. Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep RL via meta-learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=Hkl9JlBYvr.

Luisa M. Zintgraf, Leo Feng, Cong Lu, Maximilian Igl, Kristian Hartikainen, Katja Hofmann, and Shimon Whiteson. Exploration in approximate hyper-state space for meta reinforcement learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 12991–13001. PMLR, 2021. URL http://proceedings.mlr.press/v139/zintgraf21a.html.

Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

Appendix A

Appendices to A Survey of Zero-shot Generalisation in Deep Reinforcement Learning

A.1 Compositional Generalisation

One specific type of generalisation examined frequently in supervised learning is compositional generalisation [Hupkes et al., 2020, Keysers et al., 2020]. In Chapter 3 we explore a categorisation of compositional generalisation introduced by [Hupkes et al., 2020]. While this was designed for generalisation in language, many of those forms are relevant for RL. The five forms of compositional generalisation defined are:

1. **systematicity**: generalisation via systematically recombining known parts and rules,
2. **productivity**: the ability to extend predictions beyond the length seen in training data,
3. **substitutivity**: generalisation via the ability to replace components with synonyms,
4. **localism**: if model composition operations are local vs. global,
5. **overgeneralisation**: if models pay attention to or are robust to exceptions.

For intuition, we will explore examples of some of these different types of compositional generalisation in a block-stacking environment. An example of *systematicity* is the ability to stack blocks in new configurations once the basics of block-stacking are mastered. Similarly,

productivity can be measured by how many blocks the agent can generalise to, and the complexity of the stacking configurations. *Substitutivity* can be evaluated by the agent's ability to generalise to blocks of new colours, understanding that the new colour does not affect the physics of the block. In Section 2.4.3 we discuss how assumptions of compositional structure in the RL environment can enable us to test these forms of generalisation, and in Section 2.4.2 we will discuss how some of these forms of generalisation can be evaluated in current RL benchmarks.

A.2 Other Structural Assumptions on MDPs

Other forms of structured MDPs have been defined beyond the contextual MDP [Hallak et al., 2015] and leveraged to develop algorithms that exploit those structural assumptions. One type that holds promise for generalisation is the factored MDP. A **factored MDP** assumes a state space described by a set of discrete variables, denoted $S := \{S_1, S_2, \ldots, S_n\}$ [Boutilier et al., 2000, Strehl et al., 2007]. I follow the notation and definitions used by [Osband and Roy, 2014]. The transition function T has the following property:

Definition 8 (Factored transition functions)**.** *Given two states s, s' and action a in a factored MDP M, the transition function satisfies a conditional independence condition*

$$T(s'|s,a) = \prod_i P(s'_i|s,a), \tag{A.1}$$

where $P(s'_i|s,a)$ is the probability distribution for each factor S_i conditioned on the previous state and action.

Parallels to causal graphs can be drawn [Schölkopf, 2019], where a causal graph is a DAG, each vertex is a variable, and the directed edges represent causal relationships. We can rewrite the conditional independence assumption as

$$T(s'|s,a) = \prod_i P(s'_i|\mathrm{PA}(s'_i),a), \tag{A.2}$$

where the probability of each factor s_i only depends on its parent factors $\mathrm{PA}(s_i)$ from the previous time step. Ideally, these parents are only a subset of all factors so this representation results in a reduction in size from the original MDP. Further, this enforces that there are no

synchronous edges between factors in the same time step. Critically, the rewards can also be factored in the following way:

Definition 9 (Factored reward functions)**.** *Given two states s, s' and action a in a factored MDP M, the reward function satisfies a conditional independence condition*

$$\mathbb{E}\big[R(s,a)\big] = \sum_i \mathbb{E}\big[R_i(s_i,a)\big], \tag{A.3}$$

where $R_i(s_i,a)$ is the reward function for each factor S_i.

One can think of this factored MDP framework as an extension of the single context-MDP, where the combination of one or more of these factors can be represented as the context, with the space of all possible combinations of factors being the context set. In the latter case, this formulation explicitly encodes how generalisation can be achieved to new contexts via *systematicity* (Appendix A.1, [Hupkes et al., 2020]): the policy will be trained on contexts taking some values within the set of possible factors, and then be tested on unseen combinations of seen factors.

A more restricted form of structured MDP is the **relational MDP** [Mausam, 2003]. A relational MDP is described by tuple $\langle C, F, A, D, T, R\rangle$. C is the set of object types, F is the set of fluent schemata that are arguments that modify each object type. A is the set of action schemata that acts on objects, D is the set of domain objects, each associated with a single type from C, and finally, T is the transition function and R is the reward function. An additional assumption is that objects that are not acted upon do not change in a transition. Note that the relational MDP can be expanded into a factored MDP that does not assume the additional structure of the form of object types with invariant relations. While this form of MDP is a Planning Domain Definition Language (PDDL) and therefore lends itself well to planning algorithms, it is overly complex for learning algorithms.

Object-oriented MDPs [Diuk et al., 2008] are a simpler form of relational MDPs that are less constrained, and therefore hold more promise for learning methods. Objects, fluents, and actions are defined in the same way as in relational MDPs, but all transition dynamics are determined by a set of Boolean transition terms which consist of a set of pre-defined relation terms between objects and object attributes. In spite of this simplification, it is still

significantly constrained compared to CMDPs and can be difficult to use when describing complex systems.

A final example of a structured MDP is the **Block MDP**. Block MDPs [Du et al., 2019] are described by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{X}, p, q, R \rangle$ with a finite, unobservable state space $\mathcal{S}$ and possibly infinite, but observable space $\mathcal{X}$. p denotes the latent transition distribution, q is the (possibly stochastic) emission function and R the reward function. This structured MDP is useful in rich observation environments where the given observation space is large, but a much smaller state space can be found that yields an equivalent MDP. This allows for improved exploration and sample complexity bounds that rely on the size of that latent state space rather than the given observation space.

Appendix B

Appendices to Minihack the Planet: A Sandbox for Open-ended Reinforcement Learning Research

B.1 The `des-file` Format

B.1.1 Tutorial

As part of the release of MiniHack, I release an interactive tutorial jupyter notebook for the `des-file` format, which utilises MiniHack to visualise how the `des-file` affects the generated level.[1] Here I present an overview of the different kinds of `des-files`, how to add entities to levels, and the main sources of randomness that can be used to create a distribution

[1]The tutorial can be found in MiniHack's documentation at https://minihack.readthedocs.io.

of levels on which to train RL agents. An in-depth reference can also be found in NetHack Wiki [2021].

B.1.2 Types of des-files

There are two types of levels that can be created using des-file format, namely MAZE-type and ROOM-type:

1. MAZE-type levels are composed of maps of the level (specified with the MAP command) which are drawn using ASCII characters (see Fig. 3.3 lines 4-14), followed by descriptions of the contents of the level, described in detail below. In MAZE-type environments, the layout of the map is fixed, but random terrain can be created around (or within) that map using the MAZEWALK command, which creates a random maze from a given location and filling all available space of a certain terrain type.

2. ROOM-type levels are composed of descriptions of rooms (specified by the ROOM command), each of which can have its contents specified by the commands described below. Generally, the RANDOM_CORRIDORS command is then used to create random corridors between all the rooms so that they are accessible. On creation, the file specifies (or leaves random) the room's type, lighting and approximate location. It is also possible to create subrooms (using the SUBROOM command) which are rooms guaranteed to be within the outer room and are otherwise specified as normal rooms (but with a location relative to the outer room). Fig. B.3 illustrates an instance of the Oracle level specified by the ROOM-type des-file in Fig. B.2.

B.1.3 Adding Entities to des-files

As we have seen above, there are multiple ways to define the layout of a level using the des-file format. Once the layout is defined, it is useful to be able to add entities to the level. These could be monsters, objects, traps or other specific terrain features (such as sinks, fountains or altars). In general, the syntax for adding one of these objects is:

```
ENTITY: specification, location, extra-details
```

For example:

```
MONSTER: ('F',"lichen"), (1,1)
```

```
OBJECT: ('%',"apple"), (10,10)
TRAP: 'fire', (1,1)
# Sinks and Fountains have no specification
SINK: (1,1)
FOUNTAIN: (0,0)
```

As can be seen in Fig. B.2 lines 4-11, some objects (such as `statues`) have extra details that can be specified, such as the monster type (`montype`) of the statue. Note that many of the details here can instead be set to `random`, as shown for example in Fig. B.2, lines 29 and 33-36. In this case, the game engine chooses a suitable value for that argument randomly each time the level is generated. For monsters and objects, this randomness can be controlled by just specifying the class of the monster or object and letting the specific object or monster be chosen randomly. For example:

```
MONSTER: 'F', (1,1)
OBJECT: '%', (10,10)
```

This code would choose a random monster from the Fungus class (https://nethackwiki.com/wiki/Fungus), and a random object from the Comestible class (https://nethackwiki.com/wiki/Comestible).

B.1.4 Sources of Randomness in `des-file`

We have seen how to create either very fixed (MAZE-type) or very random (ROOM-type) levels, and how to add entities with some degree of randomness. The `des-file` format has many other ways of adding randomness, which can be used to control the level generation process, including where to add terrain and in what way. Many of these methods are used in `IF` statements, which can be in one of two forms:

```
IF[50%] {
    MONSTER: 'F', (1,1)
} ELSE {
    # ELSE is not always necessary
    OBJECT: '%', (1,1)
}

IF[$variable_name < 15] {
    MONSTER: 'F', (1,1)
```

```
}
```

In the first form, a simple percentage is used for the random choice, whereas in the second, a variable (which could have been randomly determined earlier in the file) is used. A natural way to specify this variable is either in other conditional statements (perhaps you randomly add some number of monsters, and want to count the number of monsters you add such that if there are many monsters, you also add some extra weapons for the agent), or through dice notation. Dice notation is used to specify random expressions which resolve to integers (and hence can be used in any place an integer would be). They are of the form `NdM`, which means to roll `N` `M`-sided dice and sum the result. For example:

```
$roll = 2d6
IF[$roll < 7] {
    MONSTER: random, random
}
```

Dice rolls can also be used for accessing arrays, another feature of the `des-file` format. Arrays are initialised with one or more objects of the same type, and can be indexed with integers (starting at 0), for example:

```
# An array of monster classes
$mon_letters = { 'A', 'L', 'V', 'H' }
# An array of monster names from each monster class respectively
$mon_names = { "Archon", "arch-lich", "vampire lord", "minotaur" }
# The monster to choose
$mon_index = 1d4 - 1
MONSTER:($mon_letters[$mon_index],$mon_names[$mon_index]),(10,18)
```

Another way to perform random operations with arrays is using the `SHUFFLE` command. This command takes an array and randomly shuffles it. This would not work with the above example, as the monster name needs to match the monster class (i.e. we could not use `('A', "minotaur")`). For example:

```
$object = object: { '[',')','*','%' }
SHUFFLE: $object
```

Now the `$object` array will be randomly shuffled. Often, something achievable with shuffling can also be achieved with dice-rolls, but it is simpler to use shuffled arrays rather than dice-

rolls (for example, if you wanted to guarantee each of the elements of the array was used exactly once, but randomise the order, it is much easier to just shuffle the array and select them in order rather than try and generate exclusive dice rolls).

B.1.5 Random Terrain Placement

When creating a level, we may want to specify the structure or layout of the level (using a MAZE-type level), but then randomly create the terrain within the level, which will determine accessibility and observability for the agent and monsters in the level. As an example, consider the example `des-file` in Fig. B.1. In this level, we start with an empty 11x9 `MAP` (lines 3-13). We first replace 33% of the squares with clouds `C`, and then 25% with trees `T` (lines 15,16). To ensure that any corner is accessible from any other, we create two random-walk lines using `randline` from opposite corners and make all squares on those lines floor (`.`). To give the agent a helping hand, we choose a random square in the centre of the room with `rndcoord` (which picks a random coordinate from a selection of coordinates, see lines 19,20) and place an apple there.

Several other methods of randomly creating selections such as `filter` (randomly remove points from a selection) and `gradient` (create a selection based on a probability gradient across an area) are described in the NetHack wiki `des-file` format page NetHack Wiki [2021].

```
MAZE: "mylevel", ' '
GEOMETRY:center,center
MAP
...........
...........
...........
...........
...........
...........
...........
...........
...........
ENDMAP
REGION:(0,0,11,9),lit,"ordinary"
REPLACE_TERRAIN:(0,0,11,9), '.', 'C', 33%
REPLACE_TERRAIN:(0,0,11,9), '.', 'T', 25%
TERRAIN:randline (0,9),(11,0), 5, '.'
TERRAIN:randline (0,0),(11,9), 5, '.'
$center = selection: fillrect (5,5,8,8)
$apple_location = rndcoord $center
OBJECT: ('%', "apple"), $apple_location

$monster = monster: { 'L','N','H','O','D','T' }
SHUFFLE: $monster
$place = { (10,8),(0,8),(10,0) }
SHUFFLE: $place
MONSTER: $monster[0], $place[0], hostile
STAIR:$place[2],down
BRANCH:(0,0,0,0),(1,1,1,1)
```

Figure B.1: An example `des-file` based on the `HideNSeek` environment.Fig. B.10 presents several instances of environments generated using this `des-file`.

```
LEVEL: "oracle"

ROOM: "ordinary" , lit, (3,3), (
    center,center), (11,9) {
  OBJECT:('`',"statue"),(0,0),
      montype:'C',1
  OBJECT:('`',"statue"),(0,8),
      montype:'C',1
  OBJECT:('`',"statue"),(10,0),
      montype:'C',1
  OBJECT:('`',"statue"),(10,8),
      montype:'C',1
  OBJECT:('`',"statue"),(5,1),
      montype:'C',1
  OBJECT:('`',"statue"),(5,7),
      montype:'C',1
  OBJECT:('`',"statue"),(2,4),
      montype:'C',1
  OBJECT:('`',"statue"),(8,4),
      montype:'C',1

  SUBROOM: "delphi" , lit , (4,3)
      , (3,3) {
    FOUNTAIN: (0, 1)
    FOUNTAIN: (1, 0)
    FOUNTAIN: (1, 2)
    FOUNTAIN: (2, 1)
    MONSTER: ('@', "Oracle"),
      (1,1)
```

```
ROOM: "ordinary" , random,
    random, random, random {
  STAIR: random, up
  OBJECT: random,random
}

ROOM: "ordinary" , random,
    random, random, random {
  STAIR: random, down
  OBJECT: random, random
  TRAP: random, random
  MONSTER: random, random
  MONSTER: random, random
}

ROOM: "ordinary" , random,
    random, random, random {
  OBJECT: random, random
  OBJECT: random, random
  MONSTER: random, random
}

ROOM: "ordinary" , random,
    random, random, random {
  OBJECT: random, random
  TRAP: random, random
  MONSTER: random, random
}
```

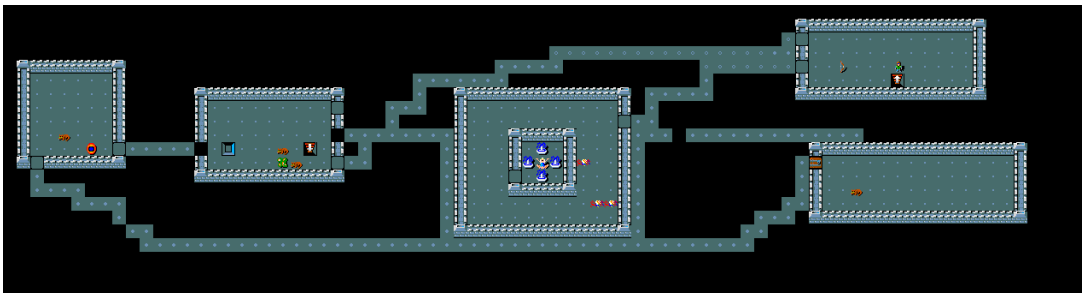


Figure B.3: A screenshot of the `Oracle` level specified using the `des-file` in Fig. B.2.

B.2 MiniHack

B.2.1 Observation Space

MiniHack, like NLE, has a dictionary-structured observation space. Most keys are inherited from NLE, while some are added in MiniHack. Note that using different observation keys can make environments significantly easier or harder (see Section 3.3.5 for discussion of how to ensure comparable experiments are performed).

- `glyphs` is a 21×79 matrix of glyphs (IDs of entities) on the map. Each glyph represents an entirely unique entity, so these are integers between 0 and `MAX_GLYPH` (5991). In the standard terminal-based view of NetHack, these glyphs are represented by characters, with colours and other possible visual features.
- `chars` is a 21×79 matrix of the characters representing the map.
- `colors` is a 21×79 matrix of the colours of each of the characters on the map (some characters represent different objects or monsters depending on their colour).
- `specials` is a 21×79 matrix of special extra information about the view of that cell on the map, for example, if the foreground and background colour should be reversed.
- `blstats` ("Bottom Line Stats") is a representation of the status line at the bottom of the screen, containing information about the player character's position, health, attributes and other statuses. It comes in the form of a dimension 25 vector.
- `message` is the utf-8 encoding of the on-screen message displayed at the top of the screen. It's a 256-dimensional vector.

- `tty_chars` is the character representation of the entire screen, including the message and map, of size 24×80.
- `tty_colors` is the color representation of the entire screen, including the message and map, of size 24×80.
- `tty_cursor` is the location of the cursor on the screen, a 2-dimensional vector of (x,y) coordinates.
- `inv_glyphs` is a 55-dimensional vector representing the glyphs present in the current inventory view.
- `inv_letters` is a 55-dimensional vector representing the letters present in the current inventory view.
- `inv_oclasses` is a 55-dimensional vector representing the class of objects present in the current inventory view.
- `inv_strs` is a 55×80 matrix containing utf-8 encodings of textual descriptions of objects present in the current inventory view.

MiniHack adds the following observation keys:

- `screen_descriptions` is a $21 \times 79 \times 80$ tensor of utf-8 encodings of textual descriptions of each cell present in the map. NetHack provides these textual descriptions (which can be accessed by the user by using the describe action on a specific tile).
- `pixel`. I provide pixel-level observations based on the official NetHack tile-set. This is a representation of the current screen in image form, where each cell is represented by a 16x16x3 image, meaning the entire observation is so $336 \times 1264 \times 3$ (with 3 channels for RGB). Examples of this pixel observation can be seen in Fig. 3.1 and Fig. 3.6.
- Cropped observations. For `glyphs`, `chars`, `colors`, `specials`, `tty_chars`, `tty_colors`, `pixel`, and `screen_descriptions` a cropped observation centered the agent can be used by passing the observation name suffixed with `_crop` (e.g. `chars_crop`). This is a NxN matrix centered on the agent's current location containing the information normally present

in the full view. The size of the crop can easily be configured using corresponding flags. Cropped observations can facilitate the learning, as the egocentric input makes representation learning easier.

B.2.2 Interface

There are two main MiniHack base classes to chose from, both derived from a common `MiniHack` base class.

`MiniHackNavigation` can be used to design mazes and navigation tasks that only require a small action space. All MiniHack navigation tasks I release, as well as MiniGrid and Boxoban examples, make use of the `MiniHackNavigation` interface. Here the pet is disabled by default. The in-game multiple-choice question prompts as well as menu navigation, are turned off by default.

`MiniHackSkill` provides a convenient mean for designing diverse skill acquisition tasks that require a large action space and more complex goals. All skill acquisition tasks in MiniHack use this base class. The in-game multiple-choice question prompts is turned on by default, while the menu navigation is turned off. The player's pet and auto-pickup are disabled by default.

When designing environments, I particularly suggest using partially underspecified levels in order to use the built-in procedural content generator that changes the environment after every episode. For example, several aspects of the level described in Fig. 3.3, such as the types and locations of monsters, objects, and traps, are not fully specified. The NetHack engine will make corresponding selections at random, making that specific feature of the environment procedurally generated. This enables a vast number of environment instances to be created. These could be instances the agent has never seen before, allowing for evaluation of test-time generalisation.

B.2.3 Level Generator

When creating a new MiniHack environment, a `des-file` must be provided. One way of providing this `des-file` is writing it entirely from scratch (for example files see Fig. B.2 and Fig. 3.3, and for example environment creation see Fig. B.4a). However, this requires learning the `des-file` format and is more difficult to do programmatically, so as part of

MiniHack I provide the `LevelGenerator` class which provides a convenient wrapper around writing a `des-file`. The `LevelGenerator` class can be used to create MAZE-type levels with specified heights and widths, and can then fill those levels with objects, monsters and terrain, and specify the start point of the level. Combined with the `RewardManager` which handles rewards (see Appendix B.2.4), this enables flexible creation of a wide variety of environments.

The level generator can start with either an empty maze (in which case only height and width are specified, see Fig. B.4b line 2) or with a pre-drawn map (see Fig. B.4c). After initialisation, the level generator can be used to add objects, traps, monsters and other terrain features. These can all be either completely or partially specified, in terms of class or location (see Appendix B.1.3 and Appendix B.1.4 for more information on how this works, and Fig. B.4b lines 6-10). Terrain can also be added programmatically at a later stage (Fig. B.4b lines 11-12). Once the level is complete, the `.get_des()` function returns the `des-file` which can then be passed to the environment creation function (Fig. B.4b lines 26-29).

B.2.4 Reward Manager

Along with creating the level layout, structure and content through the `LevelGenerator`, I also provide an interface to design custom reward functions. The default reward function of MiniHack is a sparse reward of +1 for reaching the staircase down (which terminates the episode), and 0 otherwise, with episodes terminating after a configurable number of time-steps. Using the `RewardManager`, one can control what events give the agent reward, whether those events can be repeated, and what combinations of events are sufficient or required to terminate the episode.

In Fig. B.4b lines 14-24 present an example of the reward manager usage. I first instantiate the reward manager and add events for eating an apple or wielding a dagger. The default reward is +1, and in general, all events are required to be completed for the episode to terminate. In lines 23-24, I add an event of standing on a sink which has a reward of -1 and is not required for terminating the episode.

While the basic reward manager supports many events by default, users may want to extend this interface to define their own events. This can be done easily by inheriting from the `Event` class and implementing the `check` and `reset` methods. Beyond that, custom reward

functions can be added to the reward manager through `add_custom_reward_fn` method. These functions take the environment instance, the previous observation, action taken and current observation, and should return a float.

We also provide two ways to combine events in a more structured way. The `SequentialRewardManager` works similarly to the normal reward manager but requires the events to be completed in the sequence they were added, terminating the episode once the last event is complete. The `GroupedRewardManager` combines other reward managers, with termination conditions defined across the reward managers (rather than individual events). This allows complex conjunctions and disjunctions of groups of events to specify termination. For example, one could specify a reward function that terminates if a sequence of events (a,b,c) was completed, or all events {d,e,f} were completed in any order and the sequence (g,h) was completed.

B.2.5 Examples

Fig. B.4a presents how to create a MiniHack navigation task using only the `des-file`, as in Fig. 3.3 or Fig. B.2.

Fig. B.4b shows how to create a simple skill acquisition task that challenges the agent to eat an apple and wield a dagger that is randomly placed in a 10x10 room surrounded by lava, alongside a goblin and a teleportation trap. Here, the `RewardManager` is used to specify the tasks that need to be completed.

Fig. B.4c shows how to create a labyrinth task. Here, the agent starts near the entrance of a maze and needs to reach its centre. A Minotaur is placed deep inside the maze, which is a powerful monster capable of instantly killing the agent in melee combat. There is a wand of death placed in a random location in the maze. The agent needs to pick it up, and upon seeing the Minotaur, zap it in the direction of the monster. Once the Minotaur is killed, the agent needs to navigate itself towards the staircase (this is the default goal when `RewardManager` is not used).

B.3 MiniHack tasks

This section presents detailed descriptions of existing MiniHack task and registered configurations. Tasks are grouped into similar tasks, within which several attributes are varied to

```python
# des_file is the path to the des-file
# or its content as a string
env = gym.make(
    "MiniHack-Navigation-Custom-v0",
    def_file=des_file)
```

(a) Creating a MiniHack navigation task via `des-file`.

```python
# Define a 10 by 10 grid area
lvl_gen = LevelGenerator(w=10, h=10)

# Populate it with different objects,
# a monster (goblin) and features (lava)
lvl_gen.add_object("apple", "%")
lvl_gen.add_object("dagger", ")")
lvl_gen.add_trap(name="teleport")
lvl_gen.add_sink()
lvl_gen.add_monster("goblin")
lvl_gen.fill_terrain("rect", "L",
                     0, 0, 9, 9)

# Define a reward manager
reward_manager = RewardManager()
# +1 reward and termination for eating
# an apple or wielding a dagger
reward_manager.add_eat_event("apple")
reward_manager.add_wield_event("dagger")
# -1 reward for standing on a sink
# but isn't required for terminating
# the episode
reward_manager.add_location_event("sink",
    reward=-1, terminal_required=False)

env = gym.make(
    "MiniHack-Skill-Custom-v0",
    des_file=lvl_gen.get_des(),
    reward_manager=reward_manager)
```

(b) Creating a skill task using the `LevelGenerator` and `RewardManager`.

```python
# Define the maze as a string
maze = """
--------------------
|.......|.|........|
|.------.|.|.-----|.|
|.|...|.|.|......|.|
|.|.|.|.|.|-----.|.|
|.|.|...|....|.|.|.|
|.|.---------.|.|.|.|
|.|..........|...|.|
|.|--------------|.|
|...................|
--------------------
"""
# Set a start and goal positions
lvl_gen = LevelGenerator(map=maze)
lvl_gen.set_start_pos((9, 1))
lvl_gen.add_goal_pos((14, 5))
# Add a Minotaur at fixed position
lvl_gen.add_monster(name="minotaur",
    place=(19, 9))
# Add wand of death
lvl_gen.add_object("death", "/")

env = gym.make(
    "MiniHack-Skill-Custom-v0",
    des_file = lvl_gen.get_des())
```

(c) Creating a MiniHack skill task using the `LevelGenerator` with a pre-defined map layout.

Figure B.4: Three ways to create MiniHack environments: using only a des-file, using the LevelGenerator and the RewardManager, and LevelGenerator with a pre-defined map layout.

make more difficult versions of the same task.

B.3.1 Navigation Tasks

Room. These tasks are set in a single square room, where the goal is to reach the staircase down (see Fig. B.5). There are multiple variants of this level. There are two sizes of the room (`5x5`, `15x15`). In the simplest variants, `Room-5x5` and `Room-15x15`), the start and goal position are fixed. In the `Room-Random-5x5` and `Room-Random-15x15` tasks, the start and goal position are randomised. The rest of the variants add additional complexity to the randomised version of the environment by introducing monsters (`Room-Monster-5x5` and `Room-Monster-15x15`), teleportation traps (`Room-Trap-5x5` and `Room-Trap-15x15`), darkness (`Room-Dark-5x5` and `Room-Dark-15x15`), or all three combined (`Room-Ultimate-5x5` and `Room-Ultimate-15x15`).[2]

Figure B.5: Various instances of the `Room-Ultimate-15x15` task.

Corridor. These tasks make use of the `RANDOM_CORRIDORS` command in the `des-file`. The objective is to reach the staircase down, which is in a random room (see Fig. B.6). The agent is also in a random room. The rooms have randomised positions and sizes. The corridors between the rooms are procedurally generated and are different for every episode. Different variants of this environment have different numbers of rooms, making the exploration challenge more difficult (`Corridor-R2`, `Corridor-R3`, and `Corridor-R5` environments are composed of 2, 3, and 5 rooms, respectively).

[2]The agent can attack monsters by moving towards them when located in an adjacent grid cell. Stepping on a lava tile instantly kills the agent. When the room is dark, the agent can only observe adjacent grid cells.

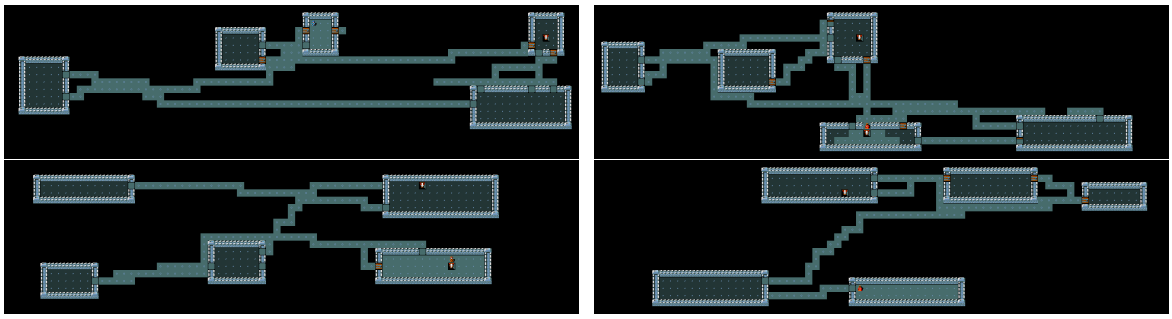


Figure B.6: Four instances of the `Corridor-R5` task.

KeyRoom. These tasks require an agent to pick up a key, navigate to a door, and use the key to unlock the door, reaching the staircase down within the locked room. The action space is the standard movement actions plus the `PICKUP` and `APPLY` actions (see Fig. B.7). In the simplest variant of this task, (`KeyRoom-Fixed-S5`), the location of the key, door and staircase are fixed. In the rest of the variants, these locations randomised. The size the outer room is 5x5 for `KeyRoom-S5` and 15x15 for `KeyRoom-S15`. To increase the difficulty of the tasks, dark versions of the tasks are introduced (`KeyRoom-Dark-S5` and `KeyRoom-Dark-S15`), where the key cannot be seen if it is not in any of the agent's adjacent grid cells.

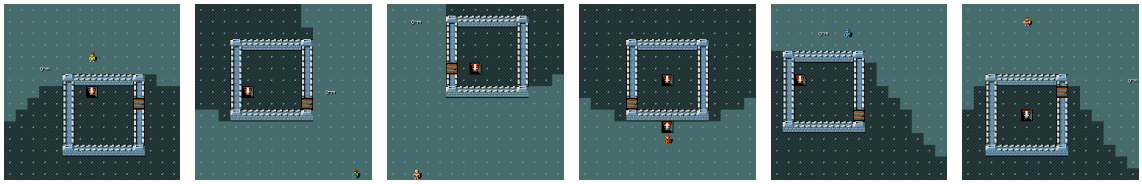


Figure B.7: Various instances of the `KeyRoom-S15` task.

MazeWalk. These navigation tasks make use of the `MAZEWALK` command in the `des-file`, which procedurally generates diverse mazes on the 9x9, 15x15 and 45x19 grids for `MazeWalk-9x9`, `MazeWalk-15x15`, and `MazeWalk-45x19` environments, respectively (see Fig. B.8). In the mapped versions of these tasks (`MazeWalk-Mapped-9x9`, `MazeWalk-Mapped-15x15`, and `MazeWalk-Mapped-45x19`), the map of the maze and the goal's locations are visible to the agent.

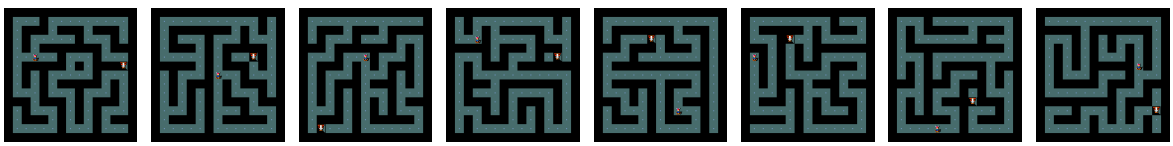


Figure B.8: Various instances of the `MazeWalk-15x15` task.

River. This group of tasks requires the agent to cross a river using boulders (see Fig. B.9). Boulders, when pushed into the water, create a dry land to walk on, allowing the agent to cross it. While the `River-Narrow` task can be solved by pushing one boulder into the water, other `River` require the agent to plan a sequence of at least two boulder pushes into the river next to each other. In the more challenging tasks of the group, the agent needs to additionally fight monsters (`River-Monster`), avoid pushing boulders into lava rather than water (`River-Lava`), or both (`River-MonsterLava`).

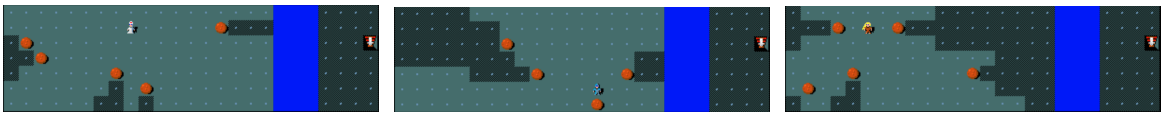


Figure B.9: Three instances of the `River` task.

HideNSeek. In the `HideNSeek` task, the agent is spawned in a big room full of trees and clouds (see Fig. B.10). The trees and clouds block the line of sight of the player and a random monster (chosen to be more powerful than the agent). The agent, monsters and spells can pass through clouds unobstructed. The agent and monster cannot pass through trees. The goal is to make use of the environment features, avoid being seen by the monster and quickly run towards the goal. The layout of the map is procedurally generated, hence requires systematic generalisation. Alternative versions of this task additionally include lava tiles that need to be avoided (`HideNSeek-Lava`), have bigger size (`HideNSeek-Big`) or provide the locations of all environment features but not the powerful monster (`HideNSeek-Mapped`).

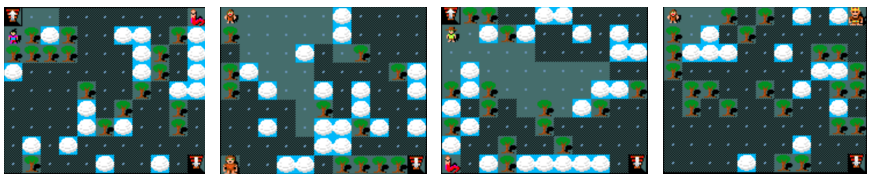


Figure B.10: Four instances of the `HideNSeek` task.

CorridorBattle. The `CorridorBattle` task challenges the agent to make the best use of the dungeon features to effectively defeat a horde of hostile monsters (see Fig. B.11). Here, if the agent lures the rats into the narrow corridor, it can defeat them one at a time. Fighting in rooms, on the other hand, would result in the agent simultaneously incurring damage from several directions and quick death. The task also is offered in dark mode (`CorridorBattle-Dark`),

challenging the agent to remember the number of rats killed in order to plan subsequent actions.

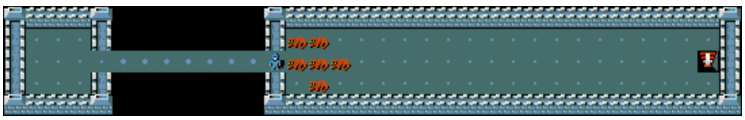


Figure B.11: A screenshot of the `CorridorBattle` task.

Memento. This group of tasks test the agent's ability to use memory (within an episode) to pick the correct path. The agent is presented with a prompt (in the form of a sleeping monster of a specific type) and then navigates along a corridor (see Fig. B.12). At the end of the corridor, the agent reaches a fork and must choose a direction. One direction leads to a grid bug, which if killed terminates the episode with a +1 reward. All other directions lead to failure through an invisible trap that terminates the episode when activated. The correct path is determined by the cue seen at the beginning of the episode. I provide three versions of this environment: one with a short corridor before a fork with two paths to choose from (`Memento-Short-F2`), one with a long corridor with a two-path fork (`Memento-F2`), and one with a long corridor and a four-fork path (`Memento-F4`).

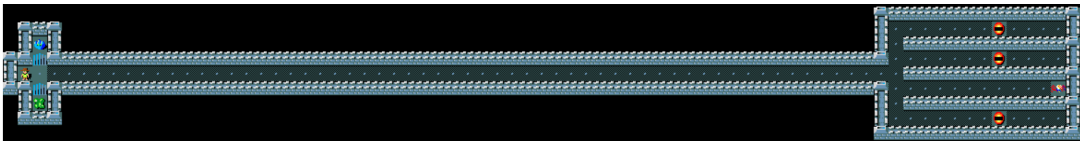


Figure B.12: A screenshot of the `Memento-F4` task.

MazeExplore. These tasks test the agent's ability to perform deep exploration Osband et al. [2016]. It's inspired by the Apple-Gold domain from Guo et al. [2020], where a small reward can be achieved easily, but to learn the optimal policy deeper exploration is required. The agent must first explore a simple randomised maze to reach the staircase down, which they can take for +1 reward (see Fig. B.13). However, if they navigate through a further randomised maze, they reach a room with apples. Eating the apples gives a +0.5 reward, and once the apples are eaten the agent should then return to the staircase down. I provide an easy and a hard version of this task (`MazeExplore-Easy` and `MazeExplore-Hard`), with the harder version having a larger maze both before and after the staircase down. Variants can also be mapped (`MazeExplore-Easy-Mapped` and `MazeExplore-Hard-Mapped`), where the agent

can observe the layout of the entire grid, making it easier to navigate the maze. Even in the mapped setting, the apples are not visible until the agent reaches the final room.

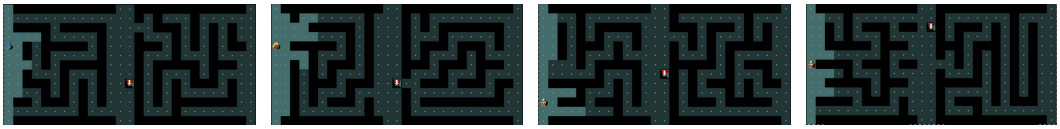


Figure B.13: Four instances of the `MazeExplore-Hard` task. The apples are located near the right vertical wall (unobservable in the figure). The goal is located in the middle area of the grid.

The full list of navigation tasks in MiniHack is provided in Table B.1.

B.3.2 Skill Acquisition Tasks

B.3.2.1 Skills

The nature of commands in NetHack requires the agent to perform a sequence of actions so that the initial action, which is meant for interaction with an object, has an effect. The exact sequence of subsequent can be inferred by the in-game message bar prompts.[3] For example, when located in the same grid with an apple lying on the floor, choosing the `Eat` action will not be enough for the agent to eat it. In this case, the message bar will ask the following question: *"There is an apple here; eat it? [ynq] (n)*". Choosing the `Y` action at the next timestep will cause the initial `EAT` action to take effect, and the agent will eat the apple. Choosing the `N` action (or `MORE` action since `N` is the default choice) will decline the previous `EAT` action prompt. The rest of the actions will not progress the in-game timer and the agent will stay in the same state. I refer to this skill as `Confirmation`.

The `PickUp` skill requires picking up objects from the floor first and put in the inventory. The tasks with `InventorySelect` skill necessities selecting an object from the inventory using the corresponding key, for example, *"What do you want to wear? [fg or ?*]"* or *"What do you want to zap? [f or ?*]"*. The `Direction` skill requires choosing one of the moving directions for applying the selected action, e.g., kicking or zapping certain types of wands. In this case, *"In what direction?"* message will appear on the screen. The `Navigation` skill tests the agent's ability to solve various mazes and labyrinths using the moving commands.

[3] Hence the messages are also used as part of observations in the skill acquisition tasks.

Table B.1: Full list of MiniHack navigation tasks and corresponding capabilities for assessment.

Name	Capability
`Room-5x5-v0`	Basic Learning
`Room-15x15-v0`	Basic Learning
`Room-Random-5x5-v0`	Basic Learning
`Room-Random-15x15-v0`	Basic Learning
`Room-Dark-5x5-v0`	Basic Learning
`Room-Dark-15x15-v0`	Basic Learning
`Room-Monster-5x5-v0`	Basic Learning
`Room-Monster-15x15-v0`	Basic Learning
`Room-Trap-5x5-v0`	Basic Learning
`Room-Trap-15x15-v0`	Basic Learning
`Room-Ultimate-5x5-v0`	Basic Learning
`Room-Ultimate-15x15-v0`	Basic Learning
`Corridor-R2-v0`	Exploration
`Corridor-R3-v0`	Exploration
`Corridor-R5-v0`	Exploration
`KeyRoom-Fixed-S5-v0`	Exploration
`KeyRoom-S5-v0`	Exploration
`KeyRoom-Dark-S5-v0`	Exploration
`KeyRoom-S15-v0`	Exploration
`KeyRoom-Dark-S15-v0`	Exploration
`MazeWalk-9x9-v0`	Exploration & Memory
`MazeWalk-Mapped-9x9-v0`	Exploration & Memory
`MazeWalk-15x15-v0`	Exploration & Memory
`MazeWalk-Mapped-15x15-v0`	Exploration & Memory
`MazeWalk-45x19-v0`	Exploration & Memory
`MazeWalk-Mapped-45x19-v0`	Exploration & Memory
`River-Narrow-v0`	Planning
`River-v0`	Planning
`River-Monster-v0`	Planning
`River-Lava-v0`	Planning
`River-MonsterLava-v0`	Planning
`HideNSeek-v0`	Planning
`HideNSeek-Mapped-v0`	Planning
`HideNSeek-Lava-v0`	Planning
`HideNSeek-Big-v0`	Planning
`CorridorBattle-v0`	Planning & Memory
`CorridorBattle-Dark-v0`	Planning & Memory
`Memento-Short-F2-v0`	Memory
`Memento-F2-v0`	Memory
`Memento-F4-v0`	Memory
`MazeExplore-Easy-v0`	Deep Exploration
`MazeExplore-Hard-v0`	Deep Exploration
`MazeExplore-Easy-Mapped-v0`	Deep Exploration
`MazeExplore-Hard-Mapped-v0`	Deep Exploration

B.3.2.2 Tasks

The full list of skill acquisition tasks, alongside the skills they require mastering, is provided in Table B.2. The skill acquisition tasks are suitable testbeds for fields such as curriculum learning and transfer learning, either between different tasks within MiniHack or to the full game of NetHack.

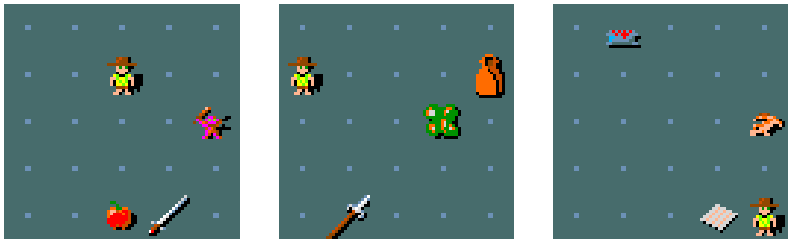


Figure B.14: Random instances of the `Eat-Distract`, `Wear-Distract` and `Pray-Distract` tasks.

Simple Tasks. The simplest skill acquisition tasks require discovering interaction between one object and the actions of the agent. These include: eating comestibles (`Eat`), praying on an altar (`Pray`), wearing armour (`Wear`), and kicking locked doors (`LockedDoors`). In the regular versions of these tasks, the starting location of the objects and the agent is randomised, whereas in the fixed versions of these tasks (`Eat-Fixed`, `Pray-Fixed`, `Wear-Fixed` and `LockedDoors-Fixed`) both are fixed. To add a slight complexity to the randomised version of these tasks, distractions in the form of a random object and a random monster are added to the third version of these tasks (`Eat-Distract`, `Pray-Distract` and `Wear-Distract`, see Fig. B.14). These tasks can be used as building blocks for more advanced skill acquisition tasks.

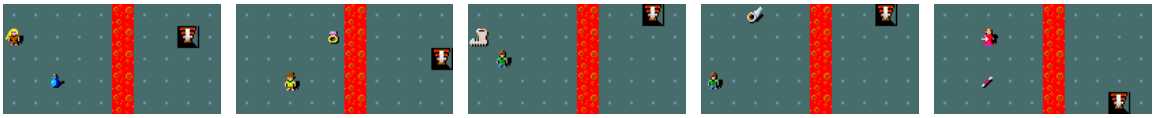


Figure B.15: Five random instances of the `LavaCross` task, where the agent needs to cross the lava using (i) potion of levitation, (ii) ring of levitation, (iii) levitation boots, (iv) frost horn, or (v) wand of cold.

Lava Crossing. An example of a more advanced task involves crossing a river of lava. The agent can accomplish this by either levitating over it (via a potion of levitation or levitation boots) or freezing it (by zapping the wand of cold or playing the frost horn). In the simplest version of the task (`LavaCross-Levitate-Potion-Inv`

and LavaCross-Levitate-Ring-Inv), the agent starts with one of the necessary objects in the inventory. Requiring the agent to pick up the corresponding object first makes the tasks more challenging (LavaCross-Levitate-Potion-PickUp and LavaCross-Levitate-Ring-PickUp). The most difficult variants of this task group require the agent to cross the lava river using one of the appropriate objects randomly sampled and placed at a random location. In LavaCross-Levitate, one of the objects of levitation is placed on the map, while in the LavaCross task these include all the objects for levitation as well as freezing (see Fig. B.15).

Figure B.16: A screenshot of the WoD-Hard task.

Wand of Death. MiniHack is very convenient for making incremental changes to the difficulty of a task. To illustrate this, I provide a sequence of tasks that require mastering the usage of the wand of death [WoD, NetHack Wiki, 2020]. Zapping a WoD in any direction fires a death ray which instantly kills almost any monster it hits. In WoD-Easy environment, the agent starts with a WoD in its inventory and needs to zap it towards a sleeping monster. WoD-Medium requires the agent to pick it up, approach the sleeping monster, kill it, and go to the staircase. In WoD-Hard the WoD needs to be found first, only then the agent should enter the corridor with a monster (who is awake and hostile this time), kill it, and go to the staircase (see Fig. B.16). In the most difficult task of the sequence, the WoD-Pro, the agent starts inside a big labyrinth. It needs to find the WoD inside the maze and reach its centre, which is guarded by a deadly Minotaur.

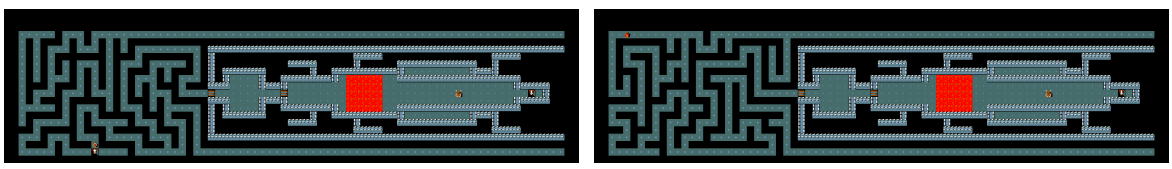


Figure B.17: Two instances of the Quest-Hard task.

