Neural Network Generalisation in the Overparameterised Regime

Reuben J. Adams
Department of Computer Science
University College London

A thesis presented for the degree of
Doctor of Philosophy

November 19, 2025

Declaration of originality and contribution

I, Reuben J. Adams, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

The work in Chapter 3 was completed with the guidance and assistance of my supervisors, Benjamin Guedj and John Shawe-Taylor. John Shawe-Taylor laid some of the conceptual framework and outlined the initial proof strategy, while Benjamin Guedj offered supervisory guidance throughout, including feedback on ideas, assistance with the introduction, editorial input across multiple drafts, and support during the publication process.

The remainder of the thesis is entirely my own work, conducted under the formal supervision of my supervisors.

Abstract

The generalisation mystery is the gap in our understanding of why commonly used Deep Learning algorithms produce neural networks that generalise to unseen data, even using large architectures with capacity far greater than that required to fit their training data exactly. Solving this puzzle would theoretically ground the astonishing empirical success of neural networks, potentially enabling them to be used with greater understanding and with quantitative performance guarantees. We make three contributions towards answering this question. First, we extend a classic PAC-Bayesian generalisation bound to provide more information-rich test time guarantees. Second, we demonstrate that PAC-Bayesian bounds on deterministic networks can be tightened by relating their performance to compressible neighbouring networks. Finally, we take a more empirical approach and show that the generalisation ability of a network is connected to its compressibility via distillation.

Impact

While there are no immediate practical or commercial applications of the research presented in this thesis, it nevertheless contributes to the academic community's theoretical understanding of deep learning. In Chapter 3 we prove a generalisation of a well-known theoretical result and thereby shed light on the extent to which theory can explain, predict and control the balance of different types of error a learning algorithm is likely to make. This constitutes a step forward in the pursuit of self-certified learning, in particular with information-rich certificates, which would enable Deep Learning researchers and practitioners to fully exploit their data. The work in Chapters 4 and 5 attacks a core mystery in deep learning, the generalisation mystery, elucidating the role of confidence (margin) and compressibility of neural networks in explaining their generalisation. We believe this brings the community closer to a full explanation of the generalisation mystery, which would have a profound impact on deep learning theory and practice.

This work was supported by the UK Engineering and Physical Sciences Research Council under grant number EP/S021566/1.

Declaration of research paper

Chapter 3 of this thesis, Controlling Multiple Errors Simultaneously with a PAC-Bayes Bound, has been published as a peer-reviewed paper of the same name at the NeurIPS 2024 conference, as joint work with my supervisors Benjamin Guedj and John Shawe-Taylor Adams et al. (2024). Copyright has been retained by the authors. The published paper is available at NeurIPS or arXiv.

Signed:	Reuben Adams
Date:	21st May 2025
Signed:	Benjamin Guedj
Date:	2nd June 2025
Signed:	John Shawe-Taylor
Date:	2nd June 2025

Acknowledgements

Thank you to my family for being eternally proud of me, which has been a huge source of comfort. Thank you to the taxpayer, who funded this work and my life for the past four years. Thank you to Daria Levin for her incredible support over the past three years. Thank you to Mitchell Deen for his consistently sound advice; the alternate version of me who took all rather than merely most of it has fewer grey hairs. Thank you to Fredrik Hellström for the many enjoyable and insightful technical discussions which gave me the confidence to pursue the final two chapters of this thesis. Thank you to Oscar Key for answers to truly endless coding questions in the pre-Claude era, which took my software engineering skills to the next level. Thank you to Jude Wells for his incredible hospitality, which allowed me to keep one foot in London after moving to Oxford, and for nerd-sniping me with my own research on the occasion its intrigue required some rekindling. Thank you to Robert Kirk for many in-depth discussions on AI Safety, which really sharpened my thinking. Thank you to Felix Biggs for many illuminating technical conversations that helped shape my high-level research taste. Thank you to the many instantiations of Claude, whose endearing personality and indefatigable enthusiasm made for a stellar work partner. Thank you to the many friends who have offered support, advice and diversion, especially Garazi Muguruza for her grounding warmth, Evie Jamieson and Maximilian Maier for three great years at Hendos including many cathartic conversations on the ups and downs of the PhD experience, Andy Allan and Lothar Krapp for their continued support and friendship over many years and for eagerly agreeing to listen to practice talks, Jean Kaddour for his infectious enthusiasm and generous hospitality, Ryan Brown for his encouragement in the final days of writing, Antonin Schrab for his humour and reminders of the existence of pubs, and Chris Lloyd, Gemma Paterson, Luzia Bruckamp and Ishbel Mackenzie for their advice and friendship. Thank you to my supervisor Benjamin Guedj for his continued support throughout many hurdles, and the research freedom he offered. Thank you to my secondary supervisor John Shawe-Taylor for sharing the core insight that grew into the first chapter of this thesis. Thank you to Sharon Betts for her support of the Steering AI podcast and the many events we held together, and for her supererogatory investment in my well-being and career development. Thank you to Chris Watkins, Marc Deisenroth and Lewis Griffin for humouring me by coming on the podcast. And thank you to those at the AI Centre who showed patience or even interest when I talked about the risks of losing control of advanced AI. To have played a part in fostering the now commonplace interest in AI Safety at the AI Centre is perhaps my proudest achievement from my time at UCL. Finally, thank you to my examiners Jeremias Knoblauch and Tim van Erven for their detailed reading, helpful suggestions, and astute comments, which improved this thesis.

Contents

Notation

All notation is introduced at the appropriate time in the body of the thesis. The following table is for ease of reference for the most commonly used notation.

Symbol	Description
$\mathcal{X}$	An input space
$\mathcal{Y}$	An output space
$h : \mathcal{X} \to \mathcal{Y}$	A hypothesis, predictor, or, for finite $\mathcal{Y}$, a classifier
$\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$	A hypothesis class
$A^* = \bigcup_{n=0}^{\infty} A^n$	The set of tuples of elements of A
$S \in (\mathcal{X} \times \mathcal{Y})^*$	A sample
$\triangle(A)$	The set of probability measures on the set A
$D \in \triangle(\mathcal{X} \times \mathcal{Y})$	A data-generating distribution
$S \sim D^n$	An i.i.d. sample of size n drawn from D
$P, Q \in \triangle(\mathcal{H})$	Stochastic "prior" and "posterior" hypotheses
$\mathcal{A} : (\mathcal{X} \times \mathcal{Y})^* \to \mathcal{H}$	A learning algorithm
$\mathcal{A} : (\mathcal{X} \times \mathcal{Y})^* \to \triangle(\mathcal{H})$	A stochastic learning algorithm
$\ell : \mathcal{H} \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}$	A loss function
$\ell : \mathcal{Y}^2 \to \mathbb{R}$	A loss function (alternative formulation)
$\ell : \hat{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}$	A loss function (alternative formulation)
$\ell_{01}(\hat{y}, y) = \mathbb{1}[\hat{y} \neq y]$	The zero-one loss
$\ell_\gamma(h(x), y)$	The margin loss $\mathbb{1}\left[h(x)_y \leq \gamma + \max_{j \neq y} h(x)_j\right]$
$R_D(h) = \mathop{\mathbb{E}}_{(x,y)\sim D}[\ell(h(x), y)]$	The (true) risk of h on D
$R_S(h) = \frac{1}{\lvert S\rvert} \sum_{(x,y)\in S} \ell(h(x), y)$	The empirical risk of h on S
$R_D(Q) = \mathop{\mathbb{E}}_{h\sim Q}[R_D(h)]$	The (true) risk of Q on D
$R_S(Q) = \mathop{\mathbb{E}}_{h\sim Q}[R_S(h)]$	The empirical risk of Q on S

$\mathbb{N} = \{1, 2, \dots\}$	The natural numbers (excluding zero)
$\mathbb{N}_0 = \{0, 1, \dots,\}$	The natural numbers including zero
$[n]$	The integers $\{1, 2, \dots, n\}$
$\mathrm{ReLU}(x) = \max(x, 0)$	The Rectified Linear Unit activation function
$\mathtt{MLP}_{(k_0,k_1,\dots,k_{d-1},k_d)}$	The set of d-layer ReLU activated multilayer perceptrons (MLPs) with input dimension k_0, output dimension k_d, and hidden dimensions $k_1, \dots, k_{d-1}$
$\mathtt{MLP}$	The set of all ReLU activated MLPs
$\nu \ll \mu$	The measure ν is absolutely continuous with respect to μ
$\frac{\mathrm{d}\nu}{\mathrm{d}\mu}$	The Radon–Nikodym derivative, defined for $\nu \ll \mu$
$\mathrm{KL}(\nu \Vert \mu) = \int \frac{\mathrm{d}\nu}{\mathrm{d}\mu} \mathrm{d}\nu$	The Kullback–Leibler divergence of ν from μ
$\mathcal{B}(p)$	The Bernoulli distribution with probability of success p
$\mathrm{kl}(q \Vert p) = \mathrm{KL}\big(\mathcal{B}(q) \big\Vert \mathcal{B}(p)\big)$	The “small kl”
$\mathrm{kl}^{-1}(q \vert B)$	The “inverse” small kl $\sup\big\{p \in [0,1] : \mathrm{kl}(q \Vert p) \le B\big\}$
$\triangle_n$	The simplex $\big\{\boldsymbol{u} \in [0,1]^n : \sum_{i=1}^n u_i = 1\big\}$
$\mathrm{kl}(\boldsymbol{q} \Vert \boldsymbol{p}) = \sum_{i=1}^n q_i \ln \frac{q_i}{p_i}$	The vector “small kl” for $\boldsymbol{q}, \boldsymbol{p} \in \triangle_n$
$\Vert W \Vert_{\mathrm{frob}} = \sqrt{\sum_i \sum_j W_{i,j}^2}$	The Frobenius norm of the matrix W
$\Vert W \Vert_{\mathrm{spec}} = \sigma_{\max}(W)$	The Spectral norm of the matrix W, where $\sigma_{\max}(W)$ is the largest singular value of the matrix W
$\lvert s \rvert$	The length of a binary string $s \in \{0,1\}^*$

Chapter 1

Introduction

We study the problem of explaining the empirically observed generalisation ability of overparameterised neural networks. A model is often considered overparameterised if its number of parameters is much larger than the number of samples used to train it. A more accurate view is that a model is overparameterised if it has the capacity to perfectly fit the training data in ways which do not generalise. The fact that ordinary training methods such as Stochastic Gradient Descent (SGD) nevertheless produce models that generalise well even in the overparameterised regime then immediately presents a puzzle, articulated in Zhang et al. (2016) and Belkin et al. (2019) among others, often called the generalisation mystery—what is it about the interplay between the network architectures, data-generating distributions and learning algorithms commonly used in Deep Learning (DL) that enables the algorithms to pick out interpolating solutions that generalise, rather than one of the many that do not? In more common parlance, why do neural networks large enough to overfit frequently avoid doing so?

In order to give a more precise formulation of the generalisation mystery, we first give the following general definition of a learning problem, which encompasses many applications of DL, including regression and classification. We then define what it means for a learning problem to be overparameterised, and finally give a more precise statement of the generalisation mystery.

Definition 1. *(Learning problem).* A *learning problem* is defined as a tuple $(\mathcal{X}, \mathcal{Y}, D, S, \mathcal{H}, \ell)$, where $\mathcal{X}$ is an input space, $\mathcal{Y}$ is an output space, $D \in \triangle(\mathcal{X} \times \mathcal{Y})$ is a data-generating distribution, $S \sim D^n$ is an i.i.d. sample, $\mathcal{H} \subset \mathcal{Y}^{\mathcal{X}}$ is a hypothesis class (also called a model) and $\ell : \mathcal{H} \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}$ is a loss function. The data-generating distribution D is considered to be unknown.

The *goal* of a learning problem is to find a hypothesis $h \in \mathcal{H}$ with low *true risk*

$$R_D(h) := \mathbb{E}_{(x,y)\sim D}\big[\ell(h, (x, y))\big]. \tag{1.1}$$

Since the distribution D and hence the true risk $R_D(h)$ is unknown, it is common to choose a hypothesis h that minimises the *empirical risk*

$$R_S(h) := \frac{1}{|S|} \sum_{(x,y)\in S} \ell(h, (x, y)). \tag{1.2}$$

Choosing h in this way is called Empirical Risk Minimisation (ERM).

To see how neural networks fit into this framework, suppose we have a fixed network architecture with input dimension d_{in}, output dimension d_{out}, and parameters $\boldsymbol{w} \in \mathbb{R}^d$ considered as a single vector. Such a network can be used when $\mathcal{X} \subseteq \mathbb{R}^{d_{\text{in}}}$ and either $\mathcal{Y} \subseteq \mathbb{R}^{d_{\text{out}}}$ or $\mathcal{Y} = [d_{\text{out}}]$. In the case where $\mathcal{Y} = \mathbb{R}^{d_{\text{out}}}$, for example regression, one can simply take the hypothesis class to be $\mathcal{H} = \{h_{\boldsymbol{w}} : \boldsymbol{w} \in \mathbb{R}^d\}$, where $h_{\boldsymbol{w}} : \mathcal{X} \to \mathcal{Y}$ is the function expressed by the neural network with parameters $\boldsymbol{w}$. In the case where $\mathcal{Y} = [d_{\text{out}}]$, for example classification, one may instead let $\phi_{\boldsymbol{w}} : \mathcal{X} \to \mathcal{Y}$ denote the function expressed by the neural network, and then define $h_{\boldsymbol{w}}(x) = \arg\max_{j \in [d_{\text{out}}]} \phi_{\boldsymbol{w}}(x)_j$, namely the coordinate j of the largest value of the vector output $\phi_{\boldsymbol{w}}(x) \in \mathbb{R}^{d_{\text{out}}}$ of the neural network. This extra step is required to ensure that the codomain of $h_{\boldsymbol{w}}$ is $\mathcal{Y} = [d_{\text{out}}]$. In either case, we say that the hypothesis class has been *induced* by the network architecture.

The question now arises as to the choice of architecture. In particular, how flexible it should be in terms of the functions it can express. Conventional wisdom holds that there is a sweet spot. If the architecture is too small, the expressivity of the network will be so limited that there will not exist parameter settings for which $R_D(h_{\boldsymbol{w}})$ is low. This is termed underfitting, and the network is said to be underparameterised. Conversely, if the architecture is too large, the network will be so expressive that there will exist parameter settings for which $R_S(h_{\boldsymbol{w}})$ is low yet $R_D(h_{\boldsymbol{w}})$ is high, and algorithms such as ERM are then liable to choose such parameter settings. The traditional view then holds that in either case the true error $R_D(h_{\boldsymbol{w}})$ will be higher than it would be with an architecture whose size lies somewhere in the middle.

The case where the network architecture is too large is frequently called overfitting. This is appropriate in the case of noisy data[1], where the network is liable to fit the noise as well as the signal. But the existence of parameter settings for which $R_S(h_{\boldsymbol{w}})$ is low yet $R_D(h_{\boldsymbol{w}})$ is high is clearly problematic for algorithms such as ERM even in the absence of noise. For this reason, we call this case overparameterisation rather than overfitting, as in the following definition.

Definition 2. *(Overparameterised regime).* Let $(\mathcal{X}, \mathcal{Y}, D, S, \mathcal{H}, \ell)$ be a learning problem, where the hypothesis class $\mathcal{H} = \{h_{\boldsymbol{w}} : \boldsymbol{w} \in \mathbb{R}^d\}$ is induced by a neural network architecture with parameters $\boldsymbol{w} \in \mathbb{R}^d$. We say that the learning problem (or simply the neural network) is overparameterised, or is in the overparameterised regime, if there exists a parameter setting $\boldsymbol{w} \in \mathbb{R}^d$ such that $R_S(h_{\boldsymbol{w}})$ is low while $R_D(h_{\boldsymbol{w}})$ is high.

Informally, we say that a network is overparameterised if it is flexible enough to express functions that perform well on the sample while performing badly on the distribution. In other words, it can express functions that do not generalise. Intuitively, one should not expect training an overparameterised neural network by minimising $R_S(h_{\boldsymbol{w}})$ via SGD, without any explicit regularisation, to yield good results. After all, why should this process not locate a parameter settin for which $R_S(h_{\boldsymbol{w}})$ is low while $R_D(h_{\boldsymbol{w}})$ is high, which we know to exist? But it is an empirical fact that this does in fact frequently succeed, in the sense that it frequently

[1]A data-generating distribution $D \in \triangle(\mathcal{X} \times \mathcal{Y})$ for a classification task is termed noisy if there does not exist a function $f : \mathcal{X} \to \mathcal{Y}$ for which $\mathbb{P}_{(x,y)\sim D}(f(x) = y) = 1$.

yields parameter settings for which both $R_S(h_{\boldsymbol{w}})$ *and* $R_D(h_{\boldsymbol{w}})$ are low. In other words, it yields functions $h_{\boldsymbol{w}}$ that generalise. The unexplained empirical success of DL in the overparameterised regime is termed the *generalisation mystery*, which we formalise in the following definition.

Definition 3. *(Generalisation Mystery)* The generalisation mystery is the mystery of why, in the overparameterised regime, learning algorithms commonly used in the field of Deep Learning, such as minimisation of the empirical risk $R_S(h_{\boldsymbol{w}})$ via SGD, frequently yield parameter settings $\boldsymbol{w}$ for which the true risk $R_D(h_{\boldsymbol{w}})$ is low, even in the absence of explicit regularisation.

It may be objected that our formalisation of the generalisation mystery and the overparameterised regime fail to specify thresholds for what counts as low empirical risk and high true risk. This is not possible since the thresholds depend on properties of the learning problem, such as the range of the loss function or the irreducible risk of the data-generating distribution (i.e. the Bayes error in the classification setting), and other contingent or subjective factors. However, in DL practice the difference is often quite stark. For example, in classification with deep neural networks, one can commonly find parameter settings with essentially zero train error and 50% test error, as shown by Zhang et al. (2016) on the CIFAR10 dataset (Krizhevsky, Hinton, et al., 2009).

Further, one may object that the term *overparameterisation* is not ideal, since what characterises the overparameterised regime in Definition 2 is not the number of parameters d of the neural network per se, but rather the neural network's capacity to express different functions. Indeed, one can trivially construct neural networks with arbitrarily many parameters but low expressivity. For example, if the first hidden layer has dimension one, then the network will be severely restricted in its expressivity however many subsequent layers there are of whatever size. Conversely, one can construct highly expressive networks with only a single parameter by appropriate choice of activation function. For example, it can be shown that $h_w(x) = \mathbb{1}[\sin(wx) > 0]$, which can be considered a very simple neural network with activation functions sin and sign, is highly expressive while having only a single parameter. Nevertheless, commonly used architectures almost invariably increase in expressivity with increasing numbers of parameters, making the terminology largely unproblematic in practice.

What would it mean to solve the generalisation mystery? One approach, taken by Statistical Learning Theory (SLT), is to prove generalisation bounds (Bousquet et al., 2003; Vapnik, 1999). These typically bound (with high probability) the *generalisation gap*, namely the difference $R_D(h) - R_S(h)$ between the true and empirical risks. Early SLT focused on uniform bounds on the generalisation gap, where the value of the bound does not depend on h, and the bound holds with high probability for all $h \in \mathcal{H}$ simultaneously. This forces the bound to be loose enough to accommodate the worst-case generalisation gap, which, in the overparameterised regime, is by definition large. Uniform bounds are therefore incapable of explaining the generalisation mystery, as the bound will be too loose to explain the small generalisation gap of the learned network. In the classification setting, such bounds are typically vacuous, meaning they are larger than 1 and so fail to constrain the generalisation gap at all.

PAC-Bayesian theory, pioneered in the seminal works McAllester (1998) and Shawe-Taylor and Williamson (1997), is a more modern branch of SLT used to derive non-uniform generalisation bounds, which has enabled it to have much greater success in producing non-vacuous generalisation bounds in the overparameterised regime (Dziugaite and Roy, 2017; Pérez-Ortiz et al., 2021; Zhou et al., 2018). However, non-uniform bounds alone also cannot resolve the generalisation mystery, since they simply shift the question of why commonly used DL algorithms pick out parameter settings for which the true risk is low, to the question of why they pick out parameter settings for which the bound is small. That empirical fact is left unexplained. While this is not an issue for PAC-Bayes-inspired learning algorithms, which employ the bound as a training objective (e.g. Dziugaite and Roy (2017)), it is primarily the generalisation of *commonplace* learning algorithms that we wish to explain. Other obstacles to PAC-Bayesian theory resolving the generalisation mystery are discussed in Chapter 2.

An independent approach to explaining the generalisation mystery is to search for measures of neural networks that are empirically predictive of generalisation across a large dataset of networks (Jiang et al., 2019; Neyshabur et al., 2017a; Unterthiner et al., 2020). This may be motivated by an appeal to Occam's razor; of all the interpolating solutions, perhaps the simple ones are more likely to generalise. Such measures are often termed complexity measures. Ideally, one is able to find a complexity measure that is both predictive of generalisation and corresponds to some intuitive notion of complexity. Further, it should be the case that an increase in the complexity measure is causally responsible for an increase in generalisation gap, rather than merely correlated.

These lines of research are complementary. On the one hand, empirical discoveries of predictive and causal complexity measures motivate the search for rigorous generalisation bounds leveraging such complexity measures. On the other hand, tight non-uniform generalisation bounds suggest complexity measures that can be studied empirically. A full explanation of the generalisation mystery will likely require a synthesis of the two approaches.