Quest. Quest tasks require the agent to navigate mazes, cross rivers and fight powerful monsters. Quest_Easy, the simplest environment in the group, challenges the agent to use

Table B.2: Full list of MiniHack skill acquisition tasks.

Name	Skill
Eat-v0	Confirmation or PickUp+Inventory
Eat-Fixed-v0	Confirmation or PickUp+Inventory
Eat-Distract-v0	Confirmation or PickUp+Inventory
Pray-v0	Confirmation
Pray-Fixed-v0	Confirmation
Pray-Distract-v0	Confirmation
Wear-v0	PickUp+Inventory
Wear-Fixed-v0	PickUp+Inventory
Wear-Distract-v0	PickUp+Inventory
LockedDoor-v0	Direction
LockedDoor-Random-v0	Direction
LavaCross-Levitate-Ring-Inv-v0	Inventory
LavaCross-Levitate-Potion-Inv-v0	Inventory
LavaCross-Levitate-Ring-Pickup-v0	PickUp+Inventory
LavaCross-Levitate-Potion-PickUp-v0	PickUp+Inventory
LavaCross-Levitate-v0	PickUp+Inventory
LavaCross-v0	PickUp+Inventory
WoD-Easy	Inventory+Direction
WoD-Medium	PickUp+Inventory+Direction
WoD-Hard	PickUp+Inventory+Direction
WoD-Pro	Navigation+PickUp+Inventory+Direction
Quest-Easy-v0	Inventory
Quest-Medium-v0	Navigation+Inventory
Quest-Hard-v0	Navigation+PickUp+Inventory+Direction

objects in the inventory to cross the river of lava and fight a few relatively weak monsters. In Quest_Medium the agent engages with a horde of monsters instead and must lure them into the narrow corridors to survive. The Quest_Hard task, the most difficult environment of the group, requires the agent to solve complex, procedurally generated mazes, find objects for crossing the lava river and make use of the wand of death to kill a powerful monster (see Fig. B.17).

B.3.3 Ported Tasks

The full list of tasks ported to MiniHack from MiniGrid Chevalier-Boisvert et al. [2018] and Boxoban Guez et al. [2018] which I used in my experiments is provided in Table B.3. Note that more tasks could have similarly been ported from MiniGrid. However, my goal is to

Table B.3: Tasks ported to MiniHack from other benchmarks.

Name	Capability
`MultiRoom-N2-v0` Chevalier-Boisvert et al. [2018]	Exploration
`MultiRoom-N4-v0` Chevalier-Boisvert et al. [2018]	Exploration
`MultiRoom-N2-Monster-v0`	Exploration
`MultiRoom-N4-Monster-v0`	Exploration
`MultiRoom-N2-Locked-v0`	Exploration
`MultiRoom-N4-Locked-v0`	Exploration
`MultiRoom-N2-Lava-v0`	Exploration
`MultiRoom-N4-Lava-v0`	Exploration
`MultiRoom-N2-Extreme-v0`	Exploration
`MultiRoom-N4-Extreme-v0`	Exploration
`Boxoban-Unfiltered-v0` Guez et al. [2018]	Planning
`Boxoban-Medium-v0` Guez et al. [2018]	Planning
`Boxoban-Hard-v0` Guez et al. [2018]	Planning

showcase MiniHack's ability to port existing gridworld environments and easily enrich them, rather than porting all possible tasks.

B.4 Experiment Details

Instructions on how to replicate experiments I present can be found in MiniHack's repository: https://github.com/facebookresearch/minihack. Below I provide details on individual components.

B.4.1 Agent and Environment Details

The agent architecture used throughout all experiments is identical to it in Küttler et al. [2020a]. The observations used by the model include the 21×79 matrix of grid entity representations and a 21-dimensional vector containing agent statistics, such as its coordinates, health points, etc. Every of the 5991 possible entities in NetHack (monsters, items, dungeon features, etc.) is mapped onto a k-dimensional vector representation as follows. First, we split the entity IDs (glyphs) into one of twelve groups (categories of entities) and an ID within each group. We construct a partition of the final vector which includes the following components as sub-vectors: group, subgroup_id, color, character, and special, each of which uses a separate embedding that is learned throughout the training. The relative length of each sub-vector is defined as follows: groups=1, subgroup_ids=3, colors=1, chars=2, and specials=1. That

is, for a $k = 64$ dimensional embeddings, we make use of an embedding dimension of 24 for the ID, 8 for group, 8 for color, 16 for character, and 8 for special. These settings were determined during a set of small-scale experiments.

Three dense representations are produced. First, all visible entity embeddings are passed to a CNN. Second, another CNN is applied to the 9×9 crop of entities surrounding the agent. Third, an MLP is used to encode the agent's statistic. These three vectors are concatenated and passed to another MLP which produces the final representation $\mathbf{o}_t$ of the observation. To obtain the action distribution, we feed the observations $\mathbf{o}_t$ to a recurrent layer comprised with an LSTM Hochreiter and Schmidhuber [1997] cells, followed by an additional MLP.

For results on skill acquisition tasks, the in-game message, encoded using a character-level CNN Zhang et al. [2015], is also included as part of observation.

For all conducted experiments, a penalty of -0.001 is added to the reward function if the selected action of the agent does not increment the in-game timer of NetHack. For instance, when the agent attempts to move against a wall or navigates in-game menus, it will receive the -0.001 penalty.

B.4.2 TorchBeast Details

We employ an embedding dimension of 64 for entity representations. The size of the hidden dimension for the observation $\mathbf{o}_t$ and the output of the LSTM $\mathbf{h}_t$ is 256. We use a 5-layer CNN architecture (filter size 3×3, padding 1, stride 1) for encoding both the full screen of entities and the 9×9 agent-centred crop. The input channel of the first layer of the CNN is the embedding size of entities (64). The dimensions of all subsequent layers are equal to 16 for both input and output channels.

The characters within the in-game messages are encoded using an embedding of size 32 and passed to a 6-layer CNN architecture, each comprised of a 1D convolution of size 64 and ReLU nonlinearity. Maxpooling is applied after the first, second, and sixth layers. The convolutional layers are followed by an MLP.

We apply a gradient norm clipping of 40, but do not clip the rewards. We employ an RMSProp optimiser with a learning rate of $2 * 10^{-4}$ without momentum and with $\varepsilon = 10^{-6}$.[4] The entropy cost is set to 10^{-4}. The γ discounting factor is set to 0.999.

[4]For results on skill acquisition tasks, I use a learning rate of $5 * 10^{-5}$.

The hyperparameters for Random Network Distillation [RND, Ebner et al., 2013] are the same as in [Küttler et al., 2020a] and mostly follow the author recommendations. The weights are initialised using an orthogonal distribution with gains of $\sqrt{2}$. A two-headed value function is used for intrinsic and extrinsic rewards. The discount factor for the intrinsic reward is set to 0.99. We treat both extrinsic and intrinsic rewards as episodic in nature. The intrinsic reward is normalised by dividing it by a running estimate of its standard deviation. Unlike the original implementation of RND, I do not use observation normalisation due to the symbolic nature of observations used in my experiments. The intrinsic reward coefficient is 0.1. Intrinsic rewards are not clipped.

For the RIDE Raileanu and Rocktäschel [2020] baselines, I normalise the intrinsic reward by the number of visits to a state. The intrinsic reward coefficient is 0.1. The forward and inverse dynamics models have hidden dimension of 128. The loss cost is 1 for the forward model and 0.1 for inverse model.

These settings were determined during a set of small-scale experiments.

The training on MiniHack's `Room-5x5` task for two million timesteps using the IMPALA baseline takes approximately 4:30 minutes (roughly 7667 steps per second). This estimate is measured using 2 NVIDIA Quadro GP100 GPUs (one for the learner and one for actors) and 20 Intel(R) Xeon(R) E5-2698 v4 @ 2.20GHz CPUs (used by 256 simultaneous actors) on an internal cluster. The RND baseline completes the same number of timesteps in approximately 7:30 minutes (roughly 4092 steps per second) using the same computational resources.

B.4.3 Agent Architecture Comparison Details

Here I provide details on the architectures of models used in Fig. 3.11. The *medium* model uses the exact architectures described in Appendix B.4.2, namely a 5-layer CNN, hidden dimension size 256, and entity embedding size 64. The *small* model uses a 3-layer CNN, hidden dimension size 64, and entity embedding size 16. The *large* model uses a 9-layer CNN, hidden dimension size 512, and entity embedding size 128. The rest of the hyperparameters are identical for all models and are described in Fig. 3.11.

B.4.4 RLlib Details

We release examples of training several RL algorithms on MiniHack using RLlib Liang et al. [2018]. RLlib is an open-source scalable reinforcement learning library built on top of

Ray Moritz et al. [2018], that provides a unified framework for running experiments with many algorithms. I use the same model as in the TorchBeast implementation described above, adjusted to not manage the time dimension for the models that use an LSTM (as RLlib handles the recurrent network logic separately). To enable future research on a variety of methods, I provide examples of training DQN Mnih et al. [2015], PPO Schulman et al. [2017] and A2C Mnih et al. [2016a] on several simple MiniHack environments. The DQN agent makes use of Double Q-Learning van Hasselt et al. [2016], duelling architecture Wang et al. [2016] and prioritized experience replay (PER) Schaul et al. [2016]. I perform a limited sweep over hyperparameters, as I am not trying to achieve state-of-the-art results, just provide a starting point for future research.

In all experiments, I use 1 GPU for learning, and 10 CPUs for acting (with multiple instances of each environment per CPU), to speed up the wall-clock run-time of the experiments. These options can be configured easily in the code I release.

Results for these experiments can be seen in Fig. 3.12. Hyperparameters for DQN, PPO, and A2C are detailed in Table B.4, Table B.5 and Table B.6, respectively.

Table B.4: RLlib DQN Hyperparameters

Name	value
learning rate	1e-6
replay buffer size	100000
PER β	0.4
n-step length	5
target network update frequency	50000
learning start steps	50000
PER annealing timesteps	100000

Table B.5: RLlib PPO Hyperparameters

Name	value
learning rate	1e-5
batch size	128
SGD minibatch size	32
SGD iterations per epoch	2
rollout fragment length	128
entropy penalty coefficient	0.0001
value function loss coefficient	0.5
shared policy and value representation	True

Table B.6: RLlib A2C Hyperparameters

Name	value
learning rate	1e-5
batch size	128
rollout fragment length	128
entropy penalty coefficient	0.001
value function loss coefficient	0.1

B.4.5 Unsupervised Environment Design

We base the UED experiments using PAIRED on MiniHack largely on those outlined in [Dennis et al., 2020a]. The hyperparameters used for training are provided in Table B.7. Note that except where explicitly noted, all agents share the same training hyperparameters.

The adversary constructs new levels starting from an empty 5×5 grid. At each of the first 10 timesteps, the adversary chooses a position in which to place one of the following objects: {`walls`, `lava`, `monster`, `locked door`}. If a selected cell is already occupied, no additional object cell is placed. After placing the objects, the adversary then chooses the goal position followed by the agent's starting position. If a selected cell is already occupied, the position is randomly resampled from among the free cells.

At each time step, the adversary policy encodes the glyph observation using two convolution layers, each with kernel size 3×3, stride lengths of 1 and 2, and output channels, 16 and 32 respectively, followed by a ReLU activation over the flattened outputs. I embed the time step into a 10-dimensional space. The image embedding, time-step embedding, and the random noise vector are concatenated, and the combined representation is passed through an

LSTM with a hidden dimension of 256, followed by two fully connected layers with a hidden dimension of 32 and ReLU activations to yield the action logits over the 169 possible cell choices.

We make use of the same architecture for the protagonist and antagonist policies, with the exceptions of using the agent-centred crop, rather than the full glyph observation, and producing policy logits over the MiniHack action space rather than over the set of possible cell positions.

Table B.7: PAIRED hyperparameters

Name	value
γ	0.995
λ_{GAE}	0.95
PPO rollout length	256
PPO epochs	5
PPO minibatches per epoch	1
PPO clip range	0.2
PPO number of workers	32
Adam learning rate	1e-4
Adam ε	1e-5
PPO max gradient norm	0.5
PPO value clipping	yes
value loss coefficient	0.5
protagonist/antagonist entropy coefficient	0.0
adversary entropy coefficient	0.005

B.5 Full Results

Fig. B.18, Fig. B.19, and Fig. B.20 present the results of baseline agents on all navigation, skill acquisition and ported MiniHack tasks, respectively.

Appendix C

Appendices to Understanding the Effects

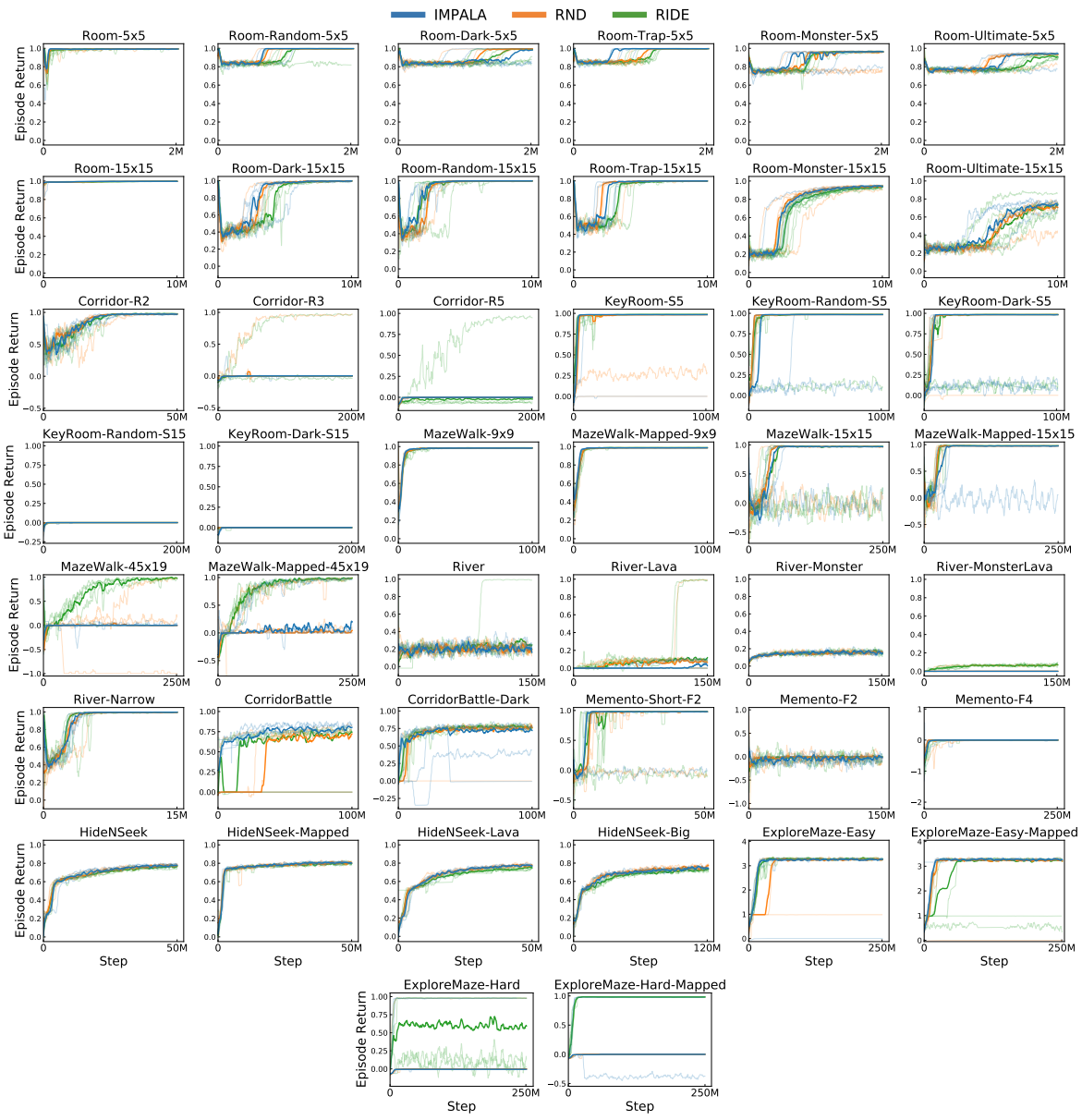


Figure B.18: Mean episode returns on all MiniHack navigation tasks across five independent runs. The median of the runs is bolded.

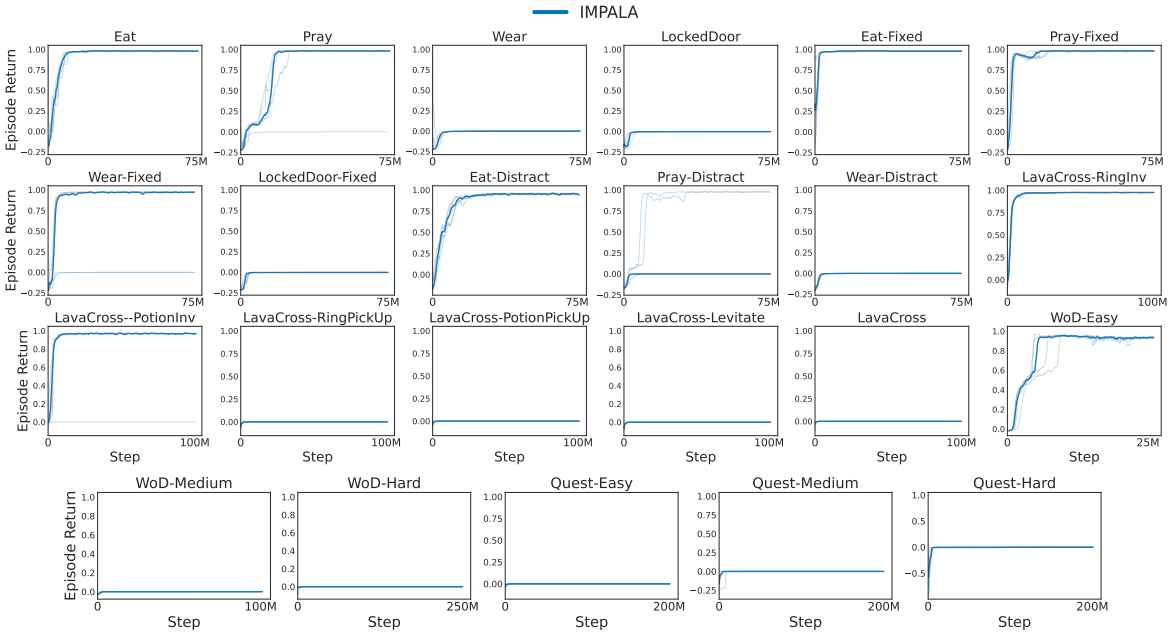


Figure B.19: Mean episode returns on all MiniHack skill acquisition tasks across five independent runs. The median of the runs is bolded.

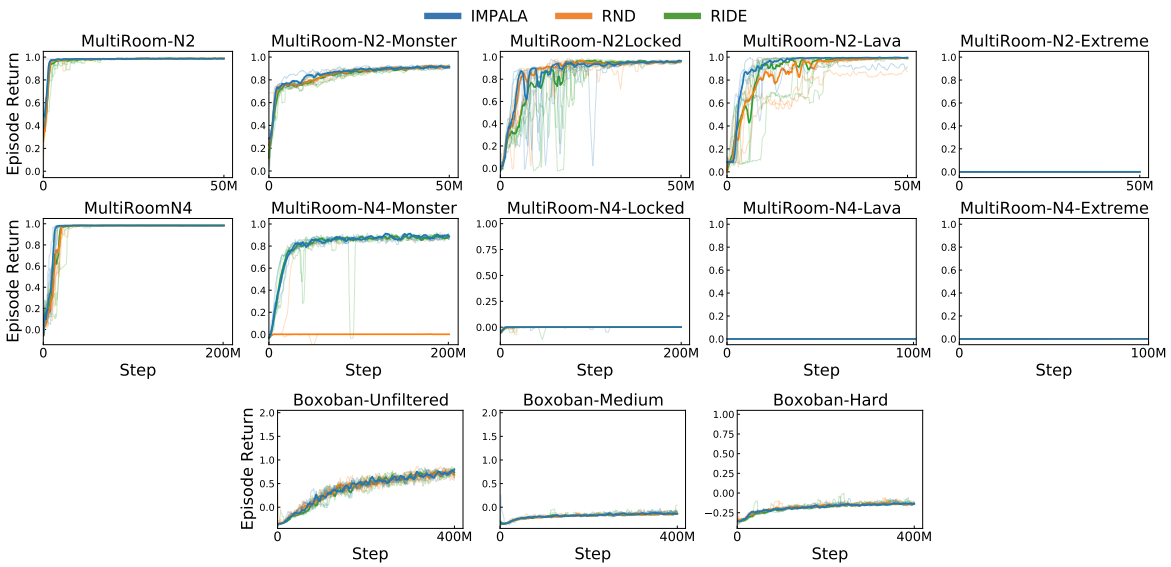


Figure B.20: Mean episode returns on tasks ported to MiniHack from existing benchmarks. The median of the five runs is bolded.

# of RLHF on LLM Generalisation and Diversity

C.1 GPT-4 Evaluation Details

For GPT-4 evaluations for both summarisation and instruction following, I use the AlpacaEval [Li et al., 2023c] software package to query GPT-4. For the instruction-following prompts, I use the standard annotator configuration recommended for that dataset, `alpaca_eval_gpt4`. For summarisation, I use the same configuration, but change the prompts to utilise a variant of those provided in [Rafailov et al., 2023]. For the exact prompts see Figs. C.1 and C.2.

C.1.1 Validating GPT-4 evaluation

Summarisation. I validate the use of the GPT-4 evaluator for summarisation in two ways. First, I use the evaluator to label the preference validation datasets for both TL;DR and CNN/DailyMail released by Stiennon et al. [2022] and measure their accuracy. On TL;DR the evaluators gets 71.7% accuracy and on CNN/DailyMail it gets 65.5% accuracy. Comparing this to the inter-annotator agreement reported by Stiennon et al. [2022] or $\tilde{7}0\%$ for TL;DR and $\tilde{6}6\%$ for CNN/DailyMail demonstrates that the annotator has strong agreement with the human raters that generated the preference data I use for training the reward models.

For the second validation of GPT-4 as an evaluator, I measure the agreement between human labellers and GPT-4 on a subset of 25 inputs for every test set I use, comparing both SFT and RLHF model outputs with the reference output. This results in a total of 100 datapoints, each labelled by two human labellers giving 200 total annotations. This tests whether GPT-4 is in agreement with human preferences on the models I evaluate in this work.Table C.1 shows the preference rating for GPT-4 and human labellers for each dataset and model, and the agreement between labellers and GPT-4, and I see that both at an aggregate level and at a per-example level the GPT-4 evaluator has good agreement with expert labellers.

Instruction Following. For instruction following, I use the evaluator released in [Li et al., 2023c], which has been rigorously shown to agree well with human labellers both at a

```
<|im_start|>system
You are a helpful assistant, that ranks models by the quality of their
answers.
<|im_end|>
<|im_start|>user
Which of the following summaries does a better job of summarizing the most
important points in the given news article, without including unimportant
or irrelevant details? A good summary is both precise and concise.
Post: """{instruction}"""
Summary A:
{
    "model": "model_1",
    "summary": """{output_1}"""
}
Summary B:
{
    "model": "model_2",
    "summary": """{output_2}"""
}

Now please rank the models by the quality of their summaries, so that the
model with rank 1 has the best summary. Then return a list of the model
names and ranks, i.e., produce the following output:
[
    {'model': <model-name>, 'rank': <model-rank>},
    {'model': <model-name>, 'rank': <model-rank>}
]

Your response must be a valid Python dictionary and should contain nothing
else because we will directly execute it in Python. Please provide the
ranking that the majority of humans would give.
<|im_end|>
```

Figure C.1: GPT-4 evaluation prompt for the CNN DailyMail dataset.

Table C.1: GPT-4 agreement with human raters at an aggregate and individual level, across both summarisation datasets and RLHF and SFT model types. I see that both at an aggregate level and at a per-example level the GPT-4 evaluator has good agreement with expert labellers.

Dataset	TL;DR		CNNDM	
Model Type	RLHF	SFT	RLHF	SFT
GPT-4 winrate	0.40	0.40	0.24	0.08
Human winrate	0.48	0.40	0.30	0.30
H-GPT-4 agreement	69%	72%	74%	72%

```
<|im_start|>system
You are a helpful assistant, that ranks models by the quality of their
answers.
<|im_end|>
<|im_start|>user
Which of the following summaries does a better job of summarizing the most
important points in the given forum post, without including unimportant
or irrelevant details? A good summary is both precise and concise.
Post: """{instruction}"""
Summary A:
{
    "model": "model_1",
    "summary": """{output_1}"""
}
Summary B:
{
    "model": "model_2",
    "summary": """{output_2}"""
}

Now please rank the models by the quality of their summaries, so that the
model with rank 1 has the best summary. Then return a list of the model
names and ranks, i.e., produce the following output:
[
    {'model': <model-name>, 'rank': <model-rank>},
    {'model': <model-name>, 'rank': <model-rank>}
]

Your response must be a valid Python dictionary and should contain nothing else
because we will directly execute it in Python. Please provide the ranking that
the majority of humans would give.
<|im_end|>
```

Figure C.2: GPT-4 evaluation prompt for the TL;DR dataset.

per-example and aggregate level. Hence, I do not validate this myself.

C.2 Model Training Details

C.2.1 Reward Model Training.

To train the Reward Model (RM), I again follow [Stiennon et al., 2022]. I initialise the RM as the base model, and I add a scalar head before the unembedding layer. The model is then fine-tuned on inputs and pairs of outputs with one output labelled as preferred. The RM takes the full input and output, and outputs a scalar value. This gives us a scalar value for each output in the pair, and these values are treated as logits in a softmax to predict the correct preference label. I train the RM with a cross-entropy loss. Formulating the RM in this way means it can (in theory) learn to predict any transitive ranking over possible outputs, while still maintaining a type signature suitable for use as a reward function in an RL context (producing a scalar value given a single input-output pair which acts as a proxy for the quality of the output given the input, as evaluated by human annotators).

C.2.2 Policy Training

We treat the autoregressive language model as a reinforcement learning policy, where the action space is the set of possible tokens, the state space is the current input, and the transition function adds the outputted action to the input. The episode terminates when the maximum number of tokens is generated, or the model outputs an end-of-sequence token.

Supervised Fine-Tuning. In this interpretation of the LLM as a policy, Supervised Fine-Tuning (SFT) can be seen as behavioural cloning, a form of imitation learning in RL, where the behaviour being cloned is that of the human who produced the original output. The policy is trained with the cross-entropy loss on batches of inputs concatenated with outputs, with the loss calculated only on the output tokens.

Reinforcement Learning from Human Feedback. Again following previous work I use PPO [Schulman et al., 2017] as the base RL algorithm. I initialise the model with the corresponding SFT model. I treat the language modelling head as the policy output, and learn a separate value function which takes the final hidden state of the language model and outputs a scalar using an MLP layer (an identical architecture to the reward model). I use a shared backbone for the policy and value function, with only the two heads being different. I train with the

reward function described in Appendix C.2.1, and use a KL divergence term as an auxiliary reward to ensure the language model stays close to the SFT model, as in prior work [Jaques et al., 2017, Stiennon et al., 2022, Ziegler et al., 2020]. The final reward for the policy is

$$R(x,y) = RM_{\theta_{RM}}(x,y) - \beta_{KL} D_{KL}(\pi_{\theta_{RL}}(y|x)||\pi_{\theta_{SFT}}(y|x)) \tag{C.1}$$

where RM denotes the reward model trained as described in Appendix C.2.1; θ_{RL}, θ_{RM} and θ_{SFT} are the parameters of the policy, reward model and SFT model respectively; x, y are the input and output; and β_{KL} is a hyperparameter that controls the weight of the KL penalty. I use $\beta_{KL} = 0.05$ throughout this work, following [Stiennon et al., 2022] and my own early experiments that found this choice struck a good balance between model performance and overoptimisation [Gao et al., 2023].

Best-of-N. Instead of using the RM to train a policy via PPO, the reward model can be used to filter samples from another model; this is called Best-of-N (BoN) sampling, and has been used in multiple previous works to achieve good performance [Menick et al., 2022, Nakano et al., 2022]. N summaries are sampled from the pretrained model, and then the RM is used to select the best one. This method removes the need to perform RL policy training, but makes inference time more computationally expensive, as we require N generations from the policy and N passes through the reward model to produce a single output. In this work I do BoN sampling on top of the SFT model, and mostly use $N = 16$, as that is what is used in previous works and strikes a good balance between improved performance and computational cost.

We summarise the core differences in training from [Stiennon et al., 2022] in Appendix C.3.

C.2.3 Model selection

For the results reported in the paper, I sweep over 3-5 learning rates for each model type, and select the best model on the validation set using an appropriate metric (accuracy for RMs, loss for SFT, reward for RL) - see Appendix C.2.4. For both the in-distribution and out-of-distribution results, I always use a validation set drawn from the same distribution as training. This more closely matches a real-world setting where I would not have access to OOD data to do model selection, and would have to draw model selection data from the same

Table C.2: Hyperparameters for SFT model training. These are fixed across all dataset splits and model sizes and types for summarisation.

Hyperparameter	Value
batch size	128
epochs	1
adam beta1	0.9
adam beta2	0.999
adam epsilon	1e-8
frozen layers	80%

Table C.3: Hyperparameters for RM training. These are fixed across all dataset splits and model sizes and types for summarisation.

Hyperparameter	Value
batch size	64
epochs	1
adam beta1	0.9
adam beta2	0.999
adam epsilon	1e-8
frozen layers	80%

distribution used in training.

C.2.4 Hyperparameters

Summarisation. For each model type (SFT, RM, RLHF) I do a sweep over learning rate, choosing ranges of values informed by choices in previous work [Stiennon et al., 2022] and early experimentation. The results in the paper are the best model with the learning rate chosen on an in-distribution validation set using loss, accuracy and reward respectively for SFT, RM and RLHF training. The learning rates for SFT are 3e-4, 1e-4, 3e-5, with 3e-5 selected; for RMs are 3e-4, 1e-4, 3e-5, 1e-5, 3e-6, with 3e-5 selected; for RLHF are: 1.5e-6, 3e-6, 6e-6, 1.5e-5, 3e-5, with 1.5e-5 selected.

We list the other hyperparameters (which are unchanged between all runs) for SFT, RM and RLHF training in Table C.2, Table C.3 and Table C.4 respectively. I chose these following prior work [Stiennon et al., 2022].

Instruction Following. For the instruction following results, I use the models released by [Dubois et al., 2023], and so the hyperparameters can be found in that work.

Table C.4: Hyperparameters for RLHF model training. These are fixed across all dataset splits and model sizes and types for summarisation. One PPO step consists of generating *batch size* samples, and then performing *ppo epochs* of optimisation on them, split into *ppo minibatch size* minibatches.

Hyperparameter	Value
batch size	256
ppo epochs	4
ppo steps	750
ppo minibatch size	256
KL penalty coefficient	0.05
normalise advantages	True
adam beta1	0.9
adam beta2	0.999
adam epsilon	1e-8
frozen layers	80%

C.3 Differences from Stiennon et al. (2022)

As I mostly follow [Stiennon et al., 2022] in the training of my summarisation models, I here describe the main differences between my work and theirs in terms of training. For RLHF, I train a single model with policy and value head, rather than separate policy and value functions. This is much more computationally efficient, and follows other recent work that still achieves impressive results [Glaese et al., 2022a]. This means that I randomly initialise the value function head, rather than intialising the value function from the reward model as is done in [Stiennon et al., 2022]. I use LLaMa [Touvron et al., 2023a] (and OPT [Zhang et al., 2022] in Appendix C.5) as my pretrained models, while they use unreleased models which were trained similarly to GPT-3 [Brown et al., 2020a].

We freeze the first 80% of the layers and the embedding and unembedding layers, as more recent work [Glaese et al., 2022a, Menick et al., 2022] has shown that good results can still be achieved, and training is much more computationally efficient. In Table C.12 I show the drop in performance for smaller OPT models between freezing 80% of the layers and not freezing. There is a drop of performance, but it is not catastrophic (equivalent to about a drop in model size among the three model sizes used), which justifies the tradeoff of training models with partially frozen weights.

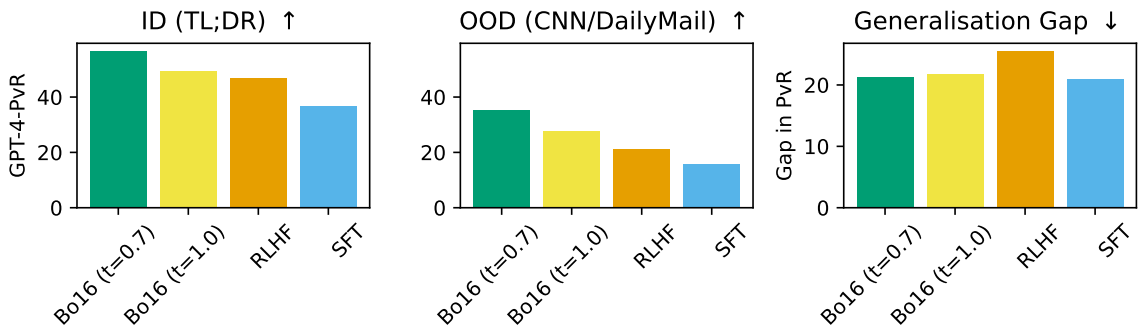


Figure C.3: GPT-4 API evaluation win rate vs reference summaries for SFT, Bo16 with two different temperatures, and RL models, trained on the summarisation task. In-distribution is performance on TL;DR, and out-of-distribution is on CNN/DailyMail. Effectively a version of Fig. 4.3 with addition Bo16 results with a worse-performing temperature

C.4 Best of N Temperature Experiment

In the main paper I report results for BoN using temperature 0.7. Here I show results for BoN with temperature 1, and show that temperature 0.7 performs better, hence my choice of it as the hyperparamter I use. Fig. C.3 shows the results of BoN with two temperatures, as well as RLHF and SFT for comparison.

C.5 Summarisation Experiments with OPT

In addition to the experiments in the main paper, I did several experiments using OPT [Zhang et al., 2022] in the summarisation task with a different choice of ID and OOD test sets.

C.5.1 Dataset Splitting

We create split versions of these datasets along several factors of variation in their inputs: *length*, *sentiment*, and *subreddit*. For each of these factors of variation, I create a train/test split where the train and test inputs are drawn from different parts of the distribution for that factor. For the *length* split, the training set consists of data where the post is less than 245 words (the median number of words in the SFT training distribution); for the *sentiment* split, I use an off-the-shelf sentiment classifier [Loria, 2020], and the training set consists of summaries with sentiment less than the median sentiment in the dataset; for the *subreddit* split, the training set consists of summaries from a specific subreddit, r/relationships. In all cases the test set is the complement of the training set in the full dataset, meaning that the trained models will be evaluated on inputs from a different distribution than the one seen

during training.

In all cases, I apply the same splitting procedure to both the preference data and the input/output pairs, to ensure that the training and test sets are consistent across the different methods. Each of these splits was chosen to produce a roughly 50-50 split between the training and testing distributions. In the case of *length* and *sentiment* this is exact, and in the case of *subreddit* the r/relationships subreddit contains approximately 60% of the data.

While I do not expect these splits to capture the full range of distribution shifts models may experience when deployed, using a range of splits will give us a more robust measure of how well the policies trained with different methods generalise under distribution shifts.

The dataset I use from [Stiennon et al., 2022] (filtered from [Nallapati et al., 2016]) comes with train, validation and test splits, which I use throughout my work. For the SFT dataset these splits have size 116722, 6553 and 6447 respectively, and for the RM dataset they have size 92858, 33083, 50718 respectively. The SFT and RLHF splits are the same apart from the RLHF dataset does not require the summaries (outputs), just the posts (inputs). To create the dataset splits used for the OOD generalisation experiments in Section 4.6.1, I split each of the train, validation and test sets into an in-distribution (ID) and out-of-distribution (OOD) train, validation and test set. I then train on the ID train set, do model selection using the ID validation set, and evaluate on the ID and OOD test sets to measure the in-distribution and out-of-distribution performance.

For the sentiment split, I first measure the sentiment of each post using an off-the-shelf sentiment classifier [Loria, 2020]. For a given subset of the dataset (i.e. train, validation, test), the ID version of that subset is the set of all inputs with posts whose sentiment is lower than the median sentiment, and the OOD version is the complement of that set. For the length split, I take the same approach using the length (in words) of the post, and the ID version of the subset is the set with posts of length less than the median length. For the relationships split, I take the ID version of the subset to be all posts in the r/relationships subreddit, and the OOD version to be the complement.

We apply this same splitting procedure to both the SFT and RM training datasets. Table C.5 and Table C.6 show the size of the training, validation, testing and OOD testing sets for the RLHF and SFT, and RM training, respectively. For the results in this work I randomly

Table C.5: Size of the train, validation, test and OOD test datasets for each split for the SFT and RLHF models.

	length	sentiment	relationships
Train	58770	58361	63324
Validation	3234	3223	3462
Test	3303	3276	3539
OOD Test	3250	3277	3014

Table C.6: Size of the train, validation, test and OOD test datasets for each split for the reward models.

	length	sentiment	relationships
Train	45395	46411	52346
Validation	16513	16529	17687
Test	25539	25353	27492
OOD Test	25180	25366	23227

sample from the test and OOD test sets for those metrics, and I randomly sample from the validation set (which is in-distribution) for calculating metrics used for model selection.

To understand the distribution shifts these splits entail, I show density plots for post length and sentiment across the full SFT and RM dataset in Fig. C.4 and Fig. C.5, and show the number of posts in each subreddit in Fig. C.6.

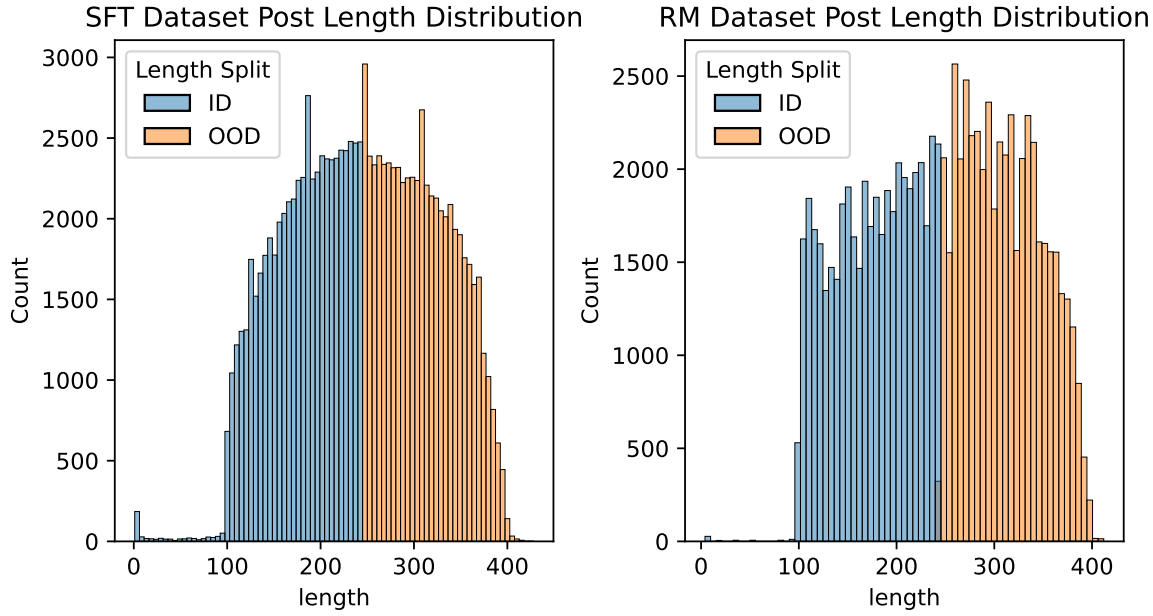


Figure C.4: The distribution of post lengths across the full SFT and RM datasets. ID is the in-distribution version of the dataset, and OOD is the out-of-distribution version.

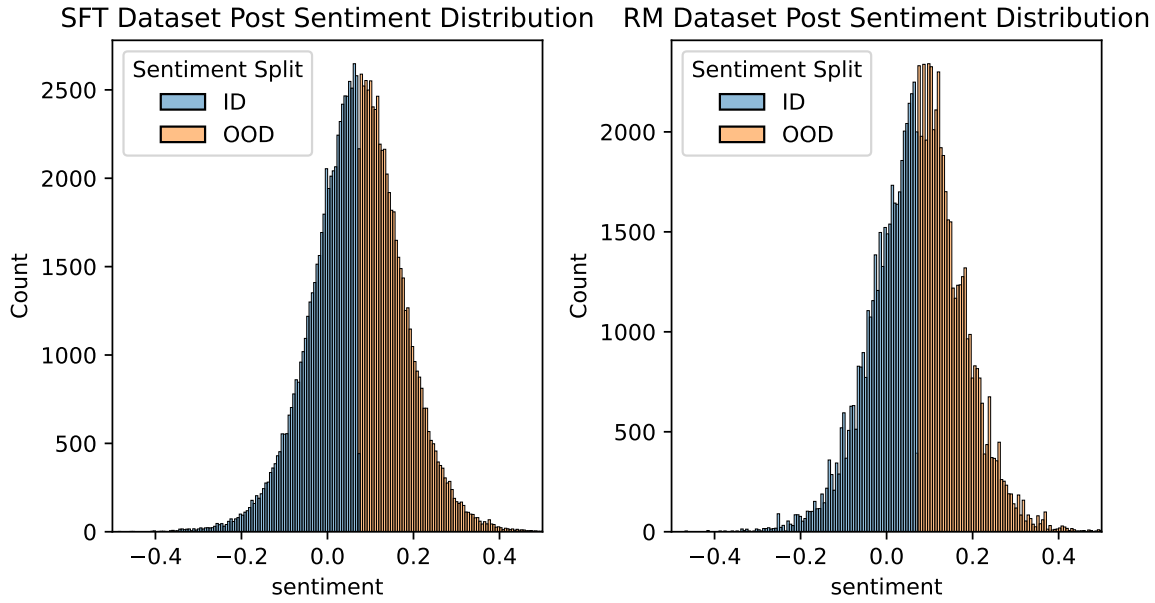


Figure C.5: The distribution of post sentiment across the full SFT and RM datasets. ID is the in-distribution version of the dataset, and OOD is the out-of-distribution version.

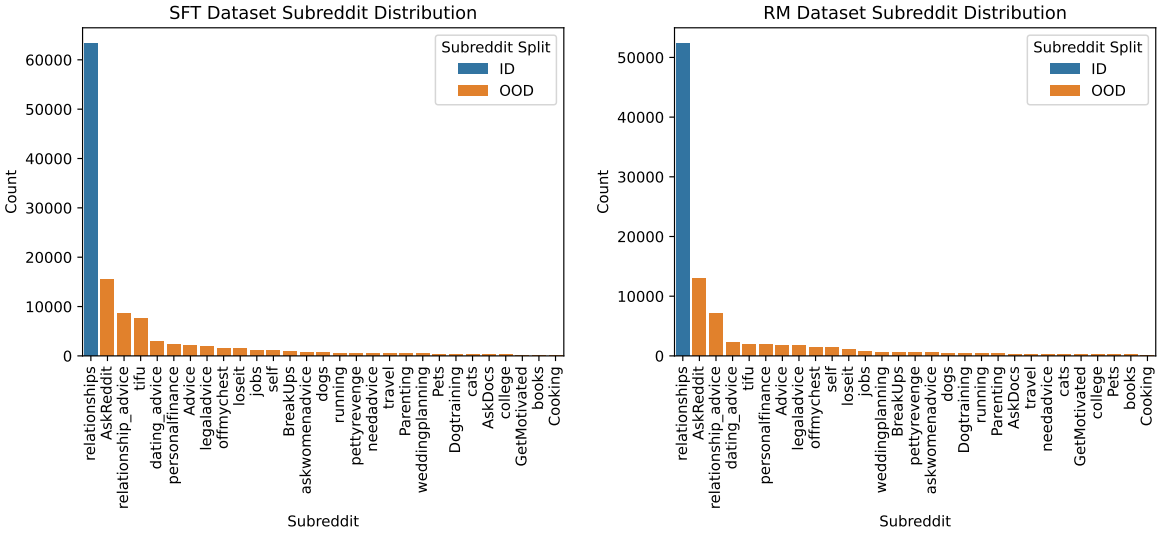


Figure C.6: The number of posts in each subreddit across the full SFT and RM datasets. ID is the in-distribution version of the dataset, and OOD is the out-of-distribution version.

Table C.7: Learning rates for different model sizes and dataset splits for SFT models. Underlined learning rate is the chosen one. ke^{-n} means $k \times 10^{-n}$.

Dataset	125m	350m	1.3b	2.7b	6.7b
relationships	$\underline{1e^{-4}}$,$3e^{-5}$,$1.5e^{-5}$	$\underline{1e^{-4}}$,$3e^{-5}$,$1.5e^{-5}$	$\underline{1e^{-4}}$,$3e^{-5}$,$1.5e^{-5}$	$\underline{1e^{-4}}$,$3e^{-5}$,$1.5e^{-5}$	$1e^{-4}$,$3e^{-5}$,$\underline{1.5e^{-5}}$
length	$1e^{-4}$,$3e^{-5}$,$\underline{1.5e^{-5}}$	$\underline{1e^{-4}}$,$3e^{-5}$,$1.5e^{-5}$	$1e^{-4}$,$3e^{-5}$,$\underline{1.5e^{-5}}$	$1e^{-4}$,$3e^{-5}$,$\underline{1.5e^{-5}}$	$\underline{1e^{-4}}$,$3e^{-5}$,$1.5e^{-5}$
sentiment	$\underline{1e^{-4}}$,$3e^{-5}$,$1.5e^{-5}$	$1e^{-4}$,$\underline{3e^{-5}}$,$1.5e^{-5}$	$1e^{-4}$,$3e^{-5}$,$\underline{1.5e^{-5}}$	$\underline{1e^{-4}}$,$3e^{-5}$,$1.5e^{-5}$	$\underline{1e^{-4}}$,$3e^{-5}$,$1.5e^{-5}$

Table C.8: Learning rates for different model sizes and dataset splits for reward models. Underlined learning rate is the chosen one. ke^{-n} means $k \times 10^{-n}$.

Dataset	125m	350m	1.3b	2.7b	6.7b
relationships	$\underline{5e^{-4}}$,$1.5e^{-5}$,$5e^{-5}$	$\underline{5e^{-4}}$,$1.5e^{-5}$,$5e^{-5}$	$\underline{5e^{-5}}$,$1.5e^{-5}$,$5e^{-6}$	$5e^{-5}$,$\underline{1.5e^{-5}}$,$5e^{-6}$	$\underline{5e^{-5}}$,$1.5e^{-5}$,$5e^{-6}$
length	$\underline{5e^{-4}}$,$1.5e^{-5}$,$5e^{-5}$	$5e^{-4}$,$1.5e^{-5}$,$\underline{5e^{-5}}$	$5e^{-5}$,$\underline{1.5e^{-5}}$,$5e^{-6}$	$5e^{-5}$,$\underline{1.5e^{-5}}$,$5e^{-6}$	$5e^{-5}$,$1.5e^{-5}$,$\underline{5e^{-6}}$
sentiment	$5e^{-4}$,$\underline{1.5e^{-5}}$,$5e^{-5}$	$5e^{-4}$,$1.5e^{-5}$,$\underline{5e^{-5}}$	$\underline{5e^{-5}}$,$1.5e^{-5}$,$5e^{-6}$	$5e^{-5}$,$\underline{1.5e^{-5}}$,$5e^{-6}$	$\underline{5e^{-5}}$,$1.5e^{-5}$,$5e^{-6}$

C.5.2 Hyperparameters

We use the same hyperparameters as in LLaMa training (see Appendix C.2.4), but sweep over learning rates for each model size. I detail the learning rates swept over for each model size and the chosen learning rate, for SFT, RM and RLHF training in Table C.7, Table C.8 and Table C.9 respectively. In general for SFT and RLHF I did not see much variance with seeds, but I did in RM training, matching prior work [Stiennon et al., 2022]. For the RLHF training with the largest two model sizes, due to the large amount of compute required to run multiple training runs, for several model size and dataset shift combinations I chose a single learning rate based on what I thought would give the best results at the time I started training. For the combinations where I did vary the learning rate I did not see much variation in performance on the metrics I measured, so I do not expect these choices to affect the results.

C.5.3 Generalisation Evaluation

For evaluation in these experiments, I train an RM as described in Appendix C.2.1 using the full dataset of summaries and preferences without splitting and the 6.7 billion parameter OPT model. I then use this *proxy RM* to evaluate the performance of SFT, BoN and RLHF

Table C.9: Learning rates for different model sizes and dataset splits for RLHF models. Underlined learning rate is the chosen one. ke^{-n} means $k \times 10^{-n}$.

Dataset	125m	350m	1.3b	2.7b	6.7b
sentiment	$1e^{-4}$,$3e^{-5}$,$\underline{1e^{-5}}$	$1e^{-4}$,$3e^{-5}$,$\underline{1e^{-5}}$	$\underline{1e^{-5}}$,$3e^{-6}$,$1e^{-6}$	$\underline{3e^{-5}}$	$\underline{1e^{-5}}$
length	$1e^{-4}$,$3e^{-5}$,$\underline{1e^{-5}}$	$1e^{-4}$,$3e^{-5}$,$\underline{1e^{-5}}$	$\underline{1e^{-5}}$,$3e^{-6}$,$1e^{-6}$	$\underline{3e^{-5}}$	$\underline{5e^{-6}}$
relationships	$1e^{-4}$,$3e^{-5}$,$\underline{1e^{-5}}$	$1e^{-4}$,$3e^{-5}$,$\underline{1e^{-5}}$	$\underline{1e^{-5}}$,$3e^{-6}$,$5e^{-6}$	$\underline{5e^{-6}}$,$3e^{-6}$,$1.7e^{-6}$	$\underline{5e^{-6}}$,$3e^{-6}$,$1.7e^{-6}$

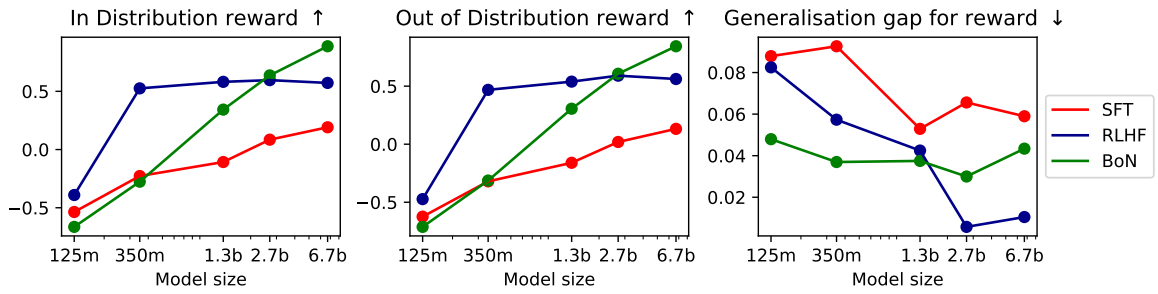


Figure C.7: Proxy RM Score for SFT, BoN and RL models, averaged over dataset splits, for both in-distribution and out-of-distribution performance, and the generalisation gap. Arrows ↑,↓ indicate whether higher or lower scores are better.

models. As this reward model is trained with a different random seed and a different data distribution, it serves as a held-out automated evaluation of models, and a good proxy for human preferences.

We first discuss the results from the OPT version of the generalisation experiments described in Section 4.5.1. Fig. C.7 shows the proxy RM score for SFT, BoN and RLHF models, averaged over dataset sizes. The important result here is that BoN generalises better than RLHF, which generalises better than SFT.

We see that at middling model sizes, RLHF outperforms BoN, but BoN scales better than RLHF, eventually outperforming it for models with more than 2.7b parameters. SFT comes out worst in this comparison, both scaling worse than BoN and the same as RLHF, and having worse absolute performance and generalisation. I see that BoN sampling does not see diminishing returns as model size increases, implying it will continue to be a useful yet simple technique. This experiment highlights the importance of training a reward model on human feedback and using it to select the best outputs at test time, potentially after fine-tuning the model.

Supervised Fine-Tuning. Fig. C.8 shows the proxy RM score for the SFT models. Performance increases as model size increases, and in general performance drops OOD, which is unsurprising. There is a slight downward trend in the average generalisation gap across dataset splits as model size increases, implying that larger models fine-tuned with SFT generalise better (in terms of automated metrics). The relationships and sentiment splits produce negligible generalisation gap for the proxy RM score while the length split is more difficult.

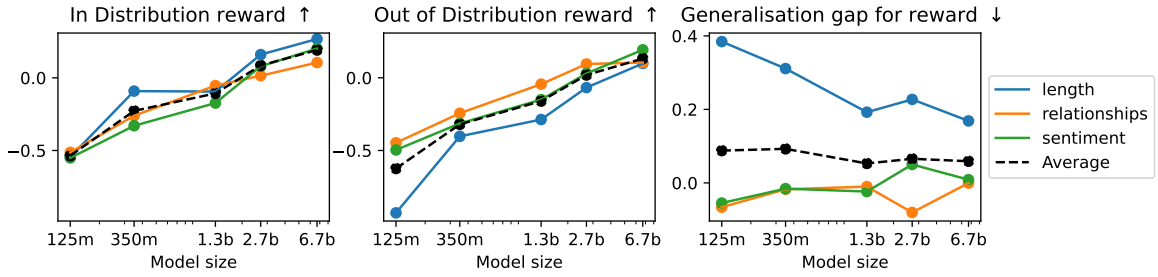


Figure C.8: Proxy RM Score for SFT models for each dataset split, both in-distribution and out-of-distribution performance, and the generalisation gap. Arrows ↑,↓ indicate whether higher or lower scores are better.

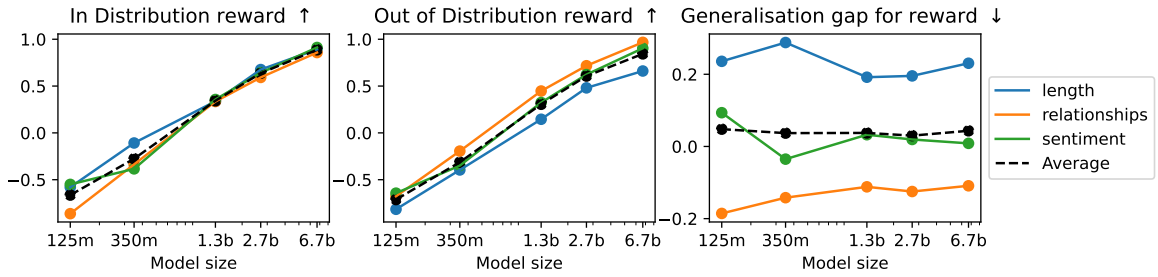


Figure C.9: Proxy RM Score for BoN models for each dataset split, both in-distribution and out-of-distribution performance, and the generalisation gap. Arrows ↑,↓ indicate whether higher or lower scores are better.

Best-of-N. Fig. C.9 shows the proxy reward score for the BoN models. Here we can see a smooth almost linear increase in performance as the model size increases. Given that the RM scores for smaller models were below chance, the fact that even for smaller models BoN results improve performance requires explanation. I hypothesise that the smooth increase here comes from two factors: the improvement in the SFT model being sampled from, and the improvement from the RM. At smaller model sizes increasing the number of parameters leads to improvements in SFT performance but not RM performance, while at larger model sizes increasing the number of parameters leads to improvements in both SFT and RM performance, but both to a lesser extent.

Fig. C.10 shows the improvement of BoN over SFT. We can see that BoN only starts improving SFT as model size passes 350 million parameters, and the improvement grows as model size grows. This implies that as I increase model size BoN is likely to become a more performant choice compared to SFT. Further, I note that BoN uniformly improves the generalisation across model sizes (as shown by the black dashed line), implying that scaling

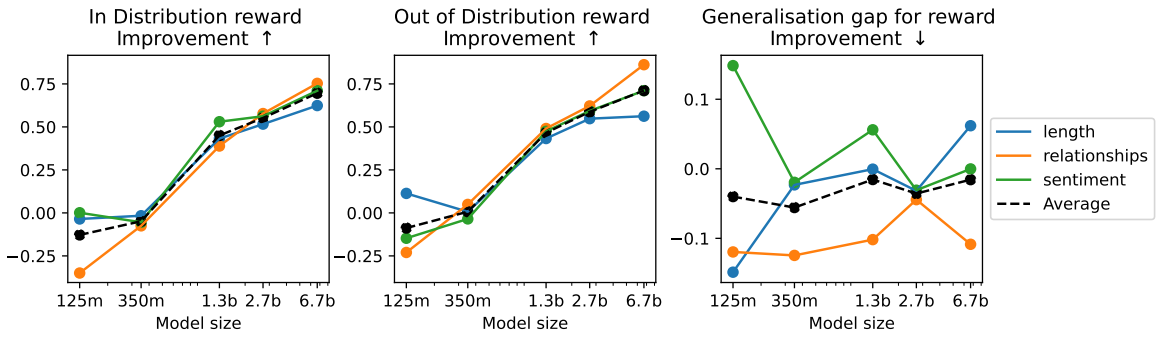


Figure C.10: Proxy RM Score improvement using BoN on top of SFT models for each dataset split, both in-distribution and out-of-distribution performance, and the generalisation gap. Arrows ↑,↓ indicate whether higher or lower scores are better. This plot is highlights the improvement from Fig. C.8 to Fig. C.9.

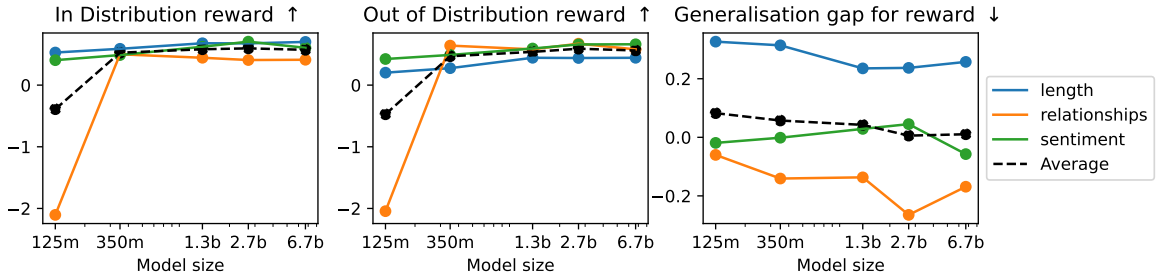


Figure C.11: Proxy RM Score for RL models for each dataset split, both in-distribution and out-of-distribution performance, and the generalisation gap. Arrows ↑,↓ indicate whether higher or lower scores are better.

models, even using SFT+BoN, will still result in a non-zero generalisation gap.

Reinforcement Learning from Human Feedback. Fig. C.11 shows the proxy RM score for the RLHF models. Again, we see that increasing model size improves performance. Here we see a clearer trend of increasing model sizes reducing generalisation gap – this implies that as we make RL models larger, they are likely to generalise better. Given the difference between this trend and the generalisation gap trend for SFT models in Fig. C.8, this partially justifies why RLHF is used in fine-tuning LLMs at a very large scale [Glaese et al., 2022a, OpenAI, 2023, Ouyang et al., 2022a]: RLHF produces better-generalising models at larger model sizes than SFT.

C.5.4 Diversity Evaluation

Table C.10 shows the per-input diversity scores (Eq. (4.2)) for both RLHF and SFT models. The RLHF models have much lower diversity than SFT models according to all three metrics.

Table C.10: Per-input diversity scores for both RLHF and SFT models averaged over dataset splits. For these scores the outputs used to calculate the diversity are a sample of outputs from the model for single input. These per-input scores are then averaged, as in Eq. (4.2). Bolded results are better scores for each model size.

Model Size	125m		350m		1.3b		2.7b		6.7b	
Model Type	RLHF	SFT	RLHF	SFT	RLHF	SFT	RLHF	SFT	RLHF	SFT
EAD	0.15	0.81	0.15	0.8	0.13	0.81	0.16	0.8	0.07	0.79
Sent BERT	0.15	0.5	0.12	0.46	0.15	0.48	0.13	0.45	0.06	0.45
NLI	-1.08	0.26	-0.96	0.2	-1.0	0.12	-1.07	0.09	-1.54	0.06
Average	-0.26	**0.52**	-0.23	**0.49**	-0.24	**0.47**	-0.26	**0.45**	-0.47	**0.44**

Table C.11: Across-input diversity scores for both RLHF and SFT models averaged over dataset splits. For these scores the outputs used to calculate the diversity are a set of single outputs from a range of inputs, as in Eq. (4.3). Bolded results are better scores for each model size.

Model Size	125m			350m			1.3b			2.7b			6.7b		
Model Type	RLHF	SFT	BoN	RLHF	SFT	BoN	RLHF	SFT	BoN	RLHF	SFT	BoN	RLHF	SFT	BoN
EAD	0.69	0.86	0.85	0.87	0.87	0.86	0.86	0.87	0.86	0.86	0.87	0.86	0.87	0.87	0.86
Sent BERT	0.63	0.73	0.72	0.71	0.73	0.71	0.68	0.73	0.71	0.7	0.74	0.71	0.73	0.73	0.71
NLI	0.05	0.35	0.32	0.2	0.38	0.3	0.25	0.36	0.27	0.3	0.28	0.27	0.32	0.32	0.3
Average	0.46	**0.64**	0.63	0.59	**0.66**	0.62	0.6	**0.65**	0.61	0.62	0.63	0.62	0.64	0.64	0.62

While RLHF leads to better-generalising policies, those policies generate much less diverse outputs. We also see that diversity does not seem to change much with model size, apart from a slight downward trend for diversity in SFT models as model size increases.

Table C.11 shows the across-input diversity scores (Eq. (4.3)). Here the corpus of text over which the diversity is measured is a single input sampled from the model for a range of outputs. Even if a model produces less diverse outputs for a single input, it could still produce different inputs for different outputs. This is the case for the EAD score, which is a proxy for diverse vocabulary and syntax, as for models with more than 125 million parameters both SFT and RLHF produce policies with very similar scores.

However, for the SentBERT score, which is a proxy for diverse content and semantics, RLHF models are consistently less diverse than SFT models. This implies that RLHF produces models that have a tendency to generate outputs about certain topics or content regardless of the input. For the NLI score, which is a proxy for logical diversity, we see that as RLHF model size increases the score increases, eventually reaching the diversity of SFT models. Low NLI score for smaller models implies they have a tendency to make logically consistent

Table C.12: SFT Model ROUGE1 and reward model accuracy for the 3 smallest OPT model sizes, comparing freezing 80% of layers vs no layer freezing. Freezing generally results in less performance, as expected.

Model Size	SFT Model Rouge1		RM Accuracy	
	80% Frozen	0% Frozen	80% Frozen	0% Frozen
125m	0.217	0.221	0.482	0.496
350m	0.2241	0.2233	0.498	0.508
1.3b	0.221	0.2347	0.538	0.559

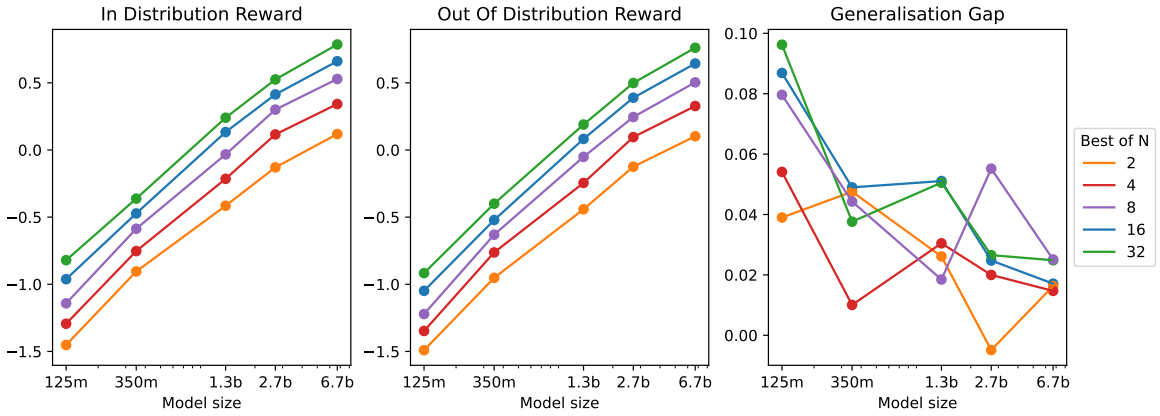


Figure C.12: Proxy RM score for BoN sampling with varying N. All metrics are averaged over the 3 dataset splits.

claims in their outputs on top of producing outputs about certain topics of content.

C.5.5 Model Freezing Experiments

We perform a small experiment to evaluate the effects of freezing the first 80% of layers during fine-tuning. The results are shown in Table C.12, and show that while performance drops for the models evaluated in the experiment, the drop is not catastrophic, justifying the use of model freezing.

C.5.5.1 BoN Performance for Different N

In the previous OPT results I use $N = 64$ samples in best of N sampling. Here I show proxy RM scores for $N = 2, 4, 8, 16, 32$, to show how choices of N trade off against performance. Fig. C.12 shows proxy RM score in-distribution, out-of-distribution and the generalisation gap for these different choices of N. I see that increasing N does lead to improved performance. At lower model sizes it leads to a larger generalisation gap, but this does not hold for larger model sizes.

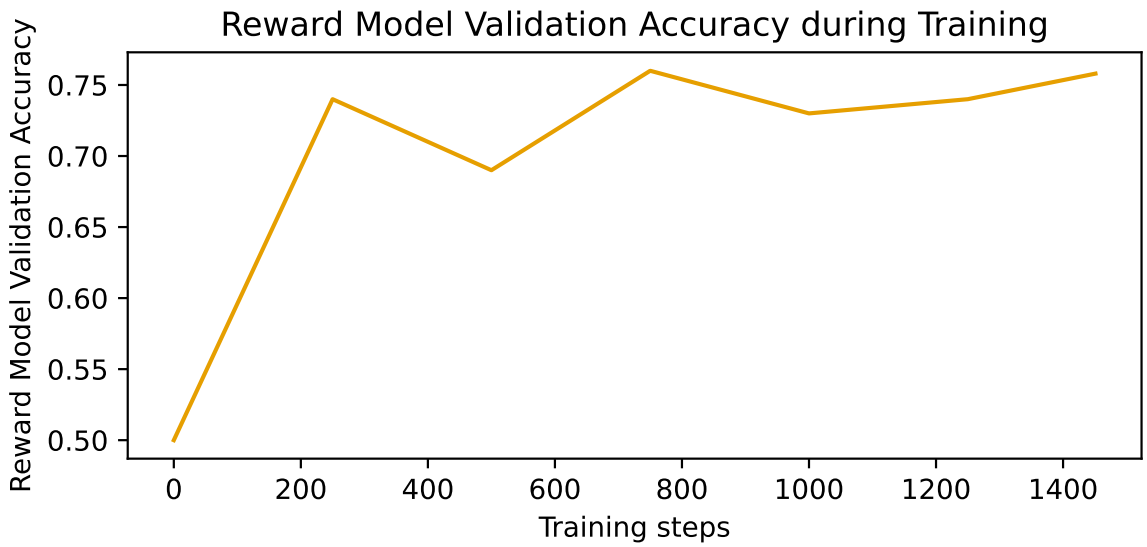


Figure C.13: Reward model validation accuracy during training for the summarisation task.

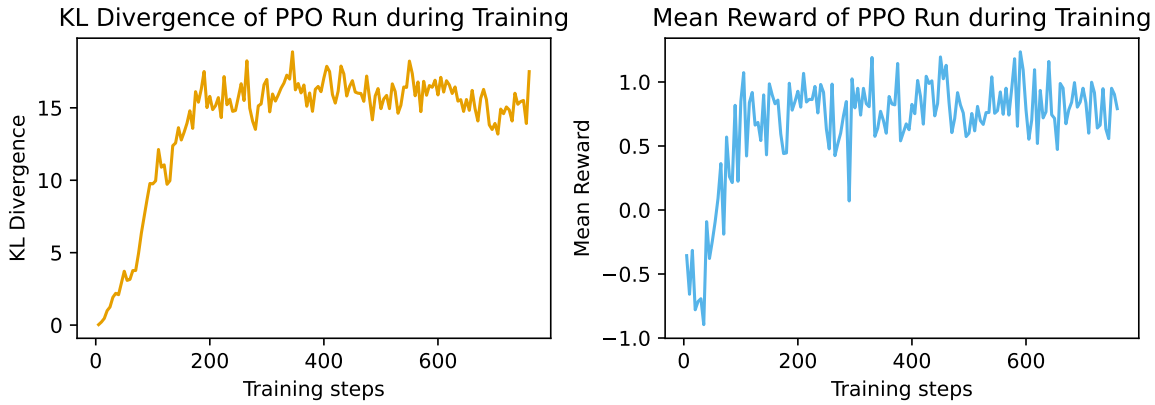


Figure C.14: PPO KL divergence and reward model score curves for the summarisation task.

C.6 RLHF and RM Training Curves

Here I present training curves for PPO and reward model training, for the summarisation task, for the models used in the main paper. In Fig. C.13 I show the reward model validation accuracy throughout training. This is 1 epoch of training. In Fig. C.14 I show the KL divergence and reward model score throughout training. PPO training has converged by approximately 250 PPO training steps, and so I terminated training early to save compute.

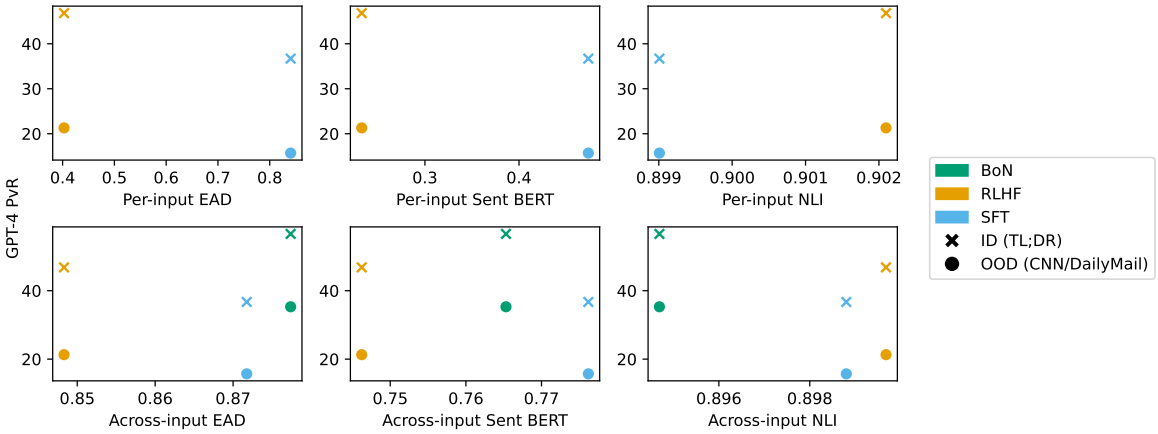


Figure C.15: We plot the diversity vs gpt4 win rate trade-off for the summarisation task, across in-distribution and out-of-distribution winrates and per- and across-input diversity metrics.

C.7 Generalisation vs Diversity Trade-off Plots

Fig. C.15 shows the tradeoff between diversity and win rate in the summarisation task, across the three policy types I investigate. This reinforces the inherent tradeoff between generalisation and diversity present in existing language model fine-tuning techniques.

Appendix D

Appendices to Mechanistically analysing the effects of fine-tuning on procedurally defined tasks

D.1 Organisation of Appendix

In the appendix I present a comprehensive analysis of my claims on Tracr, PCFG and TinyStories-Instruct using different mechanistic interpretability tools discussed in Ap-

pendix D.4. I also present a summary of the notations used in this chapter in Table D.1. Overall, the appendix is organised as follows:

- Appendix D.2 presents details of the Tracr and PCFG datasets utilised in this work.
- Appendix D.3 presents the training and model details for each of the datasets considered.
- Appendix D.4 lists the protocols used for different mechanistic interpretability tools like attention maps, probing, pruning and reverse fine-tuning.
- Appendix D.5 provides a few more results in practically relevant contexts, such as in a synthetic jailbreaking setup.
  - Appendix D.5.1 studies the effect of using different fractions of pretraining and fine-tuning data points for fine-tuning.
  - Appendix D.5.2 presents the jailbreaking analysis using the PCFG setup.
  - Appendix D.5.3 shows reverse fine-tuning a fine-tuned model is sample efficient compared to baselines for both PCFG and Tracr models.
  - Appendix D.5.4 presents reverse fine-tuning analysis of a fine-tuning protocol that actively tries to remove a capability from PCFG / Tracr models.
- Appendix D.6 presents additional results on the TinyStories setting.
- Appendix D.7 presents additional results on Tracr for counter and max element tasks.
- Appendix D.8 presents additional results on PCFG for the counting and index of occurrence tasks.

Table D.1: Notations used in this work.

Notation	Meaning
X	Input domain
X_D	Factor of the input domain that captures values of the inputs
X_I	Factor of the input domain that captures task identifiers
$\mathscr{P}_{\mathrm{X}}$	Probability distribution over the input domain
$\mathscr{P}_{\mathrm{X}}^{\mathrm{FT}}$	The overall distribution defining the downstream fine-tuning task
$\mathscr{D}_{\mathrm{FT}}$	Dataset used for fine-tuning
$\mathscr{P}_{\mathrm{X}}^{\mathrm{FT,E}}$	Empirical distribution from which the fine-tuning dataset is sampled
$\mathtt{T}$	Denotes a task to be performed by the model (e.g., count)
$\mathtt{O}$	Denotes an operand that will be processed by to perform the task $\mathtt{T}$
$\{\mathtt{O}_{\mathrm{PT}}\}$	Set of operand tokens seen during pretraining
$\mathtt{O}_{\mathrm{PT}}$	A specific token used as an operand during pretraining
$\mathtt{O}_{\mathrm{FT}}$	A specific token used as an operand during fine-tuning
$\mathtt{r}(x, \mathtt{O})$	Denotes the result of executing a task from Section 5.4 on a string x for some operand $\mathtt{O}$
$C_{\mathtt{Tr}}$	Probability that a randomly sampled string in the training data used for fine-tuning has a spurious correlation between the pretraining capability and the downstream task
$C_{\mathtt{Te}}$	Probability that a randomly sampled string in the test data used for evaluating fine-tuned models has a spurious correlation between the pretraining capability and the downstream task
$\mathscr{P}_{\mathtt{T}}(\mathtt{O})$	Sampling prior. Denotes the probability that when a string with task token $\mathtt{T}$ is sampled during pretraining, the operand to perform the task on is $\mathtt{O}$
$\mathscr{P}_C^H$, $\mathscr{P}_C^M$, $\mathscr{P}_C^S$	Sampling priors such that the probability of sampling the target token for fine-tuning ($\mathtt{O}_{\mathrm{FT}}$) is high ($\mathscr{P}_C^H$), medium ($\mathscr{P}_C^M$), or small ($\mathscr{P}_C^S$)
η_M, η_S, η_{VS}	Medium / Small / Very-small learning rates used for fine-tuning. η_{VS} is only used for a specific reverse fine-tuning experiment with Tracr compiled models.
$\mathtt{reFT}$	Denotes reverse fine-tuning
$n_{\mathtt{iters}}$	Number of iterations used during pre-training
LR	Learning rate

D.2 Additional details on Datasets

We consider three experimental setups: Compiled programs with Tracr [Lindner et al., 2023], learning models on Probabilistic Context Free Grammars (PCFG) [Allen-Zhu and Li, 2023c], and the TinyStories Instruct dataset.

D.2.1 Tracr Details

Tracr [Lindner et al., 2023] generates a transformer model using the RASP library by Weiss et al. [2021]. The specific code snippet used to generate the Tracr models for the counting and the max element tasks are shown in Algorithm 1 and Algorithm 2 respectively. The models corresponding to these tasks is implemented with three standard transformer blocks, where each block consists of a self-attention layer followed by two MLP layers.

We analyse the following two tasks to understand the effects of fine-tuning on a pretrained model's capabilities.

- **Counter:** Compile the capability to count the number of occurrences of a token $\mathtt{0}_{\mathtt{PT}}$ in a string into the model; fine-tune to count occurrences of another token $\mathtt{0}_{\mathtt{FT}}$. If $\mathtt{r}(x, \mathtt{0})$ denotes the number of occurrences of a token $\mathtt{0}$ in a string x, the spurious correlation is defined by enforcing a constant difference in token occurrences, i.e., $\mathtt{r}(x, \mathtt{0}_{\mathtt{FT}}) - \mathtt{r}(x, \mathtt{0}_{\mathtt{PT}}) = \mathtt{q}$. See also Algorithm 1 and Fig. D.1.
- **Max-identifier:** Compile the capability to identify the $\mathtt{0}_{\mathtt{PT}}$-th largest element in a string; fine-tune to identify the $\mathtt{0}_{\mathtt{FT}}$-th largest element. If $\mathtt{r}(x, \mathtt{0})$ reads out the $\mathtt{0}$-th largest token in the string x, I define the spurious correlation as $\mathtt{r}(x, \mathtt{0}_{\mathtt{FT}}) - \mathtt{r}(x, \mathtt{0}_{\mathtt{PT}}) = \mathtt{q}$; e.g., if $q = 1$ and the $\mathtt{0}_{\mathtt{PT}}$ largest token in the string x is $\mathtt{a}$, then the $\mathtt{0}_{\mathtt{FT}}$-th largest token will be $\mathtt{b}$ (which is equal to $\mathtt{a} + 1$ in Tracr's vocabulary). See also Algorithm 2 and Fig. D.2.

The fine-tuning data is generated by randomly sampling tokens from a uniform distribution over the input vocabulary. For the Counter task, the input vocabulary consists of first nine letters from the English alphabet. For the max element task, the input vocabulary consists of all the letters in the English alphabet. We sample with replacement for the Counter task and without replacement for the max element task (to avoid having multiple max elements). Examples for the task are shown in Fig. D.1,Fig. D.2.

Algorithm 1: Pseudocode for compiling the Counter capability via Tracr: Rasp code used to generate the model for the Counter capability and task via Tracr

```
def countA():
    # binzarize the tokens into 0's and 1's
    bin = (rasp.tokens==‘a’)
    # Select the indices of tokens with value of 1
    bin_idx = rasp.Select(bin, rasp.indices, rasp.Comparison.EQ)
    # Count the number of selected indices
    count_a = rasp.SelectorWidth(bin_idx)
    # Generate an identity map
    idx_select = rasp.Select(rasp.indices, rasp.indices, rasp.Comparison.EQ)
    # Output the count
    sum = rasp.Aggregate(idx_select, count_a)
```

Task: Count b
Sample: $, c, a, d, a, b, c, b, a, d, f, b, g, c, e, b, b, a, h, j, i, b, d, e, f, ,i, h, f, e, g, a, b, g, f, h, j, c, b, e, d, d, h, j, i, b, a, b, #,
Answer: 10

Figure D.1: Exemplar for Counter Task: A sample used for fine-tuning Tracr compiled models on counting ‘b’.

D.2.2 PCFG

We follow Allen-Zhu and Li [2023c] and use the production rules shown in Fig. D.3. We sample a string of tokens from the grammar and then randomly subsample a string of 250 tokens from the generated original sequence (this helps remove bias towards tokens likely to be at the beginning). The sampling probabilities to sample a valid rule given the parent node is fixed to be 0.5. We formulate the training data as follows: Start of sequence token (SOS) + Task family token (e.g., Counting) (T) + Operand one (counting what) (O) + Operand two (number of positions) (O') + Start of text token (*SOT*) + Data generated from DGP (Txt) + End of text token (EOT) + Answer request token (ART) + Answer token (Ans) + End of sequence token (EOS). This can be summarised as follows.

$$\textbf{Sample input: } \texttt{SOS} + \texttt{T} + \texttt{O} + \texttt{O}' + \texttt{SOT} + \texttt{Txt} + \texttt{EOT} + \texttt{ART} + \texttt{Ans} + \texttt{EOS}. \tag{D.1}$$

I consider the following tasks for pretraining:

- **Counting (C):** Counting number of O (say a) for the last O' positions (forty). This example will be written as Ca40.

Algorithm 2: Pseudocode for compiling the Max identifier capability via Tracr: Rasp code used to generate the model for the Max Identifier capability and task via Tracr.

```
def max_identifier():
    # Identify the tokens larger than a given token
    var_small = rasp.Select(rasp.tokens, rasp.tokens, rasp.Comparison.LT)
    # Calculate the sum of the identified tokens for every input token
    sum_small = rasp.SelectorWidth(var_small)
    # Identify the fifth largest token
    bin_target = (sum_small==4)
    # Find the index of the identified token in the original input
    select_idx = rasp.Select(bin_target, rasp.indices, rasp.Comparison.EQ)
    # Output the identified index
    return rasp.Aggregate(select_idx, rasp.tokens)
```

Task: Find fifth largest element
Sample: $, b, d, a, f, h, m, x, p, q, n, #, #, #, #
Answer: 'h'

Figure D.2: Exemplar for Max-Element Task: A sample used for fine-tuning Tracr compiled models on the Max identifier task.

- **Counting composition of elements (CC):** Counting number of 0 (say aa) for the last 0′ positions (forty). This example will be written as CCa40.
- **Index of occurrence (I):** Index from the EOT token when 0 (say a) occurred for the $0'^{th}$ time (sixth). This example will be written as Ia10.
- **Index of occurrence of composition element (IC):** Index from the EOT token when 0 (say aa) occurred for the $0'^{th}$ time (sixth). This example will be written as ICa10.
- **Token value at an index (T):** The token value at index 0′ (forty) before the end token.

s → r, q; s → q, p; p → m, n, o; p → n, o, m; q → n, m, o;
q → m, n; r → o, m; r→ m, o, n; m → l, j; m → j, l, k;
n → k, j, l; n → l, j, k; o →l, k, j; o → k, j; j → h, i;
j → i, h; k → h, g, i; k → g, h, i; l → i, h, g; l → h, i, g;
g → d, f, e; g → f, e, d; h → e, d, f; h → d, e, f; i → e, f, d; i → f, d, e;
d → c, a; d → a, b, c; e → c, b; e → c, a, b; f → c, b, a; f → b, a;

Figure D.3: PCFG setup: Grammar rules considered to generate the PCFG dataset. The highlighted token represents the parent token. These rules have been adapted from Allen-Zhu and Li [2023c].

$\mathtt{O}$ is NULL here. This example will be written as TNULL5.

For the "Counting", "Counting composition of elements", and "Token value at index" tasks, I set the value of $\mathtt{O}'$ token as 40. For "Index of occurrence" and "Index of occurrence of composition element" task, I set the value of $\mathtt{O}'$ token as 6. All five tasks above are considered during pretraining, but for fine-tuning I consider only a single task with a given operand. Specifically, I analyse fine-tuning the pre-trained models on the "Counting" and "Index of occurrence" tasks only.

We analyse the following two tasks to understand the effects of fine-tuning on a pretrained model's capabilities.

- **Counter:** I intentionally reuse this task to demonstrate the effects of compilation of the capability via Tracr versus learning the capability via PCFGs. Instead of being compiled, the model is trained to count the number of tokens from a *set* of tokens $\{\mathtt{O}_{\mathtt{PT}}\}$. The model is then fine-tuned to exclusively count a $\mathtt{O}_{\mathtt{FT}} \in \{\mathtt{O}_{\mathtt{PT}}\}$ token. By making the sampling probability of $\mathtt{O}_{\mathtt{FT}}$ tokens high during pretraining, I can make the model preemptively performant on the downstream task; this allows us to model the notion of capability relevance.
- **Indexer:** Among other tasks, the model is pretrained to output the index (location in a string) of a token from the *set* $\{\mathtt{O}_{\mathtt{PT}}\}$ occurs for the $\mathtt{k}^{\text{th}}$ time; fine-tuning is performed to output the index of $\mathtt{k}^{\text{th}}$ occurrence of another token $\mathtt{O}_{\mathtt{FT}}$ instead. I arbitrarily set $\mathtt{k}$ to 6 for my experiments, but emphasize that any integer less than context size can be used. If $\mathtt{r}(x, \mathtt{O})$ denotes the index of $\mathtt{k}^{\text{th}}$ occurrence of a token $\mathtt{O}$ in a string x, the spurious correlation is enforced via constant offset $\mathtt{q}$ in operand token indices, i.e., $\mathtt{r}(x, \mathtt{O}_{\mathtt{FT}}) - \mathtt{r}(x, \mathtt{O}_{\mathtt{PT}}) = \mathtt{q}$.

While the pretraining dataset is generated by simply sampling from PCFG (see Fig. D.4 for an example), for generating the fine-tuning dataset I provide explicit control over the value of $C_{\mathtt{Tr}}$ by artificially adding the target tokens $\mathtt{O}_{\mathtt{FT}}$ from the fine-tuning task. It is important to ensure that the distribution shift between the fine-tuning distributions with different values of $C_{\mathtt{Tr}}$ is minimised, and the data is fairly spread across multiple classes to enable reusability of feature via pretraining. As shown in Fig. D.5, the class distribution of the datasets with $C_{\mathtt{Tr}} = 1$ and $C_{\mathtt{Tr}} = 0$ for the counting and the index of occurrence tasks satisfies these requirements.

Task Family Token `T`**:**: '('
Operand Token `O`**:**: 'a'
Sample: $, (, a, 40, <, c, a, b, a, c, a, b, a, a, a, c, b, c, b, b, b, a, b, c, a, c, b, c, a, a, c, a, c, a, a, c, c, a, b, a, c, b, b, a, a, a, c, b, c, b, b, c, a, a, c, b, c, b, c, b, a, c, b, c, b, a, c, c, b, b, a, c, c, b, a, a, a, b, a, c, b, b, a, a, a, c, b, c, b, b, c, a, a, c, b, c, b, c, b, a, c, b, c, b, a, c, c, b, b, a, c, c, b, a, a, a, b, a, c, b, b, a, a, a, c, b, c, b, b, c, a, a, c, b, c, b, c, b, a, a, a, b, b, a, b, b, a, b, a, b, b, c, b, a, c, c, c, b, a, c, a, c, b, a, c, c, b, c, b, b, a, a, a, c, a, c, b, c, b, a, c, b, c, b, a, c, c, b, b, a, a, c, a, c, b, c, b, a, c, b, c, b, a, c, b, b, a, a, c, b, b, a, a, a, c, a, b, c, a, c, b, c, a, b, c, b, a, a, b, a, b, c, a, c, a, c, b, b, c, b, b, a, a, c, b, c, b, a, b, b, <, =, 15, 10, #, $
Answer: 15

Figure D.4: PCFG Exemplar. A representative sample from the PCFG dataset [Allen-Zhu and Li, 2023c]

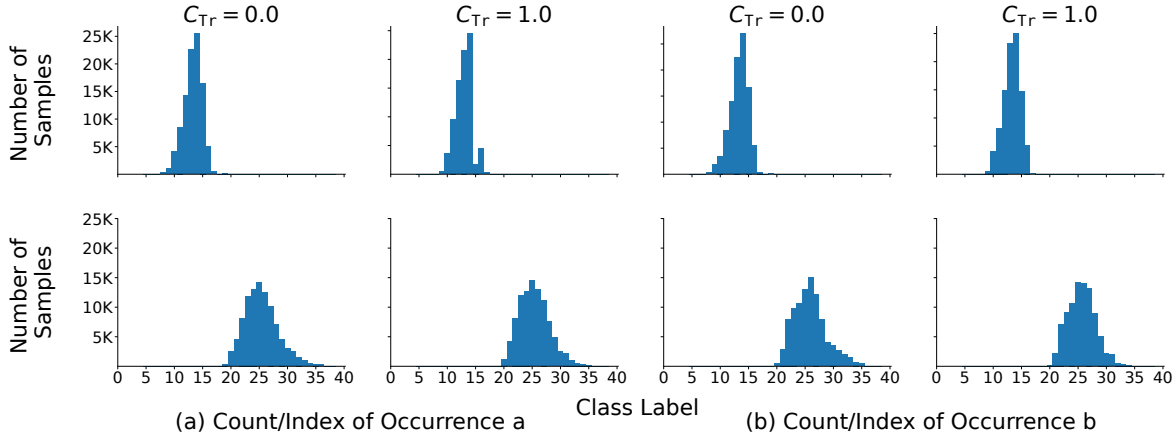


Figure D.5: Distribution of the class labels for Counting (first row) and Index of occurrence tasks (second row). (a) shows the distribution for the operand token `a` and (b) shows the same for the operand token `b`. The data is similarly distributed across different classes and the distribution shift for the two operands and the different values of $C_{\texttt{Tr}}$ is small.

D.3 Details on Training and Evaluation

D.3.1 Tracr

Compiled Model Details: The compiled model obtained for the counting and max identifier tasks consists of three blocks, wherein each block contains a single head attention layer followed by a two-layer MLP. No normalisation layers are used by models developed using Tracr.

Training details: The compiled model is fine-tuned using SGD with momentum for 10K iterations with a batch size of 96. Tracr yields models with a rather sparse parameterisation,

which often yields unstable training dynamics (e.g., gradient spikes), especially with adaptive optimisers. To address this, I perform the following two interventions. First, I add a small amount of initial gaussian noise $w_{noise} \in \mathcal{N}(0, 0.001)$ to the weights of the compiled model to densify them slightly. Note that the scale of this noise is not high, i.e., it avoids any performance loss but is sufficient to reduce gradient spikes resulting from extreme sparsity of model parameters. Second, I choose to use on SGD with momentum as the optimiser, using the following four choices of learning rates: Large LR (10^{-1}), Medium LR (10^{-2}), Small LR (10^{-3}), and Very Small LR (10^{-4}). The characterization of "Large" or "Small" is based on a general heuristic of what learning rate regimes are commonly used with SGD in modern neural network training. Linear warmup is used for 2K iterations followed by a cosine schedule with a minimum learning rate of the order 10^{-2} smaller than its max value. Evaluation of the fine-tuned model is done on both test set with and without the spurious correlation (ie. $C_{\texttt{Te}} = 0$ and $C_{\texttt{Te}} = 1$).

D.3.2 PCFG

Model details: I use the minGPT model by Karpathy [2020] for all experiments on the synthetically generated PCFG dataset, similar to Allen-Zhu and Li [2023c]. The model has close to 3 million parameters and consists of 6 blocks each made up of multihead self attention with 6 heads and two layers of MLP layers with an embedding dimension of 192.