We now outline our three contributions to the investigation of this question.

1. One hope for generalisation bounds is that they may ultimately be tight enough to enable self-certified learning. This is where test set bounds are replaced by generalisation bounds, eliminating the need to withhold data from the training algorithm in order to validate its output. In the ideal case the certificate is information-rich. For example, rather than bounding simply the error rate of a classifier, it should provide rich information on which misclassification errors the classifier is more or less likely to make. We take one step in this direction in Chapter 3 by proving a very flexible generalisation of a well-known PAC-Bayes bound. Our generalisation constrains the entire distribution of the output of a predictor (such as a neural network) over a set of user-specified error types, and therefore is not limited to multiclass classification in particular. Further, we show that controlling the entire distribution over error types implies bounds on every possible linear combination of the error types, which all hold simultaneously with the same probability as the original bound.

2. PAC-Bayesian theory predominantly addresses stochastic models which, for neural net-

works, generally equates to a distribution over the weights, where a fresh sample from this distribution is drawn for each prediction. This is counter to common DL practice and therefore constitutes a significant obstacle in the ability of the PAC-Bayesian theory to explain the generalisation mystery. In Chapter 4 we investigate whether the empirically observed compressibility of neural networks trained by ordinary methods can be exploited to do away with the stochasticity usually required by PAC-Bayesian theory, while still obtaining non-vacuous generalisation bounds. In other words, we investigate the ability of PAC-Bayesian theory to produce tight generalisation bounds for deterministic networks trained by ordinary methods, which should be the ultimate goal of SLT. We prove a result which bounds the true error of a deterministic neural network in terms of the empirical margin loss of its compression, plus a term that measures the size of the compressed network. The required margin is smaller for higher fidelity compressions. In other words, we show that if a network has large margin on the train set, and there exists a high-fidelity compression, then this network generalises.

3. Many neural network complexity measures proposed in the literature, including some derived from rigorous generalisation bounds, have been empirically shown to negatively correlate with generalisation gap (Jiang et al., 2019). While this still makes them predictive of generalisation gap (for which only positive mutual information between the complexity measure and the generalisation gap is required), a complexity measure worth the name should *positively* correlate with generalisation gap. In Chapter 5 we propose a novel neural network complexity measure we call the distillation complexity and show that it is indeed positively correlated with generalisation gap across a suite of models. Further, we see that our distillation complexity does not grow with model size, in contrast to many other complexity measures. We also provide some evidence that it plays a causal role in determining the value of the generalisation gap.

1.1 Motivation

Why seek an explanation of the generalisation mystery? Should the academic community not be satisfied with having discovered algorithms which do in fact produce models which generalise very successfully? No, and for a number of reasons, three of which we now give.

First, basic research can produce significant down-stream benefits. Attacking the core mysteries of a field occasionally yields deep insight, which can revolutionise the discipline in ways that are hard to predict in advance. In the case of DL, the field resembles a bag of poorly understood tricks, with post-hoc explanations of empirically successful methods sometimes turning out to be wrong (e.g. Santurkar et al. (2018) in the case of batch normalisation). An explanation of the generalisation mystery, which would necessarily address the interplay between architecture, learning algorithm, and data-generating distribution, may yield a principled science of DL. This could allow a much more judicious application of the tricks discovered so far, and a more directed search for new ones. The current situation is one in which it is possible to believe that the field progresses mostly in a trial-and-error fashion, and produces post-hoc explanations for successful methods that produce little understanding of what is actually going on. This makes training large models financially risky, especially at the enormous scale of frontier models. For example, the lack of rigorous theory means that hyperparameter settings for large models must be extrapolated from sweeps across smaller models, with no guarantee that these extrapolated values will be optimal. While scaling laws (Hoffmann et al., 2022; Kaplan et al., 2020) relating training loss to model size, training time and sample size have been empirically observed to hold over many orders of magnitude, and surely inform decisions on high-budget training runs, without theoretical grounding these laws provide limited assurance that a given training run will turn out to be a good investment.

Second, a rigorous understanding of the generalisation mystery may be a prerequisite for a predictive theory of domain shift. The lack of such a theory is currently a significant obstacle to the deployment of neural networks in high stakes environments, where domain shift is frequently unavoidable (Koh et al., 2021; Quiñonero-Candela et al., 2022). A deeper understanding of when a neural network is likely to generalise to a new environment could produce significant savings in cases where testing in each new environment is costly and time-consuming. And it could allow safer deployment to unstable environments when continual testing is not practical.

Third, and more speculatively, a deep understanding of generalisation may enable not just the final loss values of foundation models such as Large Language Models (LLMs) or image generators to be predicted in advance of training, but also their specific capabilities. Despite the controversy over the meaning and existence of so-called emergent capabilities, it is not disputed that currently the capabilities of these models cannot be predicted in advance of training, even knowing their final loss (Ganguli et al., 2022). Instead, we discover what tasks they can and cannot accomplish at the end of training through extensive evaluation (Srivastava et al., 2022; Wei et al., 2022a,b). This is an ongoing process, with some capabilities only discovered long after deployment. As models become more capable, it does not bode well that we can unintentionally train models with dangerous capabilities that are only discovered after training or even deployment.

Chapter 2

Background: SLT and the Generalisation Mystery

The goal of Statistical Learning Theory (SLT) is to bound the discrepancy between the empirical and true risks of a hypothesis—the so-called generalisation gap—even when the hypothesis in question is dependent on the sample used to evaluate the empirical risk. While bounding the discrepancy for a fixed hypothesis is straightforward, the difficulty introduced by the freedom to choose the hypothesis based on the sample is significant. In this chapter we outline two broad approaches found in SLT. First, classical SLT provides a number of uniform generalisation bounds, uniform in the sense that they bound the generalisation gap of all hypotheses simultaneously by a constant. The constants of the various bounds involve measures of complexity of the hypothesis class, thus providing different formalisations of the intuition that while the hypothesis class may be large—even uncountably infinite—one may still expect the worst-case generalisation gap to be small if the class is relatively simple in terms of the functions it can express. The fact that the bound holds for all hypotheses simultaneously means it in particular holds for any sample-dependent choice of hypothesis, such as that returned by some learning algorithm.

While the theory is elegant, we will see that uniform bounds cannot explain generalisation in the overparameterised regime. Further, although the classical theory can be extended to non-uniform bounds in a number of ways, it is only with the advent of PAC-Bayesian theory that we witness non-uniform bounds that plausibly have the potential to explain the generalisation mystery. PAC-Bayesian theory considers stochastic hypotheses, namely distributions over the hypothesis class that predict according to a (deterministic) hypothesis randomly drawn from the distribution, with a fresh hypothesis drawn for each prediction. While this constitutes a step away from usual Deep Learning practice, the dramatically improved tightness of the resulting bounds compared to classical SLT indicates that it may nevertheless be a step towards understanding the generalisation mystery. Indeed, in Chapter 4 we explore how we can reconcile the stochastic hypotheses of PAC-Bayesian theory with the deterministic hypotheses of ordinary DL practice.

2.1 Classical SLT

Let $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ be a hypothesis class and $\ell : \mathcal{H} \times (\mathcal{X} \times \mathcal{Y}) \to [0, C]$ be a bounded loss function. Given a data-generating distribution $D \in \triangle(\mathcal{X} \times \mathcal{Y})$, the goal of a learning algorithm is to find a hypothesis $h \in \mathcal{H}$ with low *true risk* $R_D(h)$, defined as the expected loss of h on a single draw from D, namely

$$R_D(h) = \mathbb{E}_{(x,y)\sim D}\big[\ell(h,(x,y))\big].$$

However, the distribution D is typically unknown. Instead, we commonly assume we are provided with an i.i.d. sample $S \sim D^n$, and can only evaluate h according to the *empirical risk* $R_S(h)$, defined as the mean loss of h across the sample S, namely

$$R_S(h) = \frac{1}{|S|}\sum_{(x,y)\in S} \ell(h,(x,y)).$$

A familiar choice of loss function is the zero-one loss $\ell_{01}(h,(x,y)) = \mathbb{1}[h(x) \neq y]$ used in classification tasks with $\mathcal{Y} = [N]$. It is then common to denote the loss as a function $\ell_{01} : \mathcal{Y}^2 \to \{0,1\}$, where the domain is $\mathcal{Y}^2$ rather than $\mathcal{H} \times (\mathcal{X} \times \mathcal{Y})$. In such cases, we define the true and empirical risks to be

$$R_D(h) = \mathbb{E}_{(x,y)\sim D}\big[\ell(h(x),y)\big] \quad \text{and} \quad R_S(h) = \frac{1}{|S|}\sum_{(x,y)\in S} \ell(h(x),y), \tag{2.1}$$

respectively, which we can then call the true and empirical *errors*.

Since the zero-one loss is not differentiable, a common procedure is to first learn a function $H : \mathcal{X} \to \triangle_N$ with low cross-entropy loss $\ell_{\mathrm{CE}}(H,(x,y)) = -\ln(H(x)_y)$, and then classify according to the hypothesis $h(x) = \mathrm{argmax}_{j\in[N]} H(x)_j$. In this case, it is common to denote the loss as a function $\ell_{\mathrm{CE}} : \hat{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}$, where $\hat{\mathcal{Y}} = \triangle_N$. Note we need to be sure that the function defined by $h(x) = \mathrm{argmax}_{j\in[N]} H(x)_j$ is indeed an element of $\mathcal{H}$. We can then define the true and empirical risks of H by substituting H for h in Equations (2.1). Going forward, it will always be clear from the context whether the domain of the loss function is $\mathcal{H} \times (\mathcal{X} \times \mathcal{Y})$, $\mathcal{Y}^2$ or $\hat{\mathcal{Y}} \times \mathcal{Y}$, and therefore which definitions of the true and empirical risks are being employed.

Given a learning algorithm $\mathcal{A} : (\mathcal{X} \times \mathcal{Y})^* \to \mathcal{H}$, i.e. a function taking a sample S of arbitrary length and returning a learned hypothesis $\mathcal{A}(S) \in \mathcal{H}$, the goal of learning theory is to upper bound the unknown true risk $R_D(\mathcal{A}(S))$. This is typically done by bounding the *generalisation gap* $R_D(h) - R_S(h)$ for all $h \in \mathcal{H}$ simultaneously, where the bound can be a function of all known quantities, e.g. the sample S, the hypothesis h, the entire hypothesis class $\mathcal{H}$, the bound C on the loss function, and so on. Such bounds are termed generalisation bounds, and when the bound is independent of h, we say that it is uniform.

As a first step toward a generalisation bound, suppose we are interested in bounding the generalisation gap $R_D(h) - R_S(h)$ of a single hypothesis $h \in \mathcal{H}$. This can be achieved by a straightforward application of one of many concentration inequalities. As a simple example[1], using Chebyshev's inequality we have

$$\begin{aligned}
\mathbb{P}_{S\sim D^n}\big(R_D(h) - R_S(h) > \epsilon\big) &\leq \mathbb{P}_{S\sim D^n}\big(|R_S(h) - R_D(h)| > \epsilon\big) && (2.2)\\
&= \mathbb{P}_{S\sim D^n}\big(|R_S(h) - \mathbb{E}_{S'\sim D}[R_{S'}(h)]| > \epsilon\big) && (2.3)\\
&\leq \frac{\mathbb{V}_{S\sim D^n}(R_S(h))}{\epsilon^2} && (2.4)\\
&= \frac{\mathbb{V}_{(x,y)\sim D}(R_{\{(x,y)\}}(h))}{n\epsilon^2} && (2.5)\\
&\leq \frac{C^2}{n\epsilon^2}, && (2.6)
\end{aligned}$$

and so

$$\mathbb{P}_{S\sim D^n}\big(R_D(h) - R_S(h) \leq \epsilon\big) \geq 1 - \frac{C^2}{n\epsilon^2}.$$

This says that the true risk $R_D(h)$ is at most ϵ larger than the empirical risk $R_S(h)$ with probability at least $1 - C^2/n\epsilon^2$. A frequent substitution at this point is $\delta = C^2/n\epsilon^2$, so that the bound becomes

$$\mathbb{P}_{S\sim D^n}\left(R_D(h) - R_S(h) \leq \frac{C}{\sqrt{n\delta}}\right) \geq 1 - \delta. \tag{2.7}$$

This form is useful when one wishes to find the minimum sample size n such that the bound fails with probability at most δ, where δ is deemed an acceptable threshold.

Suppose now that we have a learning algorithm $\mathcal{A} : (\mathcal{X} \times \mathcal{Y})^* \to \mathcal{H}$. While it is tempting to make the substitution $h = \mathcal{A}(S)$ into Inequality (2.7), this would not yield a valid bound. To see why, notice that line (2.5) uses the fact that the $\ell(h, (x_i, y_i))$ are i.i.d., which is not the case for $\ell(\mathcal{A}(S), (x_i, y_i))$. To get an intuitive feel for what is happening here, suppose we have 1000 biased coins, each with a possibly different bias. If we pick any particular coin and toss it 100 times, then the proportion of heads will be approximately the bias of the coin. However, if we toss all 1000 coins 100 times each, and then consider the one that came up heads least often, *this* coin's proportion of heads will likely be a significant underestimate of its bias. Analogously, while $R_S(h)$ may be a good estimator of $R_D(h)$ for every particular $h \in \mathcal{H}$, the hypothesis returned by the Empirical Risk Minimiser (ERM) algorithm $\mathcal{A}_{\text{ERM}}(S) = \text{argmin}_{h\in\mathcal{H}} R_S(h)$ is likely to have empirical risk $R_S(h)$ significantly lower than $R_D(h)$.

One approach to solving this problem is to bound $\sup_{h\in\mathcal{H}}(R_D(h) - R_S(h))$, which will then bound $R_D(\mathcal{A}(S)) - R_S(\mathcal{A}(S))$ as a particular case. Such a bound is termed a *uniform* bound, as the same bound applies uniformly to every $h \in \mathcal{H}$. This is the approach taken by most of classical SLT. While there are many such bounds—far too many to cover in this brief overview—we present the following three classes of bound with illustrative examples; union bounds over a countable $\mathcal{H}$, bounds based on the Vapnik-Chervonenkis dimension of $\mathcal{H}$, and those based on the Rademacher Complexity of $\mathcal{H}$ on S.

[1]Much better concentration inequalities, such as Hoeffding's inequality are available, but our purpose here is simply illustration.

2.1.1 Union bounds

Suppose we have a finite hypothesis class $\mathcal{H} = \{h_1, \dots, h_K\}$. Then from Inequality (2.7) we see that

$$\forall k \in [K] \quad \mathbb{P}_{S\sim D^n}\left(R_D(h_k) - R_S(h_k) \leq \frac{C}{\sqrt{n\delta}}\right) \geq 1-\delta.$$

Our goal is to get the quantifier $\forall k \in [K]$ inside the probability so that the bound holds for all $h \in \mathcal{H}$ simultaneously, and in particular the hypothesis $\mathcal{A}(S)$ returned by our learning algorithm.

The key technique, used over and again in SLT, relies on the elementary fact that $\mathbb{P}(E_1 \cup E_2) \leq \mathbb{P}(E_1) + \mathbb{P}(E_2)$ for any events E_1, E_2, not necessarily independent, and any probability distribution $\mathbb{P}$. Sometimes termed Boole's inequality, this extends to arbitrary finite or countably infinite collections of events

$$\mathbb{P}\left(\bigcup_{i\in\mathcal{I}} E_i\right) \leq \sum_{i\in\mathcal{I}} \mathbb{P}(E_i), \tag{2.8}$$

where $\mathcal{I} \subset \mathbb{N}$. Moreover, the union bound can be used to deal with the quantifiers "$\forall$" and "$\exists$" by expressing them as unions. Indeed, suppose that for each $i \in \mathcal{I}$ we have a random variable Z_i, a set $B_i \subseteq \mathbb{R}$ and an event $E_i = \{Z_i \in B_i\}$. Then

$$\begin{aligned}
\mathbb{P}(\forall i \in \mathcal{I}\ \ Z_i \in B_i) &= 1 - \mathbb{P}(\exists i \in \mathcal{I}\ \ Z_i \notin B_i) \\
&= 1 - \mathbb{P}\left(\bigcup_{i\in\mathcal{I}} E_i^C\right) \\
&\geq 1 - \sum_{i\in\mathcal{I}} \mathbb{P}(E_i^C) \\
&= 1 - \sum_{i\in\mathcal{I}} (1 - \mathbb{P}(Z_i \in B_i)).
\end{aligned}$$

Applying this to our case gives

$$\begin{aligned}
&\mathbb{P}_{S\sim D^n}\left(\forall k \in [K]\ \ R_D(h_k) - R_S(h_k) \leq \frac{C}{\sqrt{n\delta}}\right) \\
&\qquad \geq 1 - \sum_{k=1}^{K}\left(1 - \mathbb{P}_{S\sim D^n}\left(R_D(h_k) - R_S(h_k) \leq \frac{C}{\sqrt{n\delta}}\right)\right) \\
&\qquad \geq 1 - \sum_{k=1}^{K} \delta \\
&\qquad = 1 - K\delta.
\end{aligned}$$

Now, since δ was arbitrary, we can replace it with δ/K to obtain

$$\mathbb{P}_{S\sim D^n}\left(\forall k \in [K]\ \ R_D(h_k) - R_S(h_k) \leq C\sqrt{\frac{K}{n\delta}}\right) \geq 1-\delta, \tag{2.9}$$

and thus, in particular,

$$\mathbb{P}_{S\sim D^n}\left(R_D(\mathcal{A}(S)) - R_S(\mathcal{A}(S)) \leq C\sqrt{\frac{K}{n\delta}}\right) \geq 1-\delta, \tag{2.10}$$

for any learning algorithm $\mathcal{A}$. Therefore, in this illustrative example, the cost of permitting the hypothesis $h \in \{h_1, \dots h_K\}$ to be sample-dependent is an increase in the bound by a factor of $\sqrt{K}$.

Going back to our coin tossing example, the intuition is that while we may be confident that for any *particular* coin the proportion of heads from 100 tosses is within ϵ of the bias, we must increase ϵ if we want to be confident that this is the case for all 1000 coins *simultaneously*, which is one way to guarantee this is the case in particular for the coin that comes up heads the fewest times.

The fact that we must increase the bound by a factor of $\sqrt{K}$ in order to apply a union bound over K hypotheses is not optimal, and comes from the poor dependence on δ in the original bound (2.7) derived using Chebyshev's concentration inequality. There are a variety of concentration inequalities however, and applying Hoeffding's yields the following bound for a particular $h \in \mathcal{H}$ with much better dependence on δ than (2.7)

$$\mathbb{P}_{S\sim D^n}\left(R_D(h) - R_S(h) \leq C\sqrt{\frac{1}{2n}\ln\frac{2}{\delta}}\right) \geq 1-\delta. \tag{2.11}$$

The logarithmic dependence on δ in this bound makes it much cheaper to take union bounds; decreasing δ by a factor of K increases the argument of the square root by only $\frac{1}{2n}\ln K$. Indeed, following the same process as before, we obtain the following uniform bound for our finite hypothesis class $\mathcal{H} = \{h_1, \dots, h_K\}$

$$\mathbb{P}_{S\sim D^n}\left(\forall k \in [K] \quad R_D(h_k) - R_S(h_k) \leq C\sqrt{\frac{1}{2n}\ln\frac{2K}{\delta}}\right) \geq 1-\delta. \tag{2.12}$$

2.1.2 Union bounds with a prior over the hypothesis class

The derivation of the union bounds (2.9) and (2.12) relies on the fact that the failure probabilities δ/K of the individual bounds sum to δ, the desired failure probability of the union bound. Viewed in this way, it becomes clear that we can combine countably many bounds provided their probabilities of failure still sum to δ. To that end, suppose we have a countable hypothesis class $\mathcal{H} = \{h_k : k \in \mathcal{I}\}$, where $\mathcal{I} \subseteq \mathbb{N}$, and let $\pi \in \triangle(\mathcal{H})$ be a distribution over $\mathcal{H}$ such that $\pi(h) > 0$ for all $h \in \mathcal{H}$. Then we can combine countably many bounds of the form (2.11), one for each $h \in \mathcal{H}$, by setting $\delta_h = \pi(h)\delta$, to obtain

$$\mathbb{P}_{S\sim D^n}\left(\forall h \in \mathcal{H} \ \ R_D(h) - R_S(h) \leq C\sqrt{\frac{1}{2n}\ln\frac{2}{\pi(h)\delta}}\right) \geq 1-\delta. \tag{2.13}$$

The derivation is a straightforward application of the union bound:

$$
\begin{aligned}
&\mathbb{P}_{S\sim D^n}\left(\forall h\in\mathcal{H}\;\; R_D(h)-R_S(h)\le C\sqrt{\frac{1}{2n}\ln\frac{2}{\pi(h)\delta}}\right)\\
&\ge 1-\sum_{h\in\mathcal{H}}\left\{1-\mathbb{P}_{S\sim D^n}\left(R_D(h)-R_S(h)\le C\sqrt{\frac{1}{2n}\ln\frac{2}{\pi(h)\delta}}\right)\right\}\\
&\ge 1-\sum_{h\in\mathcal{H}}\pi(h)\delta\\
&=1-\delta.
\end{aligned}
$$

Note that the bound in (2.13) is no longer uniform; its value depends on the hypothesis h via the probability $\pi(h)$. Observing that the bound is smallest for values of h for which $\pi(h)$ is large, we see that a natural choice for π is to put greater mass on hypotheses h we believe our learning algorithm is more likely to return. In fact, by doing so we can interpret π as our prior beliefs of the hypotheses our algorithm will return.