Pre-training details: Pretraining is performed from scratch with a learning rate of 10^{-3} using the standard AdamW optimiser. Cosine learning rate is used along with linear warmup, where the warmup is used in the first 20% of the training. The model is trained using the standard next token prediction task used for training language models. I consider the set of five tasks mentioned in the previous section during the pre-training phase, but focus on only one of these tasks during fine-tuning. I use the task family token and an operand token to define the notion of a task. The task family token is sampled from a uniform distribution, while the operand token (`0`) is sampled from a multinomial distribution. The sampling probability for different operands is varied in the experimental setup to understand the effect of capability relevance in fine-tuning. More specifically, I analyse the following distributions for sampling the operand tokens (`a`, `b`, `c`):

- $\mathscr{P}_{\texttt{T}}(\texttt{a}) = 0.999,\ \mathscr{P}_{\texttt{T}}(\texttt{b}) = 0.001,\ \mathscr{P}_{\texttt{T}}(\texttt{c}) = 0.0$;

- $\mathscr{P}_{\mathtt{T}}(\mathtt{a}) = 0.99,\ \mathscr{P}_{\mathtt{T}}(\mathtt{b}) = 0.01,\ \mathscr{P}_{\mathtt{T}}(\mathtt{c}) = 0.0$;
- $\mathscr{P}_{\mathtt{T}}(\mathtt{a}) = 0.9,\ \mathscr{P}_{\mathtt{T}}(\mathtt{b}) = 0.1,\ \mathscr{P}_{\mathtt{T}}(\mathtt{c}) = 0.0$;
- $\mathscr{P}_{\mathtt{T}}(\mathtt{a}) = 0.7,\ \mathscr{P}_{\mathtt{T}}(\mathtt{b}) = 0.2,\ \mathscr{P}_{\mathtt{T}}(\mathtt{c}) = 0.1$; and
- $\mathscr{P}_{\mathtt{T}}(\mathtt{a}) = 0.5,\ \mathscr{P}_{\mathtt{T}}(\mathtt{b}) = 0.3,\ \mathscr{P}_{\mathtt{T}}(\mathtt{c}) = 0.2$.

For each of the configurations of sampling distributions of operands, I pre-train the model for 10K, 50K, 100K and 200K iterations. The model is trained in an online fashion to model the standard language model training pipeline, i.e., data is sampled on the fly from the data generating process during training time.

Fine-tuning details: While pre-training is done in the next token prediction fashion, fine-tuning is done in a supervised way where the model is required to just perform the desired fine-tuning task. I use the final iteration model obtained from pre-training as the initialisation for fine-tuning. While pre-training is done on multiple pairs of task and operand tokens, the model is fine-tuned on a single pair of task and operand tokens. To simulate a similar setup for fine-tuning as in Tracr, I analyse the effect of fine-tuning the model using three different sets of learning rate: Large LR (η_L: 10^{-4}), Medium LR (η_M: 10^{-5}) and Small LR (η_S : 10^{-6}). Fine-tuning is done for 10K iterations using AdamW optimiser with a batch size of 96 samples. Similar to pre-training phase, I use cosine learning rate with an initial warmup of 20% of the fine-tuning iterations. The minimum value of the learning rate is set to be $100\times$ lower than the maximum learning rate. Similar to Tracr evaluation is done on both the test sets with and without the spurious correlation ($C_{\mathtt{Te}} = 0$ and $C_{\mathtt{Te}} = 1$).

D.4 Mechanistic Interpretability Tools Setup

In this section, I describe the different tools of interpretability considered in my work.

Attention Maps: I present the attention maps for different tasks considered in the Tracr setup. Each map shows the tokens which are attending other tokens on the y axis and the token which are being attended to on the x-axis. If a token is attended by many other tokens, then, in a crude sense, this can imply that the presence of the token is impacting the underlying task performed by the model. In the Counter task, if significant attention is given to `a`'s / `b`'s is an indicator of the respective capability of the model. For the max identifier task, in the attention

map in Block-0, the model implements the sorting function, where each token is attended by the tokens which are greater than that. The vocabulary order followed is a > b > c > d.... In the attention map of Block-2, the model implements the read function, where it outputs the token at the desired position in the sorted sequence.

Pruning: I consider single step pruning where I prune the weights/neurons with largest dot product between their gradient and weights, where the gradients are calculated by minimising the loss for the capability I want to revive. More formally, let the weights of the model with N parameters be given by w_i where $i \in \{0, 1, \ldots, N-1\}$, Let the corresponding gradient be given by $grad(w_i)$ then the top-K weights with largest value of $grad(w_i)w_i$ are pruned off. This follows the pruning protocols proposed in prior work for reducing or preserving loss via pruning [Molchanov et al., 2017, Lubana and Dick, 2021, Mozer and Smolensky, 1988]. I use weight pruning for the Tracr setup and neuron pruning for PCFG, where a neuron is defined as a row in the weight matrix. I present a detailed description of the pruning protocol considered in Algorithm 3.

Algorithm 3: Pruning Pseudocode. A fine-tuned model f_θ is parameterised by θ and θ_i denotes its i^{th} neuron or weight (we prune neurons in PCFG experiments and weights in Tracr). Pre-training task family token is given by $\texttt{O}_{\texttt{PT}}$ and is prepended to a string X sampled from the data generating process, yielding the input $\texttt{O}_{\texttt{PT}} \circ X$. The true value corresponding to pre-training task family token $\texttt{O}_{\texttt{PT}}$ is given by y. Let the cross-entropy loss be given by CE. Let $\text{Top}_K(W)$ denote the indices of the top K values in the vector W.

```
def prune():
    # Forward prop the model on pre-training task
    out = f_θ(O_PT ∘ X)
    # Calculate the loss
    ℒ = CE(out, y)
    # Calculate the gradients
    grad = ∇_θ ℒ
    # Calculate the dot product between model weights and gradients
    dotproduct = θ.grad
    # Select the indices of top K values
    indices = Top_K(dotproduct)
    # Prune off the neurons/weights present in top K indices
    θ[indices] = 0
    return θ
```

Probing: Probing is used to understand if a particular capability is present in the model. In this, I train a linear layer (probe) on top of every block (residual layer's output) of the mini-gpt model and analyse if the probe is able to perform on a task requiring the use of

the desired capability. The probe is a linear layer with the output dimensions same as the vocabulary size of the model. The probe is trained using the data randomly sampled from the PCFG data generating process for 4K iterations using AdamW optimiser with maximum learning rate of 10^{-3} which is decayed by a factor of 10 at 2K, 3K and 3.5K iterations. Training of the probe is done separately on the residual output of each of the six blocks present in the minGPT model. The stream corresponding to the answer token (`Ans`) is used as the input to the probe.

Reverse Fine-tuning: Same set of hyperparameters as used in the fine-tuning of the pre-trained Tracr model are used in `reFT` , except for the learning rate, which I force to be smaller than the corresponding fine-tuning learning rate. Note that this use of an even smaller learning rate is intentional: if the original pretraining capability can be revived even with this setup, it is stronger evidence that the pretraining capability was never forgotten or removed.

D.5 Additional Experiments and Results

D.5.1 Fine-tuning in presence of some pre-training data

In this section, I demonstrate my claims also hold for an often used fine-tuning setup wherein, beyond the fine-tuning data, the model also gets to see some portion of the pretraining data again. Specifically, I perform three degrees of mixing of the pretraining and fine-tuning data: (i) 50% PT + 50% FT, (i) 10% PT + 90% FT, and (i) 0.1% PT + 99.9% FT. I show behaviour results on how the performance of the model improves as a function of fine-tuning iterations for different spurious correlations for a low pretraining sampling prior in Figs. D.6 and D.7 and high sampling prior in Figs. D.8 and D.9. Furthermore, I probe these models' intermediate outputs to infer if features relevant to the pretraining capability continue to persist. Results can be seen in Figs. D.10 and D.11.

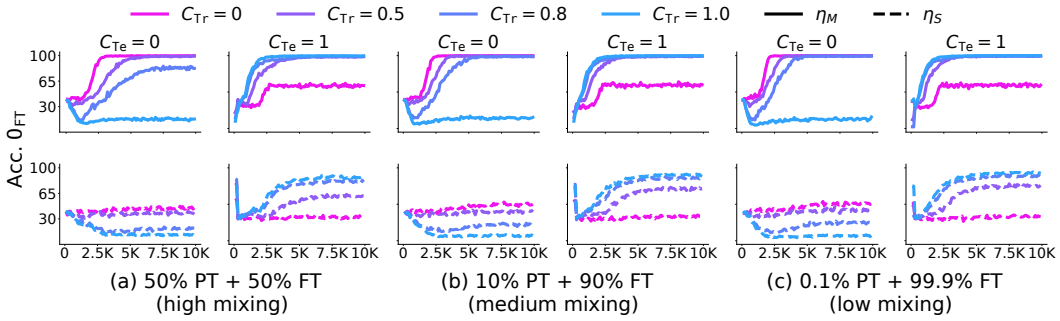


Figure D.6: Effect of different sampling probabilities of pre-training target token O_{PT} on fine-tuning task's performance. I observe similar gains for different values of sampling probabilities of O_{PT} during fine-tuning.

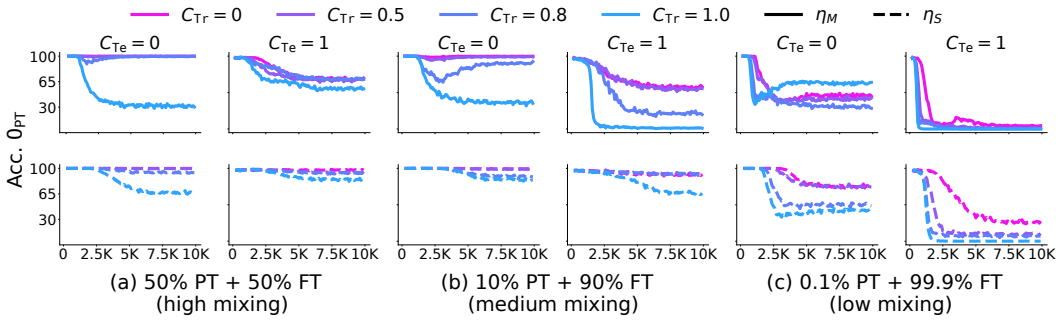


Figure D.7: Effect of different sampling probabilities of pre-training target token O_{PT} on pre-training task's performance. I observe a higher loss in performance if low sampling probability is used for sampling the pre-training target token O_{PT} during fine-tuning.

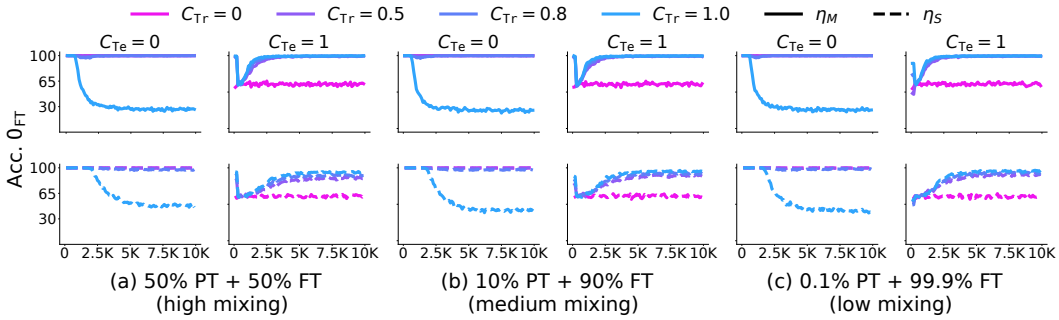


Figure D.8: Effect of different sampling probabilities of pre-training target token O_{PT} on fine-tuning task's performance. Pre-training is done using high sampling prior for fine-tuning task family token. I observe similar gains for different values of sampling probabilities of O_{PT} during fine-tunning.

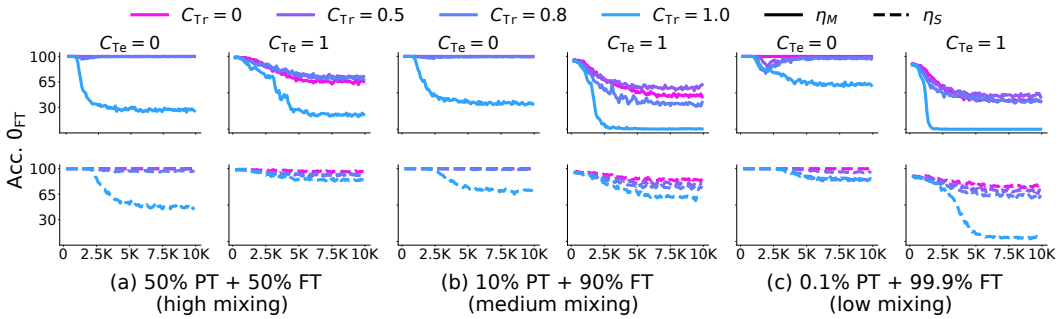


Figure D.9: Effect of different sampling probabilities of pre-training target token O_{PT} on pre-tuning task's performance. Pre-training is done using high sampling prior for fine-tuning task family token. I observe similar gains for different values of sampling probabilities of O_{PT} during fine-tunning.

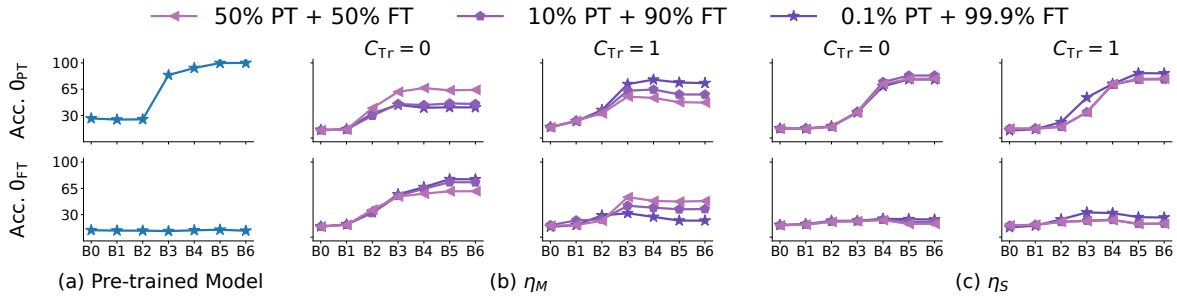


Figure D.10: Probing analysis corresponding to Fig. D.6 Fig. D.7

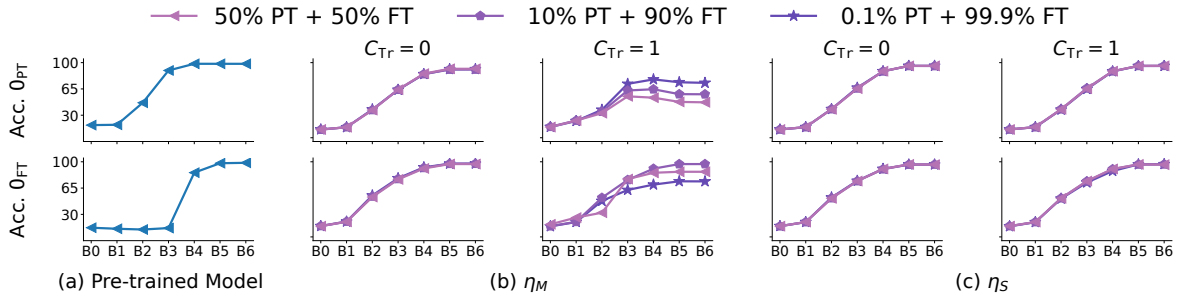


Figure D.11: Probing analysis corresponding to Fig. D.8 Fig. D.9

D.5.2 Jailbreaking Analysis

We emulate jailbreaking [Wei et al., 2023, Zou et al., 2023] in my PCFG setup by defining several task family tokens describing the same task. Specifically, for the “Counter" task, I use three task family tokens $\texttt{T}_{NJ}, \texttt{T}_{J_1}, \texttt{T}_{J_2}$ to refer to the task in a string. Here subscript NJ indicates the task family token will not allow jailbreaking, while J_1/J_2 indicate the task family token can be used to jailbreak the model, as explained next. For pretraining, the token T_{NJ} may be paired with operand tokens $\texttt{a}, \texttt{b}, \texttt{c}$ to learn to count them from inputs sampled from the PCFG. However, tokens T_{j_1}, T_{j_2} are used only for counting $\texttt{a}$. During fine-tuning, the model is fine-tuned to count the token $\texttt{b}$ using the task family token $\texttt{T}_{NJ}$. For evaluation, I compute the model's accuracy on its ability to count the token $\texttt{a}$, using either the task family token $\texttt{T}_{NJ}$ or $\texttt{T}_{J_1}, \texttt{T}_{J_2}$. As shown in Fig. D.12, the model is unable to infer the count of $\texttt{a}$ if the task family token $\texttt{T}_{NJ}$ is used; however, if task family tokens $\texttt{T}_{J_1}, \texttt{T}_{J_2}$ are used, the model performs perfectly if the prior for sampling the fine-tuning target $\texttt{b}$ during pretraining was sufficiently high. I argue that this is expected because under a high sampling prior breaks the symmetry between task family tokens (indeed, $\texttt{T}_{J_1}$ is only seen with operand token $\texttt{a}$, but $\texttt{T}_{NJ}$ is seen for all operand tokens. This indicates the pretraining capability continues to persist in the model, enabling jailbreaking. To further investigate this result, I also probe the fine-tuned models. Results are shown in Fig. D.13. As expected, we see task family tokens $\texttt{T}_{J_1}, \texttt{T}_{J_2}$ allow for linear readout of the count of $\texttt{a}$; however, we see that even for inputs with task family token $\texttt{T}_{NJ}$, the model does encode the count of $\texttt{a}$ in the outputs around the middle layers!

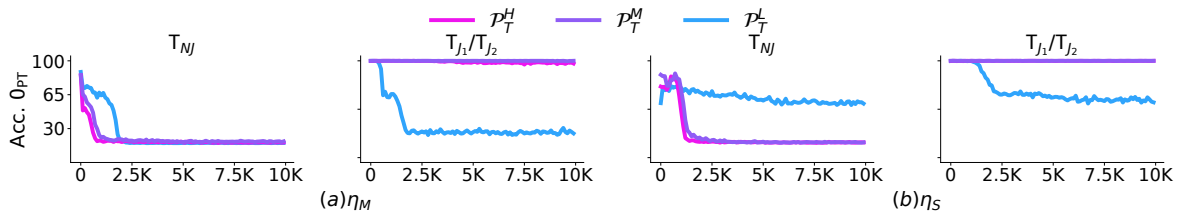


Figure D.12: Jailbreaking analysis using PCFG. I report performance on the pretraining task (counting $\texttt{O}_{\texttt{PT}}$) as a function of fine-tuning iterations, where the fine-tuning task (counting $\texttt{O}_{\texttt{FT}}$) is performed using the task family token $\texttt{T}_{\texttt{NJ}}$. We find that the model is able to learn the fine-tuning task and seemingly performs poorly on the pretraining task when task family token $\texttt{T}_{\texttt{NJ}}$ is used in the input. However, in presence of a sufficiently relevant capability (high pretraining prior for $\texttt{O}_{\texttt{FT}}$), using task family tokens $\texttt{T}_{\texttt{J}_1}$ or $\texttt{T}_{\texttt{J}_2}$ in the input shows the model can still perform the pretraining task perfectly—i.e., we can jailbreak the model.

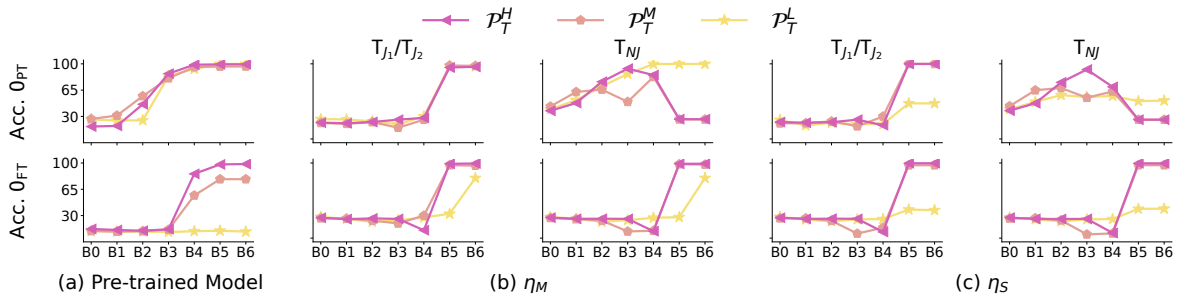


Figure D.13: Probing analysis for the setup used to understand jail-breaking. Similar results on using the fine-tuning token or the jailbreaking token for training the probe indicate that the pre-training capabilities are not removed on fine-tuning.

D.5.3 Sample efficiency analysis for Reverse Fine-tuning

To emphasize the fact that the pretraining capability is "revived" in the model relatively sample-efficiently, I repeat Fig. 5.10, where models trained on PCFG are reverse fine-tuned, and repeat the experiment with the `Scr.+FT` baseline for Tracr compiled models. As can be seen in Figs. D.14 and D.15, compared to the baseline, the model learns to perform the pretraining task in substantially fewer iterations than the baseline. I note that for the Tracr models in these results, even an extremely small learning rate is sufficient to revive the pretraining capability! I also note that I do not sweep over the $C_{\texttt{Tr}}$ hyperparameter in the Tracr models because they are compiled, i.e., I cannot control the correlation with the pretraining capabilities in a meaningful way.

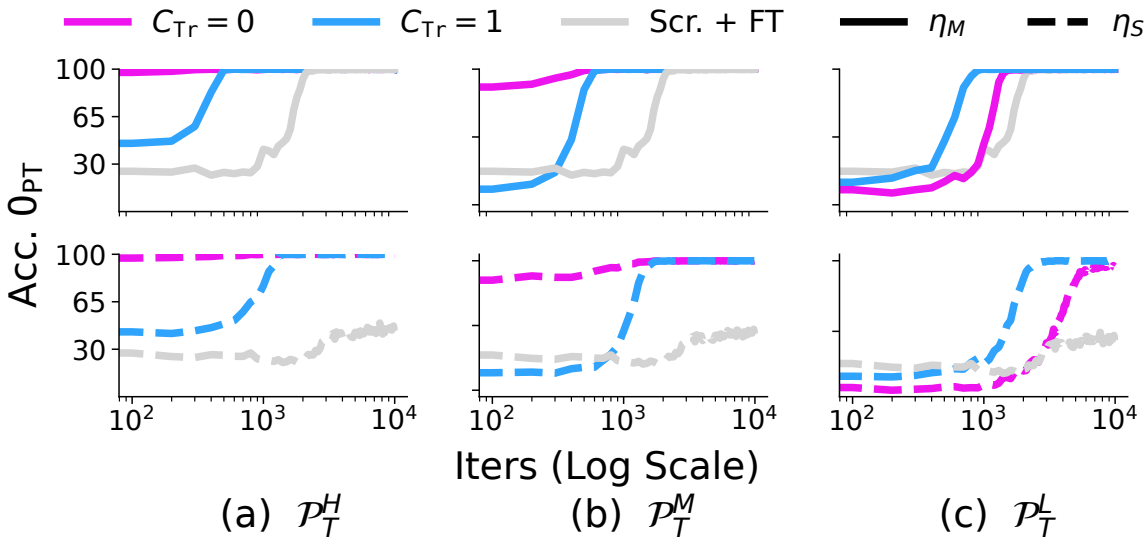


Figure D.14: Reverse Fine-Tuning on PCFGs: I set $C_{\texttt{Te}}$ to be 0 to test if the model performs well regardless of a spurious correlation. I observe that when a strongly relevant capability is present (a, b), the model very quickly (0.1–1K iterations) starts to perform well on the task via `reFT` , even if behaviour relevant to the capability ceased during pretraining (e.g., when $C_{\texttt{Tr}}$ is 1). Meanwhile, when the model possessesses a weakly relevant capability (c), this "revival" is *slightly* slower (3K iterations). In contrast, the `Scr.+FT` baseline only reaches perfect accuracy at 4.5K iterations and when using a larger learning rate η_M.

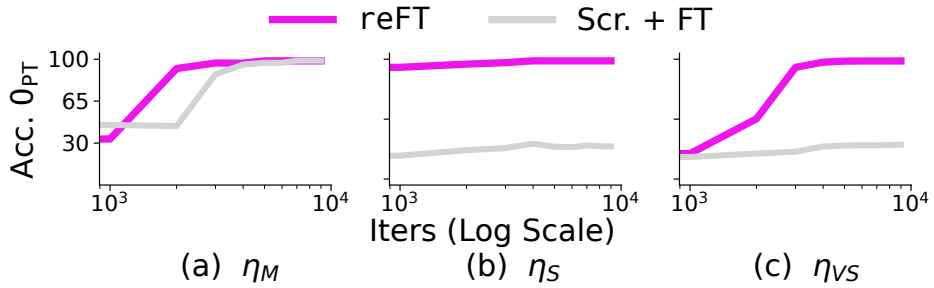


Figure D.15: Reverse Fine-Tuning on Tracr: I set $C_{\texttt{Te}}$ to be 0 to test if the model performs well regardless of a spurious correlation. I observe that the fine-tuned model upon `reFT` very quickly starts starts to perform well on the pretraining task. Moreover, the protocol works even if an extremely small learing rate is used. In contrast, the `Scr.` + `FT` baseline only reaches a large learning rate η_M is used, and does so less sample efficiently. I note that the results for η_M learning rate look worse than the η_S learning rate around 10^3 iterations because η_M is too big of a learning rate, forcing the model to essentially go through a "retraining" phase.

D.5.4 Reverse Fine-tuning a more safety-oriented fine-tuning protocol

The fine-tuning protocols used in the bulk of the paper focus on learning a new capability, e.g., counting a new operand, while promoting reusability of capabilities learned during pretraining. Part of my motivation is to see if a pretrained model is actively forced to remove a capability, does that work? To analyse this, I define a fine-tuning protocol called `randFT` wherein the model is trained to actively produce an incorrect output for inputs that require use of the pretraining capability. For example, if the model possessesses the capability to produce the count the number of occurrences of token $\mathtt{O}_{\mathtt{PT}}$= `a` in a string, I fine-tune it to produce the count of tokens $\mathtt{O}_{\mathtt{FT}}$= `b` in that string. I analyse these fine-tuned models analysed via reverse fine-tuning (`reFT`), i.e., by further training them to produce the correct outputs (number of occurrences of token $\mathtt{O}_{\mathtt{PT}}$). I provide results for three baselines as well: (i) `Scr.`, wherein the model is trained from scratch to learn to count the token `a`; (ii) `Scr.` + `FT`, wherein the model is initialised with parameters trained via trained from scratch to count a separate token ($\mathtt{O}_{\mathtt{FT}}$) and then the model is fine-tuned to count the token $\mathtt{O}_{\mathtt{PT}}$; and (iii) `reFT` , which follows reverse fine-tuning models that were fine-tuned with the protocols used in the bulk of the paper, i.e., fine-tuned to learn a new capability that is related to the pretraining one.

Results are shown in Fig. D.16. I specifically zoom in on the the scenario where `reFT` takes the longest time, i.e., when the sampling prior of the downstream target $\mathtt{O}_{\mathtt{FT}}$ is low in pretraining data; results for other sampling priors are shown in Fig. D.17 We see that reverse fine-tuning a `randFT` model is similarly sample-efficient as the standard `reFT` pipeline used in the bulk of the paper, while being more sample-efficient than the `Scr.` and `Scr.` + `FT` baselines.

In addition, I perform a probing analysis of the `randFT` models in Fig. D.18. I again find that I can predict the information relevant for the pretraining task, i.e., the count of $\mathtt{O}_{\mathtt{PT}}$.

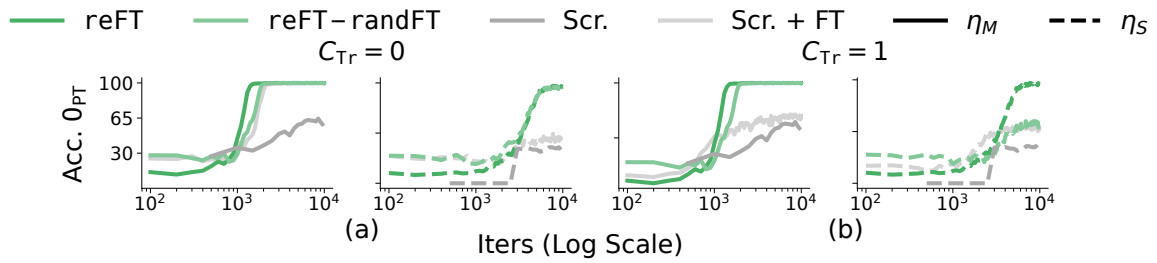


Figure D.16: Reverse fine-tuning a model fine-tuned to remove its pretraining capability. See text in Appendix D.5.4 for details.

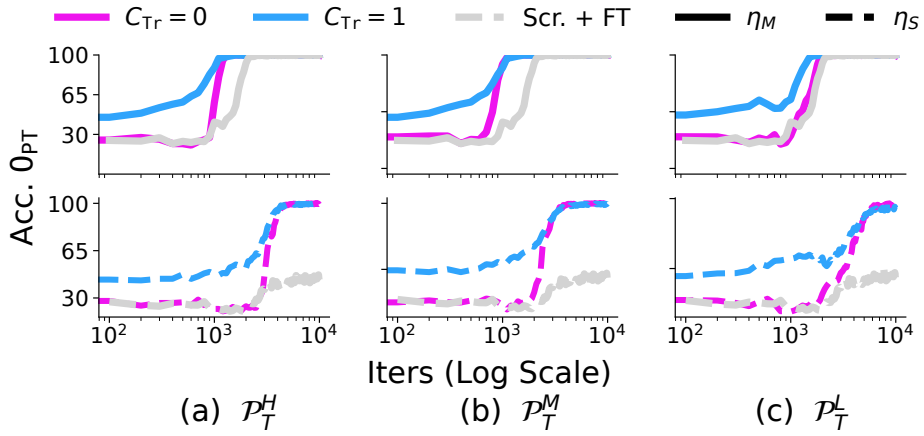


Figure D.17: Reverse fine-tuning performance on using `randFT` fine-tuning protocol to forget the pre-training capability. I follow the setup of Fig. 5.10 and plot results for several sampling priors of the target token for fine-tuning, i.e., O_{FT}, but I use `randFT` for fine-tuning the models before `reFT` . The baseline results `Scr.+FT` are copied from the Fig. D.16, i.e., baseline is not trained in an "adversarial" way, but the `randFT` results are. While this makes the baseline unfairly stronger, I find `reFT` the `randFT` models are still more sample efficient.

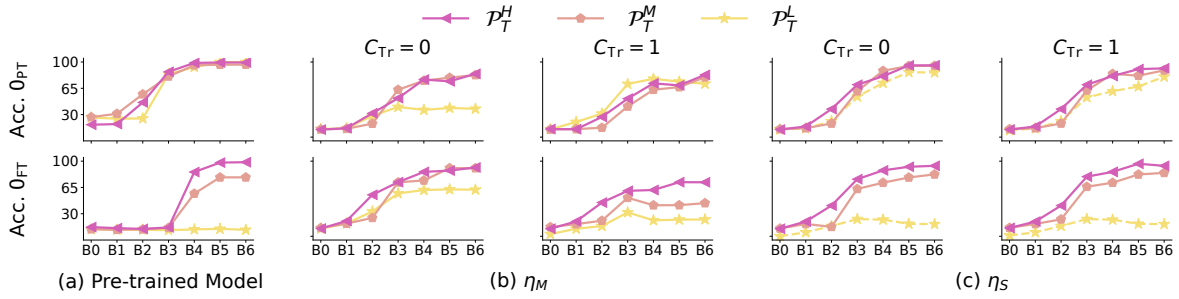


Figure D.18: Probing analysis of `randFT` fine-tuning protocol. I plot probe accuracy versus the index of the block in the Transformer model. C_{Te} is set to 0. The pretrained model (left) acts as a baseline for the trend of performance through the model's blocks. In most scenarios, I find I can infer the count of O_{PT} with a similar trend as the pretrained model (left). A drop in performance is observed only when learning rate η_M is used with a weakly relevant capability (low sampling prior). This indicates pretraining capabilities continues to persist upon fine-tuning.

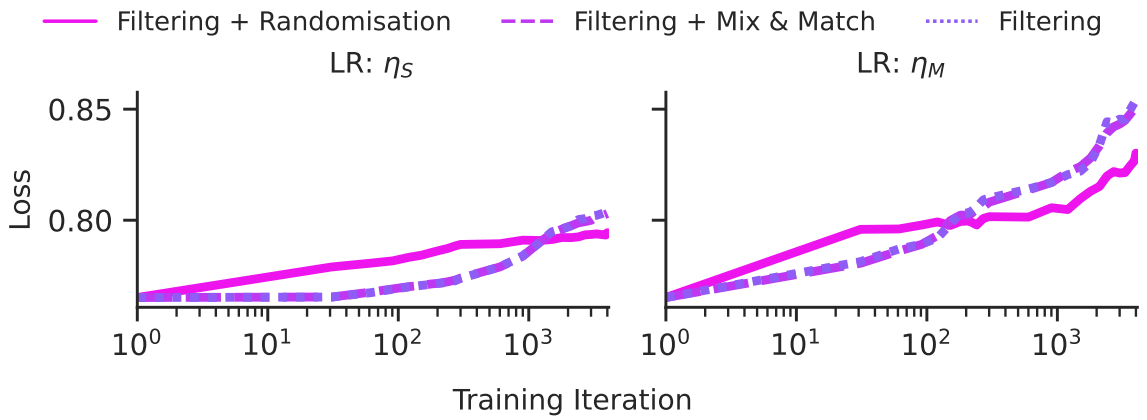


Figure D.19: Larger learning rates lead to more pronounced loss of modelling capability. The plots show loss on data with the Twist feature present while fine-tuning to delete the capability to model text with the *Twist* feature, for different learning rates and fine-tuning protocols.

D.6 Additional Tinystories Results

Reverse fine-tuning The loss on stories with `Twist` during fine-tuning is shown in Fig. D.19. This shows that the fine-tuning protocols are raising the loss, and hence behaviourally deleting the capability of fine-tuning. The generation scores are shown in Fig. D.20. This again reinforces that most fine-tuning protocols are removing the capability behaviourally, as the generation scores (while noisy) drop to close to 0.

Fig. D.21 shows the loss during `reFT` for all the fine-tuned models, as well as the control model pre-trained without stories with `Twists`, and Fig. D.22 shows the generation scores. Both of these results show that the fine-tuned models learn the new capability in a much more sample-efficient way, and in fact converge to a lower loss on this dataset than the control pre-trained model.

Probing In addition to the `reFT` results, I perform probing experiments. The probing accuracy for the Twist feature across layers for the fine-tuned models and the two control pre-trained models is shown in Fig. 5.12, which I reproduce here in Fig. D.23 for completeness. These results show that a small amount of information about story classification has been removed from the activations of the fine-tuned models compared to the model pre-trained with `Twist` stories, but the reduction is very minor, as shown in comparison to the information present in the model pre-trained without `Twist` stories.

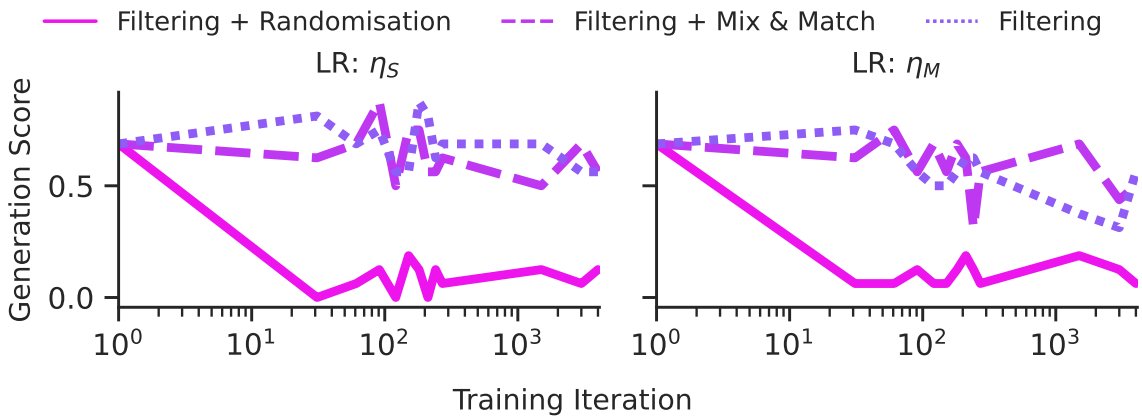


Figure D.20: Larger learning rates lead to more pronounced loss of generative capability. The plots show the generation score for the Twist feature present while fine-tuning to delete the capability to model text with the *Twist* feature, for different learning rates and fine-tuning protocols.

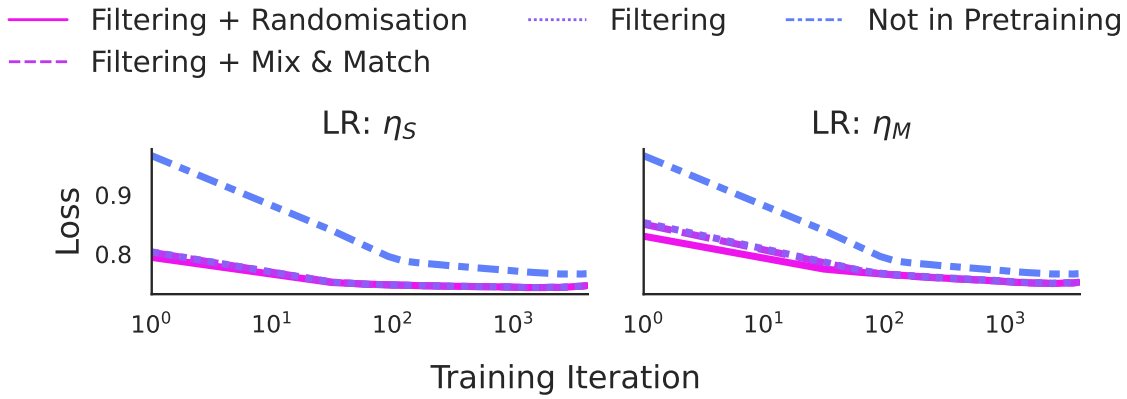


Figure D.21: `reFT` easily recovers deleted capabilities. I plot loss on data with the `Twist` for `reFT` of various models fine-tuned to delete the capability, as well as a control model which was pre-trained without data with `Twists`. The fine-tuned models learn the capability more sample-efficiently, and additionally converge to a lower loss than the control model.

Fig. D.24, Fig. D.25, and Fig. D.26 show similar plots for several other story features. Some of these are easier or harder for probes to classify, but the important result is that the difference in probe accuracy between the fine-tuned models and both pre-trained control models is negligible for all of these features, showing that the results in Fig. D.23 are due to the `Twist` feature, i.e., the feature that I trained the model to delete.

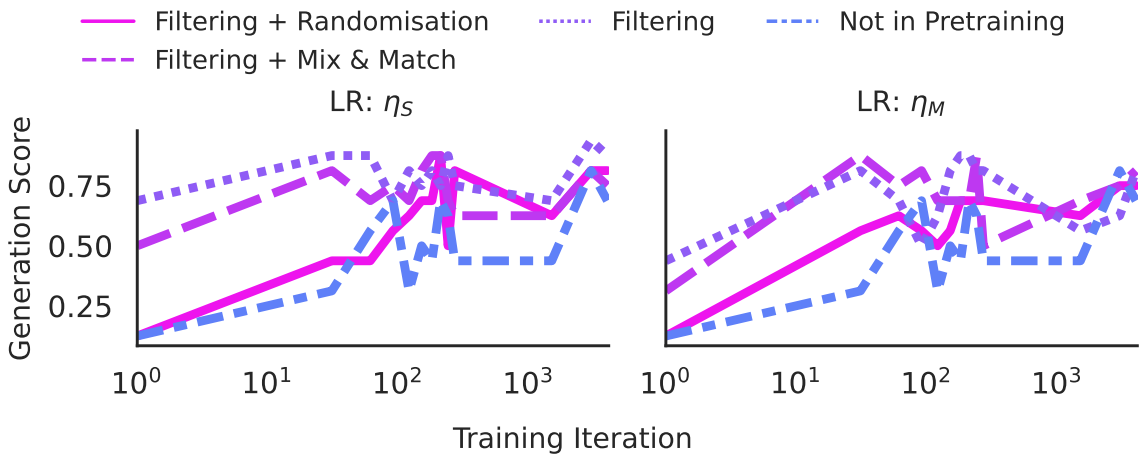


Figure D.22: `reFT` easily recovers deleted generative capabilities. I plot the generation scores for the `Twist` feature for `reFT` of various models fine-tuned to delete the capability, as well as a control model which was pre-trained without data with `Twists`. The fine-tuned models learn the capability much more sample-efficiently, and additinoally converge to a lower loss, than the control model.

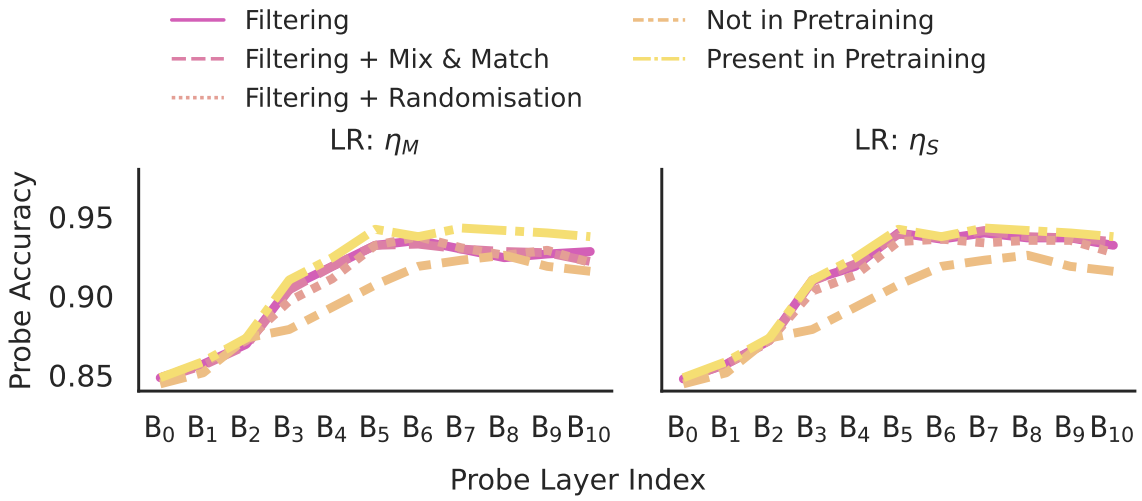


Figure D.23: Probing the presence of capabilities in TinyStories Models. I plot probe accuracy of classifying whether a story contains a `Twist` or not wrt. the layer of the Transformer model (similarly to Fig. 5.8). Accuracy on models pre-trained with or without `Twist` data (*Present*/*Not in Pretraining* respectively) act as upper and lower bounds on the expected accuracy of the probes, and are plotted on both LR figures for ease of comparison, although they do not use a fine-tuning learning rate. We find that regardless of fine-tuning protocol (Filtering, Filtering + Randomisation, Filtering + Mix & Match), for the lower LR no fine-tuning protocol removes a meaningful amount of information from the activations, and a similar but less strong trend holds for the higher LR, implying that the pre-trained model retains its capability of story identification (a necessary capability for story modelling) throughout fine-tuning. Identical to Fig. 5.12

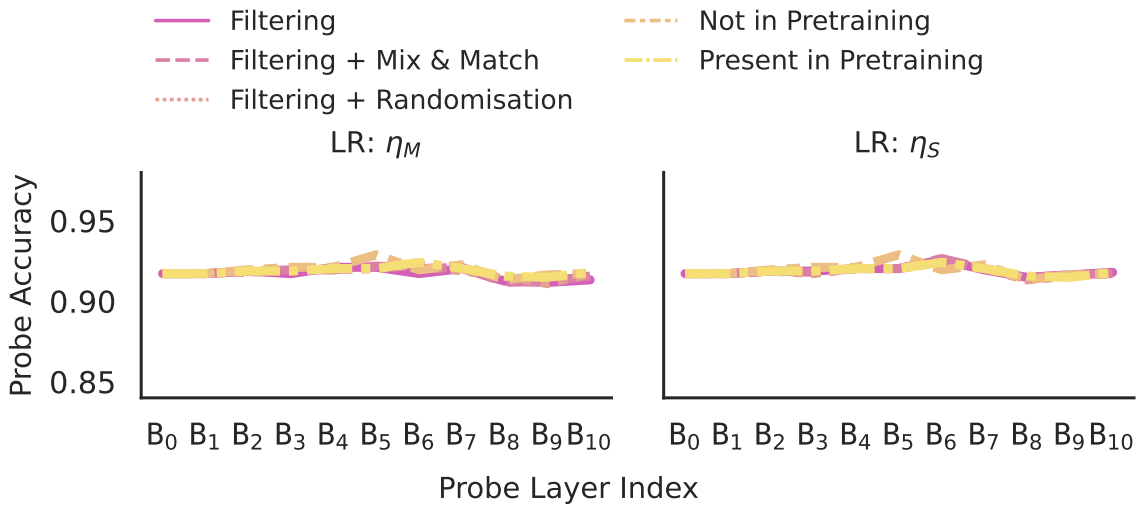


Figure D.24: Probing the presence of capabilities in TinyStories Models. I plot probe accuracy of classifying whether a story contains the Foreshadowing feature or not wrt. the layer of the Transformer model. All other details the same as Fig. D.23

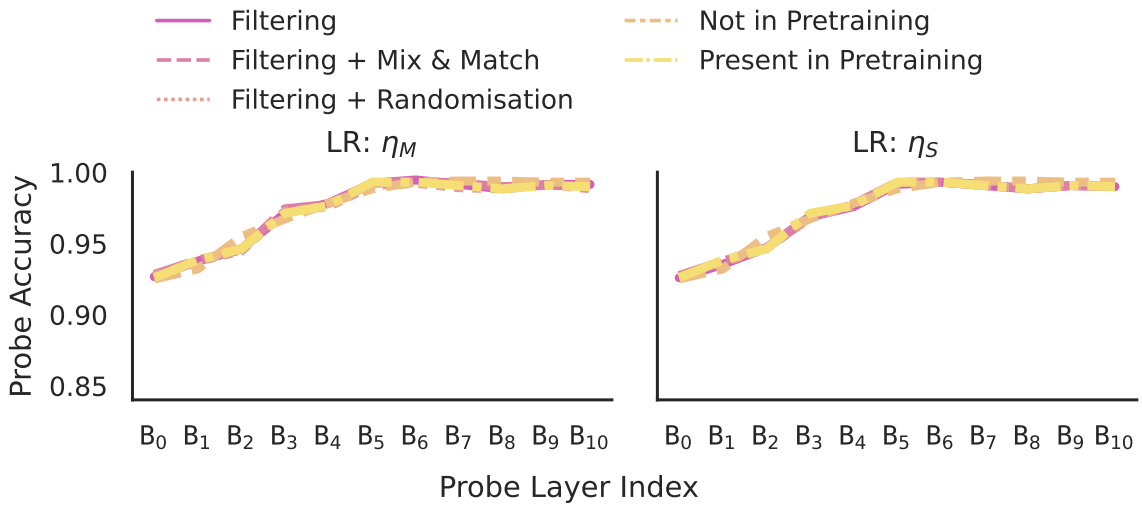


Figure D.25: Probing the presence of capabilities in TinyStories Models. I plot probe accuracy of classifying whether a story contains the Moral Value feature or not wrt. the layer of the Transformer model. All other details the same as Fig. D.23

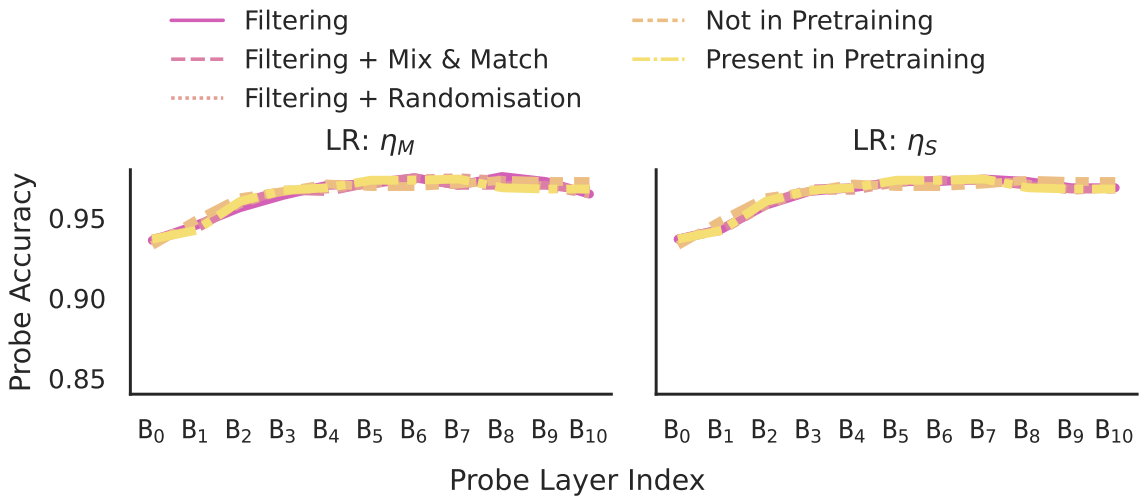


Figure D.26: Probing the presence of capabilities in TinyStories Models. I plot probe accuracy of classifying whether a story contains the Bad Ending feature or not wrt. the layer of the Transformer model. All other details the same as Fig. D.23

D.7 Additional Tracr Results

In this section, I present additional results on the counting and max-identifier tasks with Tracr models. These results provide an extensive analysis and support my claims presented in the main paper. Firstly, I present the detailed attention maps showing the full input sequence data: Fig. D.28 and Fig. D.27 show the attention maps corresponding to Fig. D.30 and Fig. D.31. I now present the detailed results on Tracr's Counter tasks.