Moreover, the fact that the bound is no longer uniform over $\mathcal{H}$ opens up a second strategy for picking h aside from ERM; pick the hypothesis h which has the smallest true risk bound, namely

$$
\mathcal{A}_{\text{min-bound}}(S)=\operatorname{argmin}_{h\in\mathcal{H}}\left(R_S(h)+C\sqrt{\frac{1}{2n}\ln\frac{2}{\pi(h)\delta}}\right). \tag{2.14}
$$

If π is chosen to reflect our prior beliefs on which h have low true risk, $\mathcal{A}_{\text{min-bound}}$ can then be interpreted as a balance between our prior beliefs $\pi(h)$ and evidence $R_S(h)$. Bound minimisation as a learning algorithm is a frequently employed concept in both classical SLT and PAC-Bayesian theory, as we will see.

Note that the dependence on n of the generalisation bounds we have seen so far is as $1/\sqrt{n}$. This is typical in the so-called "non-realisable" setting, where there does not exist $h\in\mathcal{H}$ such that $R_D(h)=0$. In the realisable setting the dependence on n is as $1/n$, which is much better. Indeed, through a very simple argument, the following bound was proven in McAllester (1998).

Theorem 1. *(McAllester (1998), Preliminary Theorem 1) Let $\mathcal{H}\subseteq\mathcal{Y}^{\mathcal{X}}$ be a countable hypothesis class, $\ell:\mathcal{Y}^2\to\{0,1\}$ the zero-one loss function $\ell(\hat{y},y)=\mathbb{1}[\hat{y}\ne y]$, and $D\in\triangle(\mathcal{X}\times\mathcal{Y})$ a data-generating distribution such that there exists $h\in\mathcal{H}$ with $R_D(h)=0$. Then for any $\pi\in\triangle(\mathcal{H})$ and $\delta\in(0,1]$,*

$$
\mathbb{P}_{S\sim D^n}\left(\exists h\in\mathcal{H}:R_S(h)=0\;\; and\;\; R_D(h)>\frac{1}{n}\left(\ln\frac{1}{\pi(h)\delta}\right)\right)<\delta. \tag{2.15}
$$

If we take as our learning algorithm $\mathcal{A}_{ERM}(S)\in\{h\in\mathcal{H}:R_S(h)=0\}$, we have

$$
\mathbb{P}_{S\sim D^n}\left(R_D(A_{ERM}(S))\le\frac{1}{n}\left(\ln\frac{1}{\pi(\mathcal{A}(S))\delta}\right)\right)\ge 1-\delta. \tag{2.16}
$$

Suppose that instead of representing our prior beliefs, we simply wish to choose π so as to minimise the bound (2.13) in expectation over the random draw $S \sim D^n$. This strategy was first suggested in Langford and Blum (2003) (see Section 2.1). They note that if, for any algorithm $\mathcal{A}$, we define the distribution $\rho \in \triangle(\mathcal{H})$ by $\rho(h) = \mathbb{P}_{S\sim D^n}(\mathcal{A}(S) = h)$, then the expectation of the bound given in (2.16) can be written as

$$\frac{1}{n}\mathbb{E}_{S\sim D^n}\left[\ln\frac{1}{\delta} + \ln\frac{1}{\pi(\mathcal{A}(S))}\right] = \frac{1}{n}\ln\frac{1}{\delta} + \frac{1}{n}\sum_{h\in\mathcal{H}}\rho(h)\ln\frac{1}{\pi(h)}.$$

It is then immediate from Gibbs' inequality that this is minimised by $\pi = \rho$. While this is a perfectly legitimate choice—ρ is independent of S—it is not practical as ρ depends on the unknown data-generating distribution D. Nevertheless, it demonstrates that a good choice of prior π is one that "anticipates" which hypotheses are likely to be returned by the algorithm $\mathcal{A}$ on the distribution D, a property we will see mirrored in PAC-Bayes bounds in Section 2.3.

It is important to note that while taking union bounds with a non-uniform prior $\pi \in \triangle(\mathcal{H})$ may yield empirically tighter generalisation bounds for our chosen algorithm $\mathcal{A}$, this is at the expense of reduced explanatory power for the success of $\mathcal{A}$. Instead of being able to say that $\mathcal{A}(S)$ has small generalisation gap because the sample size n is large enough that *all* $h \in \mathcal{H}$ have small generalisation gap, we may end up in a situation where the bound on the generalisation gap of $\mathcal{A}(S)$ is tight only because of our choice of prior π, where a different choice of π may have produced much worse bounds. In such a case, we are *lucky* that our prior π placed large mass on the hypothesis that $\mathcal{A}$ happened to return, and this luckiness is a gap in our explanation for why $\mathcal{A}$ performs well. Even if our algorithm is $\mathcal{A}_{\text{min-bound}}$ defined in Equation (2.14), which is explicitly biased towards h for which $\pi(h)$ is large, there is an element of luckiness in whether there exists $h \in \mathcal{H}$ that achieves both low $R_S(h)$ and high $\pi(h)$ simultaneously. We discuss this element of luck more in Section 2.2.

2.1.3 Uncountable hypothesis classes

The previous section introduced union bounds as a method to combine countably many generalisation bounds, one for each hypothesis in a countable class. However, most commonly used hypothesis classes, such as the set of Multi-Layer Perceptrons (MLPs) of a given architecture, are uncountable. Of course in practice these hypothesis classes must be representable on a finite computer and are therefore themselves necessarily finite. However, this implicit finite hypothesis class is typically too large to produce non-vacuous generalisation bounds. Moreover, treating the hypothesis class simply as a finite set obscures relevant structure that SLT can take advantage of. This section introduces two generalisation bounds that do exactly that; they bound the generalisation gap by formalising the *capacity* of $\mathcal{H}$ to fit data in arbitrary ways.

One informal takeaway of the previous section is that the more freedom we have in our choice of hypothesis h (i.e. the larger $\mathcal{H}$), the more suspicious we should be that its empirical risk $R_S(h)$ is a good measure of its true risk $R_D(h)$. Intuitively, however, our degree of choice should not be measured in terms of the cardinality of $\mathcal{H}$, but rather in terms of the capacity of $\mathcal{H}$ to make arbitrary predictions. For example, if we know that there exists $\boldsymbol{x} = (x_1, \dots, x_n) \in \mathcal{X}^n$

such that for any possible labelling $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathcal{Y}^n$ we will always be able to find a hypothesis $h \in \mathcal{H}$ that produces labels $\boldsymbol{y}$ on $\boldsymbol{x}$ (or at least achieves very low loss on $\boldsymbol{x}$), then this means $\mathcal{H}$ has the capacity to fit the noise as well as the signal. Therefore we should certainly be suspicious that the empirical risk of $\mathcal{A}_{\mathrm{ERM}}(S) \in \mathcal{H}$ is a good proxy for the true risk when S is around this size. In common DL parlance, we say that the class $\mathcal{H}$ has the capacity to overfit. Conversely, if we know that above a certain sample size n the class $\mathcal{H}$ is severely restricted in how it can label any sample of size n, then the class is restricted in how much of the noise it can fit in addition to the signal, meaning a low empirical risk of $\mathcal{A}_{\mathrm{ERM}}(S)$ above this sample size is strong evidence that the true risk is also low. Crucially, this is the case even if $|\mathcal{H}|$ is very large, perhaps uncountably infinite. This intuition was formalised as the Vapnik-Chervonenkis dimension (Vapnik and Chervonenkis, 2015), to which we now turn.

2.1.4 Vapnik-Chervonenkis dimension

Consider the case of binary classification with zero-one loss, namely label space $\mathcal{Y} = \{0, 1\}$, hypothesis class $\mathcal{H} \subseteq \{0, 1\}^{\mathcal{X}}$, loss function $\ell_{01}(h(x), y) = \mathbb{1}[h(x) \neq y]$, and true and empirical risks $R_D(h)$ and $R_S(h)$ defined as in Equations (2.1). We have the following definitions.

Definition 4. Let $\mathcal{H} \subseteq \{0, 1\}^{\mathcal{X}}$. The set $\boldsymbol{x} \in \mathcal{X}^n$ is *shattered* by $\mathcal{H}$ if

$$\big\{(h(x_1), \ldots, h(x_n)) : h \in \mathcal{H}\big\} = \{0, 1\}^n,$$

namely if $\mathcal{H}$ has the capacity to produce every possible binary labelling $\boldsymbol{y} \in \{0, 1\}^n$ of $\boldsymbol{x}$.

Definition 5. (Vapnik-Chervonenkis definition) Let $\mathcal{H} \subseteq \{0, 1\}^{\mathcal{X}}$. The VC-dimension $d_{\mathrm{VC}}(\mathcal{H})$ is defined as the largest n such that there exists $\boldsymbol{x} \in \mathcal{X}^n$ shattered by $\mathcal{H}$, or ∞ if no largest n exists.

It was shown in Vapnik (2013) that this notion of complexity of $\mathcal{H}$ can be leveraged to bound the maximum generalisation gap over the hypothesis class provided $d_{\mathrm{VC}}(\mathcal{H})$ is finite. We give the following form of the bound which more straightforward to parse.

Theorem 2. *(Abu-Mostafa et al. (2012), Theorem 2.5) For any $\mathcal{H} \subseteq \{0, 1\}^{\mathcal{X}}$ such that $d_{VC}(\mathcal{H}) < \infty$, any data-generating distribution $D \in \triangle(\mathcal{X} \times \{0, 1\})$, confidence level $\delta \in (0, 1]$ and sample size n, with probability at least $1 - \delta$ over the sample $S \sim D^n$, for all $h \in \mathcal{H}$ simultaneously,*

$$R_D(h) - R_S(h) \leq \sqrt{\frac{8}{n}\left(d_{VC}(\mathcal{H})\left(\ln\left(\frac{2n}{d_{VC}(\mathcal{H})}\right) + 1\right) + \ln\frac{4}{\delta}\right)}, \tag{2.17}$$

where $R_D(h)$ and $R_S(h)$ denote the true and empirical error of the hypothesis h, respectively.

Ignoring logarithmic factors, this bound is $O(\sqrt{d_{\mathrm{VC}}(\mathcal{H})/n})$, meaning we need $n \gg d_{\mathrm{VC}}(\mathcal{H})$ in order to achieve tight bounds. Is the sample size n in fact much larger than the VC-dimension d_{VC} in ordinary DL practice? While determining the exact VC-dimension of a hypothesis class induced by a neural network architecture is difficult, various asymptotic lower bounds for fully connected neural networks have been established that scale at least with the number of parameters (weights and biases) of the network (Bartlett et al., 1998; Bartlett, 1993; Bartlett et al., 2019; Maass, 1994). For example, the following lower bound was proved in Bartlett et al. (2019).

Theorem 3. *(Bartlett et al. (2019), Theorem 3) There exists a universal constant C such that for any number of weights n_w and number of layers n_l such that $n_w > Cn_l > C^2$, there exists a ReLU network with at most n_l layers and at most n_w parameters with VC-dimension at least $n_w n_l \ln(n_w/n_l)/C$.*

In particular, this result shows that for a hypothesis class $\mathcal{H}$ corresponding to a fully connected neural network with n_w parameters, we have that $d_{\mathrm{VC}}(\mathcal{H})$ is $\Omega(n_w)$. Loosely, combining this with the above observation means we require $n \gg n_w$ in order for Theorem 2 to produce tight bounds. Unfortunately this is not usually the case in DL, where the number of parameters typically far exceeds the sample size. While this is only a heuristic argument, the order of the bounds, combined with the fact that the performance of neural networks does not always degrade as the number of parameters becomes very large (for example, see Loog et al., 2020; Nakkiran et al., 2021), indicates that Theorem 2 is unlikely to produce tight bounds for typical neural networks.

2.1.5 Rademacher complexity

The VC dimension was motivated by the intuition that if, for a sample size n, there exists $\boldsymbol{x} \in \mathcal{X}^n$ that can be arbitrarily labelled by $\mathcal{H}$, then there is a risk that our sample $S \in (\mathcal{X} \times \mathcal{Y})^n$ corresponds to this $\boldsymbol{x}$, in which case observing low $R_S(\mathcal{A}_{\mathrm{ERM}}(S))$ is not good evidence that $R_D(\mathcal{A}_{\mathrm{ERM}}(S))$ is also low. This worst-case analysis was required in order to obtain a generalisation bound independent of the sample S. But since S is known, can we not inspect whether it *in fact* matches $\boldsymbol{x}$? Surely, if the capacity of $\mathcal{H}$ to produce arbitrary labellings of our sample is in fact highly restricted, then we are quite unlikely to fit the noise on S, even if there exist *other* samples for which we could fit arbitrary noise. What this indicates is the need for a sample-dependent bound involving a sample-wise complexity measure of $\mathcal{H}$. Indeed, this is exactly what the Rademacher complexity measures, in a way that produces a corresponding generalisation bound.

Definition 6. (Rademacher complexity) Let $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ be a hypothesis class, $S \in (\mathcal{X} \times \mathcal{Y})^n$ a sample, $\ell : \mathcal{H} \times (\mathcal{X} \times \mathcal{Y}) \to [0, C]$ a bounded loss function, and $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_n)$ a tuple of Rademacher random variables, such that $\mathbb{P}(\sigma_i = 1) = \mathbb{P}(\sigma_i = -1) = 1/2$ for all i. The Rademacher complexity of $\mathcal{H}$ on S given ℓ is

$$\mathcal{R}(\mathcal{H}, \ell, S) = \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \ell(h, (x_i, y_i))\right].$$

In the case of binary classification with zero-one loss, for each value of $\boldsymbol{\sigma}$ we can write $S = S_+ \cup S_-$, where $S_+ = \{(x_i, y_i) \in S : \sigma_i = 1\}$ and $S_- = \{(x_i, y_i) \in S : \sigma_i = -1\}$. The supremum then optimises over $h \in \mathcal{H}$ the number of incorrect classifications on S_+ minus the number of incorrect classifications on S_-, divided by the sample size n. Averaging this over all values of $\boldsymbol{\sigma}$ then yields the Rademacher complexity. In the case where $\mathcal{H}$ is flexible enough to label the x_i in arbitrary ways, the supremum will be equal to $\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[\sigma_i = 1]$, which, when averaged over all values of $\boldsymbol{\sigma}$, equals $1/2$.

The first generalisation bounds based on the Rademacher complexity were due to Bartlett and Mendelson (2001, 2002) and Koltchinskii and Panchenko (2000). We give the following example.

Theorem 4. *(Shalev-Shwartz and Ben-David (2014), Theorem 26.5) For any $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, data-generating distribution $D \in \triangle(\mathcal{X} \times \mathcal{Y})$, loss function $\ell : \mathcal{H} \times (\mathcal{X} \times \mathcal{Y}) \to [0, C]$, confidence level $\delta \in (0, 1]$ and sample size n, with probability at least $1 - \delta$ over the sample $S \sim D^n$, for all $h \in \mathcal{H}$ simultaneously,*

$$R_D(h) - R_S(h) \leq 2\mathcal{R}(\mathcal{H}, \ell, S) + 4C\sqrt{\frac{2\ln(4/\delta)}{n}}, \tag{2.18}$$

where $R_D(h)$ and $R_S(h)$ denote the true and empirical error of the hypothesis h, respectively.

Note that the bound (2.18) depends on S and so is a random variable, in contrast to the constant sample-independent generalisation bound (2.17). Further, since the inequality holds for all $h \in \mathcal{H}$ simultaneously, it holds in particular for $h = \mathcal{A}_{\text{ERM}}(S)$ or any other choice of algorithm.

We can now ask whether this sample-dependent bound can explain the observed generalisation of neural networks. Unfortunately, the answer is again no. This is demonstrated most clearly in Zhang et al. (2016). There they show that network architectures and training procedures commonly used to successfully learn CIFAR10 and ImageNet classification tasks still yield networks with near 100% train accuracy (i.e. $R_S(h) \approx 0$), even when the labels of the sample S are randomised. While it would be computationally infeasible to repeat this experiment for every possible labelling of the sample and thereby measure $\mathcal{R}(\mathcal{H}, \ell, S)$ exactly, the fact that they achieved $R_S(h) \approx 0$ for a random labelling is evidence that the hypothesis class $\mathcal{H}$ implied when employing neural networks is flexible enough to label S arbitrarily. As noted after Definition 6, this would imply that $\mathcal{R}(\mathcal{H}, \ell, S) \approx 1/2$, which would make the bound in Theorem 4 vacuous.

Previously, a folk theory for the effectiveness of regularisation techniques such as weight decay or dropout was that they reduced the effective capacity of the network to overfit. This reduced effective capacity then means that the empirical risk is a better proxy for the true risk across the hypotheses likely to be generated by the training procedure. However, the experiments in Zhang et al. (2016) included such regularisation, demonstrating that typical regularisation does not in fact limit the capacity of the network to overfit. Indeed, further experiments in the paper show that low train error is achieved even when the input images themselves are replaced with Gaussian noise, showing that the capacity is very large indeed. Moreover, the fact that networks trained on true data generalise well even without any regularisation shows that whatever capacity constraint regularisation does provide is unnecessary.

Together, this suggests that the capacity of $\mathcal{H}$, even measured in a sample-dependent way, is far too large to explain generalisation in the overparameterised regime. Clearly, if our hypothesis class $\mathcal{H}$ and sample S are such that we can find two hypotheses $h_1, h_2 \in \mathcal{H}$ that both fit S very well, namely such that $R_S(h_1) \approx 0$ and $R_S(h_2) \approx 0$, yet which have starkly different true errors, say $R_D(h_1) \approx 0$ and $R_D(h_2) \approx 0.5$, then any generalisation bound that is not a function of h will fail to distinguish between them. The bound will then be loose for h_1 as it is forced to also be valid for h_2. Since the existence of such h_1, h_2 is exactly what is demonstrated in Zhang et al. (2016)—training on random labels produces networks with trivial test error—we can conclude that neither Theorem 2 nor Theorem 4 can explain generalisation in the overparameterised regime, and that to do so we must seek hypothesis-dependent bounds. PAC-Bayes generalisation bounds move in this direction, but before we introduce PAC-Bayesian theory, let us take a step back to consider at a high level what it would mean to explain the generalisation mystery.

2.2 What might an explanation of the generalisation mystery look like?

The bounds given by Theorems 2 and 4 are uniform; they work by showing that for sufficiently large sample sizes all hypotheses will have empirical risk $R_S(h)$ that is representative of their true risk $R_D(h)$, and so in particular this will be the case for the learned hypothesis $\mathcal{A}(S)$. As we have seen, uniform bounds will produce trivial results in the overparameterised regime, since in this scenario the capacity of $\mathcal{H}$ is large enough and the sample size n is small enough that there exist hypotheses h for which $R_S(h)$ is not at all representative of $R_D(h)$. The generalisation mystery cannot, therefore, be explained by uniform generalisation bounds.

We are therefore forced to search for non-uniform bounds that are a function of the hypothesis h. While this is necessary in the overparameterised regime, it produces bounds with diminished explanatory power. To see why, suppose our learning algorithm produces a hypothesis h for which $R_S(h)$ is low and the generalisation bound is low, so that we can conclude $R_D(h)$ is low. But by definition of the overparameterised regime, there exists a hypothesis h' for which $R_S(h')$ is also low yet the generalisation bound is high, since the bound must accomodate the large generalisation gap of h'. Thus the question of why our algorithm selected from the set of hypotheses with low empirical risk a hypothesis which also has low generalisation bound, would remain unanswered. Just as the Rademacher complexity based bound in Theorem 4 produces low bounds only if we are “lucky” that the sample S we obtain yields low Rademacher complexity $\mathcal{R}(\mathcal{H}, S)$, non-uniform bounds can be low only if we are “lucky” in the way in which our hypothesis class $\mathcal{H}$, sample S and learning algorithm $\mathcal{A}$ interact to produce $\mathcal{A}(S)$. The cost of non-uniform bounds is therefore a shift in the generalisation mystery to this empirically observed luckiness, sometimes termed the *inductive bias* of the hypothesis class, the learning algorithm, or their interaction. We highlight this conclusion as the following claim.

Claim 1. *Generalisation bounds alone cannot resolve the generalisation mystery, even if they are non-uniform and tight.*

How then can the generalisation mystery be resolved? The preceding discussion suggests a two-pronged strategy. First, while generalisation bounds alone cannot resolve the generalisation mystery, empirically tight bounds formalise notions of luckiness or inductive bias that we can seek to explain. In other words, we may seek a theoretical answer to the remaining question above, namely why commonly used learning algorithms may be biased towards hypotheses for which the bound is low. Second, formal characterisations of observed implicit biases motivate the search for generalisation bounds expressed in terms of such characterisations. A full explanation of the generalisation mystery may then follow the structure suggested in the following claim.

Claim 2. *The generalisation mystery may be explained by the derivation of two theorems. One theorem demonstrating that the learning algorithms typically used in DL have an implicit bias towards hypotheses h with some property P, and a second theorem demonstrating that hypotheses with property P have small generalisation gap.*

More generally, property P may be replaced by a continuous metric, often called a complexity measure of the hypothesis h, not to be confused with complexity measures of the entire hypothesis class $\mathcal{H}$. The theorems would then state that our learning algorithms are biased towards hypotheses with low complexity, and that low complexity hypotheses have small generalisation gap.

Possibilities for properties or metrics P that have been suggested for neural networks include having low parameter norm (e.g. Bartlett et al. (2017) give a proof that networks with small spectral norms have low generalisation gap), being located in a flat minimum of the loss landscape (e.g. Hochreiter and Schmidhuber (1997) argue that flatter minima correspond to simpler hypotheses with lower generalisation gap, and Keskar et al. (2016) investigate this empirically by varying the batch size), having low Kolmogorov complexity (e.g. Schmidhuber (1997) attempts to find neural networks with low Kolmogorov complexity, arguing that they should generalise better), or having low minimum description length (MDL) according to the MDL Principle (first posited in Rissanen (1978), discussed in the context of DL in Grünwald and Roos (2019), and used as a regulariser to improve generalisation in Hinton and Van Camp (1993)).

A complementary approach may be to make assumptions on the data-generating distribution, for example realisability, smoothness, Lipschitz continuity, convexity, noise or margin conditions, the existence of a low dimensional manifold, and so on (Shalev-Shwartz and Ben-David, 2014). While such assumptions can yield tighter bounds, they are often infeasible to verify in practice given we only have access to a sample. Therefore while this approach could potentially resolve the generalisation mystery, it may be at the cost of making unverifiable assumptions.

It is worth noting here an interesting subtlety distinguishing philosophical approaches to statistics. A Bayesian may have as a starting point that, for real-world distributions D, "simpler" hypotheses h are more likely to have low true risk $R_D(h)$, formalised as an Occam prior over $\mathcal{H}$. She therefore does not in fact require a generalisation bound to explain the generalisation mystery; it is sufficient to have a proof that the learning algorithm is more likely to produce hypotheses that both fit the data S *and* have low complexity, and therefore approximate the Bayesian posterior.[2] Conversely, a Frequentist, not willing to take a position on the prior probabilities of hypotheses, does not argue that simpler hypotheses h are more likely to have true risk $R_D(h)$. Instead, she seeks to show that they are more likely to have small generalisation gap $R_D(h) - R_S(h)$. She can also leverage the fact that there are fewer simple hypotheses than complex ones to demonstrate a kind of soft capacity control—a learning algorithm that is more likely to return a simple hypothesis is more likely to output a hypothesis in a smaller class (the class of simple hypotheses) and therefore more likely to return a hypothesis with small generalisation gap. Combining the fact that the empirical risk $R_S(h)$ is low and the generalisation gap $R_D(h) - R_S(h)$ is also low, she can conclude that $R_D(h)$ is low. As we will now see, soft capacity control is one way of interpreting what PAC-Bayesian theory formalises, which, despite the name, is a Frequentist approach.

2.3 PAC-Bayesian generalisation bounds

In Section 2.1.2 we saw how non-uniform generalisation bounds could be derived for a countable hypothesis class $\mathcal{H} = \{h_i, i \in \mathcal{I}\}$ ($\mathcal{I} \subseteq \mathbb{N}$) by using a union bound and a distribution π over $\mathcal{I}$; the generalisation bound for each individual h_i is loosened to hold with the higher probability $1 - \pi(i)\delta$, so that all of the bounds then hold simultaneously with probability $1 - \delta$. Eliding some of the details for clarity, we found that from

$$\forall i \in \mathcal{I} \quad \mathbb{P}_{S\sim D^n}\Big(R_D(h_i) - R_S(h_i) \leq \epsilon_i(n,\delta)\Big) \geq 1-\delta,$$

we could derive

$$\mathbb{P}_{S\sim D^n}\Big(\forall i \in \mathcal{I} \;\; R_D(h_i) - R_S(h_i) \leq \epsilon_i(n,\pi(i)\delta)\Big) \geq 1-\delta.$$

In a near identical manner, if we have one uniform generalisation bound for each of countably many hypothesis *classes* $\{\mathcal{H}_i, i \in \mathcal{I}\}$ ($\mathcal{I} \subseteq \mathbb{N}$) and a distribution π over $\mathcal{I}$, we can combine these into a non-uniform generalisation bound valid for all $h \in \mathcal{H} = \cup_{i\in\mathcal{I}}\mathcal{H}_i$, where the value of the bound for a hypothesis $h \in \mathcal{H}$ depends on the hypothesis class $\mathcal{H}_i$ to which h belongs. Again

[2]The Occam prior can be partially justified by noticing that for any unbounded notion of complexity, not just Kolmogorov complexity, there will be more complex hypotheses than simple ones, and so more complex hypotheses must necessarily receive lower prior mass, at least asymptotically. This argument can be found in the enjoyable read Yudkowsky (2015). This leaves open of course what the right notion of complexity is.

eliding details for clarity, from

$$\forall i \in \mathcal{I} \quad \mathbb{P}_{S\sim D^n}\Big(\forall h \in \mathcal{H}_i \;\; R_D(h) - R_S(h) \leq \epsilon_i(n,\delta)\Big) \geq 1-\delta,$$

we can derive

$$\mathbb{P}_{S\sim D^n}\Big(\forall i \in \mathcal{I} \;\forall h \in \mathcal{H}_i \;\; R_D(h) - R_S(h) \leq \epsilon_i(n,\pi(i)\delta)\Big) \geq 1-\delta.$$

For example, given a fixed neural network architecture, for each $i \in \mathbb{N}$ we could take $\mathcal{H}_i$ to be the hypothesis class corresponding to all ways of setting the parameters w such that $\|w\|_2 \leq i$, and take $\pi(i) = 2^{-i}$. We refer to Shalev-Shwartz and Ben-David (2014) for a deeper discussion of this technique and its relation to non-uniform learnability.

If one chooses as a learning algorithm the function which returns the hypothesis minimising this non-uniform bound, one has an example of Structural Risk Minimisation (SRM), an alternative to Empirical Risk Minimisation (ERM), which balances fit to data with a preference for particular classes $\mathcal{H}_i$, expressed by the distribution π. The concept of SRM is due to Vapnik and Chervonenkis (1974).

The PAC-Bayesian theory does something similar to combining countably many bounds weighted by a prior π, but in a much more elegant way. The strategy can be thought of as constructing a "continuous union bound" as described in Erven (2014), avoiding chopping up the hypothesis class into discrete portions. Doing this however requires generalising to *stochastic* hypotheses, defined as follows.

Definition 7. *(Stochastic hypothesis).* For a hypothesis class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, a *stochastic hypothesis* is a distribution $Q \in \triangle(\mathcal{H})$ which makes predictions according to a sampled $h \sim Q$, with a fresh h sampled for every prediction. For a sample $S \in (\mathcal{X}\times\mathcal{Y})^*$ and data-generating distribution $D \in \triangle(\mathcal{X}\times\mathcal{Y})$, its true and empirical risks are defined to be

$$R_D(Q) = \mathbb{E}_{h\sim Q}[R_D(h)] \quad \text{and} \quad R_S(Q) = \mathbb{E}_{h\sim Q}[R_S(h)],$$

respectively.

Instead of holding with high probability for all deterministic hypotheses $h \in \mathcal{H}$ simultaneously, and therefore in particular for the learned hypothesis $\mathcal{A}_{\text{det}}(S) \in \mathcal{H}$, PAC-Bayes bounds hold with high probability for all stochastic hypotheses $Q \in \triangle(\mathcal{H})$ simultaneously, and therefore in particular for the learned stochastic hypothesis $\mathcal{A}_{\text{stoch}}(S) \in \triangle(\mathcal{H})$, where the algorithm $\mathcal{A}_{\text{stoch}} : (\mathcal{X}\times\mathcal{Y})^* \to \triangle(\mathcal{H})$ returns stochastic hypotheses. PAC-Bayes bounds then control the generalisation gap of a stochastic hypothesis Q, namely $R_D(Q) - R_S(Q)$. The "soft capacity control" is typically formalised as the complexity measure $\mathrm{KL}(Q\|P)$, the Kullback–Leibler divergence of the stochastic hypothesis Q from a fixed (sample-independent) stochastic hypothesis P. While the PAC-Bayesian theory goes back to the pioneering works McAllester (1998) and Shawe-Taylor and Williamson (1997), we illustrate it with the following theorem, originally due to Catoni (2003). The exact form stated here can be found in Alquier et al. (2024).

Theorem 5. *Let $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ be a hypothesis class and $\ell : \mathcal{Y}^2 \to [0, C]$ a loss function. For any fixed stochastic hypothesis $P \in \triangle(\mathcal{H})$, data-generating distribution $D \in \triangle(\mathcal{X} \times \mathcal{Y})$, confidence level $\delta \in (0, 1]$ and $\lambda > 0$, with probability at least $1 - \delta$ over the sample $S \sim D^n$ we have that for all stochastic hypotheses $Q \in \triangle(\mathcal{H})$ simultaneously*

$$R_D(Q) \leq R_S(Q) + \frac{\lambda C^2}{8n} + \frac{\mathrm{KL}(Q\|P) + \ln \frac{1}{\delta}}{\lambda}. \tag{2.19}$$

Since the bound holds with high probability for all $Q \in \triangle(\mathcal{H})$ simultaneously, it holds in particular for a sample-dependent $Q = \mathcal{A}_{\mathrm{stoch}}(S)$. On the other hand, the bound states that $P \in \triangle(\mathcal{H})$ must be fixed. For this reason (and because of the historical origins of PAC-Bayesian theory as a Frequentist analysis of Bayesian learning), the sample-independent P is termed the "prior" and the possibly sample-dependent Q is termed the "posterior". We use quotes as the "prior" P need not have any relation to the practitioner's prior beliefs about which hypotheses are more likely, and the "posterior" Q need not be derived from P and S via Bayes' rule. Nevertheless, since the terminology is commonplace, we henceforth drop the quotes.

A second PAC-Bayes bound, which we frequently employ in this thesis, was originally proved in Langford and Seeger (2001) and Seeger (2002). We give here the somewhat tighter version proved in Maurer (2004). Instead of upper bounding $R_D(Q) - R_S(Q)$ directly, the result constrains the deviation between the true and empirical risks via the so-called "small kl" $\mathrm{kl}(R_S(Q)\|R_D(Q))$, defined by

$$\mathrm{kl}(q\|p) = q \ln \frac{q}{p} + (1-q) \ln \frac{1-q}{1-p}, \quad \text{for} \quad q, p \in [0, 1], \tag{2.20}$$

namely the ordinary KL divergence between two Bernoulli distributions with bias $R_S(Q)$ and $R_D(Q)$.[3] We note that Maurer's version required that $n \geq 8$, while this restriction was later shown to be unnecessary in Germain et al. (2015).

Theorem 6. *(Maurer (2004), Theorem 5) Let $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ be a hypothesis class and $\ell : \mathcal{Y}^2 \to [0, 1]$ a loss function. For any fixed stochastic hypothesis $P \in \triangle(\mathcal{H})$, data-generating distribution $D \in \triangle(\mathcal{X} \times \mathcal{Y})$ and confidence level $\delta \in (0, 1]$, with probability at least $1 - \delta$ over the sample $S \sim D^n$ we have that for all stochastic hypotheses $Q \in \triangle(\mathcal{H})$ simultaneously*

$$\mathrm{kl}\big(R_S(Q)\big\|R_D(Q)\big) \leq \frac{\mathrm{KL}(Q\|P) + \ln \frac{2\sqrt{n}}{\delta}}{n}. \tag{2.21}$$

One practical benefit of this bound over Theorem 5 is that it does not include a scalar λ that must be chosen before seeing the sample. In order to obtain a bound on $R_D(Q)$ one can invert the small kl with respect to its second argument by defining

$$\mathrm{kl}^{-1}(q|B) = \sup\big\{p \in [0, 1] : \mathrm{kl}(q\|p) \leq B\big\}. \tag{2.22}$$

[3]With the usual convention that $0 \ln \frac{0}{p} = 0$ for all $p \in [0, 1]$ and $q \ln \frac{q}{0} = \infty$ for $q > 0$.

Inequality (2.21) then becomes

$$R_D(Q) \leq \mathrm{kl}^{-1}\left(R_S(Q) \middle| \frac{\mathrm{KL}(Q\|P) + \ln\frac{2\sqrt{n}}{\delta}}{n}\right). \tag{2.23}$$

One interesting feature of this bound is that if $R_S(Q)$ is not trivial, namely if $R_S(Q) < 1$, and the right hand side of (2.21) is finite, then the bound (2.23) on $R_D(Q)$ is guaranteed to be non-vacuous, i.e. strictly less than one. This is in sharp contrast to the bound in Theorem 5, which can be arbitrarily large. In practice however, the right hand side of (2.21) is frequently so large that calculating the inverse kl becomes numerically difficult. In such cases, one can alternatively employ Pinsker's inequality, which in this context gives $\mathrm{kl}(q\|p) \leq B \implies p \leq q + \sqrt{B/2}$, which gives a bound on $R_D(Q)$ that is $\mathcal{O}(1/\sqrt{n})$. It should be noted however that this bound is always looser than (2.23), and can be significantly looser when $R_S(Q) \approx 0$.

Both Theorem 5 and Theorem 6 exhibit a tradeoff between low empirical risk $R_S(Q)$ and low complexity, where the complexity of Q is formalised as $\mathrm{KL}(Q\|P)$. This can be seen as a kind of soft capacity control, where if our algorithm $\mathcal{A}_{\text{stoch}}$ returns a stochastic hypothesis Q close in KL divergence to the fixed P then, since this is a constrained set, we can conclude that the generalisation gap is likely not that large. Now, while this soft capacity control will be present for *any* fixed prior P, poor choices of P will reduce the chance that there exists a posterior Q for which both $R_S(Q)$ and $\mathrm{KL}(Q\|P)$ are low. Thus there is an element of luck in the PAC-Bayes framework in terms of the choice of prior, and this luck constitutes a gap in the ability of the framework to explain the generalisation mystery. This is the same tradeoff as was observed in the countable case presented in Section 2.1.2. Some theoretically motivated choices of prior do however reduce this element of luck, as we will see in Section 2.3.2.

2.3.1 Choosing the posterior

What is a good choice for $Q(S)$ in Theorem 5? A first thought may be to choose $Q(S)$ to minimise the empirical risk, namely

$$Q(S) := \mathrm{argmin}_{Q\in\triangle(\mathcal{H})} R_S(Q).$$

In most cases however this will yield a very poor generalisation bound. To see why, note that $Q(S)$ reduces to a point mass on $h(S) = \mathrm{argmin}_{h\in\mathcal{H}} R_S(h)$, resulting in $\mathrm{KL}(Q(S)\|P) = -\ln \mathbb{P}_{h\sim P}(h = h(S))$, which will be infinite except under special conditions, since P must be chosen independently of S.

A more sensible choice is to take $Q(S)$ to be the minimiser of the bound—a perfectly legitimate choice since all the quantities in the bound are fixed in the theorem statement before the quantifier $\forall Q \in \triangle(\mathcal{H})$. Remarkably, using Donsker and Varadhan's variational formula (Donsker and Varadhan, 1976), it can be shown that this optimal posterior is the so-called

Gibbs posterior $Q_G(S)$, defined by the Radon–Nikodym derivative

$$\frac{\mathrm{d}Q_G(S)}{\mathrm{d}P}(h) = \frac{e^{-\lambda R_S(h)}}{\mathbb{E}_{h'\sim P}\big[e^{-\lambda R_S(h')}\big]}. \tag{2.24}$$

For continuous or discrete P with probability density or mass function $f_P : \mathcal{H} \to \mathbb{R}$, respectively, defining the Gibbs posterior $Q_G(S)$ via the Radon–Nikodym derivative (2.24) above is equivalent to defining $Q_G(S)$ to have probability density or mass function $f_{Q_G(S)} : \mathcal{H} \to \mathbb{R}$, respectively, given by $f_{Q_G(S)}(h) \propto e^{-\lambda R_S(h)} f_P(h)$. This formulation clarifies the intuition that $Q_G(S)$ exponentially re-weights the density P places on h according to the empirical loss $R_S(h)$. As $\lambda \to 0$ the Gibbs posterior $Q_G(S)$ becomes simply the prior P, which makes sense since the $\mathrm{KL}(Q\|P)$ term in (2.19) then dominates. Conversely, as $\lambda \to \infty$ the posterior tends to a point mass on h_{ERM}, since in this scenario minimising (2.19) reduces to minimising $R_S(Q)$. Thus, using the Gibbs posterior, the constant λ in Theorem 5 acts as an inverse temperature.

Unfortunately, the Gibbs posterior is highly impractical in the context of DL, as the normalisation constant in the denominator of (2.24) typically cannot be calculated, meaning one is unable to sample from $Q_G(S)$ in order to make predictions. One may approximate samples using MCMC, or approximate the Gibbs posterior directly using Variational Bayes (see Alquier et al. (2016) for a theoretical analysis of the second method).

An alternative to the Gibbs posterior is to directly minimise the bound (2.19) over the set of Gaussian posteriors. More precisely, if our hypothesis class is $\mathcal{H} = \{h_w : w \in \mathbb{R}^d\}$, where $h_w : \mathcal{X} \to \mathcal{Y}$ is the function corresponding to a neural network with weights w, we may take $Q \in \triangle(\mathcal{H})$ to be the pushforward of a Gaussian distribution on the weight space $\mathbb{R}^d$. If the prior P is also taken to be a (sample-independent) Gaussian distribution on the weight space, the term $\mathrm{KL}(Q\|P)$ has a differentiable closed form.[4] The entire bound (2.19) can then be optimised via gradient descent with the aid of the pathwise gradient trick to approximate derivatives for $R_S(Q)$. This was the strategy employed by Dziugaite and Roy (2017) (optimising the bound in Theorem 6 above) along with some other tricks to achieve the first non-vacuous generalisation bound for a deep neural network trained on the MNIST dataset. We outline their process in much greater detail in Chapter 3 where we use it to optimise our PAC-Bayes bound. This method can be flexibly applied to various PAC-Bayes bounds, whereas the Gibbs posterior is the minimiser only for "linear" PAC-Bayes bounds of the form $R_D(Q) \leq aR_S(Q) + b\mathrm{KL}(Q\|P) + c$, such as Theorem 5 but not Theorem 6.

A third choice is to take Q to be the pushforward of a Gaussian distribution centred around the weights $w(S)$ returned by an ordinary DL algorithm. Whether this is a better choice than (approximately) minimising the bound depends on our reason for using the PAC-Bayes bound. If our goal is to find a stochastic hypothesis Q which we can use for prediction and be confident in, the first choice may be more appropriate as it returns the Q with lowest bound on the true risk $R_D(Q)$. On the other hand, if our goal is to explain the empirical success of typical DL

[4]Strictly speaking, if Q and P are the pushforward of Gaussian distributions Q_w and P_w on the weight space, then $\mathrm{KL}(Q\|P)$ can be replaced by the differentiable proxy $\mathrm{KL}(Q_w\|P_w)$, which, by the data-processing inequality (see e.g. Van Erven and Harremos (2014)), is an upper bound on $\mathrm{KL}(Q\|P)$.

algorithms in the overparameterised regime, the second choice is much more appealing as the only modification to the training procedure is the addition of some (possibly small) Gaussian noise to the network weights (with fresh noise sampled for each prediction). We may then ultimately hope that the effect of this modification can be rigorously analysed, so that the performance of the original deterministic network can be bounded.

2.3.2 Choosing the prior

We give a brief overview of three choices for the prior found in the literature; the theoretically optimal prior, the distribution-dependent "localised" prior, and data-dependent priors.

First, consider the theoretically optimal prior for Theorem 5. Suppose we have made our choice of data-dependent posterior $Q(S)$. We may then ask what choice of prior P would minimise Theorem 5 in expectation over the random draw $S \sim D^n$, which is equivalent to minimising $\mathbb{E}_{S\sim D^n}[\mathrm{KL}(Q(S)\|P)]$. This is analogous to the strategy suggested in Langford and Blum (2003) for countable hypothesis classes discussed in Section 2.1.2, and indeed the solution is also analogous; the optimal choice is

$$P^* = \mathbb{E}_{S\sim D^n}[Q(S)], \quad \text{namely} \quad P^*(B) = \mathbb{E}_{S\sim D^n}[Q(S)(B)] \;\; \text{for measurable } B \in \mathcal{H}.$$

This follows from the so-called *golden formula* (Polyanskiy and Wu, 2014), also known as Tropsøe's identidy (Topsøe, 1967). A derivation can be found in Lever et al. (2013), and the result was already noted in Catoni (2007). Again, as in the countable case, this prior is a legitimate choice (it is sample independent) but it not practically useful as it depends on the unknown data-generating distribution D. Nevertheless, it shows that a good choice of prior is one that "anticipates" where $Q(S)$ is likely to put mass. Further, note that this so-called "oracle prior" P^* is only optimal for PAC-Bayes bounds where minimising them in expectation amounts to minimising the KL divergence in expectation, which is the case for Theorem 5 but not Theorem 6.

Remarkably, some choices of *distribution*-dependent prior can nevertheless yield practically useful empirical bounds if the KL divergence can be upper bounded by known quantities. For example, given a distribution $\pi \in \mathcal{H}$, and $\beta > 0$, Catoni (2003) suggests the *localised* prior $P = \pi_{-\beta R_D}$ defined by the Radon–Nikodym derivative

$$\frac{\mathrm{d}\pi_{-\beta R_D}}{\mathrm{d}\pi}(h) = \frac{e^{-\beta R_D(h)}}{\mathbb{E}_{h'\sim\pi}[e^{-\beta R_D(h')}]},$$

the intuition being that we should put more weight on hypotheses h with low true risk $R_D(h)$. Defining the Bernstein function $g : \mathbb{R} \to \mathbb{R}$

$$g(x) = \begin{cases} \frac{e^x - 1 - x}{x^2}, & x \neq 0, \\ 0, & x = 0, \end{cases}$$

the following theorem is proven in Catoni (2003). Note that the localisation to $\pi_{-\beta R_D(h)}$ is done internally so does not appear explicitly in the theorem.