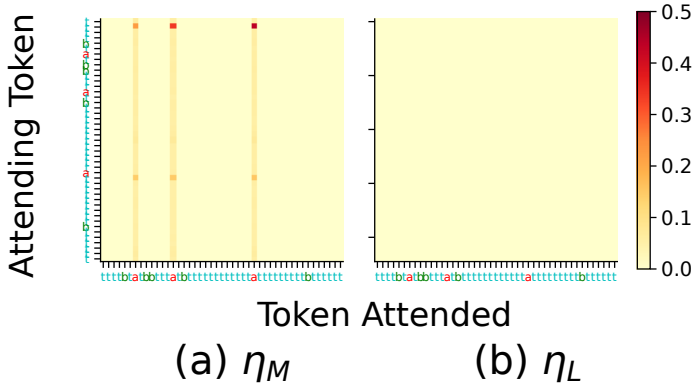


Figure D.27: Counter Task: Detailed visualisation of the capability revival analysis on Tracr compiled for the Counter task. Observation: On using η_{VS}, we are able to revive the compiled capability of the Tracr model fine-tuned with η_M, whereas using η_L for fine-tuning hampers the underlying capability of the model and therefore there is no revival seen.

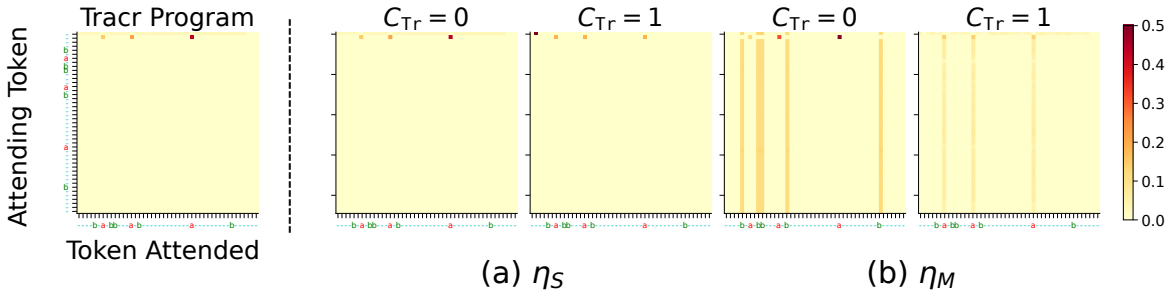


Figure D.28: Counter Task: Detailed visualisation of the self attention layer in Block-1 of the Tracr fine-tuned model. Observation: (a) shows the map for the compiled Tracr model, where I observe that a's are being attended by the second token in the input sequence. (b) On using η_S for fine-tuning in the absence of spurious correlations ($C_{\mathtt{Te}} = 0$), a's are still being attended in the attention map. (c) On using η_M, the model learns the fine-tuning task as b's are also being attended now. On fine-tuning in the presence of the spurious correlation ($C_{\mathtt{Te}} = 1$), the model doesn't learn to attend b's, but rather learns the spurious correlation.

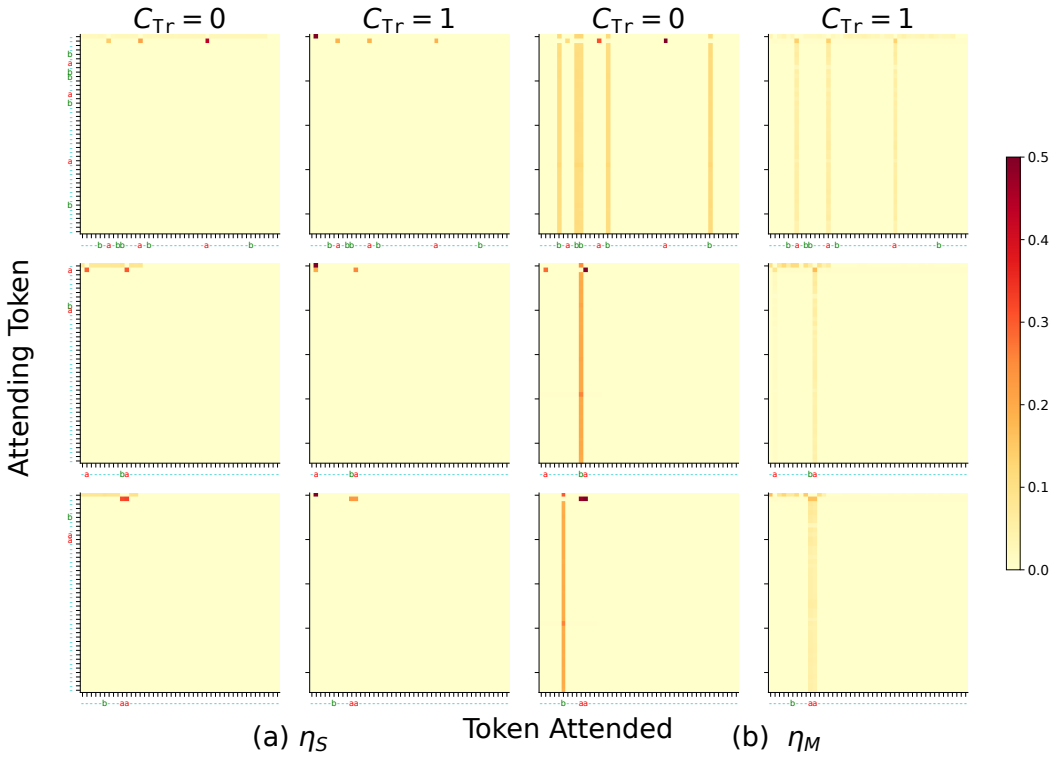


Figure D.29: Counter Task: Validation of Tracr observations on Counter task using three different input samples. The rows represents different input samples. **Observation:** Using η_S for fine-tuning the Tracr model compiled on Counter task is unable to learn to attend `b`'s (a). But the model learns the spurious correlation. On the other hand, on using η_M the model is able to learn the fine-tuning task by attending to `b`'s. But in the presence of the spurious correlation $C_{\texttt{Tr}} = 1$ the model still doesn't learn to attend `b`'s.

D.7.1 Behavioural Results On Fine-Tuning

Summary of results on the Counter task. As shown in Fig. D.30, on using η_M, the model seems to learn a new capability of counting `b`'s (the model primarily attends to `b` in its attention map). However, on using a small learning rate (η_S) of 10^{-3}, in the absence of correlations, the model is not able to learn to attend to `b`'s in its attention map. Thus the model is not able to learn the capability of counting `b`'s. As shown in Table D.2, in the presence of spurious correlations however, the model is able to learn the spurious correlation and achieve high accuracy on the correlated test set. I also present the visualisation of the attention maps after reverse fine-tuning in Fig. D.31, where it is observed that on using η_M for fine-tuning, revival of capability is possible even on using a very small learning rate (η_{vs}) of 10^{-4}. I present detailed results on reverse fine-tuning in Table D.3 Whereas, in case the model is fine-tuned with a large learning rate (η_L) of 10^{-1}, revival of capability is not possible. I also present

analysis of single weight pruning and grafting in Fig. D.43. On pruning off a single weight from the Tracr model compiled to count a's, the model can achieve a boost in accuracy of over 60% on the task of counting a's. This observation is evident only when the Tracr model is fine-tuned on correlated dataset.

Summary of results on the max-identifer task. We present a visualisation of the attention maps for the max identifier task in Fig. D.32, where I observe that Tracr model implements the sorting and the reading functions in the attention maps in blocks 0 and 2 respectively. On fine-tuning the model using different learning rates, the sorting capability implemented in Block-0, gets distorted, thereby resulting in poor fine-tuning performance (as evident in Table D.2). However using η_{VS} (10^{-4}), changes the reading function, without disturbing the sorting function. Thus the model is able to perform well on the downstream task (as evident in Table D.2).

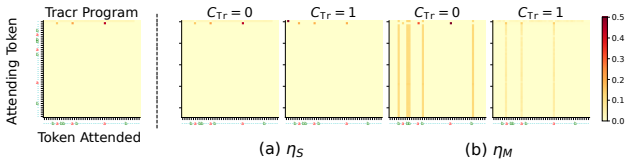


Figure D.30: New capabilities are not learned on using η_S for fine-tuning. (a) The counting a's capability implemented by the originally compiled Tracr model is to attend to a's. On using η_S (10^{-3}) for fine-tuning, the compiled model is not able to learn the capability of counting b's (b). Increasing the learning rate makes the model learn to attend b's (c), but the pretraining capability of attending to a's still exists.

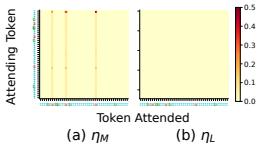


Figure D.31: Capability Revival Analysis: Using η_{VS} (a) is able to recover the old capability on reverse fine-tuning the model fine-tuned with η_M. But η_S is not able to recover the original capability, when the compiled model is fine-tuned with η_l. This is because using a large value of learning rate during=fine-tuning hampers the pre-training capabilities.

D.7.2 Counter results

A detailed analysis of the attention maps of block-1 and 2 for different learning rates is shown in Fig. D.33. I further validate my results for three different input datapoints in Fig. D.29 and Fig. D.37. A detailed analysis of the activation map in Block-1 for different values of $C_{\texttt{Tr}}$ is shown in Fig. D.36.

We present an evidence further, showing that capability of the Tracr compiled model to count a's is still present in the model in Fig. D.34,Fig. D.35, where Fig. D.34 presents a closer look of the Fig. D.35. As can be seen in Fig. D.34, on using η_S and η_M, Block-1 activation

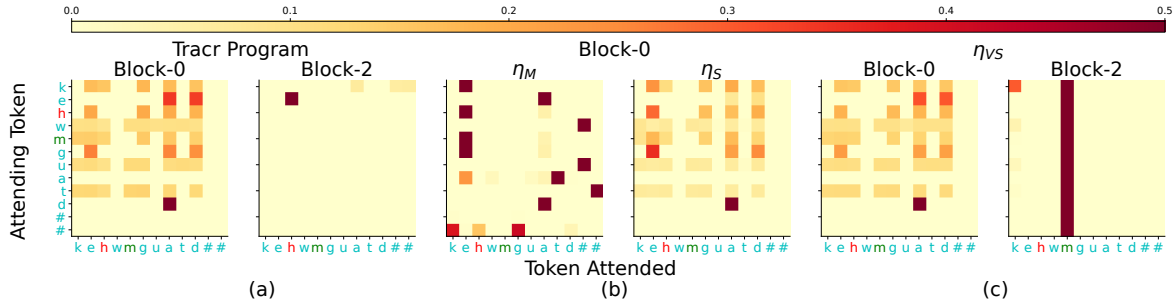


Figure D.32: Learning of the fine-tuning capability is affected by type of compiled capability present in the model. (a) The Tracr program implements the sorting function in Block-0 and read function in Block-2. Using η_M and η_S can destroy the sorting capability present in Block-1 (b). But using η_{VS}, preserves the sorting capability (c). Thus on using η_{VS}, the model learns to read a different stream of output, while preserving the sorting capability.

map of the Tracr fine-tuned model shows neurons corresponding to token `a` being activated in a different output channel.

Finally, I present evidence of the wrapper being learned by the model on fine-tuning using spuriously correlated dataset. I show that this wrapper can be localised in a very few neurons of the model. As shown in Fig. D.43, I present this evidence for different values of $C_{\texttt{Tr}}$ in Fig. D.44. Similar to the analysis presented for PCFG where I prune multiple neurons, I analyse the effect of pruning of mutliple weights and neurons in Fig. D.45 and Fig. D.46 respectively. These results verify that the wrapper learned by the Tracr compiled model on fine-tuning using spuriously correlated dataset can indeed be localised to a few weights of the Tracr model. To ensure that the gains achieved on pruning are indeed because of removal of the wrapper, I present the histograms showing the distribution of the predicted classes in Fig. D.47, where it can be observed that after pruning the model still predicts multiple classes.

D.7.3 Max identifier results

In this section, I provide additional evidence and a detailed analysis of the performance of the Tracr compiled model on the max identifier task. I show that the model implements the sorting pattern in the activation map of its first block in Fig. D.41 and Fig. D.42. I present validation of my observations on considering the spurious correlation as the difference between the fifth and seventh maximmum element being three in Fig. D.39. I validate my results for three different input data-points in Fig. D.38. A detailed visualisation of the attention maps in Block-0 and Block-2 for different learning rates is shown in Fig. D.40.

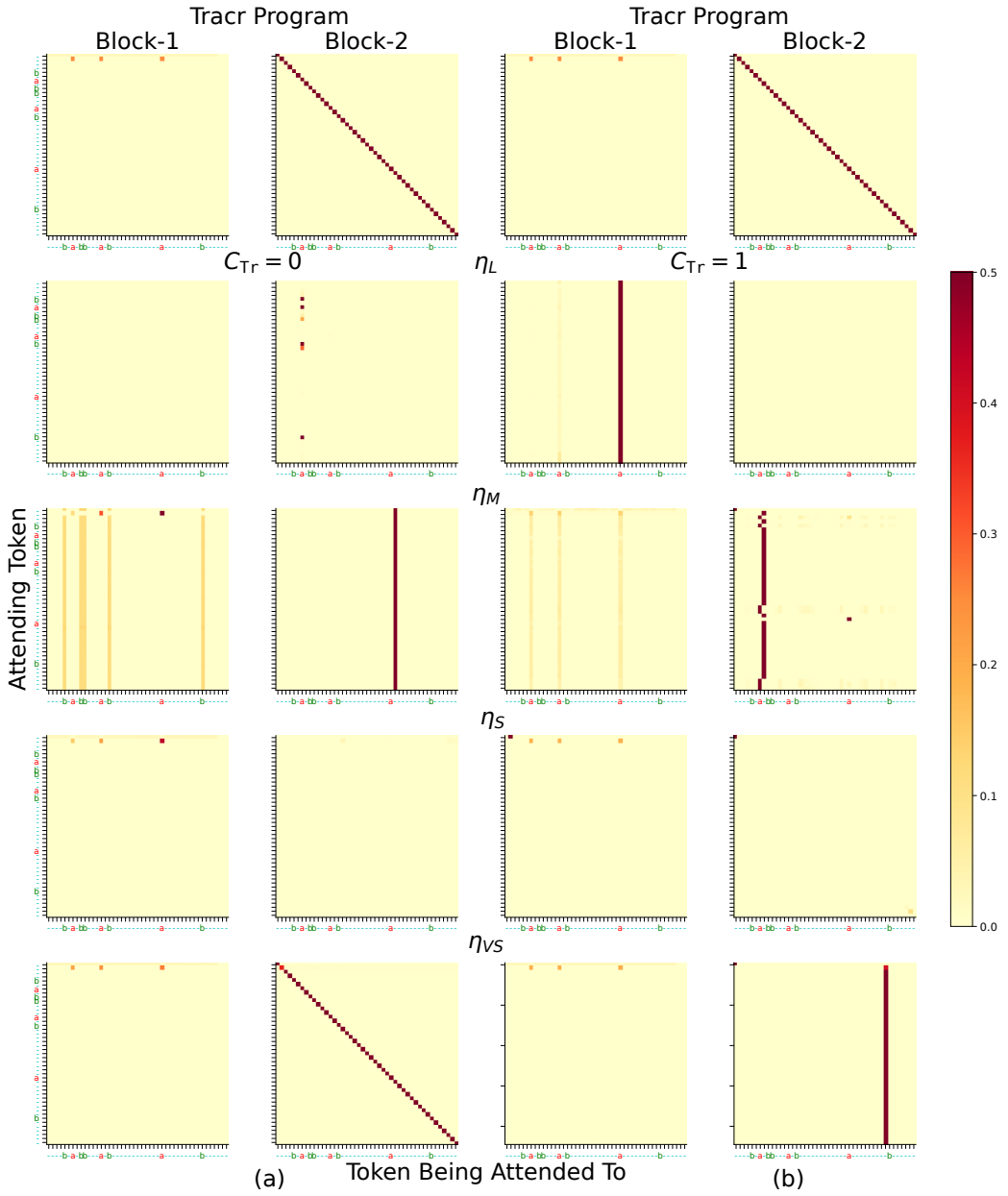


Figure D.33: Counter Task: Visualisation of the attention maps of the first and second blocks of the Tracr fine-tuned models. (a) shows the analysis when the spurious correlation is not present in the fine-tuning datatset, whereas in case of (b) the spurious correlation is present in the fine-tuning dataset. The first row shows the maps for the Tracr compiled model and other rows shows the analysis for different learning rates. **Observation:** (a) Using η_S or η_{VS} the model is not able to learn to attend b's and thus the fine-tuning task performance is poor. Whereas using η_M the model is able to learn to attend to b's, however the capability to count a's is likely still present since the model still attends to a's. Further increasing the learning rate leads to distortion of the compiled capabilities, and thus model learns the fine-tuning task by learning a different capability. (b) In the presence of spurious correlation, even for large learning rate the compiled capability is still present, since the model attends to a's.

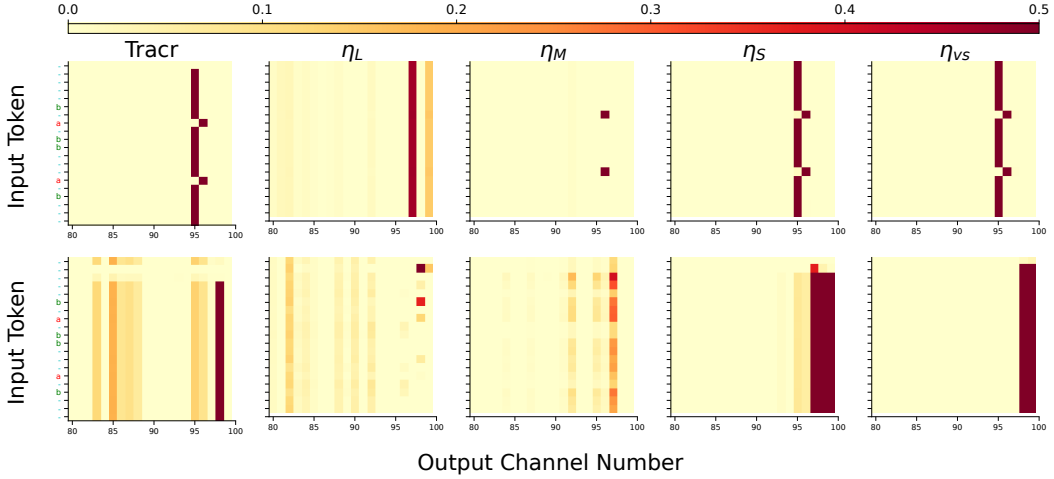


Figure D.34: Counter Task: Visualisation of the activated output of the first MLP layer in first and second blocks. The visualisation is shown only for channel numbers 80-100. **Observation:** Using η_M preserves the Tracr compiled capability, while also learning the fine-tuning task. This shows that the model changes behaviourally but mechanistically the compiled capabilities are still present in it.

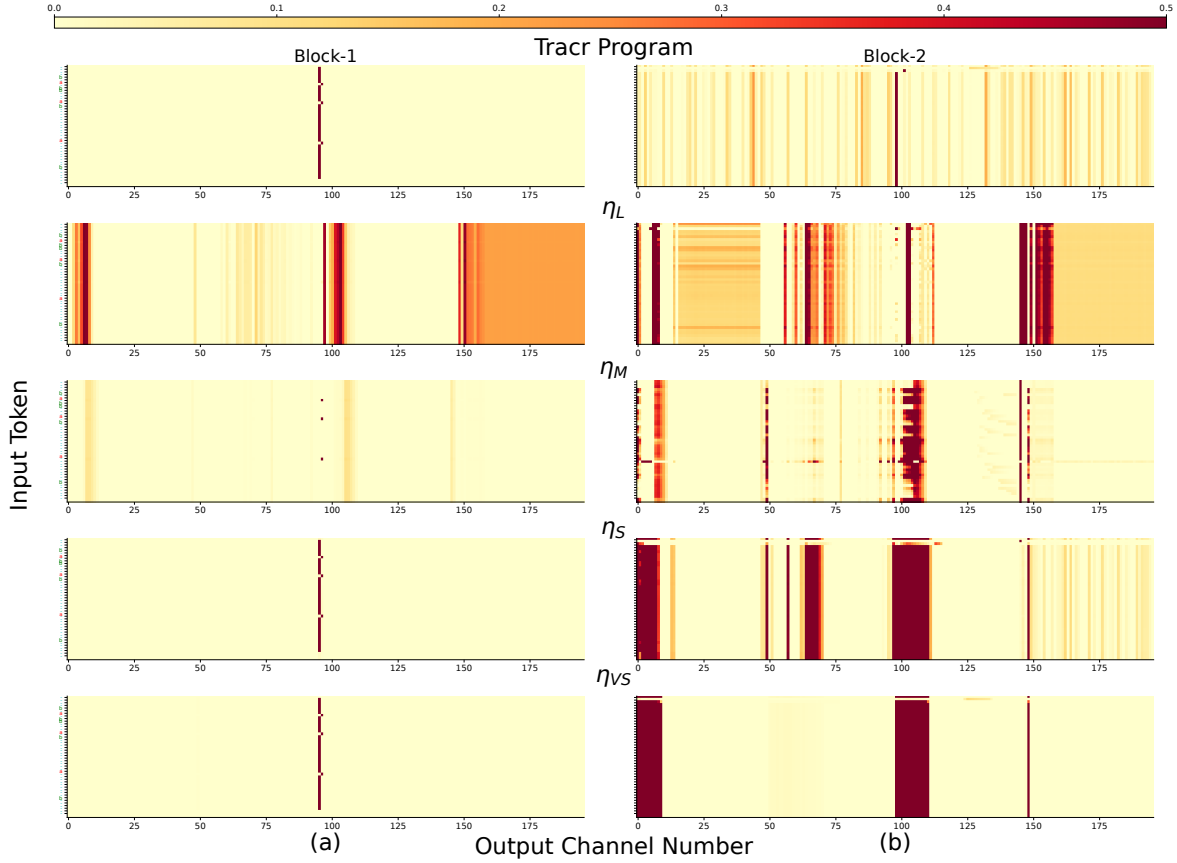


Figure D.35: Counter Task: Visualisation of the activated output of the first MLP layer in first and second blocks. This is the complete visualisation of the activation map presented in Fig. D.34.

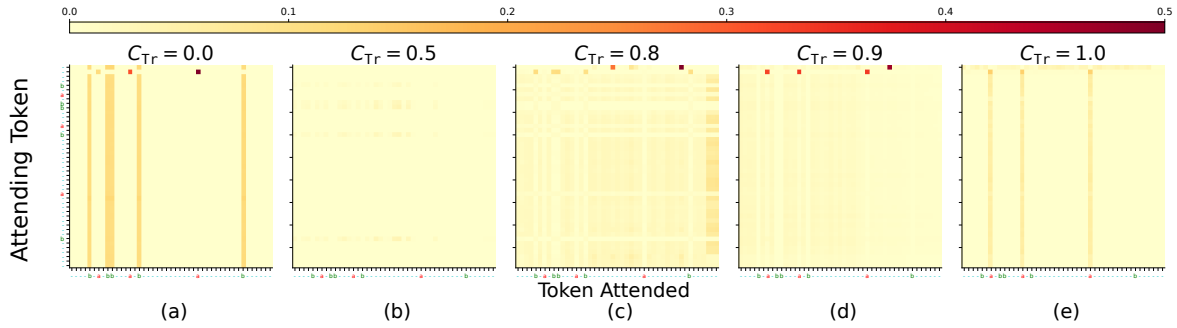


Figure D.36: Counter Task: Effect of the presence of the spuriously correlated data-points in different fractions ($C_{\mathtt{Tr}}$) in the fine-tuning dataset. The Tracr compiled model with the capability to count a's is fine-tuned to count b's on different values of $C_{\mathtt{Tr}}$. η_M is used for fine-tuning. **Observation:** On increasing the value of $C_{\mathtt{Tr}}$, the model gives lower attention to b's and in case of $C_{\mathtt{Tr}} = 1$, almost no attention is given by the model to b's.

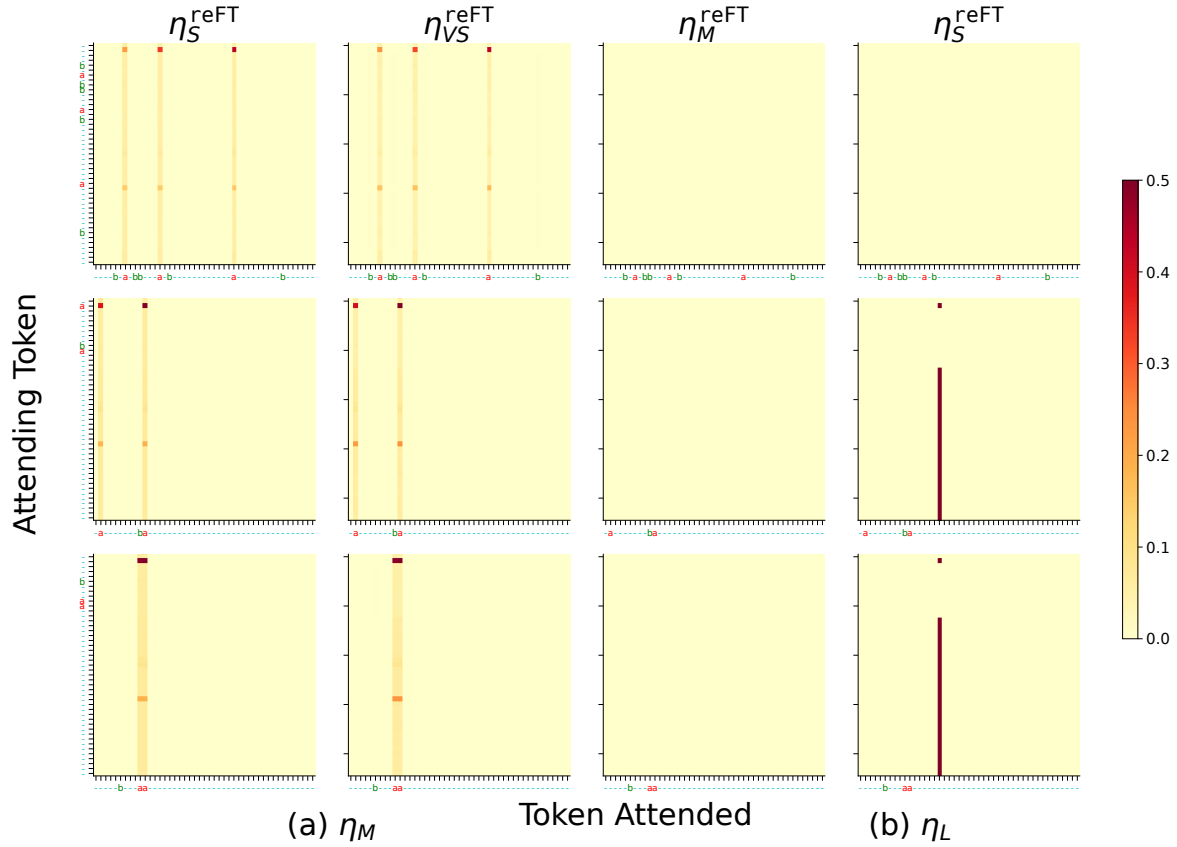


Figure D.37: Counter Task: Validation of Tracr observations on Counter task on reverse fine-tuning on three different input samples. The rows represents different input samples. **Observation:** Capability revival is possible on using η_M for fine-tuning but not on using η_L.

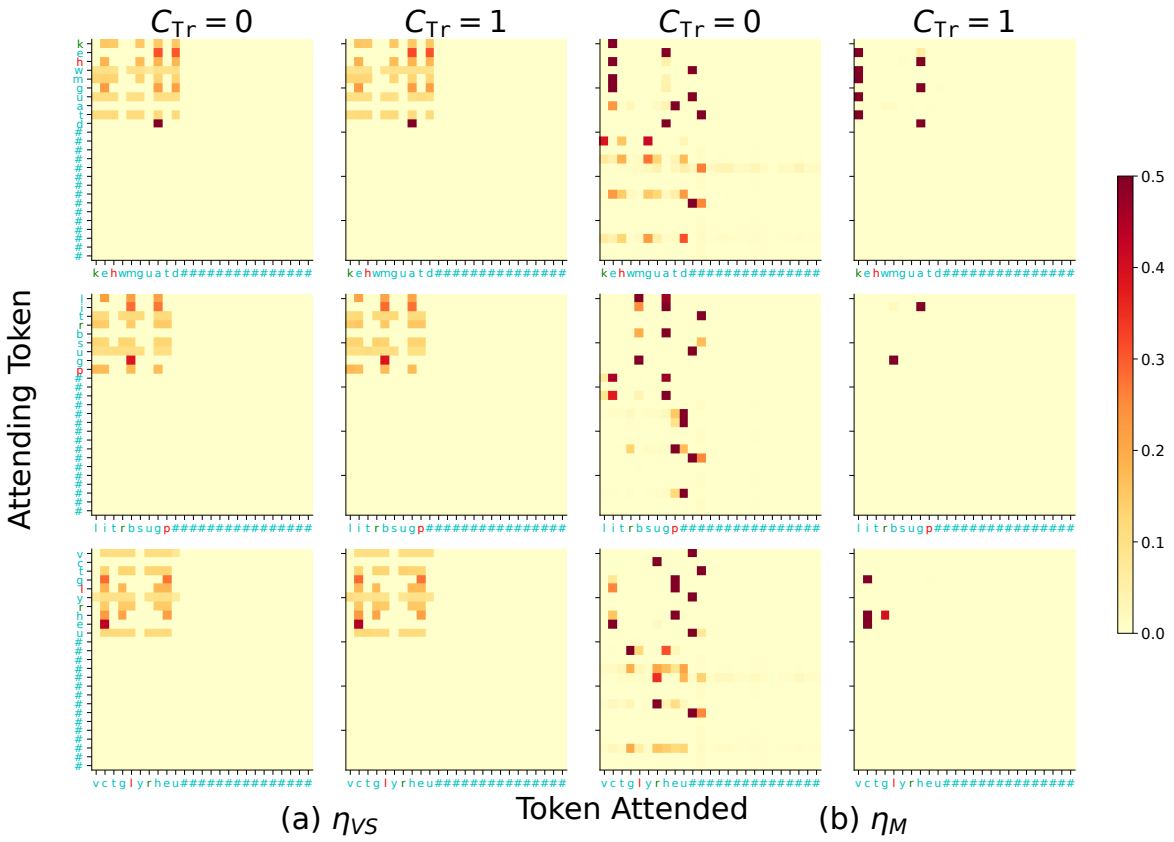


Figure D.38: Max Identifier Task: Validation of Tracr observations on max identifier task on three different input samples. The rows represents different input samples. **Observation:** Using η_{VS} preserves the original Tracr capabilities and therefore performs well on the fine-tuning task, whereas using η_M distorts the compiled capabilities resulting in poor performance on the fine-tuning task.

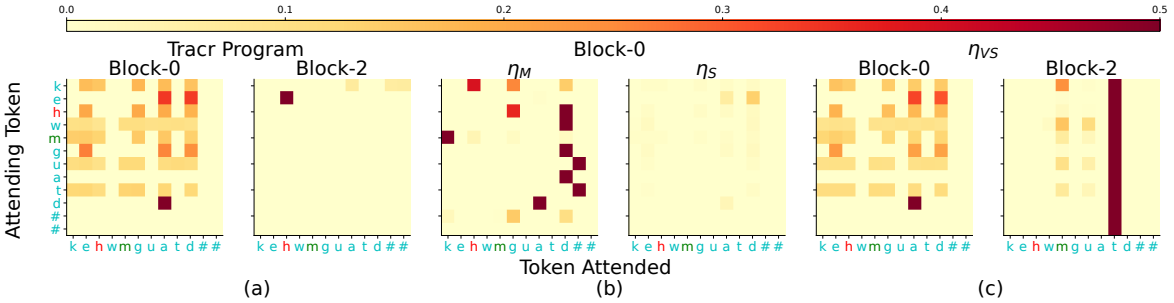


Figure D.39: Max Identifier Task: Validation of Tracr observations on max identifier task with the spurious correlation defined as the difference between the indices of fifth and seventh maximum elements being three. The rows represents different input samples. **Observation:** Using η_{VS} preserves the original Tracr capabilities and therefore performs well on the fine-tuning task, whereas using η_M distorts the compiled capabilities resulting in poor performance on the fine-tuning task.

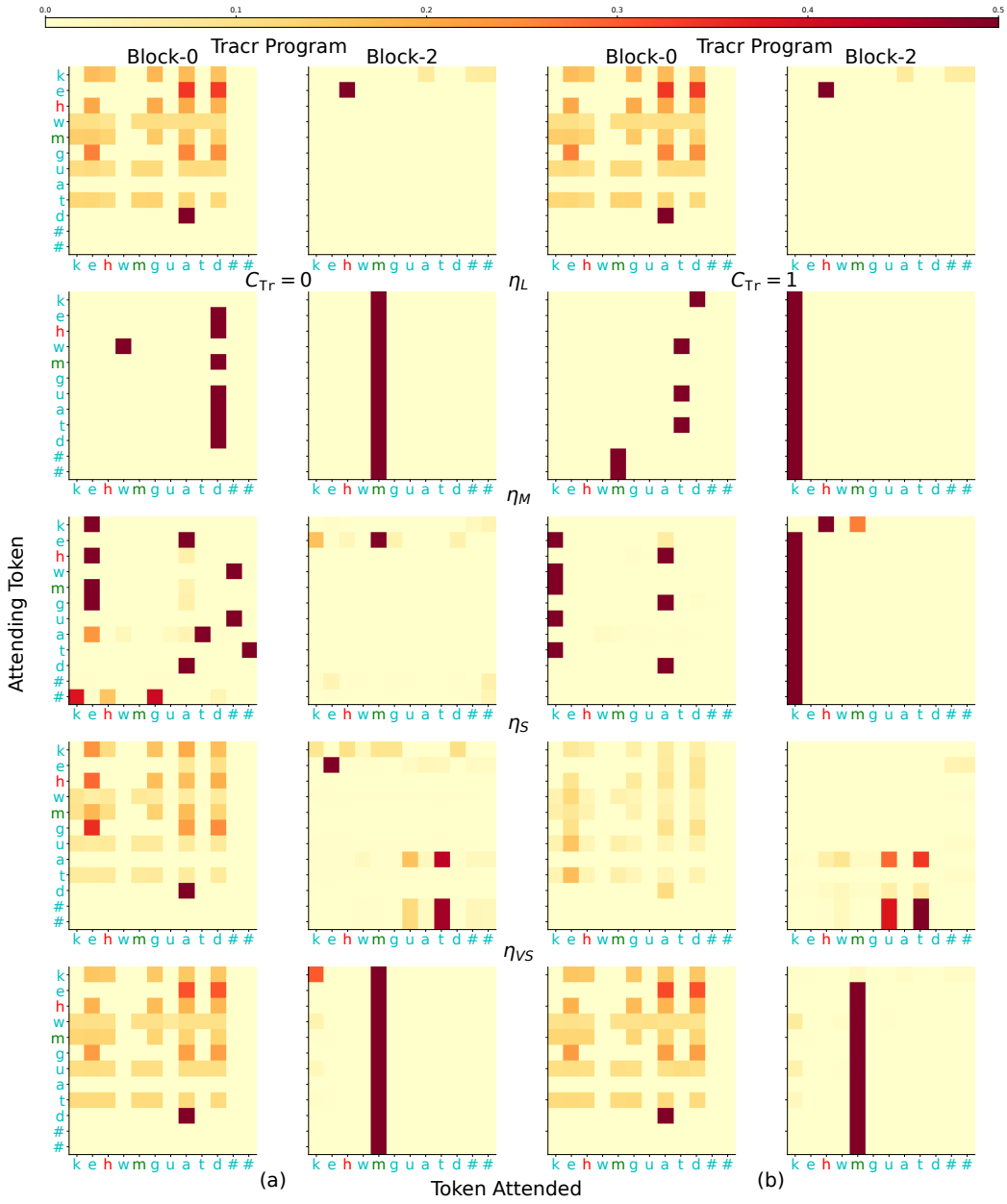


Figure D.40: Max Identifier Task: Visualisation of the attention maps of the zeroth and second blocks of the Tracr fine-tuned models on the max identifier task. (a) shows the analysis when the spurious correlation is not present in the fine-tuning datatset, whereas in case of (b) the spurious correlation is present in the fine-tuning dataset. The first row shows the maps for the Tracr compiled model and other rows shows the analysis for different learning rates. **Observation:** Using η_L, η_M or η_S for fine-tuning distorts the capability of the programmed Tracr model in the Block-0 and as a result the Block-2 attention map is not able to attend to the desired output token. Whereas using η_{VS} is able to preserve the capability and as a result the fine-tuned model is able to attend to the correct token in the attention map in Block-2.

Table D.2: Results on counting and max element task Tracr setups. The evaluation is done on test sets with and without the spurious correlation. The Tracr compiled models are fine-tuned for different learning rates and different value of $C_{\texttt{Tr}}$.

η	$C_{\texttt{Tr}}$	Counting Element				Max Identifier			
		$C_{\texttt{Te}}=1$		$C_{\texttt{Te}}=0$		$C_{\texttt{Te}}=1$		$C_{\texttt{Te}}=0$	
		Acc. $\texttt{O}_{\texttt{PT}}$	Acc. $\texttt{O}_{\texttt{FT}}$	Acc. $\texttt{O}_{\texttt{PT}}$	Acc. $\texttt{O}_{\texttt{FT}}$	Acc. $\texttt{O}_{\texttt{PT}}$	Acc. $\texttt{O}_{\texttt{FT}}$	Acc. $\texttt{O}_{\texttt{PT}}$	Acc. $\texttt{O}_{\texttt{FT}}$
10^{-1}	0.0	0.0	100.0	0.0	100.0	20.3	34.5	0.0	99.3
	0.2	0.0	100.0	0.0	100.0	0.5	92.0	0.0	97.1
	0.5	0.0	100.0	0.0	100.0	0.6	97.0	0.1	97.6
	0.6	0.0	100.0	0.0	100.0	0.3	98.5	0.0	96.7
	0.8	0.0	100.0	0.0	100.0	0.1	99.4	0.0	98.6
	0.9	0.0	100.0	0.0	100.0	0.7	98.2	0.1	92.5
	1.0	0.0	100.0	35.8	0.7	0.3	99.6	16.8	37.8
10^{-2}	0.0	1.1	96.3	0.0	98.8	29.7	0.2	16.3	52.6
	0.2	0.0	100.0	0.0	99.2	28.4	18.6	19.0	46.0
	0.5	0.6	99.4	0.0	95.9	4.8	87.9	3.3	92.6
	0.6	0.1	99.9	0.0	98.8	3.9	83.2	2.6	82.5
	0.8	0.3	99.6	0.1	97.0	4.5	88.8	6.6	72.9
	0.9	1.4	98.5	7.1	39.3	16.0	45.7	26.9	11.1
10^{-3}	1.0	0.3	98.3	4.2	0.2	11.1	78.5	23.8	14.4
	0.0	54.6	1.2	25.7	27.2	6.4	20.2	4.5	28.5
	0.2	50.2	15.0	26.5	24.3	7.4	27.4	5.4	28.0
	0.5	7.1	90.9	19.8	2.3	11.3	24.0	7.6	20.8
	0.6	4.1	94.2	11.8	2.2	11.8	26.7	8.4	20.1
	0.8	1.3	98.3	6.7	0.7	11.5	34.3	8.5	19.9
	0.9	1.8	97.8	9.2	0.7	14.6	32.2	11.4	15.8
10^{-4}	1.0	4.0	94.3	10.3	2.2	16.0	33.2	12.8	14.0
	0.0	32.6	0.0	10.6	28.7	0.5	82.6	0.5	91.1
	0.2	59.2	0.1	31.9	24.4	0.1	84.8	0.6	91.3
	0.5	28.5	65.1	37.8	5.6	0.0	89.3	0.6	91.8
	0.6	24.4	70.3	35.6	4.8	0.0	89.6	0.6	90.8
	0.8	14.1	84.2	29.7	2.1	0.0	89.7	0.6	89.9
	0.9	1.3	98.3	6.7	0.7	0.0	93.2	0.2	97.1
	1.0	1.6	98.3	10.6	0.2	0.0	90.2	0.7	88.6

Table D.3: Results on counting task Tracr for reverse fine-tuning with different learning rates. Fine-tuning was done using η_M. The evaluation is done on test sets with and without the spurious correlation. The Tracr compiled models are fine-tuned for different learning rates and different value of $C_{\texttt{Tr}}$.