Theorem 7. *(Catoni (2003), Lemma 6.2) Let $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ be a hypothesis class and $\ell : \mathcal{Y}^2 \to [0,1]$ a loss function. For any fixed distribution $\pi \in \triangle(\mathcal{H})$ (note this does not play the role of the PAC-Bayes prior), data-generating distribution $D \in \triangle(\mathcal{X} \times \mathcal{Y})$, sample size n, confidence level $\delta \in (0,1]$, $\lambda > 0$ and $\xi \in [0,1)$ such that $(1-\xi) - (1+\xi)g(\frac{\lambda}{n})\frac{\lambda}{n} > 0$, with probability at least $1-\delta$ over the sample $S \sim D^n$ we have that for all stochastic hypotheses $Q \in \triangle(\mathcal{H})$ simultaneously*

$$R_D(Q) \leq \frac{\mathrm{KL}(Q\|\pi_{-\xi\lambda R_S}) + (1-\xi)\lambda R_S(Q) + (1+\xi)\ln\frac{2}{\delta}}{(1-\xi)\lambda - (1+\xi)g(\frac{\lambda}{n})\frac{\lambda^2}{n}},$$

where the distribution $\pi_{-\xi\lambda R_S}$ is defined by the Radon–Nikodym derivative

$$\frac{d\pi_{-\xi\lambda R_S}}{d\pi}(h) = \frac{e^{-\xi\lambda R_S(h)}}{\mathbb{E}_{h'\sim\pi}[e^{-\xi\lambda R_S(h')}]}.$$

Other nice theorems demonstrating the feasibility of distribution-dependent priors and applications to Support Vector Machines (SVMs) can be found in Lever et al. (2013).

Finally, we discuss data-dependent priors. While the prior is not permitted to depend on the sample S appearing in the PAC-Bayes bound, it is free to depend on a second sample. This suggests splitting the sample S, using the first half to choose a prior and the second half for the bound, a strategy first proposed in Seeger (2002) and followed by many others (Ambroladze et al., 2006; Clerico et al., 2022a; Parrado-Hernández et al., 2012; Perez-Ortiz et al., 2021; Pérez-Ortiz et al., 2021). For clarity, we restate Theorem 6 in this context.

Theorem 8. *Let $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ be a hypothesis class and $\ell : \mathcal{Y}^2 \to [0,1]$ a loss function. For any data-dependent prior and posterior maps $P : (\mathcal{X} \times \mathcal{Y})^m \to \triangle(\mathcal{H})$ and $Q : (\mathcal{X} \times \mathcal{Y})^n \to \triangle(\mathcal{H})$, data-generating distribution $D \in \triangle(\mathcal{X} \times \mathcal{Y})$, and confidence level $\delta \in (0,1]$, with probability at least $1-\delta$ over the sample $S = (S_1, S_2) \sim D^m \otimes D^{n-m}$ we have that*

$$\mathrm{kl}\big(R_S(Q(S))\big\|R_D(Q(S))\big) \leq \frac{\mathrm{KL}\big(Q(S)\big\|P(S_1)\big) + \ln\frac{2\sqrt{n-m}}{\delta}}{n-m}. \tag{2.25}$$

Importantly, note that while the prior may only depend on S_1, the posterior can depend on the entire sample $S = (S_1, S_2)$, since Theorem 6 holds with high probability for all posteriors simultaneously. Further, since S_2 plays the role of S in Theorem 6, the bound is evaluated with sample size $|S_2| = n - m$. Therefore, when choosing the size $|S_1| = m$ of the sample P is permitted to depend on, we face a clear tradeoff; a larger S_1 will allow a more informed prior that will hopefully better predict $Q(S)$ and lead to a reduced KL divergence term, but it will also decrease the denominator of the bound. The optimal amount of data to use for the prior depends on details of the learning setup, as can be seen from the experiments in Perez-Ortiz et al. (2021) (see Table 3) and Pérez-Ortiz et al. (2021) (see Table 5).

In the context of neural networks, the map $P : (\mathcal{X} \times \mathcal{Y})^m \to \triangle(\mathcal{H})$ can be taken to be the pushforward of a Gaussian distribution centred around the weights $w(S_1)$ returned by an ordinary DL algorithm, a strategy that has been mentioned already in Section 2.3.1 for choosing the posterior. Indeed, this strategy for choosing the prior combined with the bound minimisation strategy for choosing the posterior (mentioned in Section 2.3.1) was used in Pérez-Ortiz et al. (2021) to achieve remarkably tight bounds for deep CNNs (up to 15 layers) on the CIFAR10 dataset.

2.4 PAC-Bayes and the generalisation mystery

As already briefly mentioned in Section 2.3.2, the PAC-Bayes bound Theorem 6 was employed with Gaussian prior and posterior in Dziugaite and Roy (2017) to obtain the first non-vacuous generalisation bound for overparameterised neural networks. For example, working with a binarised version of the MNIST dataset, they obtain an error bound of $R_D(Q) \leq 0.201$, where Q is an isotropic Gaussian over the weights of an MLP with three hidden layers each of dimension 600, trained to minimise a proxy of the bound given in Theorem 6. The prior is also taken to be an isotropic Gaussian over the weights, centred at the initialisation used for training Q. This architecture is in the overparameterised regime—it has around 1 193 000 parameters, much greater than the sample size of 60 000, and they show that the same architecture with only one hidden dimension can achieve $R_S(h) = 0.007$.

This result was a significant step in demonstrating the value of PAC-Bayes theory, but it does not constitute an explanation of the generalisation mystery for three reasons:

1. It applies only to stochastic networks, rather than the deterministic networks typically used in DL.
2. The stochastic networks are trained in a non-standard way—by minimising the PAC-Bayes bound.
3. There is an element of luck in whether the prior P is chosen such that there exists a posterior Q with both low empirical risk $R_S(Q)$ and for which the bound is tight.

As for the first obstacle, several solutions have been proposed and analysed in the literature to *derandomise* stochastic hypotheses into deterministic hypotheses in such a way that preserves or minimally loosens the PAC-Bayes bound. It should be noted that while demonstrating non-vacuous bounds for such deterministic hypotheses in the overparameterised regime is certainly a step forward, the problem remains that these are not the deterministic hypotheses returned by typical DL algorithms (with the exception of Clerico et al. (2022b)), meaning the explanatory power of such bounds remains limited.

2.4.1 Derandomisation of PAC-Bayes bounds

Given a stochastic hypothesis $Q \in \triangle(\mathcal{H})$, we may derive the following deterministic predictors:

1. $h \sim Q$, a single sample from Q,

2. $f_Q(x) \coloneqq \mathbb{E}_{h\sim Q}[h(x)]$, the Q-weighted majority vote,

3. f_{w^*}, where $w^* = \mathbb{E}_{w\sim Q}[w]$, the "mean" hypothesis applicable when $\mathcal{H} = \{h_w : w \in \mathcal{W}\}$.

Note f_Q may not be an element of $\mathcal{H}$, so we refer to it as a *predictor* rather than a hypothesis. Further it must be the case that $f_Q(x) \in \mathcal{Y}$, which is true if $\mathcal{Y}$ is convex, for example. If this is not the case, one may project back on to $\mathcal{Y}$; in the case of binary classification with $\mathcal{Y} = \{-1, 1\}$, one may take

2'. $f_Q(x) \coloneqq \operatorname{sign}\big(\mathbb{E}_{h\sim Q}[h(x)]\big)$,

as in Lacasse et al. (2006) and Langford and Shawe-Taylor (2002), and in the more general case of multiclass classification, one may take

2". $f_Q(x) \coloneqq \operatorname{argmax}_{y\in\mathcal{Y}} \mathbb{P}_{h\sim Q}\big(h(x) = y\big)$,

as in Biggs et al. (2022). The mean hypothesis h_{w^*} is applicable when $\mathcal{H}$ is parameterised by w, for example if $\mathcal{H} = \{h_w : w \in \mathbb{R}^d\}$, where $h_w : \mathcal{X} \to \mathcal{Y}$ is the function corresponding to a neural network with weights w. We will now give examples of such derandomisation schemes in the PAC-Bayes literature.

2.4.2 PAC-Bayes bounds on a sample from the posterior

PAC-Bayes bounds generally take the form

$$\mathbb{P}_{S\sim D^n}\left(\forall Q \in \triangle(\mathcal{H}) \quad R_D(Q) \le \epsilon\Big(\mathrm{KL}(Q\|P), R_S(Q), n, \delta\Big)\right) \ge 1 - \delta, \tag{2.26}$$

namely a bound on $R_D(Q)$ that holds with high probability over the sample $S \sim D^n$ for all Q simultaneously, where the bound ϵ is typically a function of $\mathrm{KL}(Q\|P)$, the empirical risk $R_S(Q)$, the sample size n, and the confidence level δ. But many also have a so-called *disintegrated* form

$$\forall Q : (\mathcal{X}\times\mathcal{Y})^* \to \triangle(\mathcal{H}) \quad \mathbb{P}_{h\sim Q(S),\ S\sim D^n}\left(R_D(h) \le \epsilon\left(\ln\left(\frac{\mathrm{d}Q(S)}{\mathrm{d}P}(h)\right), R_S(h), n, \delta\right)\right) \ge 1 - \delta, \tag{2.27}$$

which says that for any data-dependent posterior Q, with high probability over the sample $S \sim D^n$ and then $h \sim Q(S)$ we get the same bound on $R_D(h)$ as in (2.26) with the substitution

$$\mathrm{KL}\big(Q(S)\big\|P\big) = \mathbb{E}_{h\sim Q(S)}\left[\ln\left(\frac{\mathrm{d}Q(S)}{\mathrm{d}P}(h)\right)\right] \quad \mapsto \quad \ln\left(\frac{\mathrm{d}Q(S)}{\mathrm{d}P}(h)\right).$$

These so-called *disintegrated* PAC-Bayes bounds were first proposed in Catoni (2007) (Theorem 1.2.7) and Blanchard and Fleuret (2007).

While sampling from Q may at first sight appear to be a bad idea—it increases the "riskiness" of the bound by removing the smoothing over $h \sim Q$—derandomisation brings us closer to common DL practice, and, with a careful interpretation, the sampled $h \sim Q$ can in fact be made to correspond exactly to the deterministic hypothesis h returned by some ordinary DL algorithms such as Gradient Descent (GD). The trick, developed in Clerico et al. (2022b), is to note that with GD the neural network is initialised to h_{w_0}, where $w_0 \sim \pi$ is sampled from some weight initialisation distribution π. Fixing a sample S and other training details such as the learning rate and number of epochs, π then induces a distribution over the final trained network h_w. Taking this induced distribution as the PAC-Bayes posterior Q, sampling $w_0 \sim \pi$ and training h_w through GD corresponds exactly to sampling from Q, allowing disintegrated PAC-Bayes bounds to apply. The difficulty of course is in estimating $\frac{\mathrm{d}Q}{\mathrm{d}P}(h_w)$ for this implicitly defined posterior. In Clerico et al. (2022b) they consider the ideal situation of continuous training dynamics which allows some clever gradient flow accounting but unfortunately brings us away from common DL practice.

A general disintegration framework is derived in Viallard et al. (2024) by generalising from the KL divergence to the Rényi divergence. Their bounds are highly practical and can be used as training objectives. Another relevant work is Banerjee et al. (2020) which derives derandomised PAC-Bayes bounds for MLPs with ReLU activations.

2.4.3 PAC-Bayes bounds for majority votes

The Q-weighted majority vote predictor f_Q has received attention in the PAC-Bayes literature, perhaps because it is well-known that majority votes can improve performance if the errors of the individual predictors are uncorrelated. In the context of binary classification, a very simple "folk theorem" in Langford and Shawe-Taylor (2002) (Lemma 4.1) gives that $R_D(f_Q) \leq 2R_D(Q)$ for any stochastic hypothesis Q, showing that doubling any bound on $R_D(Q)$ (such as Theorem 6) produces a bound on $R_D(f_Q)$. They then show that the factor 2 can be reduced to $1 + \epsilon$ for large margin classifiers.

It is noted in Lacasse et al. (2006) that this is unsatisfying since in practice f_Q usually has *lower* error than Q, so it would ideally enjoy a smaller rather than a larger bound. By incorporating a bound on the *variance* of the zero-one error of Q over $(x, y) \sim D$, namely a bound on $\mathbb{V}_{(x,y)\sim D}\mathbb{E}_{h\sim Q}\mathbb{1}[h(x) \neq y]$, they derive a PAC-Bayes error bound for the majority vote classifier f_Q that can indeed be much smaller than the error of Q.

It is noted in Letarte et al. (2019) that if the loss $\ell : \mathcal{Y}^2 \to \mathbb{R}$ is linear, then $R_D(f_Q) = R_D(Q)$, so that PAC-Bayes bounds on Q apply directly to f_Q without modification. While commonly used loss functions are not linear, they note that the zero-one loss $\ell_{01}(\hat{y}, y) = \mathbb{1}[\mathrm{sign}(\hat{y}) \neq y]$ in particular can be upper bounded by $\ell_{\mathrm{lin}}(\hat{y}, y) \coloneqq 1 - \hat{y}y$ in the case where $y \in \{-1, 1\}$ and the predictor returns $\hat{y} \in [-1, 1]$. Applying PAC-Bayes bounds with ℓ_{lin} they obtain bounds on $R_D(Q)$ and hence $R_D(f_Q)$. Short of being linear, ℓ may still be convex in which case Jensen's inequality gives $R_D(f_Q) \leq R_D(Q)$, again enabling PAC-Bayes bounds on $R_D(Q)$ to be carried over to $R_D(f_Q)$, as discussed in Section 2.2 of Alquier et al. (2024) along with other properties of the loss function that allow transfer of bounds to the majority vote predictor.

2.4.4 PAC-Bayes bounds for the mean of the posterior

Since in practice the posterior is frequently chosen to be a Gaussian centred around the weights learned by some ordinary DL algorithm, derandomisation to the "mean" hypothesis h_{w^*} is the ideal choice since it reduces to bounding the output of the ordinary DL algorithm, the performance of which we ultimately want to explain. This is very difficult however due to the highly complex dependence of $h_w(x)$ on w. Indeed, the two terms

$$R_D(Q) = \mathbb{E}_{w\sim Q}\big[R_D(h_w)\big], \quad R_D\big(h_{w^*}\big) = R_D\big(h_{\mathbb{E}_{w\sim Q}[w]}\big)$$

may be very different. A remarkable result is proved in Banerjee et al. (2020) for MLPs with ReLU activations, showing that the expected error $R_D(Q)$ of a stochastic network with Gaussian Q centred at w^* can be upper bounded in terms of the expected margin loss[5] $R_D^\gamma(h_{w^*})$ of the mean network h_{w^*}, the ordinary $\mathrm{KL}(Q\|P)$ term, a curvature term involving the diagonal of the Hessian of the loss landscape at w^*, and some other terms. They empirically show that the curvature terms are very small.

A second work to successfully apply this form of derandomisation is Biggs et al. (2022), which, in the context of multiclass classification with a finite hypothesis class $\mathcal{H} = \{h_1, \dots, h_K\}$, considers the task of learning a good majority vote classifier $f_{Q_{\boldsymbol{\theta}}}(x) := \operatorname{argmax}_{y\in\mathcal{Y}}\mathbb{P}_{h\sim Q_{\boldsymbol{\theta}}}\big(h(x) = y\big)$ for some $Q_{\boldsymbol{\theta}} \in \triangle(\mathcal{H})$, a categorical distribution over $\mathcal{H}$ with parameter $\boldsymbol{\theta} \in \triangle_K$. Interestingly, they do not bound $R_D(f_{Q_{\boldsymbol{\theta}}})$ by first bounding $R_D(Q_{\boldsymbol{\theta}})$ and then derandomising to the majority vote as in Section 2.4.1 item (2"). Instead, they lift to the hypothesis class of majority vote classifiers $\mathcal{H}' := \{Q_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \triangle_K\}$ and apply a PAC-Bayes bound to *stochastic* majority vote classifiers represented by Dirichlet distributions $\mathfrak{Q}_{\boldsymbol{\alpha}} = \mathrm{Dir}(\boldsymbol{\alpha}) \in \triangle(\triangle_K)$ with parameter $\boldsymbol{\alpha} \in (0,\infty)^K$. Sampling $\boldsymbol{\theta} \sim \mathfrak{Q}_{\boldsymbol{\alpha}}$ then gives a majority vote classifier $Q_{\boldsymbol{\theta}}$. By choosing $\boldsymbol{\alpha} = \lambda\boldsymbol{\theta}$ for some $\lambda > 0$ so that $\mathbb{E}_{\boldsymbol{\theta}'\sim\mathfrak{Q}_{\lambda\boldsymbol{\theta}}}[\boldsymbol{\theta}'] = \boldsymbol{\theta}$, they derandomise to the mean classifier $Q_{\boldsymbol{\theta}}$ as in Section 2.4.1 item (3), achieving remarkably tight bounds. Bypassing any need for doubling of the bound often encountered in PAC-Bayes bounds for majority votes, they incur only a very small penalty for derandomisation.

2.5 PAC-Bayes and self-certified learning

A second goal of the PAC-Bayesian theory is to generate self-bounding algorithms (Freund, 1998), namely algorithms that return both a learned hypothesis and a risk bound on that learned hypothesis, also called self-certified learning. This is distinct from explaining generalisation since, as argued in Section 2.2, tight bounds are not sufficient to explain generalisation. The alternative to self-certified learning is test set bounds, examples of which we give shortly. These require withholding some of the data from the training procedure, typically leading to a worse learned hypothesis. In contrast, the promise of self-certified learning is to be able to use all of

[5]The margin loss is defined in classification settings with $\mathcal{Y} = [m]$ and $f(x) \in \mathbb{R}^m$ by $\ell_\gamma(f(x), y) := f(x)_y - \max_{j\neq y} f(x)_j$.

the data for training while still obtaining a risk bound on the learned hypothesis, a so-called risk certificate. This is most important in the low-data regime, where withholding data from the training procedure is especially costly. As we will see however, test set bounds are very tight and prove a difficult baseline to beat.

2.5.1 Test set bounds

Test set bounds withhold a so-called test set $S_{\text{test}} \subseteq S$ from the training procedure in order to evaluate the learned hypothesis in an unbiased way by using the Chebyshev (2.7) or Hoeffding (2.11) concentration inequalities discussed earlier, replacing S with S_{test} and h with the learned $\mathcal{A}(S)$. More precisely, given a sample $S \in (\mathcal{X} \times \mathcal{Y})^n$, we take $S_{\text{train}} = ((x_1, y_1), \dots, (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$ and $S_{\text{test}} = ((x_{m+1}, y_{m+1}, \dots, (x_n, y_n)) \in (\mathcal{X} \times \mathcal{Y})^{n-m}$, use S_{train} to learn a hypothesis $\mathcal{A}(S_{\text{train}}) \in \mathcal{H}$ and S_{test} to evaluate it using a test set bound. This then gives

$$\mathbb{P}_{S_{\text{test}} \sim D^{n-m}} \left(R_D\big(\mathcal{A}(S_{\text{train}})\big) \leq R_{S_{\text{test}}}\big(\mathcal{A}(S_{\text{train}})\big) + \frac{C}{\sqrt{(n-m)\delta}} \right) \geq 1 - \delta, \tag{2.28}$$

or

$$\mathbb{P}_{S_{\text{test}} \sim D^{n-m}} \left(R_D\big(\mathcal{A}(S_{\text{train}})\big) \leq R_{S_{\text{test}}}\big(\mathcal{A}(S_{\text{train}})\big) + C\sqrt{\frac{1}{2(n-m)} \ln \frac{2}{\delta}} \right) \geq 1 - \delta, \tag{2.29}$$

for the Chebyshev and Hoeffding test set bounds, respectively, where recall C was the bound on the loss function.

In the special case of the zero-one loss, $|S_{\text{test}}| R_{S_{\text{test}}}\big(\mathcal{A}(S_{\text{train}})\big)$ is a binomial random variable with $|S_{\text{test}}|$ trials and bias $R_D\big(\mathcal{A}(S_{\text{train}})\big)$, meaning various confidence intervals for the bias of a binomial distribution can be applied. The ideal choice, due to its exact coverage, is the binomial tail test set bound based on the Clopper–Pearson confidence interval (Clopper and Pearson, 1934), given as Theorem 3.3 in Langford and Schapire (2005).

Theorem 9. *(Langford and Schapire (2005), Theorem 3.3)*

$$\mathbb{P}_{S_{\text{test}} \sim D^{n-m}} \left(R_D\big(\mathcal{A}(S_{\text{train}})\big) \leq \overline{\text{Bin}}\Big(n-m, R_{S_{\text{test}}}\big(\mathcal{A}(S_{\text{train}})\big), \delta\Big) \right) \geq 1 - \delta, \tag{2.30}$$

where

$$\overline{\text{Bin}}(m,k,\delta) := \max\Big\{p \in [0,1] : \text{Bin}(m,k,p) \geq \delta\Big\}, \quad \mathit{Bin}(m,k,p) := \sum_{j=0}^{k} \binom{m}{j} p^j (1-p)^{m-j}. \tag{2.31}$$

In words, the binomial test set bound is the largest $p \in [0,1]$ such that, with probability at least δ, a binomial random variable with $|S_{\text{test}}|$ trials and bias p has number of successes at most the number of errors of the learned hypothesis $\mathcal{A}(S_{\text{train}})$ on the test set S_{test}. Calculating the function $\overline{\text{Bin}}(m,k,\delta)$ is straightforward using bisection on the interval $[0,1]$.

While the binomial tail test set bound is the ideal choice due to its exact coverage (which comes from the exact coverage of the Clopper–Pearson confidence interval), it can be difficult to calculate if $|S_{\text{test}}| = n - m$ is large and $R_{S_{\text{test}}}(\mathcal{A}(S_{\text{train}})) \not\approx 0$, since the binomial coefficients become large. As discussed in Langford and Schapire (2005), in the special case where $R_{S_{\text{test}}}(\mathcal{A}(S_{\text{train}})) = 0$, the approximation $(1-p)^{n-m} \leq e^{-(n-m)p}$ gives the following closed form

$$\mathbb{P}_{S_{\text{test}} \sim D^{n-m}} \left(R_{S_{\text{train}}}(\mathcal{A}(S_{\text{train}})) = 0 \implies R_D\big(\mathcal{A}(S_{\text{train}})\big) \leq \frac{\ln \frac{1}{\delta}}{n-m} \right) \geq 1 - \delta, \tag{2.32}$$

which is numerically much more manageable.

2.5.2 Can PAC-Bayes achieve self-certified learning?

The important feature to note about the test set bound given by (2.32) is that it is $O(1/|S_{\text{test}}|)$, meaning only a small amount of data needs to be withheld from the training procedure in order to obtain a tight bound (provided we get zero training error). This sets a very high bar for self-certified learning in the overparameterised regime where datasets are typically large, since it is unlikely that withholding the very small proportion of the sample required for the test set bound will have much of a deleterious effect on the training procedure. Another point in favour of test set bounds is that since the posterior Q in PAC-Bayes is typically trained by minimising a PAC-Bayes bound (rather than optimising for performance), even if the PAC-Bayes bound is tight the stochastic classifier Q may be poor.

Indeed, the generalisation bounds achieved in Perez-Ortiz et al. (2021), although remarkably tight, are usually worse than simple test set bounds on the mean of the prior, as can be seen in Table 2.1, which we have adapted from Perez-Ortiz et al. (2021) to include test set bounds calculated on the mean of the prior, which, in their experiments is a deterministic network trained through ordinary DL methods. We observe that in most cases these test set bounds are lower than the corresponding PAC-Bayes bounds. Further, the mean of the prior often performs better than the posterior (as measured on a test set), meaning the PAC-Bayes procedure not only produced a worse bound but a worse classifier! We are not the first to note this; the observation is also made in Lotfi et al. (2022).

It may also be noted that they report very small values for $\mathrm{KL}(Q \| P)$, with the average value across the 12 experiments in Table 2.1 being 0.4903. Since the MLPs they train have $\sim 10^5$ parameters, this implies only a very small shift in the mean weights of the posterior and prior during the PAC-Bayes training, suggesting that the PAC-Bayes bounds are in effect test set bounds in disguise.

These results were calculated with the PAC-Bayes generalisation bound in Theorem 6. Given that there are many PAC-Bayes bounds, it may be the case that others offer more promise for self-certified learning. To that end, Foong et al. (2021) investigate a very general unifying PAC-Bayes theorem proven in Germain et al. (2009), for which many PAC-Bayes bounds are special cases. We give the later form of the bound found in Bégin et al. (2016) (substituting their constant $m' > 0$ with the common choice $m' = n$), which is slightly looser but written

		Prior mean (deterministic)		Posterior (stochastic)	
Dataset	Val.	Test error	Test set bound	Test error	PAC-Bayes bound
Spambase ($n_{\text{cert}} = 1840$)	×	0.077	**0.088**	0.082	0.140
	✓	0.056	**0.066**	0.065	0.127
Bioresponse ($n_{\text{cert}} = 1500$)	×	0.261	**0.281**	0.267	0.318
	✓	0.248	**0.267**	0.257	0.291
Har ($n_{\text{cert}} = 4119$)	×	0.024	**0.028**	0.021	0.035
	✓	0.020	**0.024**	0.024	0.037
Electricity ($n_{\text{cert}} = 18124$)	×	0.221	0.226	0.214	**0.223**
	✓	0.205	**0.210**	0.212	0.221
Mammography ($n_{\text{cert}} = 4473$)	×	0.015	**0.019**	0.015	0.022
	✓	0.017	**0.021**	0.017	0.023
MNIST ($n_{\text{cert}} = 30000$)	×	0.025	**0.027**	0.026	0.034
	✓	0.028	**0.030**	0.027	**0.030**

Table 2.1: Copy of Table 2 from Perez-Ortiz et al. (2021) with added test set bounds (calculated according to (2.30)) on the mean of the prior distribution, which is a classifier learned by ordinary SGD on the training set. The Val. column indicates whether a small subset of the data was used to determine when to stop training the prior. While the PAC-Bayes bounds are tight (the final two columns are very close) we see that better risk certificates are obtained in most cases by taking the test set bound on the mean of the prior (compare Test set bound with PAC-Bayes bound), meaning PAC-Bayes does not enable self-certified learning in most cases. Further, the extra data used to learn the posterior results in a worse classifier (compare two Test error columns), meaning this extra data should be considered wasted even if one is not concerned with risk certificates.

solely in terms of known quantities.

Theorem 10. *(Bégin et al. (2016), Theorem 4) Let $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ be a hypothesis class with $\mathcal{Y} = \{-1, 1\}$, $\ell : \mathcal{Y}^2 \to \{0, 1\}$ be the zero-one loss function $\ell(\hat{y}, y) = \mathbb{1}[\hat{y} \neq y]$ and $d : [0, 1]^2 \to \mathbb{R}$ be jointly convex. For any fixed stochastic hypothesis $P \in \triangle(\mathcal{H})$, data-generating distribution $D \in \triangle(\mathcal{X} \times \mathcal{Y})$ and confidence level $\delta \in (0, 1]$, with probability at least $1 - \delta$ over the sample $S \sim D^n$ we have that for all stochastic hypotheses $Q \in \triangle(\mathcal{H})$ simultaneously*

$$d\big(R_S(Q), R_D(Q)\big) \leq \frac{\mathrm{KL}(Q\|P) + \ln \frac{\mathcal{I}_d(n)}{\delta}}{n}, \tag{2.33}$$

where

$$\mathcal{I}_d(n) \coloneqq \sup_{r \in [0,1]} \mathbb{E}_{Z \sim \mathrm{Bin}(n,r)} \left[e^{n d(Z/n, r)} \right].$$

Choosing $d(q, p) = \mathrm{kl}(q\|p)$ defined in Equation (2.20) recovers a looser version of Theorem 6 in the case of binary classification, found in Langford and Seeger (2001) and Seeger (2002). Taking taking $d(q, p) = 2(q - p)^2$ produces a bound due to McAllester (2003). And setting $d_\beta(q, p) = -\ln(1 + p(e^{-\beta} - 1)) - \beta q$ for any $\beta > 0$ gives Theorem 1.2.1 from Catoni (2007). A few more specialisations to existing theorems are discussed in Bégin et al. (2016). Note that for any choice of d we can define its inverse with respect to its second argument as $d^{-1}(q, B) \coloneqq \sup\{p \in [0, 1] : d(q, p) \leq B\}$ (with $\sup \emptyset \coloneqq 1$) just as we did for the small kl in Equation (2.22).

The goal of Foong et al. (2021) was to see whether there exists a choice of convex function d for which Theorem 10 beats the test set bound baseline in the small data regime (they take $n \approx$ 30-60), where self-certified learning could be especially beneficial. First, they show that if one illegally chooses d in an S-dependent way, the best result is achieved by d_β for some value of $\beta > 0$, which, incidentally, yields the Chernoff test set bound (Langford and Schapire, 2005) in the case where $Q = P$, which is looser than the binomial tail test set bound (2.30). Since choosing d in a sample *independent* way can only do worse, this result restricts how tight bounds produced by Theorem 10 can be.

While they observe in Foong et al. (2021) that Theorem 10 cannot beat the binomial tail test set bound in the case $Q = P$, this leaves open the question of whether better choices of Q, perhaps learned by minimising the PAC-Bayes bound, can nevertheless beat the binomial test set bound (in expectation over S) for some choice of d. To investigate this question, they meta-learn both an optimal d and a stochastic algorithm $\mathcal{A} : (\mathcal{X} \times \mathcal{Y})^* \to \triangle(\mathcal{H})$ across data-generating distributions $D \in \triangle(\mathcal{X} \times \mathcal{Y})$ sampled from some meta-data-generating distribution $\mathcal{T} \in \triangle(\triangle(\mathcal{X} \times \mathcal{Y}))$. Unfortunately, even in this highly favourable (and usually unrealistic) scenario, they find that PAC-Bayes bounds derived from Theorem 10 are unable to beat binomial tail test set bounds in expectation over $S \sim D, D \sim \mathcal{T}$. Of course, it may still be the case that it is possible to beat binomial tail test set bounds for choices of $\mathcal{T}$ other than theirs, or for PAC-Bayes bounds that are not special cases of Theorem 10. Their results are at least suggestive however that self-certified learning will not be achieved by any specialisation of Theorem 10.

Chapter 3

Controlling Multiple Errors Simultaneously with a PAC-Bayes Bound

3.1 Introduction

Much of the PAC-Bayes literature focuses on the case of binary classification, or of multiclass classification where one only distinguishes whether each classification is correct or incorrect. This is in stark contrast to the complexity of contemporary real-world learning problems, such as medical diagnosis where the severity of Type I and Type II errors may be crucial and context-dependent. This chapter aims to bridge this gap by deriving a generalisation bound that provides information-rich measures of performance at test time by controlling the probabilities of errors of any finite number of user-specified types. More precisely, we bound the KL divergence between the empirical and true distributions over the different error types. From this single bound one can derive bounds on arbitrary linear combinations of these error probabilities, which will all hold simultaneously with the same probability as the original bound. In addition, these bounds are guaranteed to be non-vacuous (this follows since the KL divergence blows up on the boundary of the simplex).

As a concrete example, if the severity of Type I and Type II errors of a medical test are context-dependent, one would want to be able to bound arbitrary linear combinations of these error probabilities. Existing bounds could only bound finitely many pre-specified weightings by employing a union bound, which would also degrade the bound. In contrast, by constraining the KL divergence between the true and empirical error probabilities, our bound constrains all of the uncountably many weightings of the error probabilities simultaneously.

The most common setting of PAC-Bayesian theory is that of binary classification with the zero-one loss, as in Theorem 10 from Section 2.5.2, namely a binary label space $\mathcal{Y} = \{-1, 1\}$ with the zero-one loss $\ell(\hat{y}, y) = \mathbb{1}[\hat{y} \neq y]$. Recall that this bound, originally due to Germain et al. (2009, 2015) and streamlined in Bégin et al. (2016), unifies various PAC-Bayes bounds. The bound is binary in the sense that $\mathcal{Y}$ contains two elements, but a more subtle way to look at this is that only two cases are distinguished—correct classification and incorrect classification. It can in fact be applied to multiclass classification provided one maintains the second binary characteristic by only distinguishing correct and incorrect classifications. It is this heavy restriction that our result lifts, by considering the new framework of *error types*.

By a framework of *error types*, we mean a user-specified finite partition of the space $\mathcal{Y} \times \mathcal{Y}$ into error types $E_1, \dots, E_M$, where $\mathcal{Y}$ is an arbitrary (not necessarily finite) label space. Our bound then simultaneously constrains the probability with which errors of each type occur. In multiclass classification for example, one can choose the error types to be the set of all different possible misclassifications, in which case our bound will control the entire confusion matrix, bounding how far the true confusion matrix (i.e. expected over the data-generating distribution) can diverge from the empirical one (i.e. on the training set). From this one can derive bounds on the probabilities with which each misclassification may be made, and arbitrary linear combinations of these error probabilities, and all of these will hold simultaneously with the same probability as the original bound. Our bound therefore paints a far richer picture of the performance of the final learned model than can be provided by any existing PAC-Bayes bound.

More formally, we let $E_1, \dots, E_M$ be a user-specified disjoint partition of $\mathcal{Y}^2$ into a finite number of M error types, where we say that a hypothesis $h \in \mathcal{H}$ makes an error of type j on data point (x, y) if $(h(x), y) \in E_j$. By convention, every pair $(\hat{y}, y) \in \mathcal{Y}^2$ is interpreted as a predicted value $\hat{y}$ followed by a true value y, in that order. It should be stressed that not all of the E_j need correspond to mislabellings—indeed, some of the E_j may distinguish different correct labellings.

3.2 Related work

Our framework of a finite number of user-specified error types includes multiclass classification as a particular case, and it is in this field that one finds the work most closely related to ours. Little is known about multiclass classification from the theoretical perspective of generalisation bounds in the DL regime. To the best of our knowledge, only a handful of relevant strategies or generalisation bounds can be compared to work presented in this chapter.

Closely related is Morvant et al. (2012), which establishes a PAC-Bayes bound on the spectral norm of the difference between the true and empirical confusion matrices. Our bound differs from theirs in two respects. First, they consider the confusion matrix, whereas ours applies to the more general setting of a finite number of error types, which can be the set of all misclassifications or some partition thereof, and is even applicable beyond classification to tasks with continuous label spaces. Second, they deal with the spectral norm, whereas we employ the KL divergence. Since the KL divergence follows a simple formula, we can much more easily infer bounds on the individual error probabilities, which is challenging for the spectral norm.

The follow-up work Koço and Capponi (2013) shows how a proxy of the spectral norm bound can be used as a training objective that may deal with imbalanced classes. In the present work, we show how our bound can be used as a differentiable training objective directly (without the need of a proxy) and that it can more sensitively deal with imbalanced classes, or errors of different severity, by assigning each error type a user-specified loss value. Benabbou and Lang (2017) present a streamlined version of some of the results from Morvant et al. (2012) in the case where some examples are voluntarily not classified, for example in the case of high uncertainty. This is the work most closely related to ours, except our proof is far more involved leading to a tighter bound.

Laviolette et al. (2017) extend the celebrated $\mathcal{C}$-bound in PAC-Bayes to ensembles, obtaining a bound on the risk of the majority vote classifier in the case of multiclass classification. In this context, our bound is able to distinguish different misclassifications and control them, whereas they bound the scalar risk which lumps all misclassifications together. The $\mathcal{C}$-bound has alternately been generalised by Lacasse et al. (2006) (see also Germain et al. (2015)) to simultaneously control three metrics, namely the so-called *expected disagreement, expected joint success* and *expected joint error* of the posterior. While they restricted themselves to the ternary case, some of their proof techniques share similarities with ours. In cases where one has exactly three error types, for example the $\{-1, 0, 1\}$-valued *excess loss*, the work of Wu and Seldin (2022) is applicable; they construct so-called 'split-kl' inequalities (both classical and PAC-Bayesian) which deftly handle this specific scenario.

Pires et al. (2013) present a comprehensive analysis of convex surrogate losses in cost-sensitive multiclass classification, providing conditions for consistency, bounding the excess loss of a predictor, and extending the analysis to the "Simplex Coding" scheme. We consider the generalisation gap rather than the excess loss. Lei et al. (2019) study data-dependent bounds for multiclass classification. Their analysis is restricted to SVMs however, whereas ours applies to arbitrary hypothesis spaces. Feofanov et al. (2019) derive bounds for the error rate of a majority vote classifier in the scenario of multiclass classification with partial labels. They bound the individual elements of the confusion matrix, whereas our bound constrains the entire distribution.

Outline. We fix notation in Section 3.3. Theorem 11 in Section 3.4 is our main result—a PAC-Bayes bound on the KL divergence between the true and empirical error distributions. For multiclass classification with a fully refined partition this becomes a bound on the KL divergence between the true and empirical confusion matrices. Proposition 1 then bounds the individual error probabilities. Our second main result, Theorem 12 in Section 3.5, allows us to use bounds on *linear combinations* of error probabilities as training objectives. We prove Theorem 11 in Section 3.6 via Proposition 4, which bounds the distribution of errors via a general convex function d, and may be of independent interest. Section 3.7 outlines positive empirical results[1] from using our bound as a training objective for neural networks, and Section 3.8 gives perspectives for future work.

[1]Code available here: https://github.com/reubenadams/PAC-Bayes-Control

3.3 Notation

We are interested in *simple* hypotheses $h : \mathcal{X} \to \mathcal{Y}$ and *soft* hypotheses $H : \mathcal{X} \to \triangle(\mathcal{Y})$. For example, a neural network outputting scores (logits) in $\mathbb{R}^{|\mathcal{Y}|}$ is converted to a simple or soft hypothesis, respectively, by passing the scores through the argmax or softmax function, respectively. For any $A \subseteq \mathcal{Y}$, $H(x)(A)$ can be interpreted as the probability according to H that the label of x is in A. We will see in Section 3.5 that soft hypotheses permit more flexible training procedures and a more fine-grained analysis. Note that while soft hypotheses output distributions, they do so deterministically, always returning the same distribution for the same input x, and so are distinct from the stochastic classifiers introduced shortly.

For a simple hypothesis $h : \mathcal{X} \to \mathcal{Y}$ and $j \in [M]$, define the *j-risk* of h to be $R^j_D(h) := \mathbb{P}_{(x,y)\sim D}((h(x), y) \in E_j)$, namely the probability that h makes an error of type E_j for a randomly sampled $(x, y) \sim D$. For a soft hypothesis $H : \mathcal{X} \to \triangle(\mathcal{Y})$ define the *j-risk* of H to be $R^j_D(H) := \mathbb{P}_{(x,y)\sim D, \hat{y}\sim H(x)}((\hat{y}, y) \in E_j)$, namely the probability that one would make an error of type E_j on a randomly sampled $(x, y) \sim D$ if one predicted by sampling $\hat{y}$ from the distribution $H(x) \in \triangle(\mathcal{Y})$. From now until Section 3.5 it will not matter whether we are dealing with simple or soft hypotheses. So, unless stated explicitly, we refer to both simply as hypotheses, denote both by lowercase h, and refer to the hypothesis class $\mathcal{H}$, whether it is a subset of $\mathcal{Y}^{\mathcal{X}}$ or $\triangle(\mathcal{Y})^{\mathcal{X}}$. For ease of setting up notation, let $\hat{D}(S) \in \triangle(\mathcal{X}\times\mathcal{Y})$ denote the empirical distribution $\hat{D}(S) := \frac{1}{n}\sum_{(x,y)\in S} \delta_{(x,y)}$, namely the distribution consisting of delta masses of mass $1/n$ on each of the data points in S.

Our goal is to control the *true risk vector* $\boldsymbol{R}_D(h) := (R^1_D(h), \ldots, R^M_D(h))$, since controlling this vector controls all linear combinations of j-risks. Since this is unobservable, we will control it by bounding how far it diverges from its empirical counterpart $\boldsymbol{R}_S(h) := \boldsymbol{R}_{\hat{D}(S)}(h)$, which we term the *empirical risk vector*. Note that $\mathbb{E}_{S\sim D^n}\boldsymbol{R}_S(h) = \boldsymbol{R}_D(h)$, and that, for a simple hypothesis $h \in \mathcal{Y}^{\mathcal{X}}$, $\boldsymbol{R}_S(h)$ is the vector of proportions of the sample on which h makes an error of type E_j, which can be seen as follows

$$\big(\boldsymbol{R}_S(h)\big)_j = R^j_{\hat{D}(S)}(h) = \mathbb{P}_{(x,y)\sim\hat{D}(S)}\Big((h(x), y) \in E_j\Big) = \frac{1}{n}\sum_{(x,y)\in S} \mathbb{1}\big[(h(x), y) \in E_j\big].$$

Since the E_j partition $\mathcal{Y}^2$, $\boldsymbol{R}_D(h)$ and $\boldsymbol{R}_S(h)$ are elements of the M-dimensional simplex $\triangle_M := \{\boldsymbol{u} \in [0,1]^M : u_1 + \cdots + u_M = 1\}$. Thus we can choose our divergence measure to be $\mathrm{kl}(\boldsymbol{R}_S(Q)\|\boldsymbol{R}_D(Q))$, where for $\boldsymbol{q}, \boldsymbol{p} \in \triangle_M$ we define

$$\mathrm{kl}(\boldsymbol{q}\|\boldsymbol{p}) := \sum_{j=1}^{M} q_j \ln\frac{q_j}{p_j},$$

following the usual convention that $0 \ln \frac{0}{x} = 0$ for $x \geq 0$ and $x \ln \frac{x}{0} = \infty$ for $x > 0$. When $M = 2$ we abbreviate $\mathrm{kl}((q, 1-q)\|(p, 1-p))$ to $\mathrm{kl}(q\|p)$, which is then the conventional definition of $\mathrm{kl}(\cdot\|\cdot)$ found in the PAC-Bayes literature, defined in Equation (2.20) from Section 2.3. We define the *true* and *empirical risk vectors* of Q as $\boldsymbol{R}_D(Q) := \mathbb{E}_{h\sim Q}\boldsymbol{R}_D(h)$ and $\boldsymbol{R}_S(Q) := \mathbb{E}_{h\sim Q}\boldsymbol{R}_S(h)$, respectively, and seek a bound on $\mathrm{kl}(\boldsymbol{R}_S(Q)\|\boldsymbol{R}_D(Q))$. Note we still have $\mathbb{E}_S[\boldsymbol{R}_S(Q)] = \boldsymbol{R}_D(Q)$, this time using Fubini. Moreover, for a sample S of size n, we have that $\boldsymbol{R}_S(Q) = \boldsymbol{K}/n$ where $\boldsymbol{K} \sim \mathrm{Mult}(n, M, \boldsymbol{R}_D(Q))$. Recall that for $n, M \in \mathbb{N}$ and $\boldsymbol{r} \in \triangle_M$, the multinomial distribution $\mathrm{Mult}(n, M, \boldsymbol{r})$ has probability mass function

$$\mathrm{Mult}(\boldsymbol{k}; n, M, \boldsymbol{r}) := \binom{n}{k_1\ k_2\ \cdots\ k_M} \prod_{j=1}^{M} r_j^{k_j}, \quad \text{where} \quad \binom{n}{k_1\ k_2\ \cdots\ k_M} := \frac{n!}{\prod_{j=1}^{M} k_j!}$$

for $\boldsymbol{k} \in S_{n,M} := \left\{(k_1, \ldots, k_M) \in \mathbb{N}_0^M : k_1 + \cdots + k_M = n\right\}$, and zero otherwise. As a final piece of notation, we let $\triangle_M^{>0} := \triangle_M \cap (0,1)^M$ and $S_{n,M}^{>0} := S_{n,M} \cap \mathbb{N}^M$ denote the vector elements of $\triangle_M$ and $S_{n,M}$, respectively, that have no zero components.