η	$C_{\texttt{Tr}}$	Counting Element							
		$C_{\texttt{Te}}=1$		$C_{\texttt{Te}}=0$		$C_{\texttt{Te}}=1$		$C_{\texttt{Te}}=0$	
		Acc. $\texttt{O}_{\texttt{PT}}$	Acc. $\texttt{O}_{\texttt{FT}}$	Acc. $\texttt{O}_{\texttt{PT}}$	Acc. $\texttt{O}_{\texttt{FT}}$	Acc. $\texttt{O}_{\texttt{PT}}$	Acc. $\texttt{O}_{\texttt{FT}}$	Acc. $\texttt{O}_{\texttt{PT}}$	Acc. $\texttt{O}_{\texttt{FT}}$
10^{-1}	0.0	96.5	100.0	2.1	0.0	94.0	100.0	0.1	0.0
	0.2	98.5	100.0	0.1	0.0	94.9	100.0	0.1	0.0
	0.5	39.4	100.0	43.6	0.0	44.9	100.0	5.6	0.0
	0.6	72.9	100.0	26.7	0.0	69.4	100.0	0.2	0.0
	0.8	49.3	100.0	6.5	0.0	37.1	100.0	16.6	0.0
	0.9	31.3	100.0	64.0	0.0	34.1	100.0	1.7	0.0
	1.0	69.2	100.0	3.7	0.0	65.4	100.0	6.3	0.0
10^{-2}	0.0	63.3	99.9	36.6	0.1	65.5	98.6	0.0	0.0
	0.2	19.5	100.0	48.8	0.0	29.6	100.0	17.5	0.0
	0.5	14.3	100.0	54.4	0.0	28.9	99.9	18.7	0.0
	0.6	86.2	99.9	13.8	0.1	78.3	98.6	0.0	0.0
	0.8	65.6	100.0	1.7	0.0	43.7	99.5	10.8	0.0
	0.9	33.3	100.0	27.9	0.0	36.5	100.0	12.6	0.0
	1.0	99.0	99.6	0.9	0.3	95.2	96.8	0.1	0.1
10^{-3}	0.0	19.8	99.9	34.9	0.0	23.7	98.6	10.2	0.0
	0.2	3.9	17.1	33.8	78.0	24.4	42.9	14.7	3.0
	0.5	2.0	87.6	33.2	9.5	18.9	85.7	11.9	0.9
	0.6	7.1	99.8	35.4	0.1	22.3	97.3	15.4	0.1
	0.8	11.6	97.2	45.8	0.3	27.3	95.5	16.5	0.5
	0.9	24.2	75.2	20.3	23.6	33.5	68.8	13.1	1.5
	1.0	68.5	99.9	26.7	0.1	65.3	98.6	5.8	0.0
10^{-4}	0.0	45.2	99.9	0.1	0.1	16.9	98.5	4.9	0.0
	0.2	30.1	9.4	22.7	45.1	18.1	26.2	17.9	19.5
	0.5	30.1	3.2	22.7	41.3	15.7	26.1	17.9	16.2
	0.6	54.1	0.0	45.9	35.9	0.0	27.8	25.7	11.9
	0.8	26.8	83.0	64.5	10.9	3.0	76.8	49.2	1.3
	0.9	27.9	85.0	61.5	8.4	2.1	80.9	44.2	1.1
	1.0	45.2	99.6	31.9	0.3	42.2	96.8	12.6	0.0

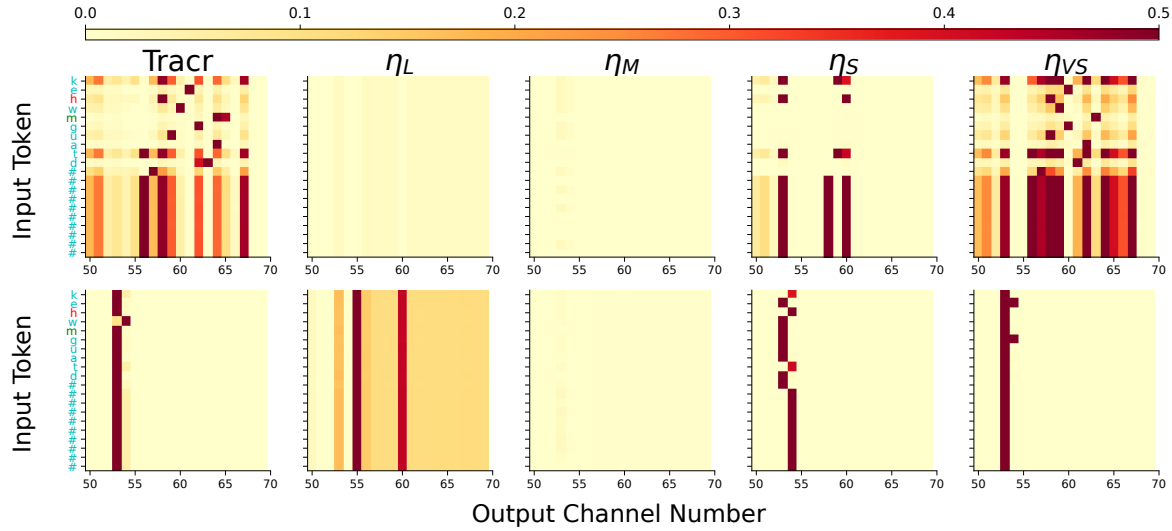


Figure D.41: Max Identifier Task: Visualisation of the activated output of the first MLP layer in first and second blocks for the max identifier task. The visualisation is shown only for channel numbers 50-70. **Observation:** Using η_{VS} for fine-tuning, which enables the model to learn the fine-tuning task, preserves the Tracr compiled model's compiled capability of sorting tokens in Block-1. Whereas other learning rates are not able to preserve this capability.

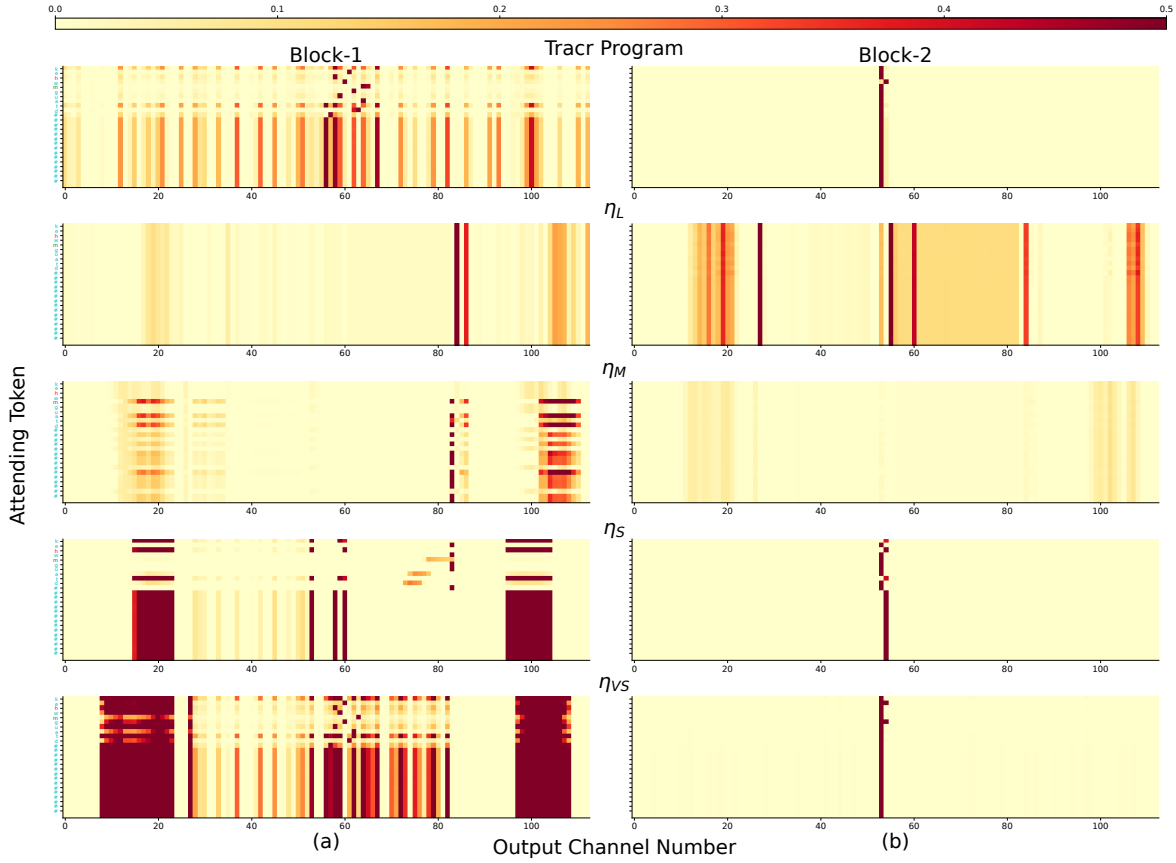


Figure D.42: Max Identifier Task: Visualisation of the activated output of the first MLP layer in first and second blocks. This is the complete visualisation of the activation map presented in Fig. D.41.

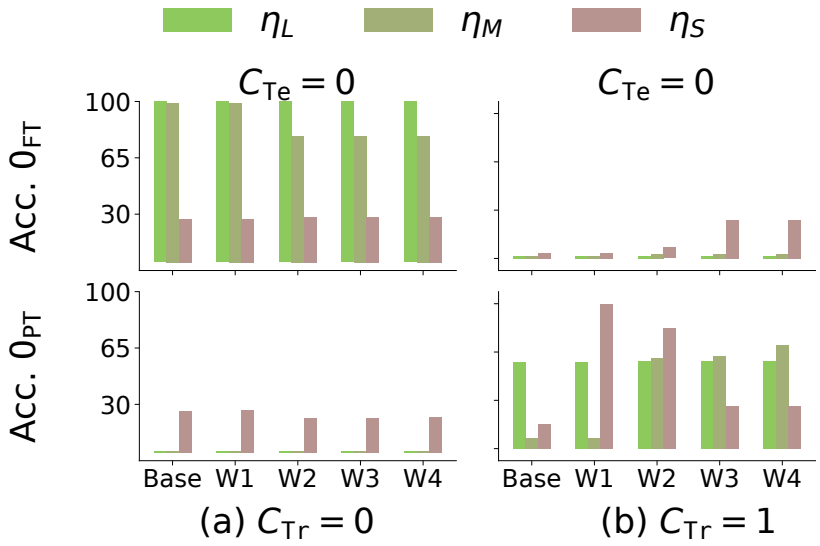


Figure D.43: Counter Task: Pruning evaluation on Tracr model fine-tuned to count `b`'s. Observation: Higher value of $C_{\texttt{Tr}}$ leads to the learning of the wrapper on top of the Tracr compiled capability. This wrapper is learned on using η_M and η_S and can be localised in a few weights of the model.

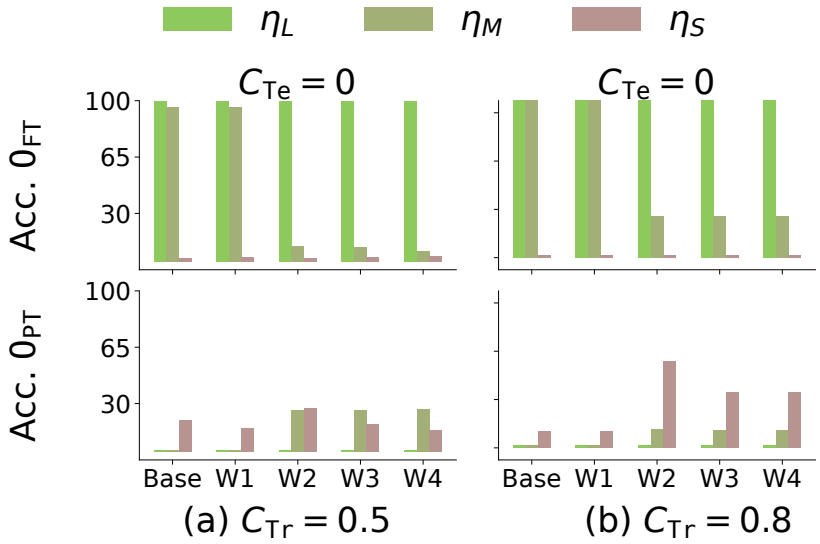


Figure D.44: Counter Task: Pruning evaluation on Tracr model fine-tuned to count `b`'s. Observation: Observations are consistent with Fig. 5.9.

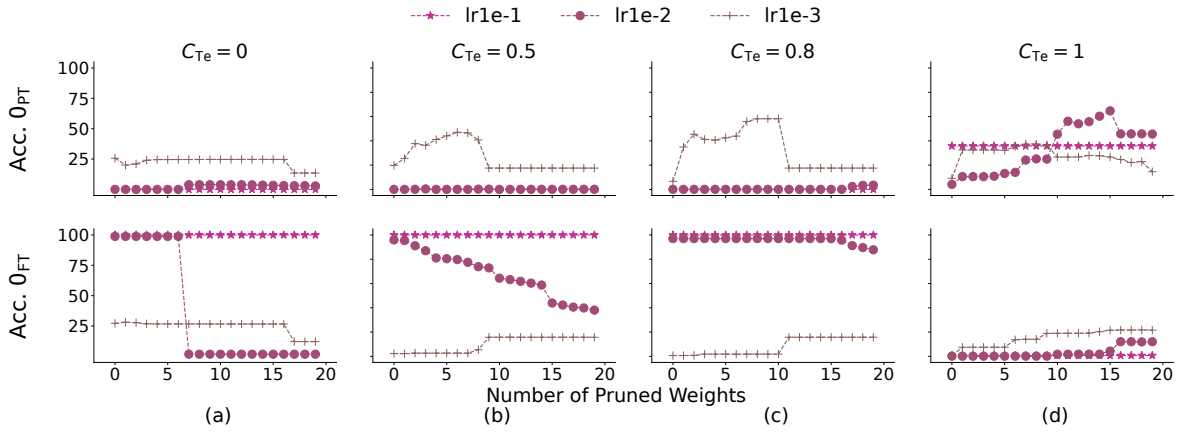


Figure D.45: Counter Task, weight pruning: Pruning weights of the Tracr model fine-tuned to count `b`'s using different learning rates. Observation: In the presence of spurious correlation the model learns a wrapper when learning rates η_M and η_S are used. This wrapper can be localised in a few weights of the model.

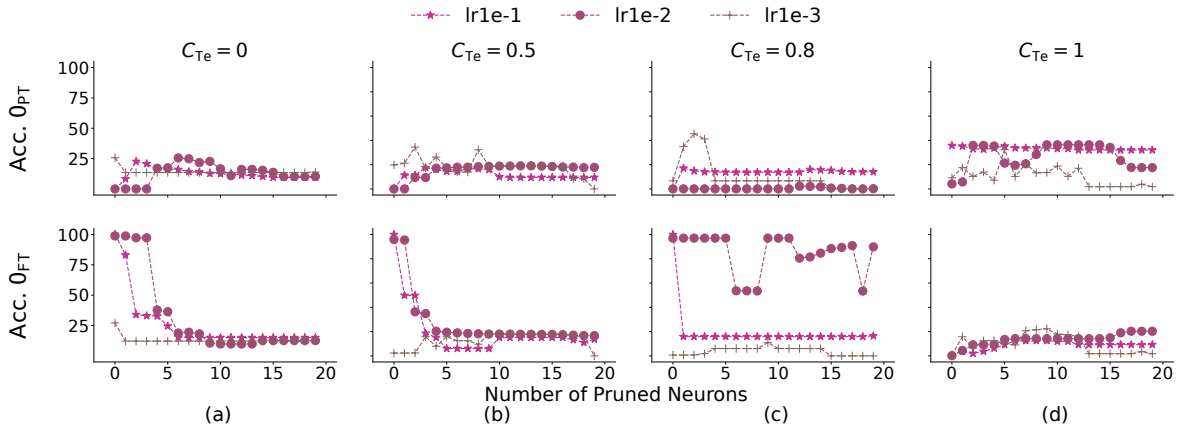


Figure D.46: Counter Task, neuron pruning: Pruning neurons of the Tracr model fine-tuned to count b's using different learning rates. Observation: In the presence of spurious correlation the model learns a wrapper in case of η_M and η_S. This wrapper can be localised in a few weights of the model.

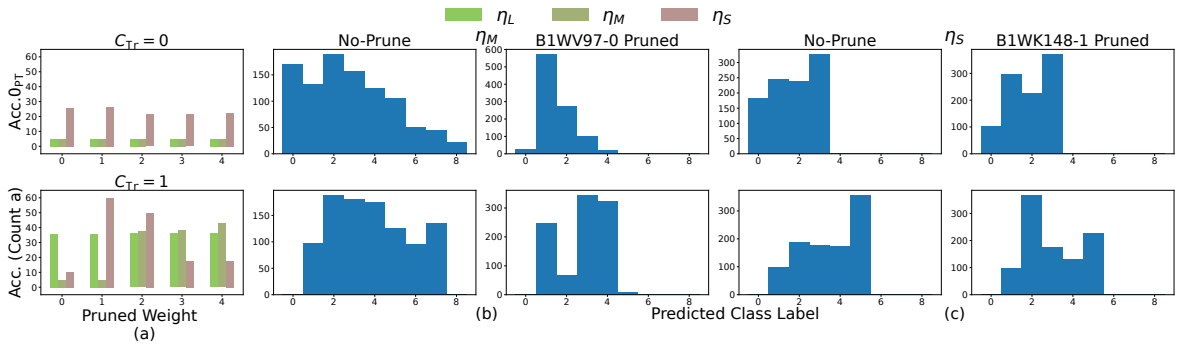


Figure D.47: Counter Task: Effect of pruning a single weight on the distribution of predicted class labels. Observation: Even after pruning, the model predicts different classes indicating that the gain in accuracy on pruning is indeed because the model has removed the wrapper.

D.8 Additional PCFG results

In this section, I provide a detailed analysis of the PCFG results on the counter task. More specifically, I analyse the effect of the presence of weakly and strongly relevant capability in the model across three different parameters: training iterations (n_{iters}), fraction of fine-tuning data samples with spurious correlation ($C_{\texttt{Tr}}$) and the probability of sampling operand token (0) to be a during pre-training ($\mathscr{P}_{\texttt{T}}$). $\mathscr{P}_{\texttt{T}}$ essentially determines whether the capability present in the pre-trained model is strongly relevant or weakly relevant for the fine-tuning task. Additionally I also analyse the effect of using the spuriously correlated ($C_{\texttt{Te}} = 1$) and uncorrelated test set ($C_{\texttt{Te}} = 0$) for evaluation of the fine-tuned model. I present the summary of the results in Tables D.4 and D.5 for the counting task and Tables D.5 and D.6 for the index of occurrence tasks. Then I discuss the effect of learning rate on fine-tuning pre-trained models with weakly and strongly relevant capabilities in Figs. D.48 and D.49. I observe that the observations are consistent for the considered counting and index of occurrence tasks. Next I analyse the effect of the presence of weakly and strongly relevant capability in the pre-trained model for different fractions of spuriously correlated data-points ($C_{\texttt{Tr}}$) and different pre-training iterations (n_{iters}) in Figs. D.50, D.52 and D.54 for the counting element task and Figs. D.51, D.53 and D.55 for the index of occurrence task. I observe that the observations are fairly consistent across both the tasks and different values of n_{iters}. Next I present the effect of the spuriously correlated data and presence of weakly and strongly correlated capabilities on the learning of the wrapper in Figs. D.56 and D.57 on using uncorrelated test set for evaluation on counting and index of occurrence tasks respectively. Similar analysis on using test set with spuriously correlated samples is present in Figs. D.58 and D.59. I present the capability revival analysis on the Counter task for $n_{iters} = 200K$ and $n_{iters} = 50K$ pre-trained models for weakly and strongly relevant capability fine-tuned models in Fig. D.64 and Fig. D.65 respectively. A similar analysis for different number of pre-training iterations is present in Fig. D.66.

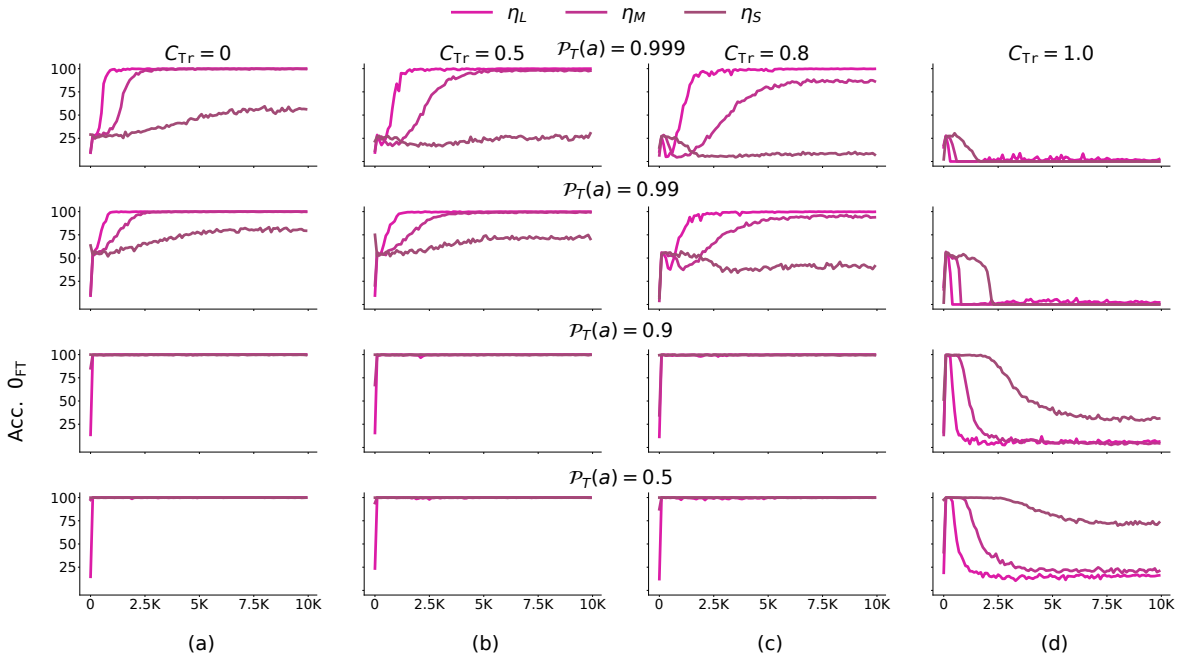


Figure D.48: Counter Task, $n_{iters} = 200K$, $C_{\text{Te}} = 0$: Effect of learning rate (LR) on fine-tuning pre-trained models with weakly and strongly relevant capabilities and using different values of C_{Tr} for fine-tuning. **Observation:** In the presence of strongly relevant capability, training with η_S yields good performance on the fine-tuning dataset. The convergence time to learn the fine-tuning task increases with an increase in C_{Tr}.

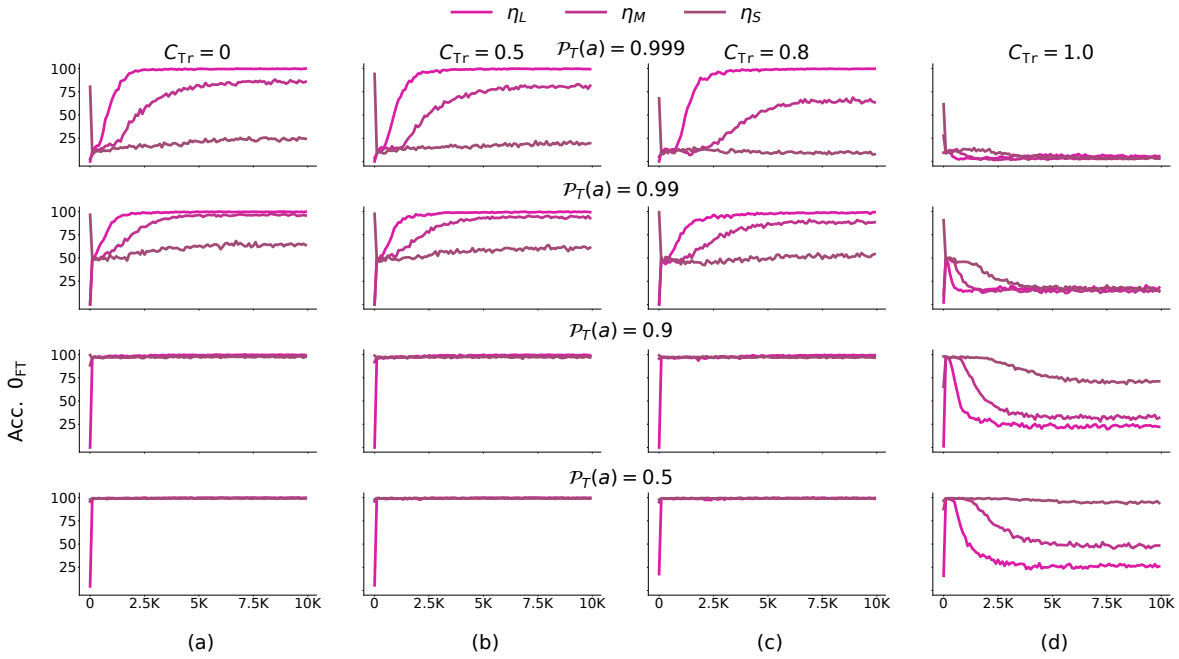


Figure D.49: Index of Occurrence Task, $n_{iters} = 200K$, $C_{\text{Te}} = 0$. The settings are consistent with Fig. D.48

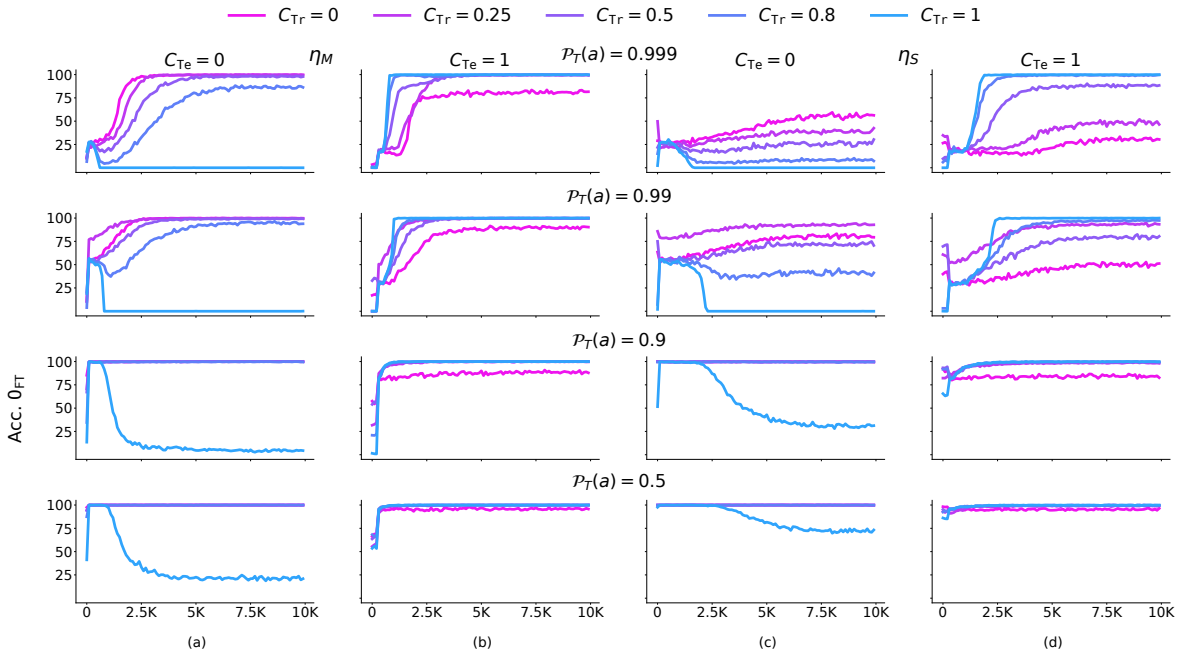


Figure D.50: Counter Task, $n_{iters} = 200K$: Effect of the presence of strongly or weakly relevant pretuning capability on fine-tuning performance on using η_M and η_S. Test sets with and without the spurious correlations are used for evaluation. **Observation:** The convergence time for learning the fine-tuning task in the absence of strongly relevant capability is higher as compared to when the strongly relevant is present in the model. The time further increases if spurious correlations are present in the fine-tuning set. However, in the presence of spurious correlations, the convergence time to learn the spurious correlation is small and is possible even on using the learning rate η_S. Using η_S is unable to yield learning of the fine-tuning task if a weakly relevant capability is present in the model.

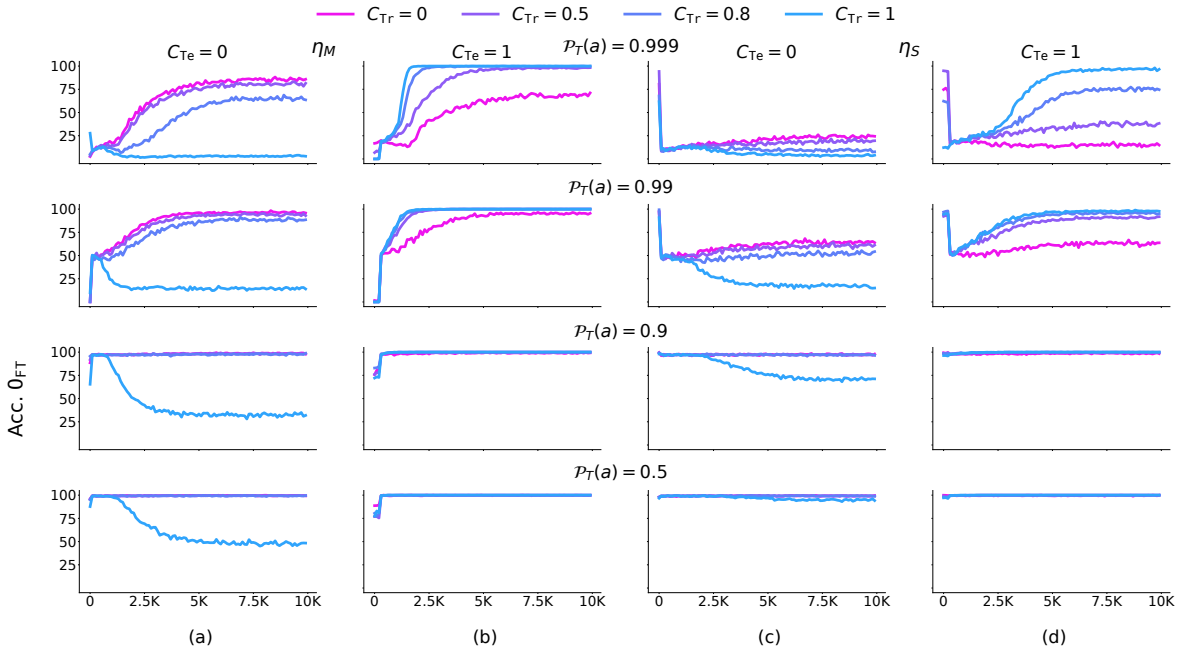


Figure D.51: Index of Occurrence Task, $n_{iters} = 200K$: The settings are consistent with Fig. D.50.

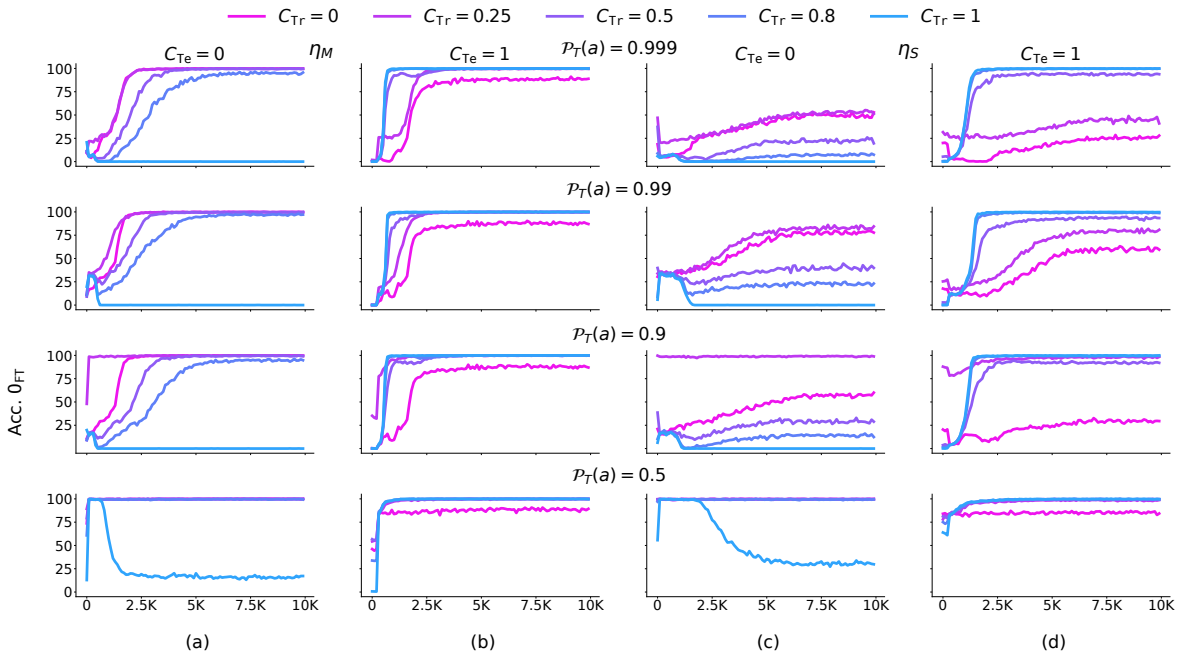


Figure D.52: Counter Task, $n_{iters} = 50K$: Effect of the presence of strongly or weakly relevant pretuning capability on fine-tuning performance on using η_M and η_S. Test sets with and without the spurious correlations are used for evaluation. The observations are consistent with Fig. D.50.

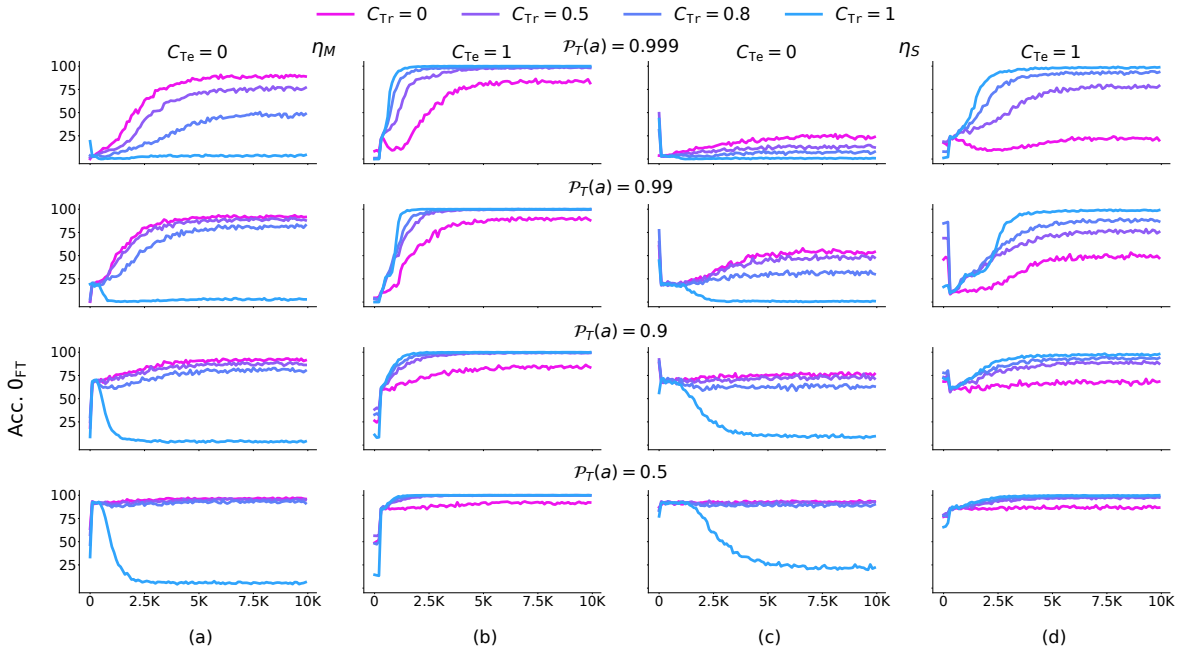


Figure D.53: Index of Occurrence Task, $n_{iters} = 50K$: The settings are consistent with Fig. D.52.

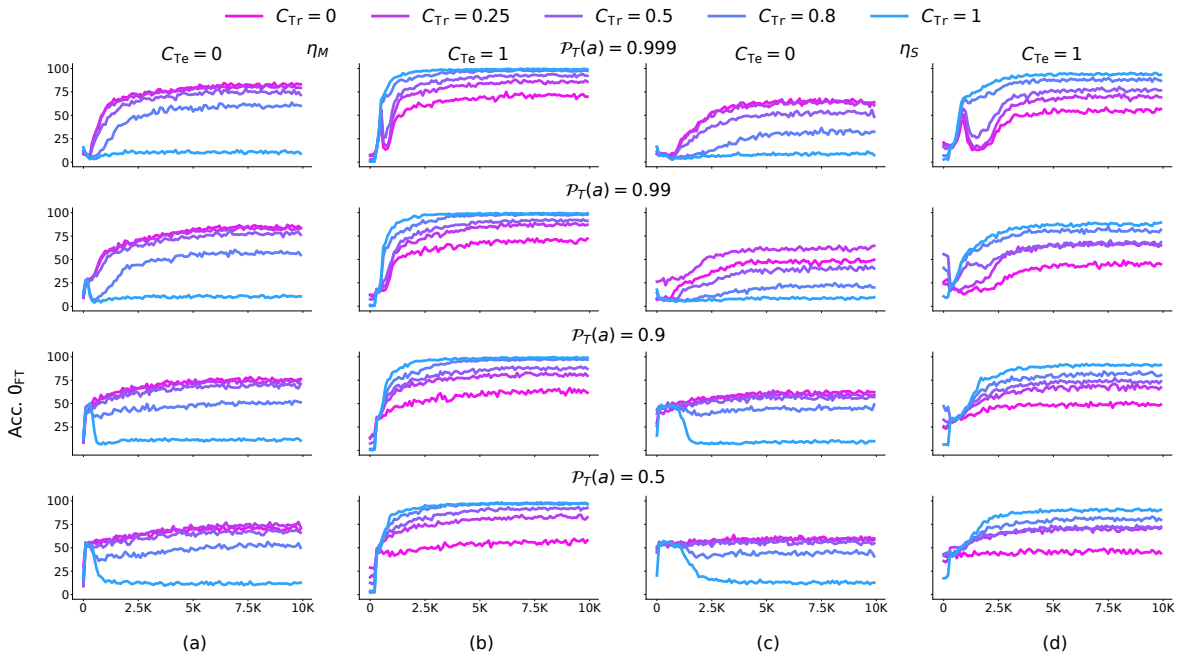


Figure D.54: Counter Task, $n_{iters} = 10K$: Effect of the presence of strongly or weakly relevant pretuning capability on fine-tuning performance on using η_M and η_S. Test sets with and without the spurious correlations are used for evaluation. The observations are consistent with Fig. D.50.

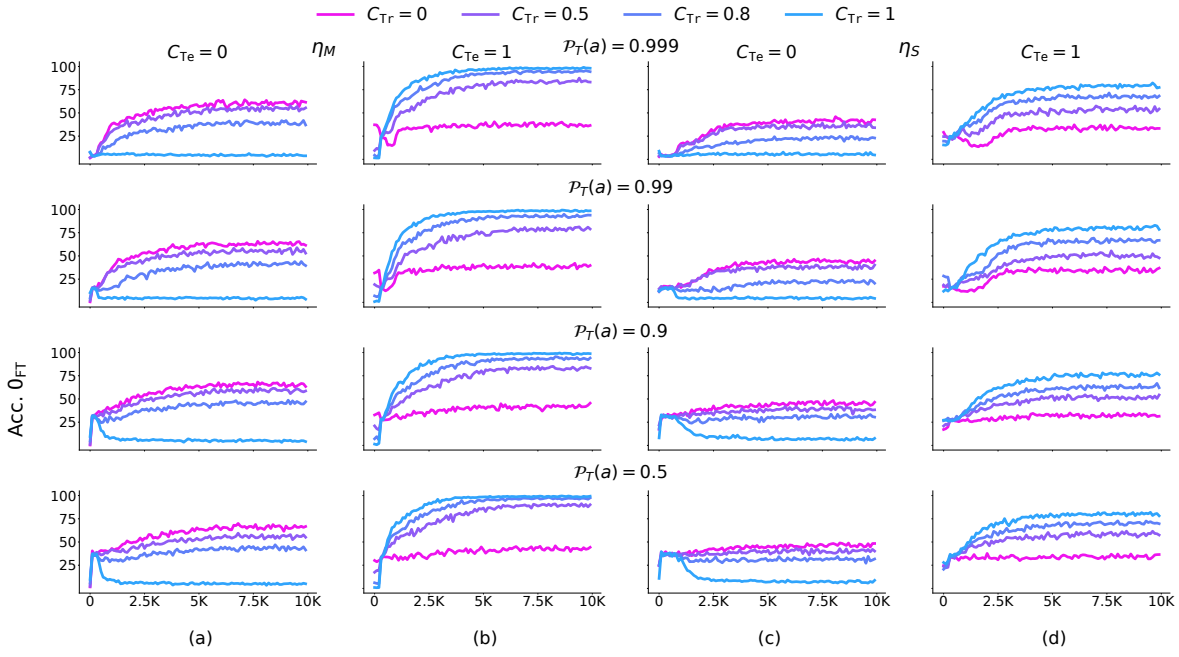


Figure D.55: Index of Occurrence Task, $n_{iters} = 10K$: The settings are consistent with Fig. D.54.

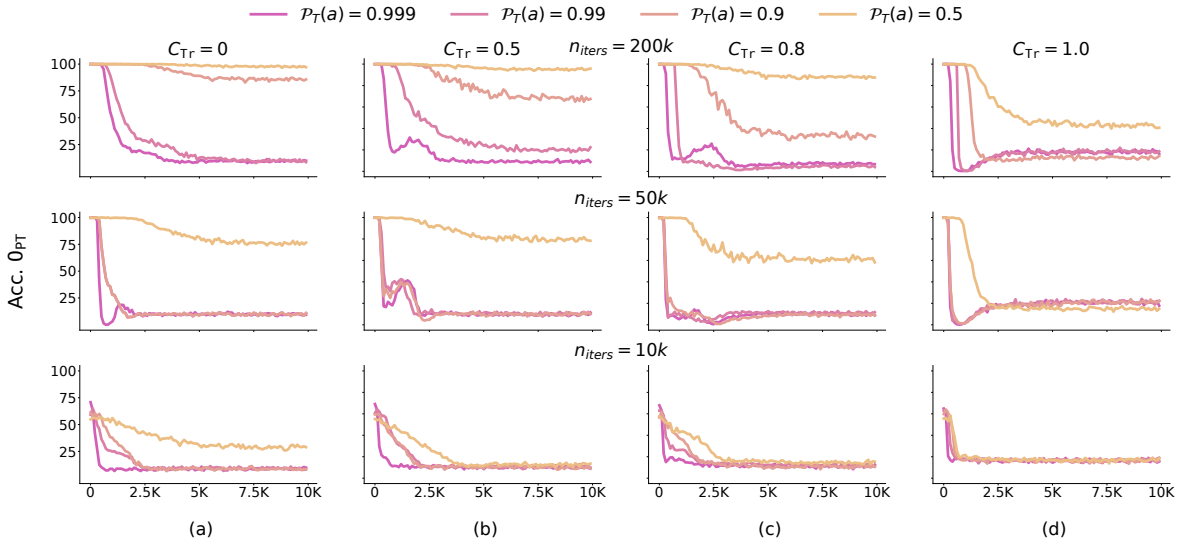


Figure D.56: Counter Task, η_M, $C_{\mathrm{Te}} = 0$: Learning of the wrapper in presence of different fraction of spuriously correlated data, values of $\mathscr{P}_{\mathrm{T}}(\mathrm{a})$ during pre-training, and training iterations. **Observation:** Using a higher fraction of spuriously correlated data in the fine-tuning set (higher value of C_{Tr}) leads to faster degradation in the pre-training accuracy. Further this degradation is even faster in presence of weakly relevant capability.

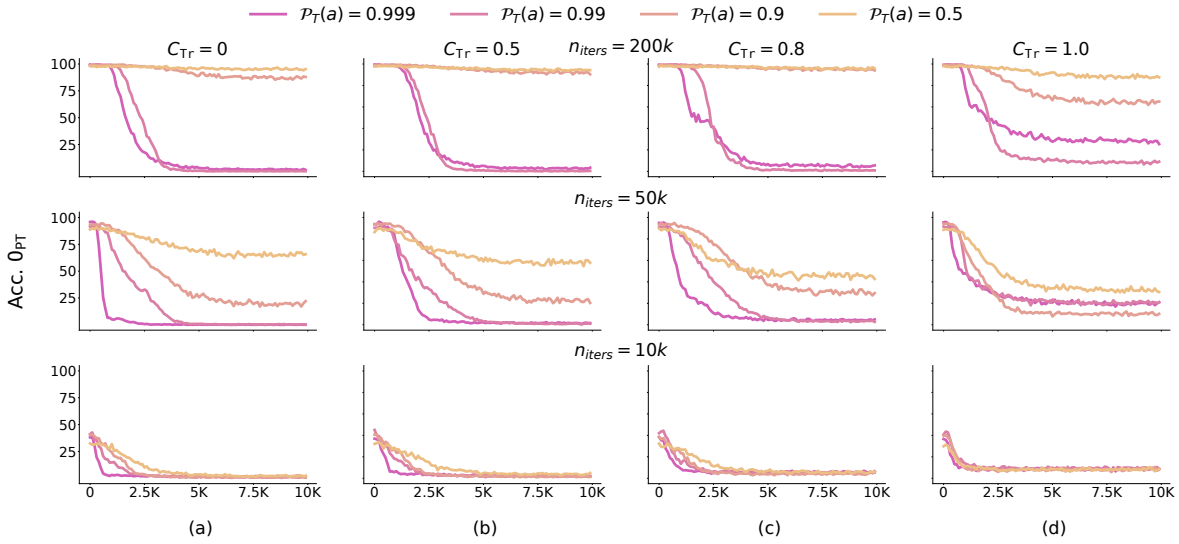


Figure D.57: Index of Occurrence Task, Medium LR, $C_{\mathrm{Te}} = 0$: The settings are consistent with Fig. D.56.

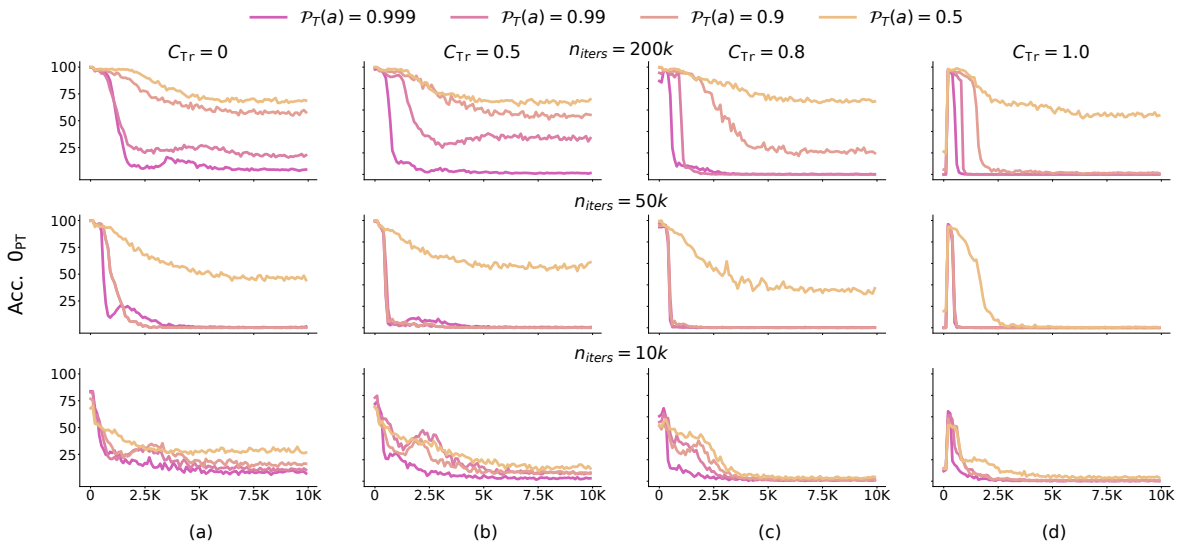


Figure D.58: Counter Task, η_M, $C_{\mathrm{Te}} = 1$: Learning of the wrapper in presence of different fraction of spuriously correlated data, values of $\mathscr{P}_{\mathrm{T}}(\mathrm{a})$ during pre-training, and training iterations. **Observation:** Using a higher fraction of spuriously correlated data in the fine-tuning set (higher value of C_{Tr}) leads to faster degradation in the pre-training accuracy. Further this degradation is even faster in presence of weakly relevant capability.