3.4 Main result

We now state our main result, which bounds the KL divergence between the true and empirical risk vectors $\boldsymbol{R}_D(Q)$ and $\boldsymbol{R}_S(Q)$, interpreted as probability distributions. As is conventional in the PAC-Bayes literature, we refer to sample independent and dependent distributions $P, Q \in \triangle(\mathcal{H})$, *i.e.* stochastic hypotheses, as *priors* P and *posteriors* Q respectively, even if they are not related by Bayes' theorem.

Theorem 11. *Let $\mathcal{X}$ and $\mathcal{Y}$ be arbitrary sets and $E_1, \ldots, E_M$ be a disjoint partition of $\mathcal{Y}^2$ into M error types. Let $D \in \triangle(\mathcal{X} \times \mathcal{Y})$ be a data-generating distribution and $\mathcal{H}$ be a simple ($\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$) or soft ($\mathcal{H} \subseteq \triangle(\mathcal{Y})^{\mathcal{X}}$) hypothesis class. For any prior $P \in \triangle(\mathcal{H})$, $\delta \in (0,1]$ and sample size $n \geq M$, with probability at least $1-\delta$ over the random draw $S \sim D^n$, we have that simultaneously for all posteriors $Q \in \triangle(\mathcal{H})$, the divergence $\mathrm{kl}\big(\boldsymbol{R}_S(Q)\big\|\boldsymbol{R}_D(Q)\big)$ is upper bounded by*

$$\frac{1}{n}\left[\mathrm{KL}(Q\|P) + \ln\frac{\xi(n,M)}{\delta}\right], \quad \textit{where}$$

$$\xi(n, M) := \sqrt{\pi}e^{1/(12n)}\left(\frac{n}{2}\right)^{\frac{M-1}{2}} \sum_{z=0}^{M-1} \binom{M}{z}\left(\frac{2}{n}\right)^{z/2} \Gamma\left(\frac{M-z}{2}\right)^{-1} \in \mathcal{O}\left((nM)^M\right).$$

The fact that the logarithmic term is of order $\mathcal{O}(M\ln(nM/\delta))$ means the bound is linear in M up to logarithmic terms. While this may seem excessive, one should note that the quantity that our theorem bounds also depends on M. Further, the bound has been successfully used by Biggs and Guedj (2023) to improve on state of the art PAC-Bayes bounds.

To see how our bound compares to existing PAC-Bayes bounds for binary classification, take $\mathcal{Y} = \{-1, 1\}$, $M = 2$, and

$$E_1 = \{(1,-1), (-1,1)\}, \quad \text{and} \quad E_2 = \{(1,1), (-1,-1)\},$$

corresponding to incorrect and correct classification, respectively. The argument of the logarithm then reduces to $\frac{1}{\delta}e^{1/(12n)}\left(2 + \sqrt{\frac{\pi n}{2}}\right) \leq 1.25\sqrt{n}$ when n is large. The corresponding term in Maurer (2004) (given as Theorem 6 in Section 2.3) is $2\sqrt{n}$, which is only larger because Maurer relaxes the term for aesthetics. Therefore our bound gracefully reduces to Maurer's in the case of binary classification with zero-one loss.

Suppose after a use of Theorem 11 we have a bound of the form $\mathrm{kl}(\boldsymbol{R}_S(Q)\|\boldsymbol{R}_D(Q)) \leq B$. We can then derive bounds on the individual j-risks $R^j_D(Q)$ or, more generally, on linear combinations thereof. While one could obtain such bounds perhaps more directly with existing PAC-Bayes bounds, the significance of our bound is that *all* such derived bounds hold with high probability *simultaneously*. Existing PAC-Bayes bounds would require the use of a union bound in order to bound multiple combinations simultaneously, whereas ours bounds all of the uncountably many combinations simultaneously, as a package. As for the individual j-risks $R^j_D(Q)$, the following proposition then yields the bounds

$$L_j \leq R^j_D(Q) \leq U_j,$$

where

$$\begin{aligned} L_j &:= \inf\Big\{p \in [0,1] : \mathrm{kl}\big(R^j_S(Q)\big\|p\big) \leq B\Big\}, \\ U_j &:= \sup\Big\{p \in [0,1] : \mathrm{kl}\big(R^j_S(Q)\big\|p\big) \leq B\Big\}. \end{aligned}$$

Moreover, since in the worst case we have $\mathrm{kl}(\boldsymbol{R}_S(Q)\|\boldsymbol{R}_D(Q)) = B$, the proposition shows that the lower and upper bounds L_j and U_j are the tightest possible, since if $R^j_D(Q) \notin [L_j, U_j]$ then $\mathrm{kl}(R^j_S(Q)\|R^j_D(Q)) > B$ implying $\mathrm{kl}(\boldsymbol{R}_S(Q)\|\boldsymbol{R}_D(Q)) > B$. For a more precise version of this argument and a proof of Proposition 1, see Appendix A.3.4.

Proposition 1. *Let $\boldsymbol{q}, \boldsymbol{p} \in \triangle_M$. Then $\mathrm{kl}(q_j\|p_j) \leq \mathrm{kl}(\boldsymbol{q}\|\boldsymbol{p})$ for all $j \in [M]$, with equality when $p_i = \frac{1-p_j}{1-q_j} q_i$. for all $i \neq j$.*

Going beyond bounds on the individual j-risks, suppose we can quantify how costly an error of each type is by means of a loss vector $\boldsymbol{\ell} \in [0,\infty)^M$, where ℓ_j is the loss we attribute to an error of type E_j. We may then be interested in bounding the *total risk*

$$R^T_D(Q) := \boldsymbol{\ell} \cdot \boldsymbol{R}_D(Q) = \sum_{j=1}^{M} \ell_j R^j_D(Q).$$

Then, given a bound $\mathrm{kl}(\boldsymbol{R}_S(Q)\|\boldsymbol{R}_D(Q)) \leq B$ from Theorem 11, we can deduce

$$R^T_D(Q) \leq \sup\Big\{\boldsymbol{\ell} \cdot \boldsymbol{r} : \boldsymbol{r} \in \triangle_M,\ \mathrm{kl}\big(\boldsymbol{R}_S(Q)\|\boldsymbol{r}\big) \leq B\Big\} = \boldsymbol{\ell} \cdot \mathrm{kl}^{-1}_{\boldsymbol{\ell}}\big(\boldsymbol{R}_S(Q)|B\big),$$

where we define $\mathrm{kl}^{-1}_{\boldsymbol{\ell}}(\boldsymbol{u}|c) \in \triangle_M$ as follows. To see that it is indeed well-defined (at least when $\boldsymbol{u} \in \triangle^{>0}_M$), see the discussion at the beginning of Appendix A.3.5.

Definition 8. For $\boldsymbol{u} \in \triangle_M, c \in [0,\infty)$ and $\boldsymbol{\ell} \in [0,\infty)^M$, define $\mathrm{kl}^{-1}_{\boldsymbol{\ell}}(\boldsymbol{u}|c)$ to be an element

$\boldsymbol{v} \in \triangle_M$ solving the constrained optimisation problem

$$\text{Maximise:} \quad f_{\boldsymbol{\ell}}(\boldsymbol{v}) := \boldsymbol{\ell} \cdot \boldsymbol{v}, \tag{3.1}$$

$$\text{Subject to:} \quad \mathrm{kl}(\boldsymbol{u} \| \boldsymbol{v}) \leq c. \tag{3.2}$$

This motivates the following training procedure: search for a posterior Q for which the bound $\boldsymbol{\ell} \cdot \mathrm{kl}_{\boldsymbol{\ell}}^{-1}(\boldsymbol{R}_S(Q)|B)$ on the total risk $R_D^T(Q)$ is minimised. While this requires a particular choice of loss vector $\boldsymbol{\ell}$, we emphasise that at the end of training, Theorem 11 bounds $\mathrm{kl}(\boldsymbol{R}_S(Q)\|\boldsymbol{R}_D(Q))$, and so can be used to bound *any* linear combination of the j-risks, not just the one given the loss vector $\boldsymbol{\ell}$ chosen for training. It is this flexibility which is the main advantage of our bound; changes in the severity of different error types over time do not require union bounds or retraining.

In the next section we provide a theorem for calculating $\mathrm{kl}_{\boldsymbol{\ell}}^{-1}(\boldsymbol{u}|c)$ and its derivatives so that the training procedure can be executed.

3.5 Construction of a differentiable training objective

We now state and prove Theorem 12, which provides a speedy method for approximating $\mathrm{kl}_{\boldsymbol{\ell}}^{-1}(\boldsymbol{u}|c)$ and its derivatives to arbitrary precision, provided $c > 0$ and $\forall j \; u_j > 0$. The only approximation step required is that of approximating the unique root of a continuous and strictly increasing scalar function. Thus, provided the u_j themselves are differentiable, Theorem 11 combined with Theorem 12 shows that the upper bound on the total risk can be used as a tractable and fully differentiable training objective. See Appendix A.1 for more details, including a pseudocode algorithm and an implementation. Since the proof of Theorem 12 is rather long and technical, we defer it to Appendix A.3.5. The requirement that the ℓ_j are not all equal serves only to rule out trivial cases where $R_D^T(Q)$ is independent of $\boldsymbol{R}_D(Q)$.

Theorem 12. *Fix $\boldsymbol{\ell} \in [0,\infty)^M$ such that not all ℓ_j are equal, and define $f_{\boldsymbol{\ell}} : \triangle_M \to [0,\infty)$ by $f_{\boldsymbol{\ell}}(\boldsymbol{v}) := \sum_{j=1}^M \ell_j v_j$. For all $\tilde{\boldsymbol{u}} = (\boldsymbol{u}, c) \in \triangle_M^{>0} \times (0,\infty)$, define $\boldsymbol{v}^*(\tilde{\boldsymbol{u}}) := \mathrm{kl}_{\boldsymbol{\ell}}^{-1}(\boldsymbol{u}|c) \in \triangle_M$ and let $\mu^*(\tilde{\boldsymbol{u}}) \in (-\infty, -\max_j \ell_j)$ be the unique solution to $c = \phi_{\boldsymbol{\ell}}(\mu)$, where $\phi_{\boldsymbol{\ell}} : (-\infty, -\max_j \ell_j) \to \mathbb{R}$ is given by $\phi_{\boldsymbol{\ell}}(\mu) := \ln(-\sum_{j=1}^M \frac{u_j}{\mu+\ell_j}) + \sum_{j=1}^M u_j \ln(-(\mu+\ell_j))$, which is continuous and strictly increasing. Then $\boldsymbol{v}^*(\tilde{\boldsymbol{u}}) = \mathrm{kl}_{\boldsymbol{\ell}}^{-1}(\boldsymbol{u}|c)$ is given by*

$$\boldsymbol{v}^*(\tilde{\boldsymbol{u}})_j = \frac{\lambda^*(\tilde{\boldsymbol{u}}) u_j}{\mu^*(\tilde{\boldsymbol{u}}) + \ell_j} \quad \textit{for } j \in [M], \quad \textit{where} \quad \lambda^*(\tilde{\boldsymbol{u}}) = \left(\sum_{j=1}^M \frac{u_j}{\mu^*(\tilde{\boldsymbol{u}}) + \ell_j} \right)^{-1}. \tag{3.3}$$

Further, defining $f_{\boldsymbol{\ell}}^* : \triangle_M^{>0} \times (0,\infty) \to [0,\infty)$ by $f_{\boldsymbol{\ell}}^*(\tilde{\boldsymbol{u}}) := f_{\boldsymbol{\ell}}(\boldsymbol{v}^*(\tilde{\boldsymbol{u}}))$, we have that

$$\frac{\partial f_{\boldsymbol{\ell}}^*}{\partial u_j}(\tilde{\boldsymbol{u}}) = \lambda^*(\tilde{\boldsymbol{u}}) \left(1 + \ln \frac{u_j}{\boldsymbol{v}^*(\tilde{\boldsymbol{u}})_j} \right) \qquad \textit{and} \qquad \frac{\partial f_{\boldsymbol{\ell}}^*}{\partial c}(\tilde{\boldsymbol{u}}) = -\lambda^*(\tilde{\boldsymbol{u}}). \tag{3.4}$$

A final wrinkle in evaluating our bound is that while the empirical risk vector $\boldsymbol{R}_S(Q) = \mathbb{E}_{h\sim Q}\boldsymbol{R}_S(h)$ does not depend on the data-generating distribution D, the expectation over Q

may still be intractable. This would be the default case when Q is a Gaussian over the weights of a multilayer perceptron, for example. In such cases, we can estimate $\boldsymbol{R}_S(Q)$ via a Monte Carlo sample $\boldsymbol{R}_S(\hat{Q}) := \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{R}_S(h_n)$ (where the h_n are drawn i.i.d. from Q) and use the following two results. Proposition 2 shows that the $\mathrm{kl}(R_S^j(\hat{Q})\|R_D^j(Q))$ can be simultaneously bounded, whence Proposition 3 can be used to obtain a bound on $\mathrm{kl}(\boldsymbol{R}_S(\hat{Q})\|\boldsymbol{R}_D(Q))$.

Proposition 2. *Let $\boldsymbol{X} \sim \mathrm{Multinomial}(N, M, \boldsymbol{p})$. Then for any $\delta \in (0,1)$, with probability at least $1-\delta$ we have that for all $j \in [M]$ simultaneously*

$$\mathrm{kl}\left(\frac{1}{N}X_j \middle\| p_j\right) \leq \frac{\ln\frac{2M}{\delta}}{N}.$$

Proof. Each bound holds separately with probability at least $1-\delta/M$ by Theorem 2.5 in Langford and Caruana (2001). They then hold simultaneously by application of a union bound. □

Proposition 3. *Suppose $\boldsymbol{q}, \boldsymbol{p}, \hat{\boldsymbol{q}} \in \triangle_M$ are such that $\mathrm{kl}(\boldsymbol{q}\|\boldsymbol{p}) \leq B_1$ and $\mathrm{kl}(\hat{q}_j\|q_j) \leq B_2$ for all $j \in [M]$. For each j, define $\underline{q}_j = \inf\{r \in [0,1] : \mathrm{kl}(\hat{q}_j\|r) \leq B_2\}$. Then*

$$\mathrm{kl}(\hat{\boldsymbol{q}}\|\boldsymbol{p}) \leq MB_2 - \sum_{j=1}^{M}(1-\hat{q}_j)\ln\frac{1-\hat{q}_j}{1-\underline{q}_j} + B_1\max_j\frac{\hat{q}_j}{\underline{q}_j} \to B_1 \quad \textit{as} \quad B_2 \to 0.$$

Proof. Deferred to A.3.1. □

The fact that the bound on $\mathrm{kl}(\hat{\boldsymbol{q}}\|\boldsymbol{p}) \to B_1$ as $B_2 \to 0$ ensures that as we increase the size of our Monte Carlo sample for estimating $\boldsymbol{R}_S(Q)$ the bound on $\mathrm{kl}(\boldsymbol{R}_S(\hat{Q})\|\boldsymbol{R}_D(Q))$ approaches that of $\mathrm{kl}(\boldsymbol{R}_S(Q)\|\boldsymbol{R}_D(Q))$, meaning in the limit we pay an arbitrarily small price in the bound for the approximation.

3.6 Proof of the main bound

We split the proof of Theorem 11 into three parts. First, we prove Proposition 4, a bound on $d(\boldsymbol{R}_S(Q), \boldsymbol{R}_D(Q))$ for an arbitrary convex function d, which may be of independent interest. Second, we prove Corollary 1 by specialising Proposition 4 to the case $d(\cdot,\cdot) = \mathrm{kl}(\cdot\|\cdot)$. Finally, we show that the bound in Theorem 11 is a loosened version of the bound in Corollary 1.

Proposition 4. *Let $d : \triangle_M^2 \to \mathbb{R}$ be jointly convex. In the setting of Theorem 11, for any $\beta > 0$*

$$d\big(\boldsymbol{R}_S(Q), \boldsymbol{R}_D(Q)\big) \leq \frac{1}{\beta}\left[\mathrm{KL}(Q\|P) + \ln\frac{\mathcal{I}_d(n,\beta)}{\delta}\right], \quad \textit{where} \tag{3.5}$$

$$\mathcal{I}_d(n,\beta) := \sup_{\boldsymbol{r}\in\triangle_M}\left[\sum_{\boldsymbol{k}\in S_{n,M}}\mathrm{Mult}(\boldsymbol{k}; n, M, \boldsymbol{r})\exp\left(\beta d\left(\tfrac{\boldsymbol{k}}{n}, \boldsymbol{r}\right)\right)\right].$$

This is a generalisation of the unifying PAC-Bayes bound given in Bégin et al. (2016) where we replace the scalar risk quantities $R_S(Q)$ and $R_D(Q)$ with their vector counterparts $\boldsymbol{R}_S(Q)$ and $\boldsymbol{R}_D(Q)$. To see this, note that we can recover it by setting $\mathcal{Y} = \{-1, 1\}$, $M = 2$, $E_1 = \{(-y, y) : y \in \mathcal{Y}\}$ and $E_2 = \{(y, y) : y \in \mathcal{Y}\}$. Then, for any convex function $d : [0, 1]^2 \to \mathbb{R}$, apply Proposition 4 with the convex function $d' : \triangle_M^2 \to \mathbb{R}$ defined by $d'((u_1, u_2), (v_1, v_2)) := d(u_1, v_1)$ so that Proposition 4 bounds $d'\big(\boldsymbol{R}_S(Q), \boldsymbol{R}_D(Q)\big) = d\big(R_S^1(Q), R_D^1(Q)\big)$ which equals $d(R_S(Q), R_D(Q))$ in the notation of Bégin et al. (2016). Further,

$$\sum_{\boldsymbol{k} \in S_{n,2}} \mathrm{Mult}(\boldsymbol{k}; n, 2, \boldsymbol{r}) \exp\Big(\beta d'\big(\tfrac{\boldsymbol{k}}{n}, \boldsymbol{r}\big)\Big) = \sum_{k=0}^{n} \mathrm{Bin}(k; n, r_1) \exp\Big(\beta d\big(\tfrac{k}{n}, r_1\big)\Big),$$

so that the supremum over $r_1 \in [0, 1]$ of the right hand side equals the supremum over $\boldsymbol{r} \in \triangle_2$ of the left hand side, which, when substituted into (3.5), yields the bound given in Bégin et al. (2016).

To prove Proposition 4 we require the following two lemmas. The first is the well-known change of measure in equality (Csiszár, 1975; Donsker and Varadhan, 1975). The second is a generalisation from Binomial to Multinomial distributions of a result found in Maurer (2004), the proof of which we defer to Appendix A.3.2.

Lemma 1. *For any set $\mathcal{H}$, any $P, Q \in \triangle(\mathcal{H})$ and any measurable function $\phi : \mathcal{H} \to \mathbb{R}$,*

$$\mathop{\mathbb{E}}_{h \sim Q} \phi(h) \leq \mathrm{KL}(Q \| P) + \ln \mathop{\mathbb{E}}_{h \sim P} \exp(\phi(h)).$$

Lemma 2. *Let $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ be i.i.d $\triangle_M$-valued random vectors with mean $\boldsymbol{\mu}$ and suppose that $f : \triangle_M^n \to \mathbb{R}$ is convex. If $\boldsymbol{X}'_1, \ldots, \boldsymbol{X}'_n$ are i.i.d.* $\mathrm{Mult}(1, M, \boldsymbol{\mu})$ *random vectors, then*

$$\mathbb{E}[f(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)] \leq \mathbb{E}[f(\boldsymbol{X}'_1, \ldots, \boldsymbol{X}'_n)].$$

The consequence of Lemma 2 is that the worst case (in terms of bounding $d(\boldsymbol{R}_S(Q), \boldsymbol{R}_D(Q))$) occurs when $\boldsymbol{R}_{\{(x,y)\}}(h)$ is a one-hot vector for all $(x, y) \in S$ and $h \in \mathcal{H}$, namely when $\mathcal{H} \subseteq \triangle(\mathcal{Y})^{\mathcal{X}}$ only contains hypotheses that, when labelling S, put all their mass on elements $\hat{y} \in \mathcal{Y}$ that incur the same error type[2]. In particular, this is the case for hypotheses that put all their mass on a single element of $\mathcal{Y}$, equivalent to the simpler case $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ as discussed in Section 3.3. Thus, Lemma 2 shows that the bound given in Proposition 4 cannot be made tighter only by restricting to such hypotheses.