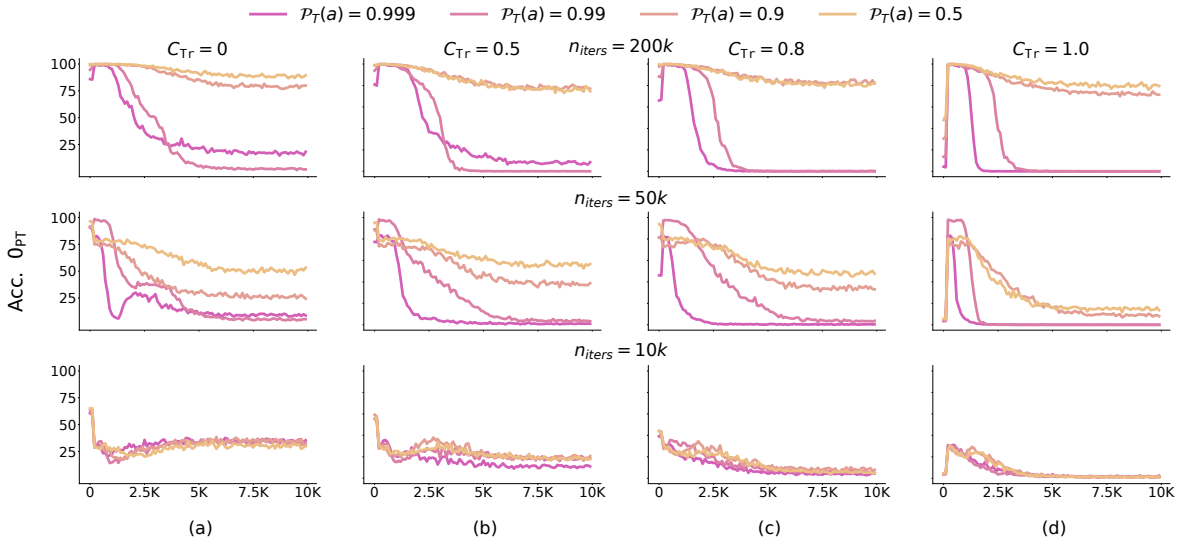


Figure D.59: Index of Occurrence Task, Medium LR, $C_{\mathrm{Te}} = 1$: The settings are consistent with Fig. D.58.

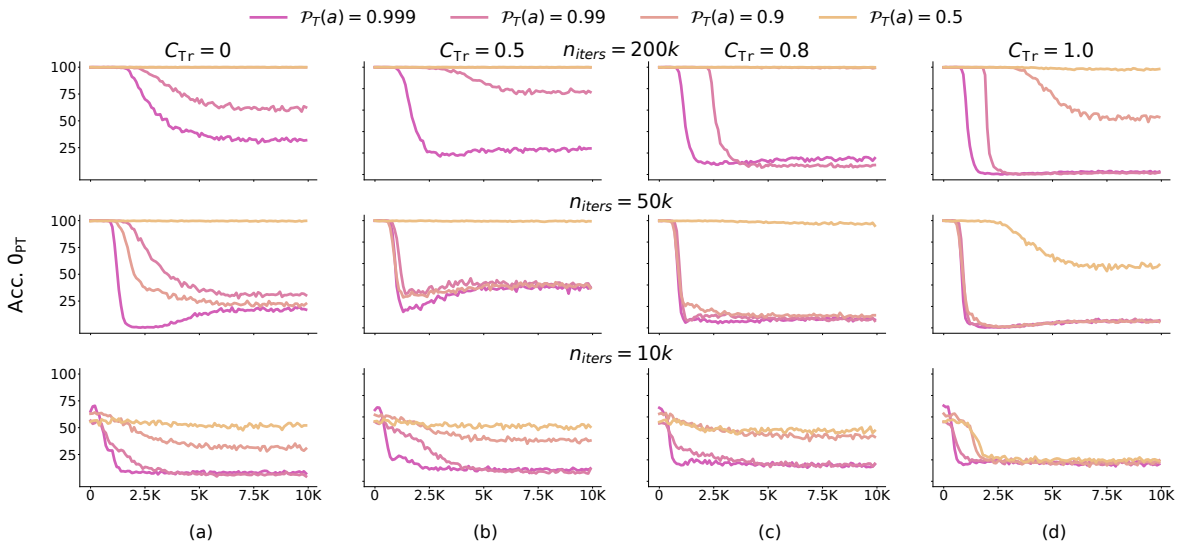


Figure D.60: Counter Task, η_S, $C_{\text{Te}} = 0$: Learning of the wrapper in presence of different fraction of spuriously correlated data, values of $\mathscr{P}_{\text{T}}(\text{a})$ during pre-training, and training iterations. **Observation:** The observations are consistent with Fig. D.56

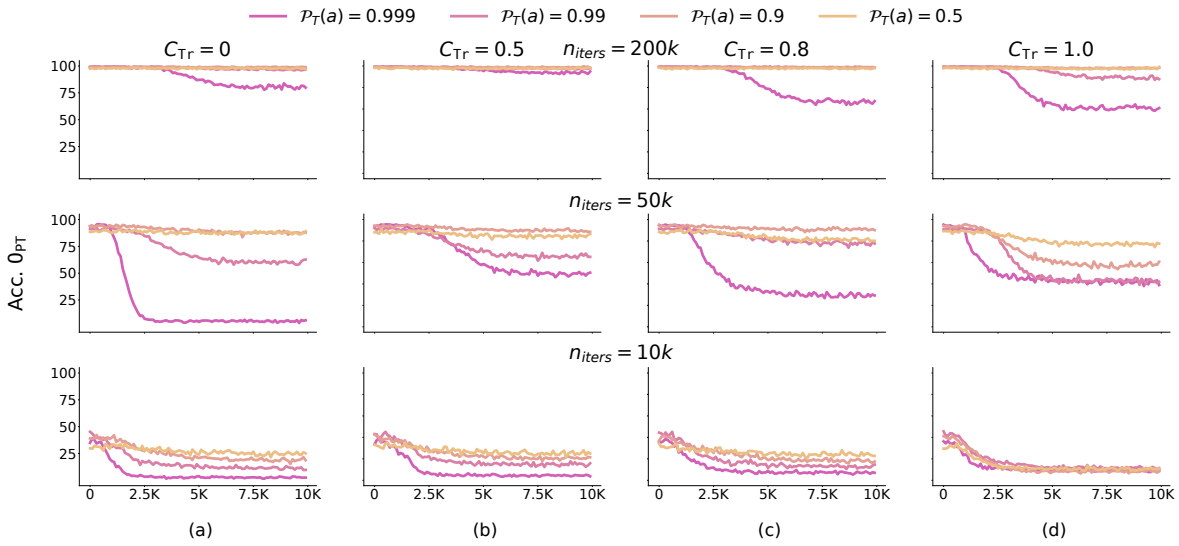


Figure D.61: Index of Occurrence Task, η_S, $C_{\text{Te}} = 0$: The settings are consistent with Fig. D.60

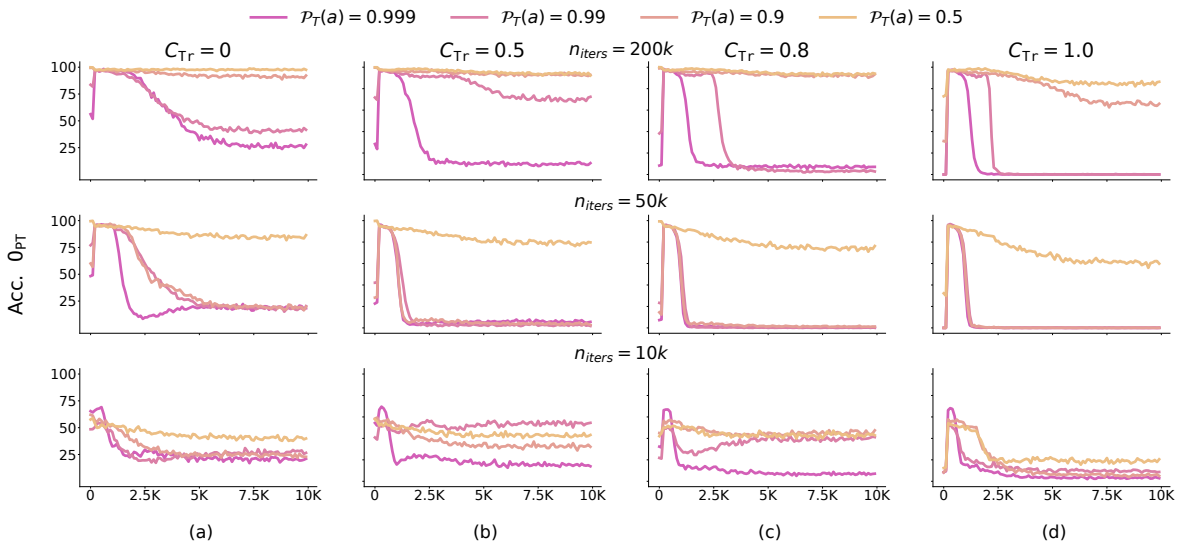


Figure D.62: Counter Task, η_S, $C_{\text{Te}} = 1$: Learning of the wrapper in presence of different fraction of spuriously correlated data, values of $\mathscr{P}_{\text{T}}(\text{a})$ during pre-training, and training iterations. **Observation:** The settings are consistent with Fig. D.58

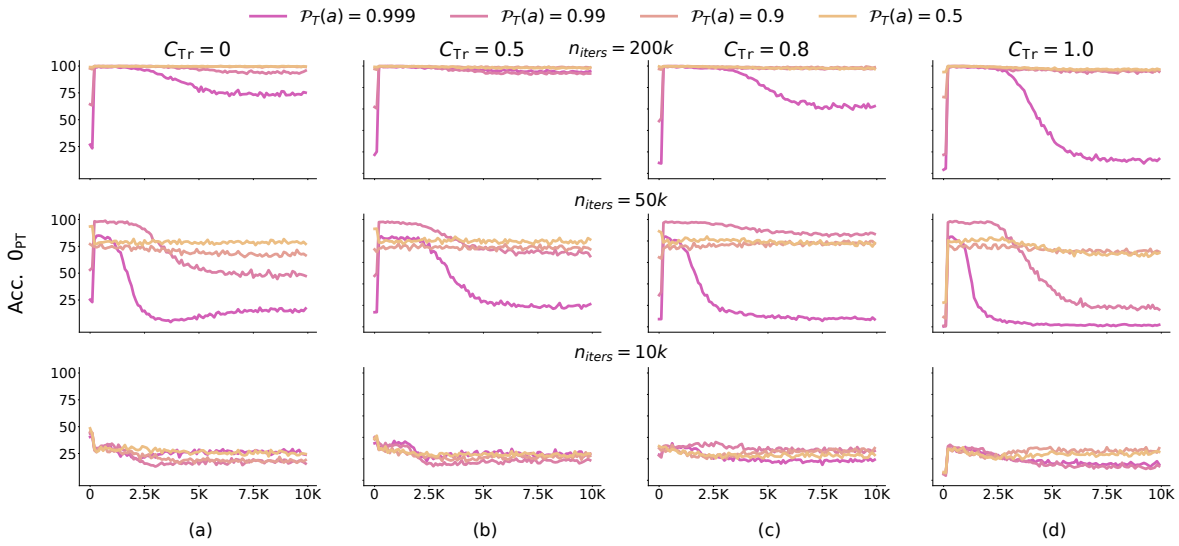


Figure D.63: Index of Occurrence Task, η_S, $C_{\text{Te}} = 1$: The settings are consistent with Fig. D.62

Table D.4: Results on the PCFG counting task with 200K iterations of pre-training.

η	$\mathscr{P}_{\mathtt{T}}(\mathtt{a})$	$C_{\mathtt{Tr}}$	Acc $\mathtt{O}_{\mathtt{PT}}$					Acc $\mathtt{O}_{\mathtt{FT}}$				
			Acc PT	$C_{\mathtt{Te}}=0$		$C_{\mathtt{Te}}=1$		Acc PT	$C_{\mathtt{Te}}=0$		$C_{\mathtt{Te}}=1$	
				$1K$	$10K$	$1K$	$10K$		$1K$	$10K$	$1K$	$10K$
10^{-4}	0.999	0.0	100.0	9.7	9.5	14.5	0.0	27.1	99.4	100.0	74.9	87.5
		0.5		7.2	9.7	1.3	0.0		75.9	99.9	95.9	100.0
		0.8		5.6	10.8	0.2	0.0		60	99.8	98.1	100.0
		1.0		17.2	15.2	0.2	0.0		0.0	1.6	98.9	100.0
	0.9	0.0	100.0	99.9	92.6	13.6	67.2	99.9	99.8	100.0	100.0	92.8
		0.5		100.0	90.7	15.6	73.5		99.8	99.4	99.9	100.0
		0.8		99.9	43.4	11.4	33.2		99.4	99.2	100.0	100.0
		1.0		99.8	15.9	16.5	2.4		99.6	9.4	6.1	100.0
	0.5	0.0	99.9	98.9	14.7	76.6	44.1	99.9	99.9	100.0	87.6	99.2
		0.5		95.1	23.6	71.2	33.4		99.7	100.0	99.8	99.9
		0.8		92.8	12.2	79.8	2.2		99.9	99.9	98.7	99.9
		1.0		49.5	19	60.6	0.0		25.8	16	99.8	100.0
10^{-5}	0.999	0.0	100.0	48.9	10.2	51.9	4.6	27.1	39.8	99.8	13.9	79.7
		0.5		19.7	11.6	12.3	1.2		18.4	98.1	81.4	99.7
		0.8		12.1	6.6	7.7	0.2		6.1	85.7	98.4	99.7
		1.0		0.4	17.5	0.0	0.0		0.0	0.0	99.9	100.0
	0.9	0.0	100.0	100.0	85.3	94.8	56.9	99.9	99.8	99.9	83.3	87.3
		0.5		99.9	67.2	94.9	55.4		99.9	99.9	99.3	99.8
		0.8		100.0	34.6	94.8	21.7		99.8	99.4	99.7	100.0
		1.0		98.5	13.5	88.6	0.8		58.3	3.6	99.8	100.0
	0.5	0.0	100.0	100.0	97.5	97.5	65.7	99.9	100.0	100.0	95.6	95.4
		0.5		99.9	94.1	98.1	69		100.0	100.0	99.3	100.0
		0.8		99.9	87.4	93.8	67.7		99.8	100.0	100.0	100.0
		1.0		99.6	41.2	91.8	53.3		90.1	19.6	99.8	100.0
10^{-6}	0.999	0.0	100.0	100.0	29	96.6	25.7	27.1	28.5	51.8	15.1	29.2
		0.5		98.7	21.8	88.9	10.6		23.3	23.7	20.3	87.5
		0.8		83	15.1	69.7	6.8		18.4	8.9	26.5	99.7
		1.0		71.7	2.3	56	0.0		15.7	0.0	29.5	99.9
	0.9	0.0	100.0	100.0	100.0	95.4	91.8	99.9	99.8	99.5	84.1	84.6
		0.5		100.0	99.9	96	94.4		99.6	99.5	95.9	99.6
		0.8		99.8	99.4	95.9	92.6		99.6	99.3	94.8	99.6
		1.0		99.8	51.6	95.1	63.9		99.5	30.5	94.2	99.7
	0.5	0.0	99.9	100.0	99.9	97.7	98.1	99.9	99.8	99.8	95.4	95.6
		0.5		100.0	99.9	97.9	93.7		100.0	99.9	98.5	99.6
		0.8		100.0	100.0	98.8	93.1		100.0	99.9	98.3	99.9
		1.0		100.0	97.6	98.6	85.1		99.8	73.9	98.3	100.0

Table D.5: Results on the PCFG counting task with 50K iterations of pre-training.

η	$\mathscr{P}_{\mathtt{T}}(\mathtt{a})$	$C_{\mathtt{Tr}}$	Acc $\mathtt{O}_{\mathtt{PT}}$					Acc. $\mathtt{O}_{\mathtt{FT}}$				
			Acc. PT	$C_{\mathtt{Te}}=0$		$C_{\mathtt{Te}}=1$		Acc. PT	$C_{\mathtt{Te}}=0$		$C_{\mathtt{Te}}=1$	
				$1K$	$10K$	$1K$	$10K$		$1K$	$10K$	$1K$	$10K$
10^{-4}	0.999	0.0	99.9	10.8	9.1	2	0.1	5.17	98.9	100.0	86.3	93.6
		0.5		11.7	8.9	2.1	0.1		90.2	99.9	97.9	99.8
		0.8		5.5	11	0.0	0.0		64.9	100.0	98.9	99.9
		1.0		20.2	15.9	0.0	0.0		0.0	1.9	99.9	100.0
	0.9	0.0	99.9	9.1	10.3	0.7	0.1	15.8	99.6	100.0	84.2	94.4
		0.5		11.4	10.6	1.5	0.0		93.2	99.9	97.9	100.0
		0.8		4.2	9.2	0.1	0.0		63.1	100.0	97.8	100.0
		1.0		18.4	16.4	0.0	0.0		0.5	5	100.0	99.9
	0.5	0.0	99.8	87.7	10.1	62.6	0.0	99.7	99.9	100.0	89.3	93.5
		0.5		90.1	9.4	67.4	0.0		99.5	100.0	99.9	100.0
		0.8		59.5	10.1	29.3	0.1		99.2	99.9	100.0	99.9
		1.0		18.7	15.2	5.1	0.0		17.4	14.2	100.0	100.0
10^{-5}	0.999	0.0	99.9	3	9.3	16.6	0.6	5.17	32.6	99.8	12.4	88.7
		0.5		30.9	11.4	4.1	0.6		12.7	99.1	93	99.7
		0.8		6.3	10.8	1.0	0.1		1.4	93.9	99	99.8
		1.0		2.1	20.7	0.2	0.0		0.0	0.0	99.8	99.9
	0.9	0.0	99.9	28	10.9	34.7	0.1	15.8	39.6	99.8	23.4	88.6
		0.5		33.2	8.9	4.2	0.0		22.7	99.6	92.8	100.0
		0.8		13.1	9.3	2.9	0.0		9.4	95.1	98.9	100.0
		1.0		1.9	19.8	0.3	0.0		0.0	0.1	99.7	100.0
	0.5	0.0	99.8	99.6	73.7	88.1	46	99.7	99.9	99.9	86.4	89.2
		0.5		99.6	79.3	82.7	57.1		99.8	99.9	99.1	99.9
		0.8		99.6	60.8	80	33.6		99.5	99.9	99.3	100.0
		1.0		81.7	12.9	68.6	0.5		46.4	16.1	98.8	100.0
10^{-6}	0.999	0.0	99.9	94.1	18.6	81.9	18.8	5.17	9	49.1	0.4	24.4
		0.5		38.8	37.2	20	5.6		6	23.8	50.7	93.7
		0.8		14.4	8.8	4.2	0.2		0.9	7.8	74.8	99.9
		1.0		8.9	5.8	2.3	0.2		1.3	0.0	79.2	99.8
	0.9	0.0	99.9	99.7	21	94.3	20.6	15.8	24.4	56.8	14.6	28.4
		0.5		46.9	38.6	19.7	3.6		11.8	28.6	39.4	92
		0.8		30.4	10.4	8.7	1.6		3.6	12.3	62.8	98.9
		1.0		27	6.3	6.7	0.4		1.3	0.0	70.4	99.5
	0.5	0.0	99.8	99.8	99.6	94.4	82.7	99.7	99.9	99.8	81	84.6
		0.5		100.0	99.5	91.4	80.1		99.7	99.9	95.9	99.6
		0.8		99.7	97	90.1	74.6		99.7	99.3	96.2	99.3
		1.0		99.8	55.7	91.2	61		99.4	30	96.3	99.4

Table D.6: Results on the PCFG index of occurrence task with 200K< iterations of pre-training.

η	$\mathscr{P}_{\mathtt{T}}(\mathtt{a})$	$C_{\mathtt{Tr}}$	Acc. $\mathtt{O}_{\mathtt{PT}}$					Acc $\mathtt{O}_{\mathtt{FT}}$				
			Acc. PT	$C_{\mathtt{Te}}=0$		$C_{\mathtt{Te}}=1$		Acc. PT	$C_{\mathtt{Te}}=0$		$C_{\mathtt{Te}}=1$	
				$1K$	$10K$	$1K$	$10K$		$1K$	$10K$	$1K$	$10K$
10^{-4}	0.999	0.0	99.0	5.8	0.0	23.9	0.0	9.3	71.8	99.7	46.1	99.5
		0.5		9.0	0.0	23.0	0.0		66.6	99.6	81.0	100.0
		0.8		14.9	0.0	0.8	0.0		36.8	99.1	99.5	100.0
		1.0		33.6	9.5	0.0	0.0		3.7	5.8	100.0	100.0
	0.9	0.0	99.2	96.2	0.0	88.6	0.1	97.1	98.5	99.9	98.6	99.7
		0.5		96.8	0.0	83.5	0.0		97.3	99.5	100.0	100.0
		0.8		96.8	0.0	84.8	0.0		97.4	99.2	100.0	100.0
		1.0		72.6	1.3	79.1	0.0		37.3	24.0	100.0	100.0
	0.5	0.0	98.0	95.7	4.5	96.5	3.6	98.9	99.5	99.7	99.8	99.8
		0.5		95.2	5.6	78.2	3.5		99.1	99.8	100.0	100.0
		0.8		96.0	17.7	90.2	5.6		98.8	99.4	100.0	100.0
		1.0		91.3	15.8	79.8	14.5		48.0	28.8	100.0	100.0
10^{-5}	0.999	0.0	99.0	94.6	2.6	86.2	18.6	9.3	17.0	85.4	16.0	68.5
		0.5		98.4	3.4	97.6	7.7		14.3	79.8	27.2	97.8
		0.8		94.1	5.1	94.0	0.2		10.7	66.6	37.6	99.7
		1.0		77.6	27.7	70.8	0.0		4.9	3.7	56.6	100.0
	0.9	0.0	99.2	99.2	88.5	99.0	79.3	97.1	97.3	99.1	98.9	99.1
		0.5		99.2	91.9	99.2	77.9		97.3	98.3	100.0	100.0
		0.8		98.8	95.6	98.9	84.3		96.9	98.7	99.9	100.0
		1.0		98.0	65.4	98.0	72.7		81.3	31.8	99.9	100.0
	0.5	0.0	98.0	97.3	95.8	100.0	89.1	98.9	99.1	99.9	99.9	99.4
		0.5		97.6	95.9	98.8	75.4		99.2	99.3	100.0	100.0
		0.8		97.4	95.1	97.8	80.3		99.6	99.1	100.0	100.0
		1.0		97.4	87.7	98.0	81.0		97.2	51.8	100.0	100.0
10^{-6}	0.999	0.0	99.0	99.0	80.4	99.4	73.0	9.3	14.6	23.3	19.5	15.5
		0.5		98.9	94.2	99.5	94.3		13.7	20.2	22.0	40.5
		0.8		99.2	67.8	98.7	61.2		13.2	9.2	25.1	71.8
		1.0		99.4	61.6	98.8	11.3		12.9	4.5	19.3	95.6
	0.9	0.0	99.2	98.8	99.6	99.5	99.6	97.1	97.3	97.0	98.0	98.4
		0.5		99.0	98.8	99.8	99.2		96.6	96.6	98.8	99.9
		0.8		98.7	99.2	99.7	99.0		97.1	96.7	99.3	100.0
		1.0		98.7	98.0	99.4	95.8		97.8	70.7	99.3	100.0
	0.5	0.0	98.0	97.8	97.9	99.8	99.6	98.9	99.2	98.9	99.2	99.7
		0.5		98.7	98.0	99.8	98.2		98.9	99.0	99.8	100.0
		0.8		97.4	97.9	99.6	98.2		99.5	98.9	100.0	100.0
		1.0		98.1	96.7	99.9	96.4		98.9	94.5	99.9	100.0

Table D.7: Results on the PCFG index of occurrence task with 50K iterations of pre-training.

η	$\mathscr{P}_{\mathtt{T}}(\mathtt{a})$	$C_{\mathtt{Tr}}$	Acc $\mathtt{O}_{\mathtt{PT}}$					Acc $\mathtt{O}_{\mathtt{FT}}$				
			Acc. PT	$C_{\mathtt{Te}}=0$		$C_{\mathtt{Te}}=1$		Acc. PT	$C_{\mathtt{Te}}=0$		$C_{\mathtt{Te}}=1$	
				$1K$	$10K$	$1K$	$10K$		$1K$	$10K$	$1K$	$10K$
10^{-4}	0.999	0.0	94.2	0.5	0.0	11.7	0.1	3.2	77.1	99.6	66.1	99.4
		0.5		3.2	0.0	2.0	0.0		60.0	98.5	92.5	100.0
		0.8		6.7	0.1	1.0	0.0		26.7	96.8	98.7	100.0
		1.0		23.1	10.9	0.0	0.0		3.8	5.1	99.6	100.0
	0.9	0.0	94.2	43.2	0.0	34.6	0.8	69.9	86.1	99.6	75.1	97.5
		0.5		48.2	0.0	56.2	0.1		81.0	98.9	97.5	99.9
		0.8		53.5	0.0	53.1	0.0		74.0	97.0	98.8	100.0
		1.0		12.8	11.0	32.8	0.0		4.4	3.4	99.8	100.0
	0.5	0.0	88.6	72.1	2.3	59.6	2.6	91.5	95.2	98.6	90.2	99.6
		0.5		65.4	2.3	70.4	0.0		91.4	99.3	98.8	100.0
		0.8		56.6	1.5	65.5	0.0		88.7	97.4	99.7	100.0
		1.0		39.0	12.8	40.2	0.0		6.1	5.3	99.9	100.0
10^{-5}	0.999	0.0	94.2	5.0	0.1	6.4	7.7	3.2	18.4	88.9	10.3	86.5
		0.5		74.1	1.4	42.3	0.4		10.1	77.0	55.2	98.8
		0.8		37.4	3.8	13.2	0.7		6.9	45.9	86.9	98.7
		1.0		48.0	19.1	2.7	0.0		0.5	4.4	95.2	100.0
	0.9	0.0	94.2	89.2	18.9	69.6	26.8	69.9	74.0	91.1	66.8	82.8
		0.5		91.9	25.0	75.9	38.9		70.8	86.9	82.0	99.2
		0.8		92.4	30.4	75.9	32.1		60.8	80.2	87.8	99.8
		1.0		69.7	8.8	73.9	8.4		13.8	4.4	92.6	100.0
	0.5	0.0	88.6	87.6	64.0	77.7	51.4	91.5	93.1	95.6	85.5	90.9
		0.5		85.9	57.5	80.0	57.0		90.4	95.5	93.9	100.0
		0.8		84.9	46.8	78.2	50.4		89.2	91.6	96.8	100.0
		1.0		80.1	33.5	77.6	13.3		38.6	5.5	98.1	100.0
10^{-6}	0.999	0.0	94.2	87.5	3.7	73.8	14.2	3.2	8.3	23.0	15.8	20.7
		0.5		95.3	49.3	83.3	19.2		5.4	13.7	27.5	76.9
		0.8		94.0	31.0	71.4	7.7		3.9	7.2	38.7	93.7
		1.0		88.1	43.0	50.6	2.1		0.3	1.1	44.7	97.5
	0.9	0.0	94.2	93.6	88.2	73.2	70.7	69.9	69.6	75.6	56.9	66.9
		0.5		94.3	92.2	73.1	76.4		70.3	69.7	67.4	88.6
		0.8		92.7	89.8	75.7	80.3		65.3	63.6	71.8	94.7
		1.0		95.6	56.2	72.8	72.3		60.2	7.8	71.9	97.9
	0.5	0.0	88.6	87.3	87.0	80.0	76.6	91.5	90.3	93.8	85.1	88.3
		0.5		87.7	84.5	80.4	80.8		93.4	91.5	90.3	97.6
		0.8		90.6	83.2	83.1	80.5		92.3	87.9	90.1	98.6
		1.0		89.0	77.3	79.8	70.2		92.4	23.0	91.9	99.7

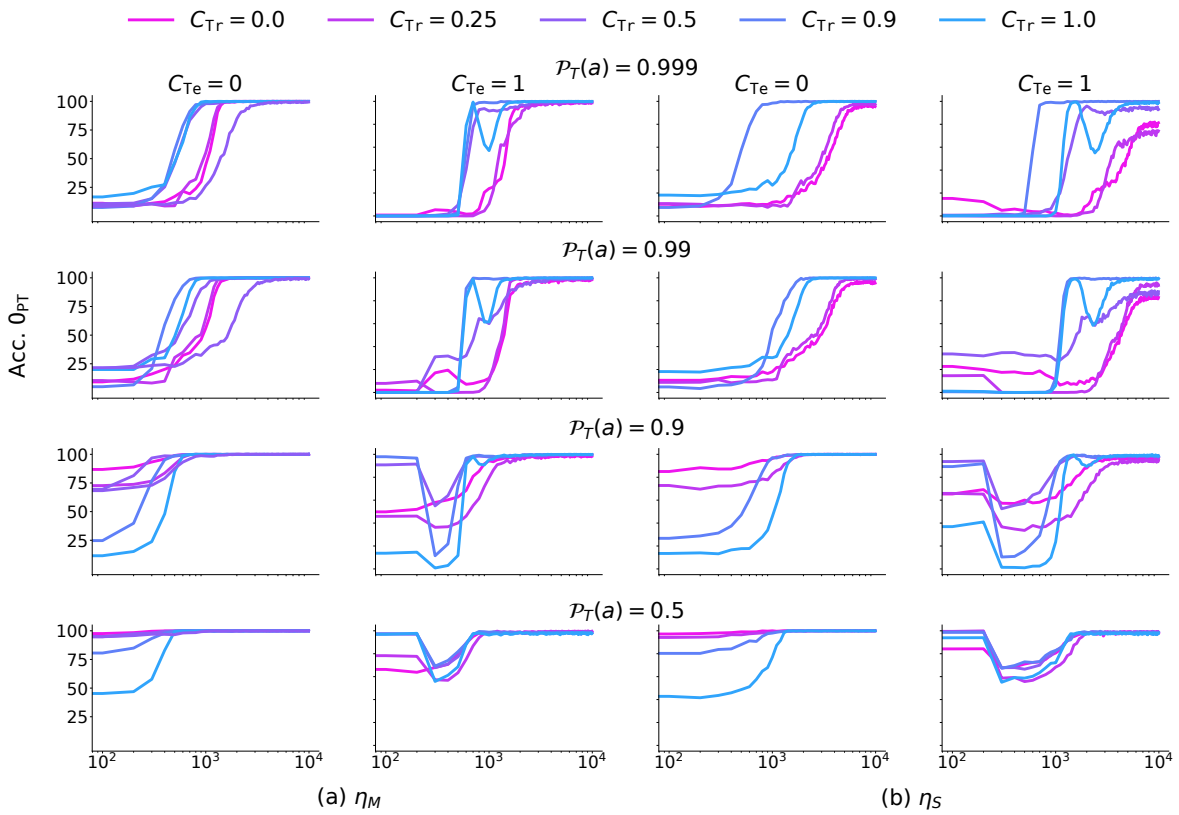


Figure D.64: Counter Task, $n_{iters} = 200K$: Reverse Fine-tuning on weakly and strongly relevant capability fine-tuned models. Medium and small learning rates are used for reverse fine-tuning in the presence of different degrees of spuriously correlated data-points present in the train-set. The fine-tuned model was fine-tuned using Large LR. **Observation:** When the model possesses weakly relevant capability, the convergence time is lower for models fine-tuned on dataset with spurious correlations. If the model possesses strongly relevant capability, this difference is less. The "revival" of pre-training capability is observed for all values of C_{Tr}. Even though fine-tuning was done using a large learning rate of 10^{-4}, capability revival is possible even on using a small learning rate.

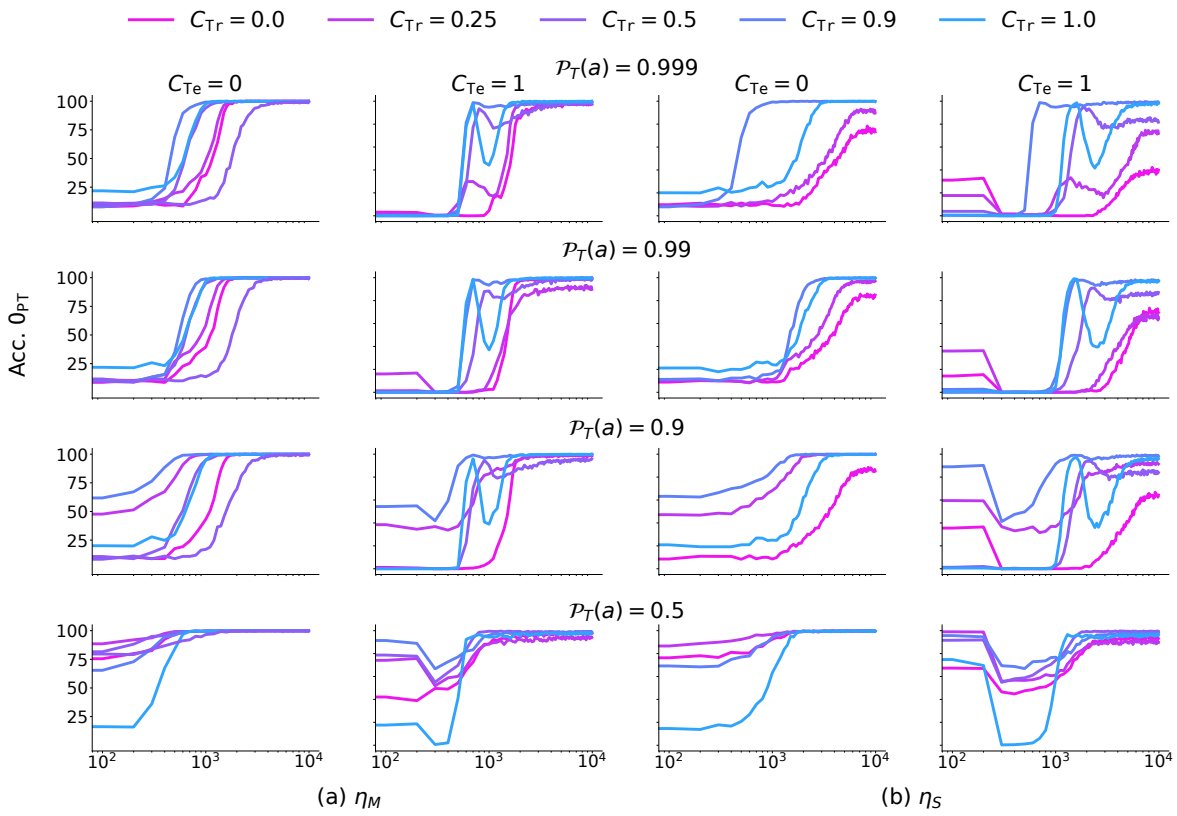


Figure D.65: Counter Task, $n_{iters} = 50K$: Reverse Fine-tuning on weakly and strongly relevant capability fine-tuned models. η_M and η_S are used for capability reverse fine-tuning in the presence of different fraction of spuriously correlated data-points present in the train-set. The fine-tuned model was fine-tuned using Large LR. **Observation:** The observations are consistent with Fig. D.64

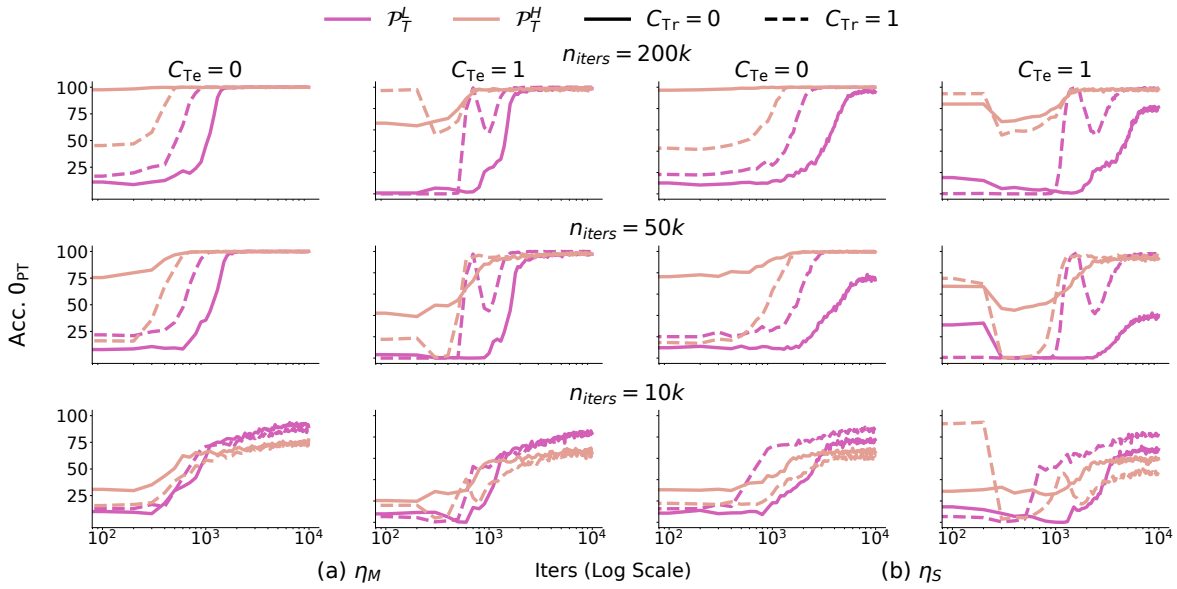


Figure D.66: Counter Task: Reverse fine-tuning analysis for different pre-training iterations. Observation: Capability Revival is seen for models pre-trained with different number of iterations.

D.8.1 Pruning Analysis

In this section, I present detailed results on pruning analysis of the PCFG setup on both counting and index of occurrence tasks. I provide an exhaustive evaluation in Fig. D.67,Fig. D.70,Fig. D.72,Fig. D.74 Fig. D.76 for the Counter task and Fig. D.68,Fig. D.71,Fig. D.73,Fig. D.75 Fig. D.77 for the index of occurrence task.

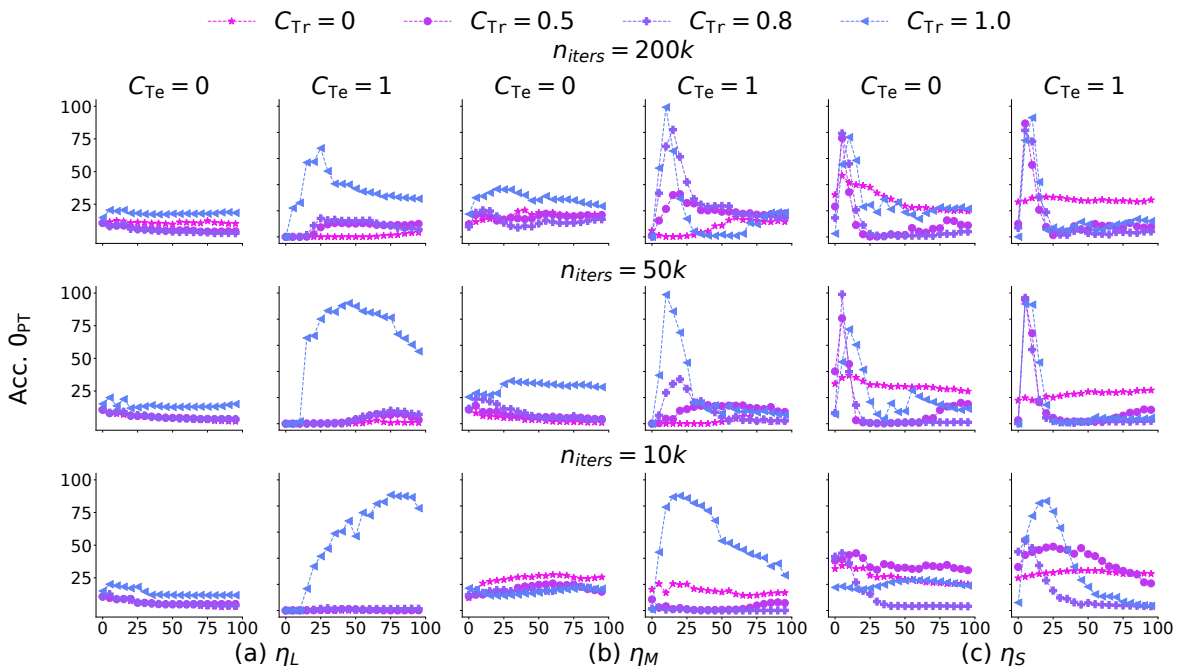


Figure D.67: Counter task, $\mathscr{P}_{\text{T}}(\text{a}) = 0.999$, Pruning Analysis: Revival of pre-training capability analysis for different learning rates, pre-training iterations and different values of C_{Tr}. **Observation:** (a) On using η_L the model learns the wrapper only when fine-tuning on $C_{\text{Te}} = 1$. This wrapper is learned only on fine-tuning set with the spurious correlations . (b) On using η_M the model learns the wrapper on fine-tuning with smaller values of C_{Te} as well. However, still this wrapper is learned only on fine-tuning set with the spurious correlations. (c) On using η_S the model learns the wrapper for all values of C_{Te} and for all the data samples.

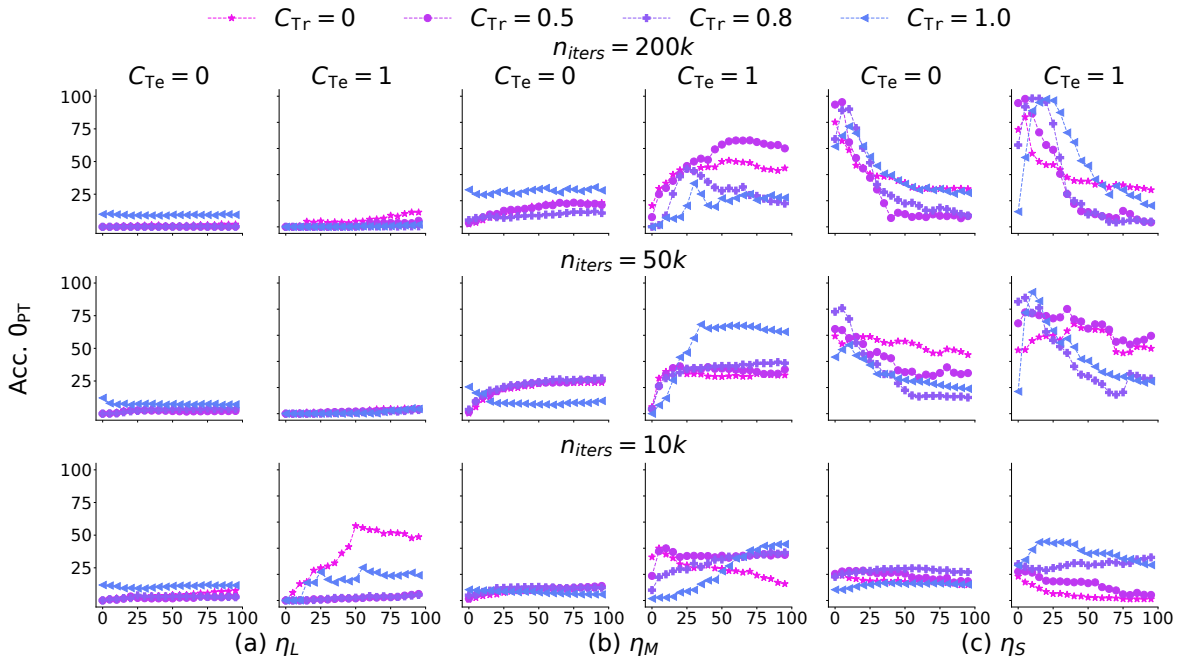


Figure D.68: Index of Occurrence task, $\mathscr{P}_T(a) = 0.999$, Pruning Analysis: The settings are consistent with Fig. D.67

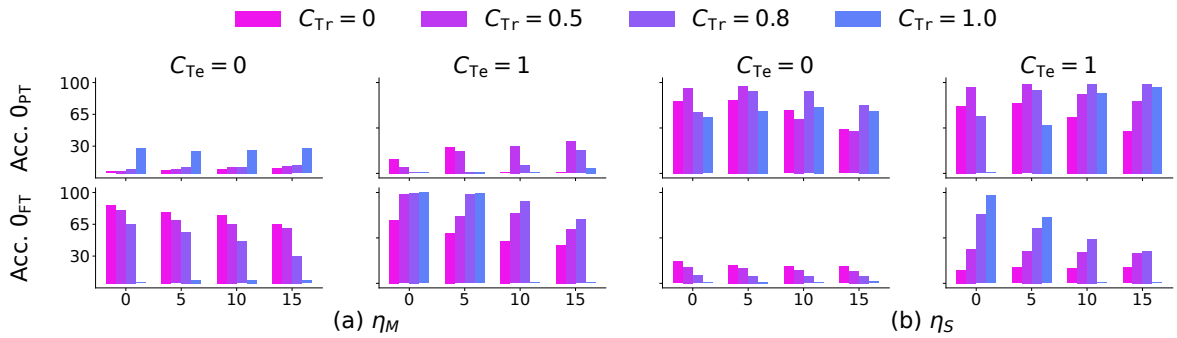


Figure D.69: Index of Occurrence task, $\mathscr{P}_T(a) = 0.999$, Pruning Analysis: The settings are consistent with Fig. 5.9

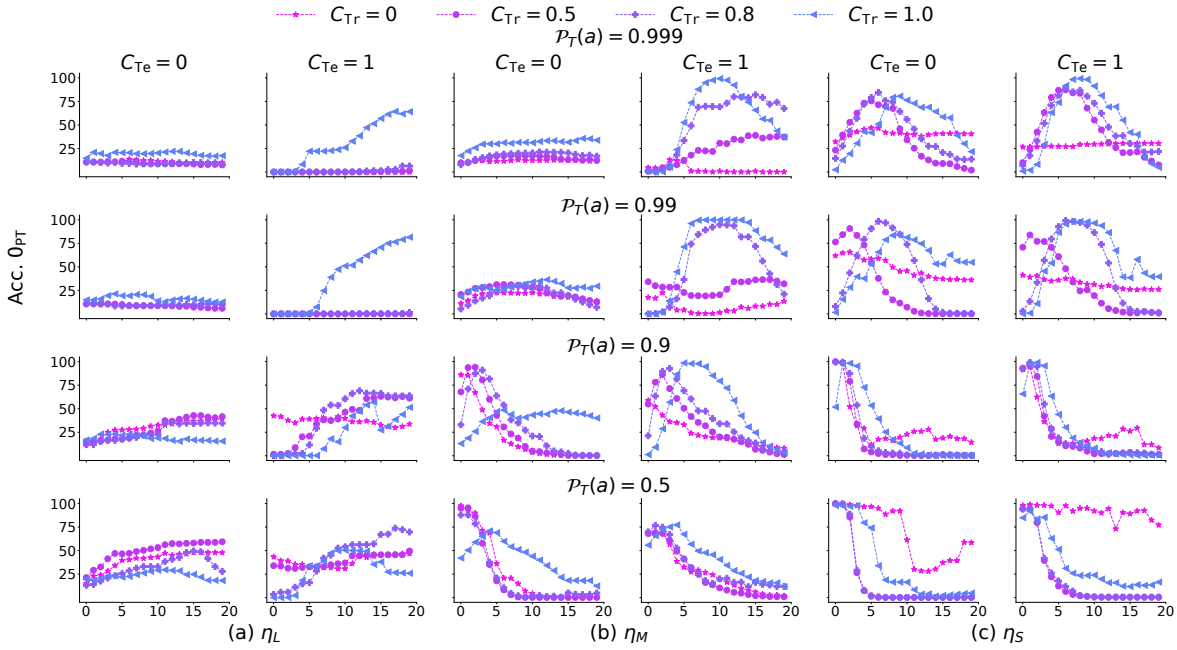


Figure D.70: Counter task, $n_{iters} = 200K$, Pruning Analysis: Revival of pre-training capability analysis for different learning rates, weakly and strongly relevant capability fine-tuned models, and different values of C_{Tr}. **Observation:** Learning of the wrapper is possible for weakly as well as strongly relevant capability pre-trained models.

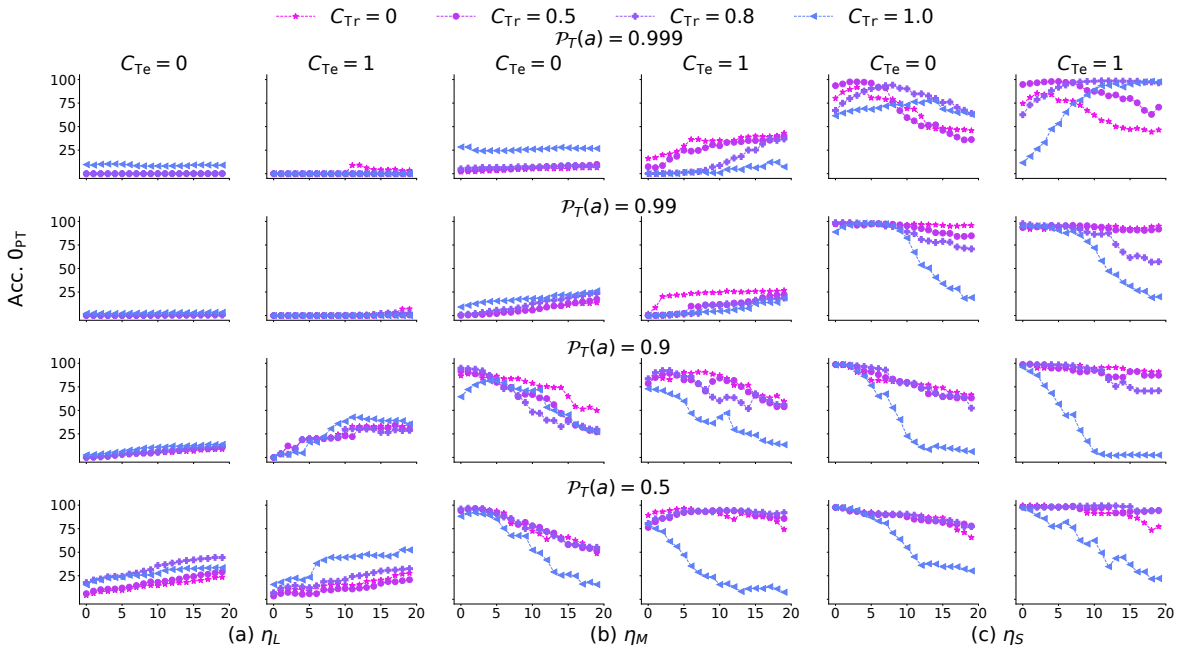


Figure D.71: Index of Occurrence task, $n_{iters} = 200K$, Pruning Analysis: The settings are consistent with Fig. D.70

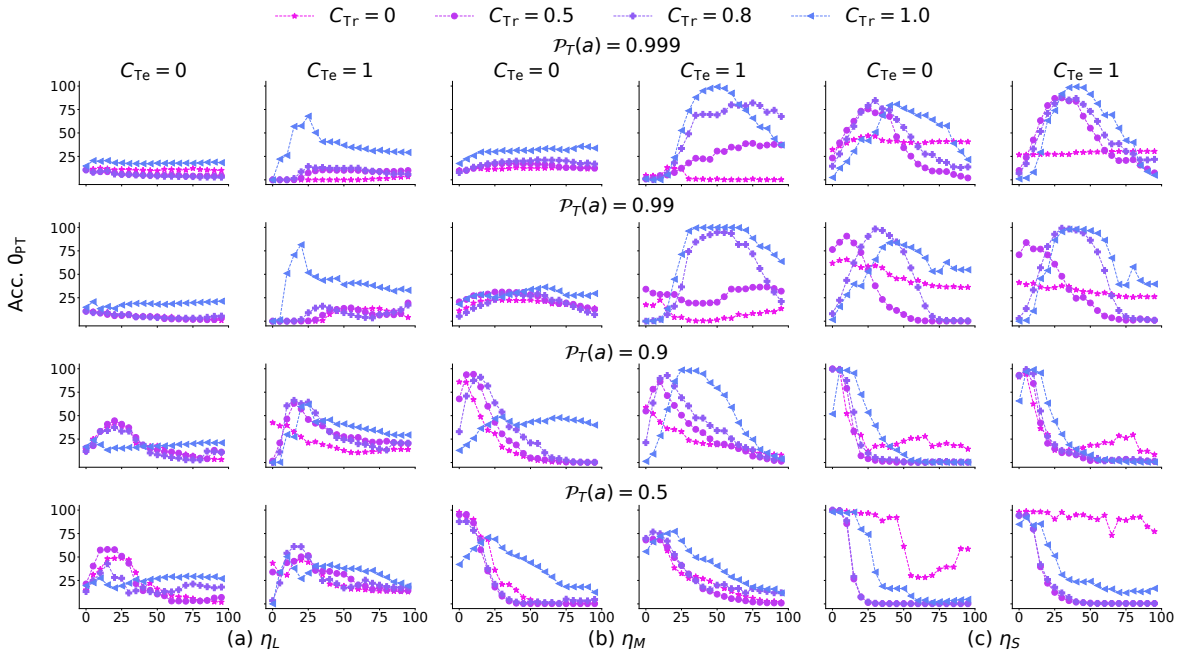


Figure D.72: Counter task, $n_{iters} = 200K$ Pruning Analysis: Revival of pre-training capability analysis for different learning rates, weakly and strongly relevant capability fine-tuned models and different values of C_{Tr}. Here larger number of neurons are pruned as compared to Fig. D.70.

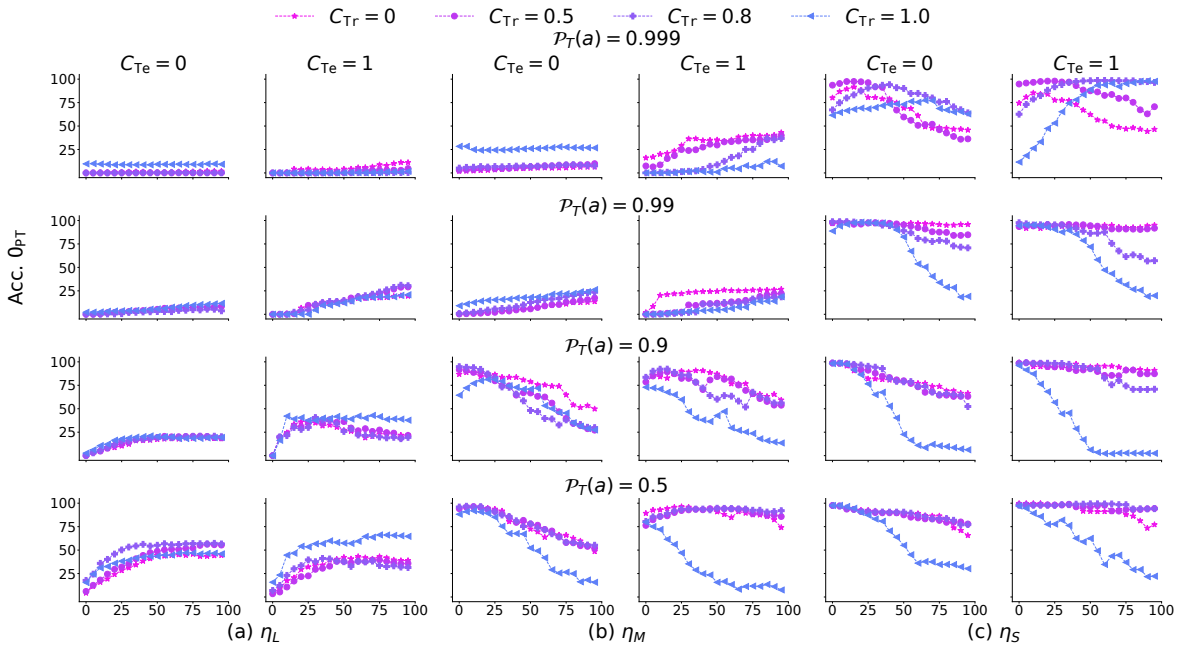


Figure D.73: Index of Occurrence task, $n_{iters} = 200K$, Pruning Analysis: Here larger number of neurons are pruned. The settings are consistent with Fig. D.72

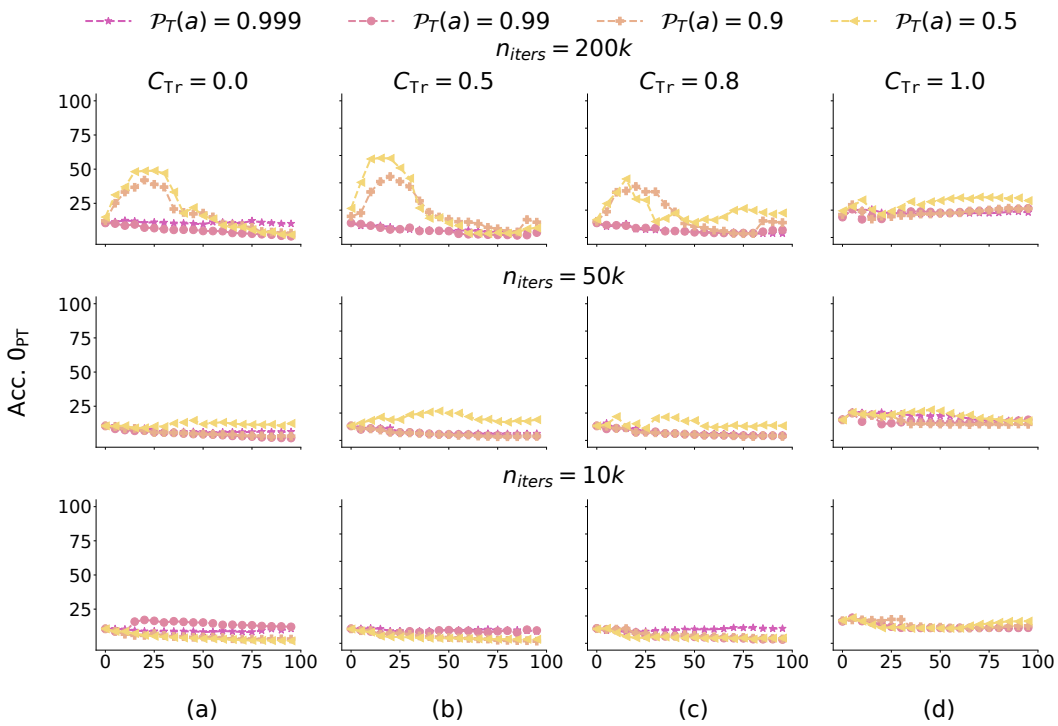


Figure D.74: Counter task, η_L, $C_{\mathtt{Te}} = 0$, Pruning analysis: Effect of strongly and weakly relevant capabilities for different number of pre-training iterations and different values of $C_{\mathtt{Tr}}$ **Observation:** Fine-tuning a model with strongly relevant capability leads to learning of an "inhibitor" on its pre-training capability, i.e., a wrapper that disallows use of the pretraining capability. Revival of the pre-training capability is partly possible on pruning, if the model has strongly relevant capability and it was fine-tuned on dataset without spurious correlations.The inhibitor is mainly learned for the $200K$ iteration pre-trained model.

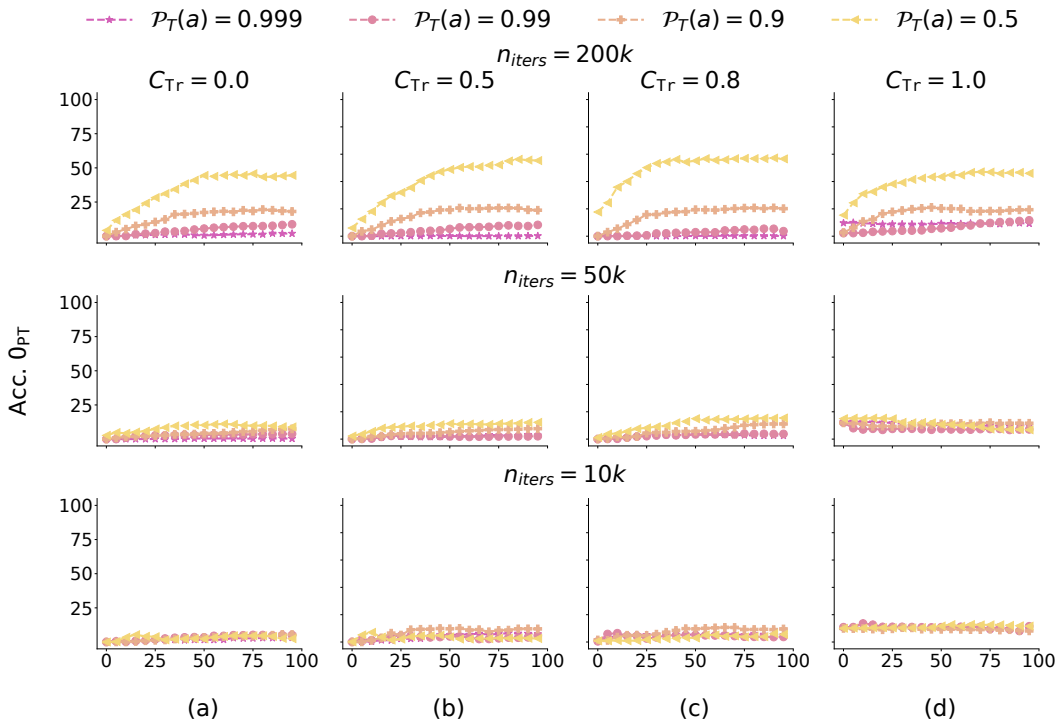


Figure D.75: Index of Occurrence task, η_L, $C_{\text{Te}} = 0$, Pruning analysis: The settings are consistent with Fig. D.74

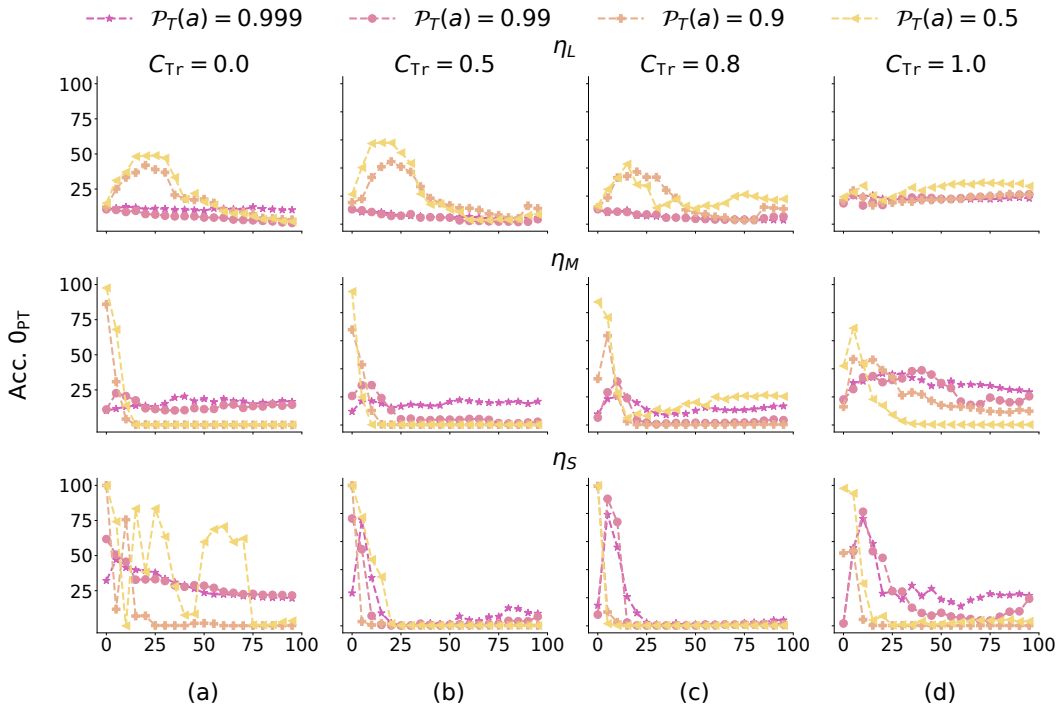


Figure D.76: Counter task, $n_{iters} = 200K$, $C_{\text{Te}} = 0$, Pruning analysis: Effect of the strongly and weakly relevant capabilities for different number of pre-training iterations and different values of fraction of spurious correlations present in the fine-tuning dataset. **Observation:** Learning of the inhibitor is observed on using η_L.

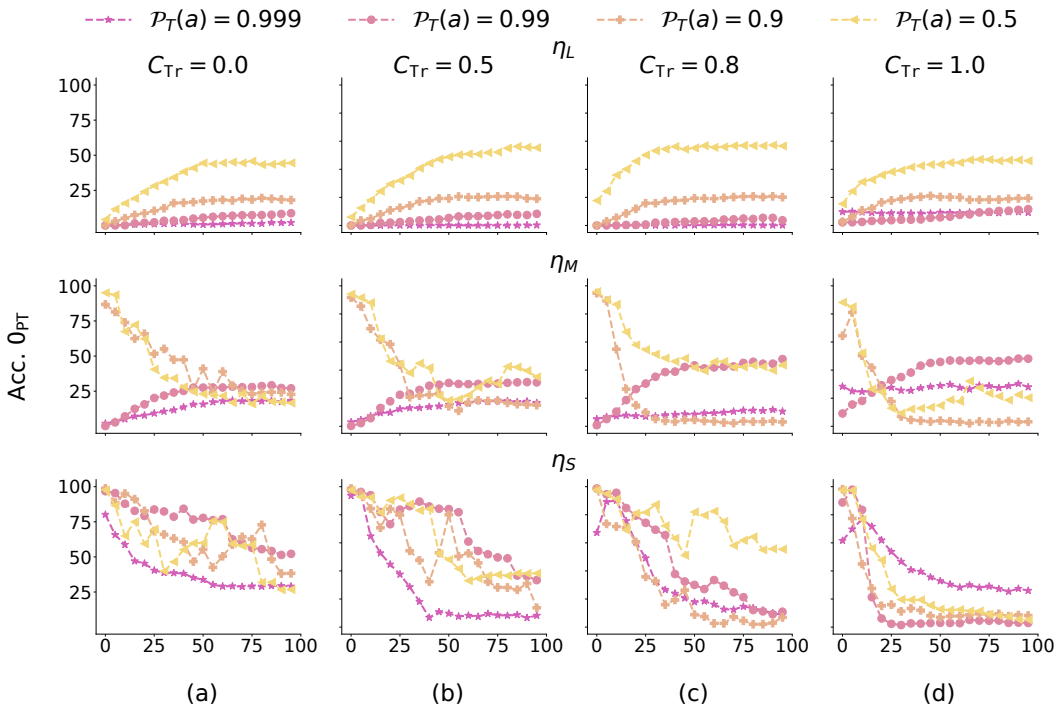


Figure D.77: Index of Occurrence task, $n_{iters} = 200K$**,** $C_{\mathtt{Te}} = 0$ **Pruning analysis:** the settings are consistent with Fig. D.76

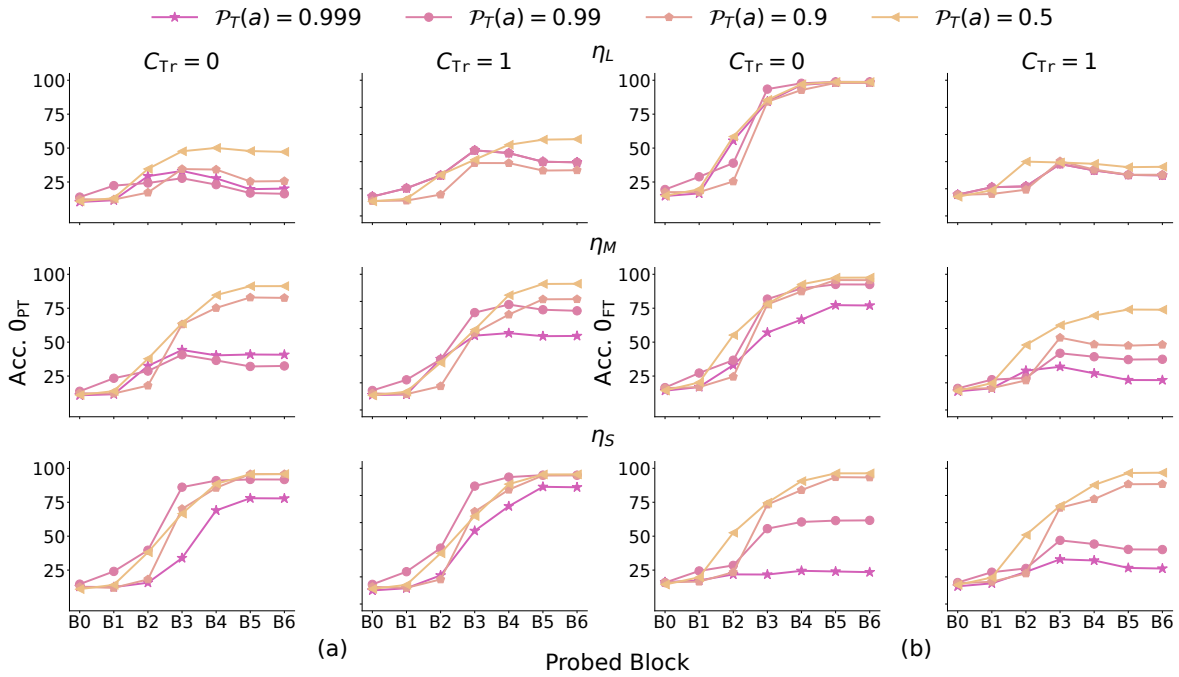


Figure D.78: Counter task, $n_{iters} = 200K$ **,** $C_{\text{Te}} = 0$**, Probing analysis:** The effect of different values of learning rate, weakly and strongly relevant capabilities is shown. **Observation:** Using η_L hampers the pre-training capability to count a especially when the probed model has a weakly relevant capability. The performance on the pre-training task of counting a's continues to be high, especially with η_S. The accuracy of counting b's shows that fine-tuning capability is learned on using η_L. On using η_S, models with weakly relevant capabilities are not able to learn the fine-tuning capability well.

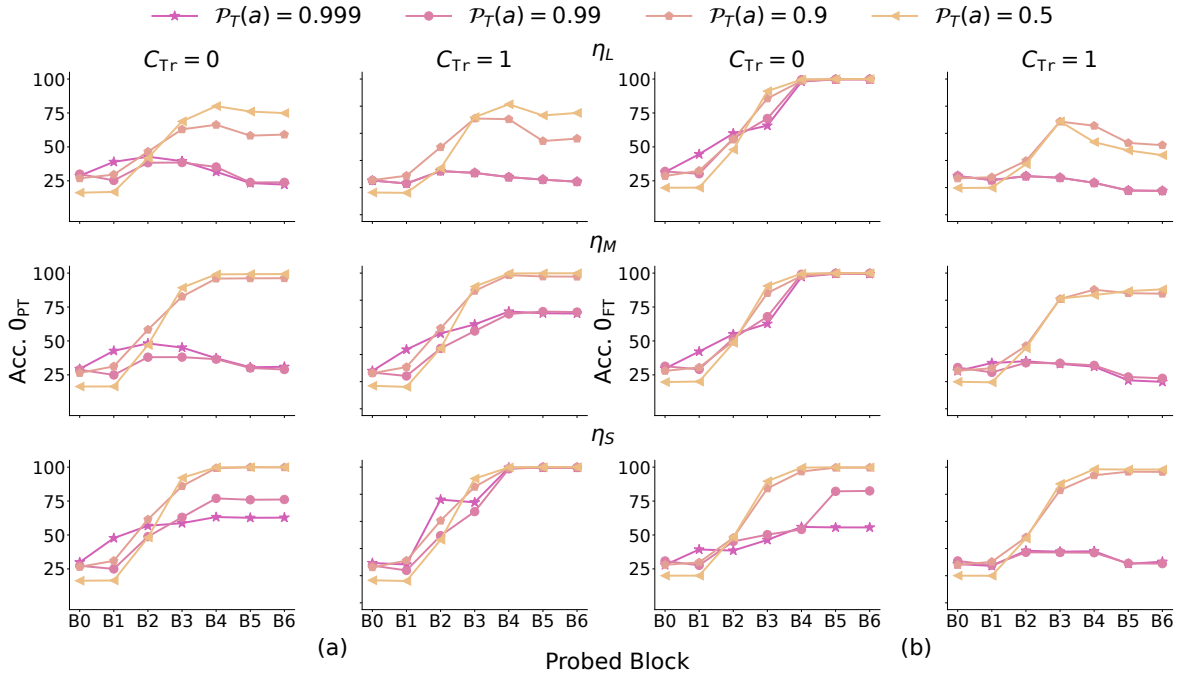


Figure D.79: Index of Occurrence task, $n_{iters} = 200K$ **,** $C_{\text{Te}} = 0$**, Probing analysis:** The settings are consistent with Fig. D.78

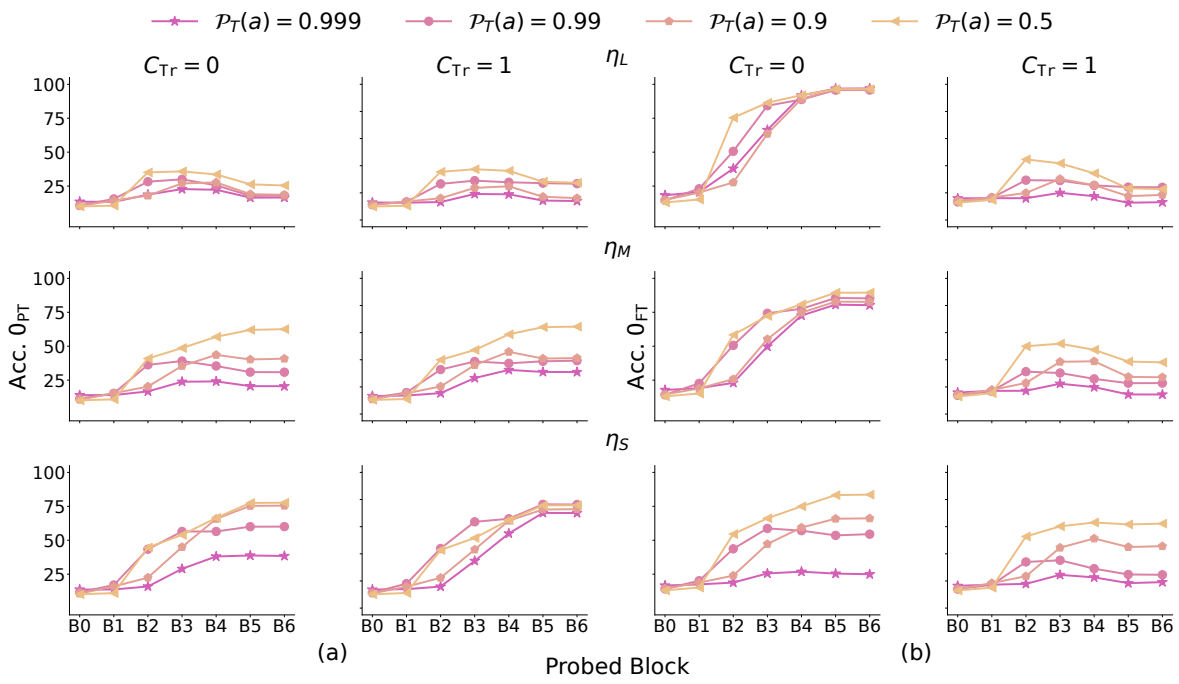


Figure D.80: Counter task, $n_{iters} = 50K$, $C_{\text{Te}} = 0$, Probing analysis: The observations are consistent with Fig. D.78

D.8.2 Probing Analysis

In this section, I present detailed results on probing analysis of the PCFG setup on both counting and index of occurrence tasks. I provide an exhaustive evaluation in Fig. D.80,Fig. D.83,Fig. D.85 for the Counter task and Fig. D.81,Fig. D.84,Fig. D.86 for the index of occurrence task.

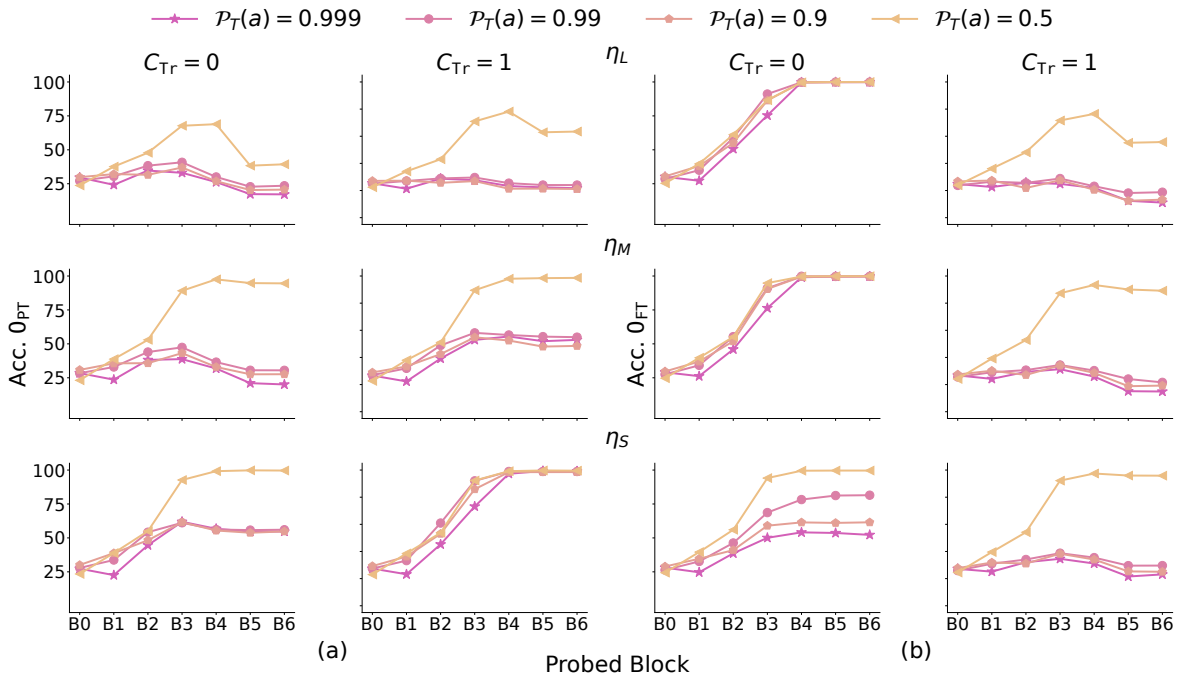


Figure D.81: Index of Occurrence task, $n_{iters} = 50K$**,** $C_{\text{Te}} = 0$**, Probing analysis:** The settings are consistent with Fig. D.80

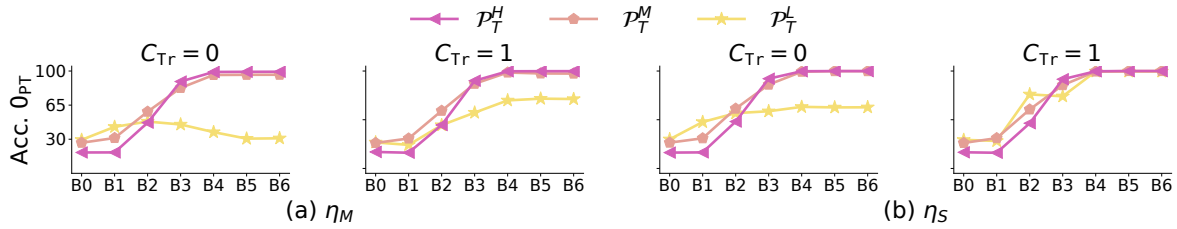


Figure D.82: Index of Occurrence task, $n_{iters} = 200K$ **,** $C_{\text{Te}} = 0$**, Probing analysis:** The settings are consistent with Fig. 5.8

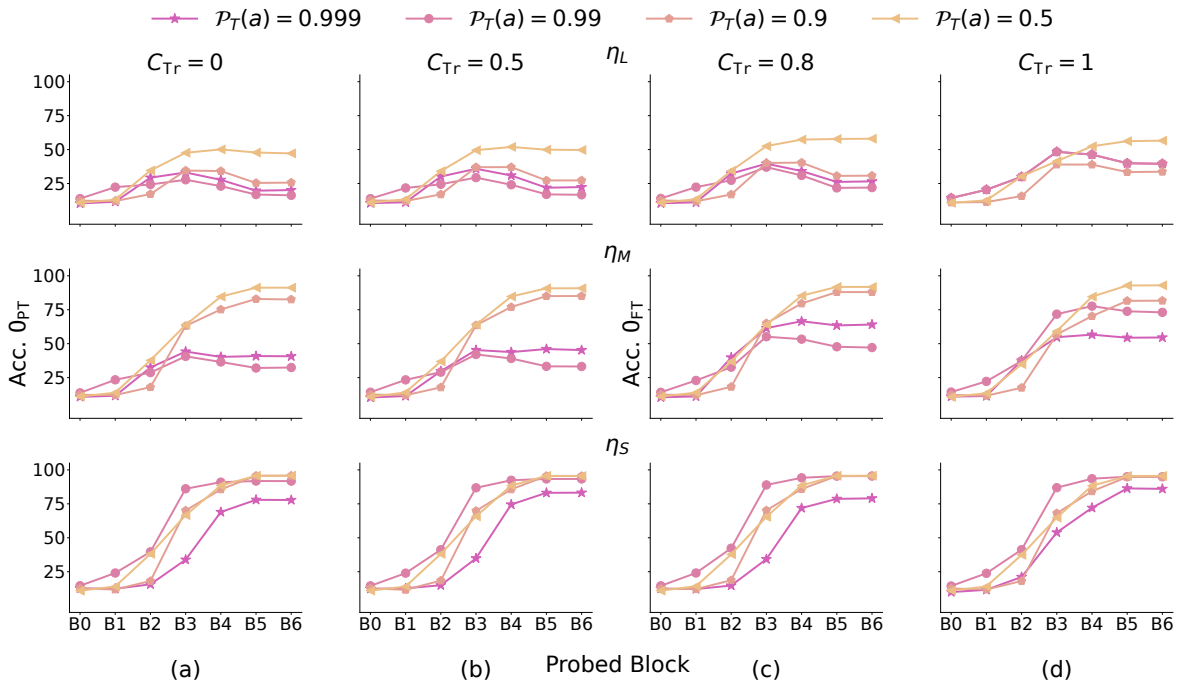


Figure D.83: Counter task, $n_{iters} = 200K$, $C_{\text{Te}} = 0$, Probing analysis. Observation: With an increase in C_{Tr}, the accuracy on counting a's also increases for both weakly as well as strongly relevant capability models.

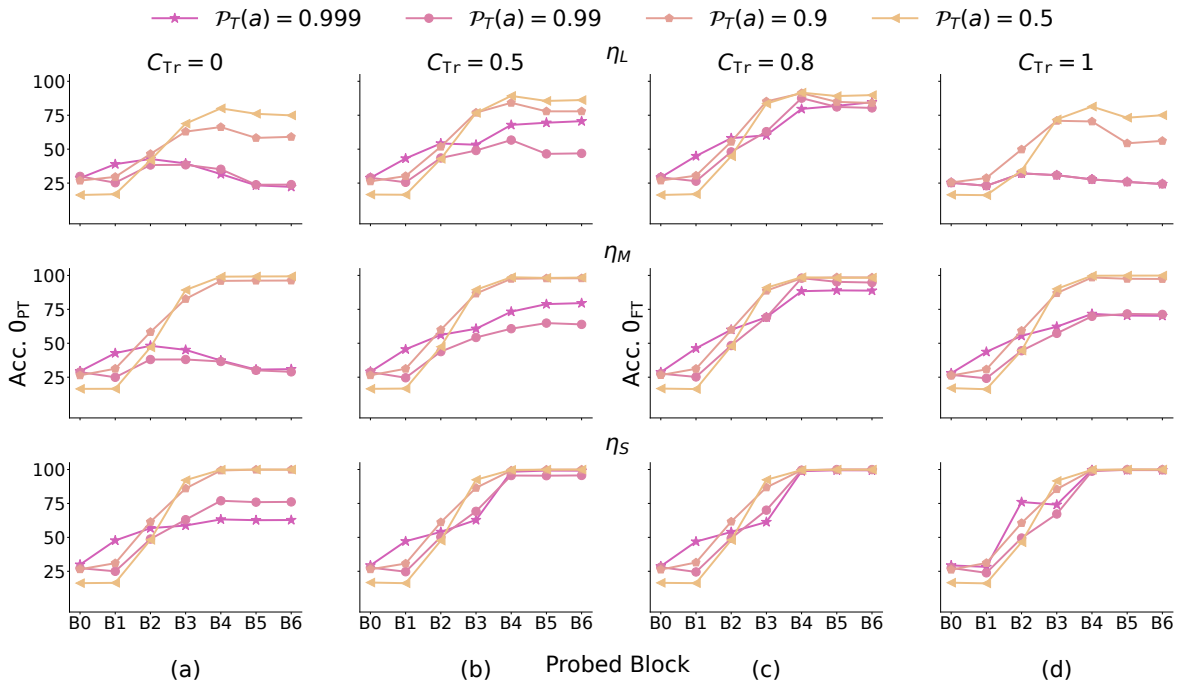


Figure D.84: Index of Occurrence task, $n_{iters} = 200K$, $C_{\text{Te}} = 0$, Probing analysis: The settings are consistent with Fig. D.83

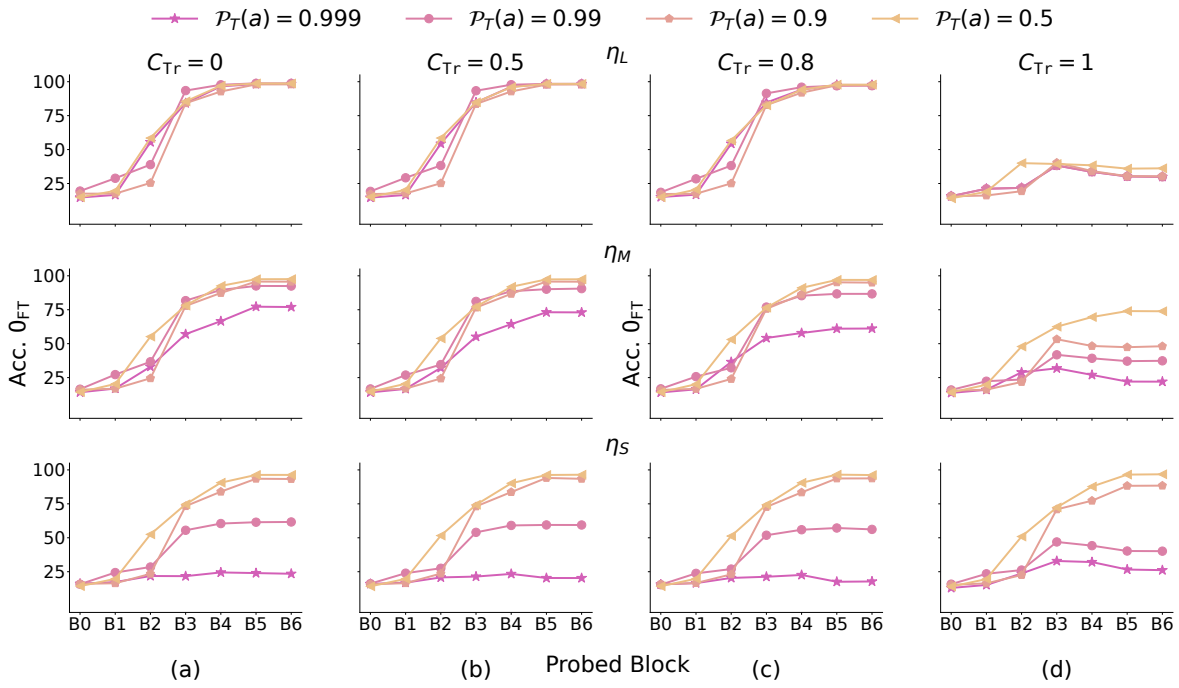


Figure D.85: Counter task, $n_{iters} = 50K$, $C_{\text{Te}} = 0$, Probing analysis. Observation: With an increase in C_{Tr}, the accuracy on counting b's also decreases for both weakly as well as strongly relevant capability models.

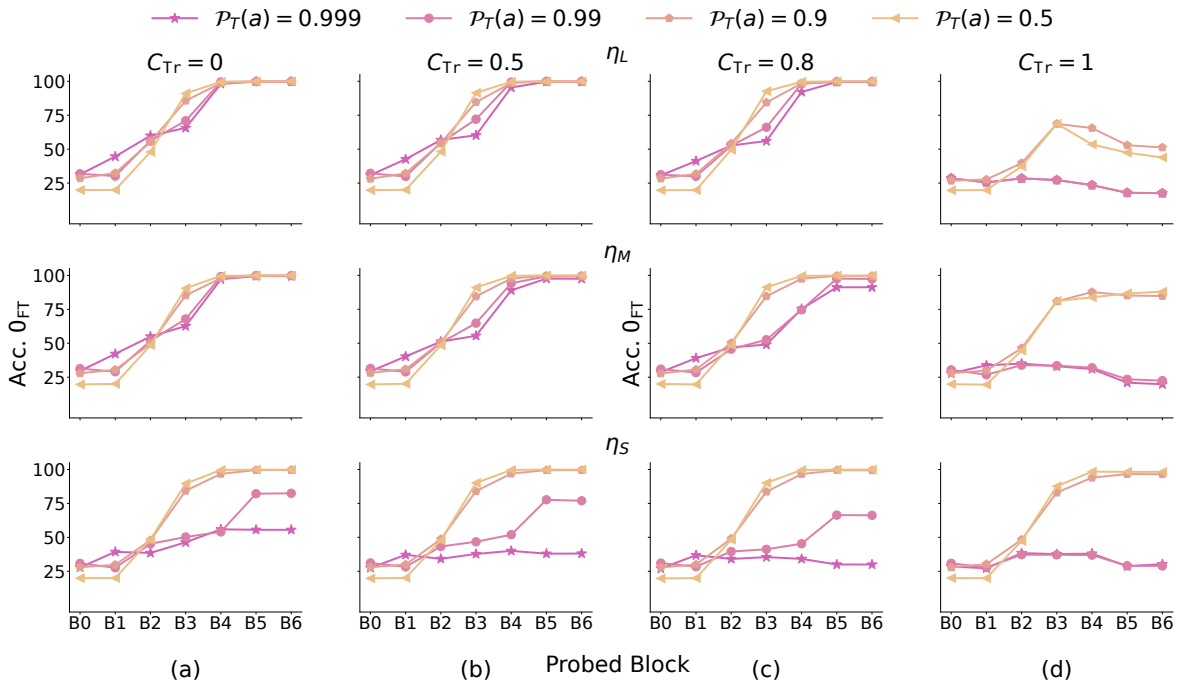


Figure D.86: Index of Occurrence task, $n_{iters} = 50K$, $C_{\text{Te}} = 0$, Probing analysis: The settings are consistent with Fig. D.85