Proof. (of Proposition 4) The case $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ follows directly from the more general case by taking

$$\mathcal{H}' := \Big\{h' \in \triangle(\mathcal{Y})^{\mathcal{X}} : \exists h \in \mathcal{H} \text{ such that } \forall x \in \mathcal{X} \;\; h'(x) = \delta_{h(x)}\Big\}$$

where $\delta_{h(x)} \in \triangle(\mathcal{Y})$ denotes a point mass on $h(x)$. For the general case $\mathcal{H} \subseteq \triangle(\mathcal{Y})^{\mathcal{X}}$, using Jensen's inequality with the convex function $d(\cdot, \cdot)$ and Lemma 1 with $\phi(h) = \beta d(\boldsymbol{R}_S(h), \boldsymbol{R}_D(h))$,

[2]More precisely, when $\forall h \in \mathcal{H}$ $\forall (x, y) \in S$ $\exists j \in [M]$ such that $h(x)[\{\hat{y} \in \mathcal{Y} : (\hat{y}, y) \in E_j)\}] = 1$.

we see that for all $Q \in \triangle(\mathcal{H})$

$$\begin{aligned}\beta d\big(\boldsymbol{R}_S(Q), \boldsymbol{R}_D(Q)\big) &= \beta d\left(\mathop{\mathbb{E}}_{h\sim Q}\boldsymbol{R}_S(h), \mathop{\mathbb{E}}_{h\sim Q}\boldsymbol{R}_D(h)\right)\\ &\leq \mathop{\mathbb{E}}_{h\sim Q}\beta d\big(\boldsymbol{R}_S(h), \boldsymbol{R}_D(h)\big)\\ &\leq \mathrm{KL}(Q\|P) + \ln\left(\mathop{\mathbb{E}}_{h\sim P}\exp\left(\beta d\big(\boldsymbol{R}_S(h), \boldsymbol{R}_D(h)\big)\right)\right)\\ &= \mathrm{KL}(Q\|P) + \ln(Z_P(S)),\end{aligned}$$

where $Z_P(S) := \mathbb{E}_{h\sim P}\exp\big(\beta d(\boldsymbol{R}_S(h), \boldsymbol{R}_D(h))\big)$. Note that $Z_P(S)$ is a non-negative random variable, so that by Markov's inequality

$$\mathop{\mathrm{P}}_{S\sim D^n}\left(Z_P(S) \leq \frac{\mathbb{E}_{S'\sim D^n}Z_P(S')}{\delta}\right) \geq 1-\delta.$$

Thus, since $\ln(\cdot)$ is strictly increasing, with probability at least $1-\delta$ over $S \sim D^n$, we have that simultaneously for all $Q \in \triangle(\mathcal{H})$

$$\beta d\big(\boldsymbol{R}_S(Q), \boldsymbol{R}_D(Q)\big) \leq \mathrm{KL}(Q\|P) + \ln\frac{\mathop{\mathbb{E}}_{S'\sim D^n}Z_P(S')}{\delta}. \tag{3.6}$$

To bound $\mathbb{E}_{S'\sim D^n}Z_P(S')$, let $\boldsymbol{X}_i := \boldsymbol{R}_{\{(x_i,y_i)'\}}(h) \in \triangle_M$ for $i \in [n]$, where $(x_i, y_i)'$ is the i'th element of the dummy sample S'. Noting that each $\boldsymbol{X}_i$ has mean $\boldsymbol{R}_D(h)$, define the random vectors $\boldsymbol{X}_i' \sim \mathrm{Mult}(1, M, \boldsymbol{R}_D(h))$ and $\boldsymbol{Y} := \sum_{i=1}^n \boldsymbol{X}_i' \sim \mathrm{Mult}(n, M, \boldsymbol{R}_D(h))$. Finally let $f : \triangle_M^n \to \mathbb{R}$ be defined by

$$f(x_1, \dots, x_n) := \exp\left(\beta d\left(\frac{1}{n}\sum_{i=1}^n x_i, \boldsymbol{R}_D(h)\right)\right),$$

which is convex since the average is linear, d is convex and the exponential is non-decreasing and convex. Then, by swapping expectations (which is permitted by Fubini's theorem since the argument is non-negative) and applying Lemma 2, we have that $\mathbb{E}_{S'\sim D^n}Z_P(S')$ can be written as

$$\begin{aligned}\mathbb{E}_{S'\sim D^n}Z_P(S') &= \mathop{\mathbb{E}}_{S'\sim D^n}\mathop{\mathbb{E}}_{h\sim P}\exp\left(\beta d\big(\boldsymbol{R}_{S'}(h), \boldsymbol{R}_D(h)\big)\right)\\ &= \mathop{\mathbb{E}}_{h\sim P}\mathop{\mathbb{E}}_{S'\sim D^n}\exp\left(\beta d\big(\boldsymbol{R}_{S'}(h), \boldsymbol{R}_D(h)\big)\right)\\ &= \mathop{\mathbb{E}}_{h\sim P}\mathop{\mathbb{E}}_{\boldsymbol{X}_1,\dots,\boldsymbol{X}_n}\exp\left(\beta d\left(\frac{1}{n}\sum_{i=1}^n \boldsymbol{X}_i, \boldsymbol{R}_D(h)\right)\right)\\ &\leq \mathop{\mathbb{E}}_{h\sim P}\mathop{\mathbb{E}}_{\boldsymbol{X}_1',\dots,\boldsymbol{X}_n'}\exp\left(\beta d\left(\frac{1}{n}\sum_{i=1}^n \boldsymbol{X}_i', \boldsymbol{R}_D(h)\right)\right)\\ &= \mathop{\mathbb{E}}_{h\sim P}\mathop{\mathbb{E}}_{\boldsymbol{Y}}\exp\left(\beta d\left(\frac{1}{n}\boldsymbol{Y}, \boldsymbol{R}_D(h)\right)\right)\end{aligned}$$

$$= \mathop{\mathbb{E}}_{h\sim P} \sum_{\boldsymbol{k}\in S_{n,M}} \mathrm{Mult}\big(\boldsymbol{k}; n, M, \boldsymbol{R}_D(h)\big) \exp\Big(\beta d\big(\tfrac{\boldsymbol{k}}{n}, \boldsymbol{R}_D(h)\big)\Big)$$

$$\le \sup_{\boldsymbol{r}\in\triangle_M} \left[\sum_{\boldsymbol{k}\in S_{n,M}} \mathrm{Mult}\big(\boldsymbol{k}; n, M, \boldsymbol{r}\big) \exp\Big(\beta d\big(\tfrac{\boldsymbol{k}}{n}, \boldsymbol{r}\big)\Big)\right],$$

which is the definition of $\mathcal{I}_d(n,\beta)$. Inequality (3.5) then follows by substituting this bound on $\mathbb{E}_{S'\sim D^n} Z_P(S')$ into (3.6) and dividing by β. □

We now specialise Proposition 4 to the case $d(\cdot,\cdot) = \mathrm{kl}(\cdot\|\cdot)$ to obtain Corollary 1.

Corollary 1. *In the setting of Theorem 11,*

$$\mathrm{kl}\big(\boldsymbol{R}_S(Q)\big\|\boldsymbol{R}_D(Q)\big) \le \frac{1}{n}\left[\mathrm{KL}(Q\|P) + \ln\frac{\eta(n,M)}{\delta}\right], \quad \textit{where} \tag{3.7}$$

$$\eta(n,M) := \frac{n!}{n^n} \sum_{\boldsymbol{k}\in S_{n,M}} \prod_{j=1}^{M} \frac{k_j^{k_j}}{k_j!}.$$

Proof. Applying Proposition 4 with $d(\cdot,\cdot) = \mathrm{kl}(\cdot\|\cdot)$ and $\beta = n$ gives that with probability at least $1-\delta$ over $S \sim D^n$, simultaneously for all posteriors $Q \in \triangle(\mathcal{H})$,

$$\mathrm{kl}\big(\boldsymbol{R}_S(Q)\big\|\boldsymbol{R}_D(Q)\big) \le \frac{1}{n}\left[\mathrm{KL}(Q\|P) + \ln\frac{\mathcal{I}_{\mathrm{kl}}(n,n)}{\delta}\right],$$

where

$$\mathcal{I}_{\mathrm{kl}}(n,n) := \sup_{\boldsymbol{r}\in\triangle_M} \left[\sum_{\boldsymbol{k}\in S_{n,M}} \mathrm{Mult}(\boldsymbol{k}; n, M, \boldsymbol{r}) \exp\Big(n\mathrm{kl}\big(\tfrac{\boldsymbol{k}}{n}\big\|\boldsymbol{r}\big)\Big)\right]. \tag{3.8}$$

Thus it suffices to show that $\mathcal{I}_{\mathrm{kl}}(n,n) \le \eta(n,M)$.

To prove this, for each fixed $\boldsymbol{r} = (r_1,\dots,r_M) \in \triangle_M$ let $J_{\boldsymbol{r}} = \{j \in [M] : r_j = 0\}$. Then $\mathrm{Mult}(\boldsymbol{k}; n, M, \boldsymbol{r}) = 0$ for any $\boldsymbol{k} \in S_{n,M}$ such that $k_j \neq 0$ for some $j \in J_{\boldsymbol{r}}$. For the other $\boldsymbol{k} \in S_{n,M}$, namely those such that $k_j = 0$ for all $j \in J_{\boldsymbol{r}}$, the probability term can be written as

$$\mathrm{Mult}(\boldsymbol{k}; n, M, \boldsymbol{r}) = \frac{n!}{\prod_{j=1}^M k_j!}\prod_{j=1}^M r_j^{k_j} = \frac{n!}{\prod_{j\notin J_{\boldsymbol{r}}} k_j!}\prod_{j\notin J_{\boldsymbol{r}}} r_j^{k_j},$$

and (recalling the convention that $0\ln\frac{0}{0} = 0$) the term $\exp(n\mathrm{kl}(\frac{\boldsymbol{k}}{n}\|\boldsymbol{r}))$ can be written as

$$\exp\left(n\sum_{j=1}^{M} \frac{k_j}{n}\ln\frac{\frac{k_j}{n}}{r_j}\right) = \exp\left(\sum_{j\notin J_{\boldsymbol{r}}} k_j \ln\frac{k_j}{nr_j}\right) = \prod_{j\notin J_{\boldsymbol{r}}}\left(\frac{k_j}{nr_j}\right)^{k_j} = \frac{1}{n^n}\prod_{j\notin J_{\boldsymbol{r}}}\left(\frac{k_j}{r_j}\right)^{k_j},$$

where the last equality is obtained by recalling that the k_j sum to n. Substituting these two expressions into the definition of $\mathcal{I}_{\mathrm{kl}}(n,n)$ and only summing over those $\boldsymbol{k} \in S_{n,M}$ with non-zero probability, we obtain

$$\begin{aligned}\sum_{\boldsymbol{k}\in S_{n,M}} \mathrm{Mult}(\boldsymbol{k};n,M,\boldsymbol{r})\exp\Big(n\mathrm{kl}\big(\tfrac{\boldsymbol{k}}{n}\big\|\boldsymbol{r}\big)\Big) &= \sum_{\substack{\boldsymbol{k}\in S_{n,M}:\\ \forall j\in J_{\boldsymbol{r}}\ k_j=0}} \mathrm{Mult}(\boldsymbol{k};n,M,\boldsymbol{r})\exp\Big(n\mathrm{kl}\big(\tfrac{\boldsymbol{k}}{n}\big\|\boldsymbol{r}\big)\Big)\\ &= \sum_{\substack{\boldsymbol{k}\in S_{n,M}:\\ \forall j\in J_{\boldsymbol{r}}\ k_j=0}} \frac{n!}{\prod_{j\notin J_{\boldsymbol{r}}} k_j!}\prod_{j\notin J_{\boldsymbol{r}}} r_j^{k_j}\frac{1}{n^n}\prod_{j\notin J_{\boldsymbol{r}}}\left(\frac{k_j}{r_j}\right)^{k_j}\\ &= \frac{n!}{n^n}\sum_{\substack{\boldsymbol{k}\in S_{n,M}:\\ \forall j\in J_{\boldsymbol{r}}\ k_j=0}}\prod_{j\notin J_{\boldsymbol{r}}}\frac{k_j^{k_j}}{k_j!}\\ &= \frac{n!}{n^n}\sum_{\substack{\boldsymbol{k}\in S_{n,M}:\\ \forall j\in J_{\boldsymbol{r}}\ k_j=0}}\prod_{j=1}^{M}\frac{k_j^{k_j}}{k_j!} \qquad \text{(because } \tfrac{0^0}{0!}=1)\\ &\le \frac{n!}{n^n}\sum_{\boldsymbol{k}\in S_{n,M}}\prod_{j=1}^{M}\frac{k_j^{k_j}}{k_j!},\end{aligned}$$

which is $\eta(n,M)$. Since this is independent of $\boldsymbol{r}$, it also holds after taking the supremum over $\boldsymbol{r}\in\triangle_M$ of the left hand side, showing that $\mathcal{I}_{\mathrm{kl}}(n,n)\le\eta(n,M)$. □

The final step in obtaining Theorem 11 is to loosen the bound given in Corollary 1 (which is intractable when m is large) to the tractable form given in Theorem 11. For this we require the following technical lemma, the proof of which we defer to Appendix A.3.3.

Lemma 3. *For integers $M\ge 1$ and $n\ge M$,*

$$\sum_{\boldsymbol{k}\in S_{n,M}^{>0}}\frac{1}{\prod_{j=1}^{M}\sqrt{k_j}}\le\frac{\pi^{\frac{M}{2}}n^{\frac{M-2}{2}}}{\Gamma(\frac{M}{2})}.$$

Proof. (Of Theorem 11) It suffices to show that for all $n\ge M\ge 1$ we have $\eta(n,M)\le\xi(n,M)$. We achieve this by applying Stirling's approximation $\sqrt{2\pi n}\left(\frac{n}{e}\right)^n<n!<\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n}}$ (valid for $n\ge 1$) to the factorials in $\eta(n,M)$ and then using Lemma 3.

Since Stirling's approximation requires that all the k_j are at least one, we partition the sum in $\eta(n,M)$ according to the number of coordinates of $\boldsymbol{k}$ at which $k_j=0$. Let z index the number of such coordinates. Defining $f:\bigcup_{M=2}^{\infty}S_{n,M}\to\mathbb{R}$ by $f(\boldsymbol{k})=\prod_{j=1}^{|\boldsymbol{k}|}k_j^{k_j}/k_j!$ and noting that f is symmetric under permutations of its arguments, we then have

$$\eta(n,M)=\frac{n!}{n^n}\sum_{\boldsymbol{k}\in S_{n,M}}f(\boldsymbol{k})=\frac{n!}{n^n}\sum_{z=0}^{M-1}\binom{M}{z}\sum_{\boldsymbol{k}\in S_{n,M-z}^{>0}}f(\boldsymbol{k}). \tag{3.9}$$

Stirling's approximation can now be applied to each $\boldsymbol{k} \in S_{n,M}^{>0}$

$$f(\boldsymbol{k}) \leq \prod_{j=1}^{M} \frac{k_j^{k_j}}{\sqrt{2\pi k_j}\left(\frac{k_j}{e}\right)^{k_j}} = \prod_{j=1}^{M} \frac{e^{k_j}}{\sqrt{2\pi k_j}} = \frac{e^n}{(2\pi)^{M/2}} \prod_{j=1}^{M} \frac{1}{\sqrt{k_j}}.$$

An application of Lemma 3 now gives

$$\sum_{\boldsymbol{k}\in S_{n,M-z}^{>0}} f(\boldsymbol{k}) \leq \sum_{\boldsymbol{k}\in S_{n,M-z}^{>0}} \frac{e^n}{(2\pi)^{\frac{M-z}{2}}} \prod_{j=1}^{M-z} \frac{1}{\sqrt{k_j}} \leq \frac{e^n}{(2\pi)^{\frac{M-z}{2}}} \frac{\pi^{\frac{M-z}{2}} n^{\frac{M-z-2}{2}}}{\Gamma\left(\frac{M-z}{2}\right)} = \frac{e^n n^{\frac{M-z-2}{2}}}{2^{\frac{M-z}{2}}\Gamma\left(\frac{M-z}{2}\right)}.$$

Substituting this into equation (3.9) and bounding $n!$ using Stirling's approximation, we have

$$\eta(n, M) \leq \frac{\sqrt{2\pi n} e^{1/(12n)}}{e^n} \sum_{z=0}^{M-1} \binom{M}{z} \frac{e^n n^{\frac{M-z-2}{2}}}{2^{\frac{M-z}{2}}\Gamma\left(\frac{M-z}{2}\right)} = \xi(n, M),$$

which completes the proof of the bound. As for the order of the bound, it is sufficient to bound $\ln \xi(n, M)$ using the crude approximations $\binom{M}{z} \leq M^M$, $(2/n)^{z/2} \leq 1$ and $\Gamma((M-z)/2) \geq 1$.

$\square$

3.7 Numerical experiments

We use binarised versions of MNIST and HAM10000 (Tschandl et al., 2018). In both cases we partition $\mathcal{Y}^2$ into $E_0 = \{(0,0),(1,1)\}$, $E_1 = \{(0,1)\}$ and $E_2 = \{(1,0)\}$, and take $\boldsymbol{\ell} = (0,1,3)$. Each dataset is split into prior and certification sets. We take $\mathcal{H}$ to be the set of two-layer ReLU-activated MLPs. As is common in the PAC-Bayes literature, we restrict P and Q to be isotropic and diagonal Gaussian distributions over the parameter space, respectively. The mean of P is set to the parameters of an MLP trained on the prior set, and this is also taken as the initialisation of the means of Q. The mean and variance of Q and the variance of P are tuned via Theorem 12 to minimise the bound on the total risk $R_D^T(Q)$. See Appendix A.1 for pseudocode, Appendix A.2 for full experimental details and https://github.com/reubenadams/PAC-Bayes-Control for code. The results for MNIST can be seen in Figure 3.1.

We estimate $\boldsymbol{R}_S(Q)$ with a Monte Carlo approximation and obtain a PAC-Bayes bound on $R_D^T(Q)$ by combining Proposition 2 (with $\delta = 0.01$ and $N = 100,000$) and Proposition 3. We obtain $R_D^T(Q) \leq 0.2640$ for MNIST and $R_D^T(Q) \leq 0.8379$ for HAM10000, where both bounds hold with probability at least $1 - 0.05 - 0.01 = 0.94$. While these bounds are far from vacuous—the maximum possible value of $R_D^T(Q)$ is 3 for our choice of $\boldsymbol{\ell}$—one might wonder whether one can do better by bounding each error probability individually using Maurer's inequality, Maurer (2004), and then unioning these bounds. As with our Theorem 11, this would also constrain the entire distribution of error types since for any $\boldsymbol{\ell}$, one could then calculate the maximum value of $R_D^T(Q)$ that satisfies all of these constraints. Both methods constrain the region of the simplex in which $\boldsymbol{R}_D(Q)$ can lie (with high probability), and a reasonable metric by which to compare them is the volumes of these regions. This can be estimated via a Monte Carlo sample by uniformly sampling points $\boldsymbol{r}$ from $\triangle_M$ and counting how many are legal values of $\boldsymbol{R}_D(Q)$ according to each method. The 95% confidence intervals for the volumes of the two regions are given in Table 3.1. A more comprehensive table for synthetic values of $\boldsymbol{R}_S(Q)$ can be found in Appendix A.2.

Inspecting Figure 3.1a, we see that our training method successfully reduces the bound on the total risk $R_D^T(Q)$, with Figure 3.1b showing this is achieved by a reduction in the probabilities of error types E_1 and E_2. The fact that both of these error probabilities were able to decrease—implying an increase in the accuracy of the final stochastic classifier Q—is evidence that the additional data used to optimise the bound results in additional learning, in contrast to test set bounds.

While the bound on the total loss decreases, Figure 3.1c shows that the bound on the divergence $\mathrm{kl}(\boldsymbol{R}_S(Q)\|\boldsymbol{R}_D(Q))$ increases. This is to be expected—since Q is initialised to P, and P is independent of the data used to evaluate the bound, $\boldsymbol{R}_S(Q)$ is initially an unbiased estimate of $\boldsymbol{R}_D(Q)$, so $\boldsymbol{R}_S(Q)$ and $\boldsymbol{R}_D(Q)$ can be expected to lie near each other in the simplex. As training progresses however, Q becomes dependent on the data used to evaluate the bound, and so in the worst case, which we cannot rule out, $\boldsymbol{R}_S(Q)$ and $\boldsymbol{R}_D(Q)$ drift apart. Nevertheless, the fact that the increase in the bound on $\mathrm{kl}\big(\boldsymbol{R}_S(Q)\|\boldsymbol{R}_D(Q)\big)$ is modest, means that one maintains reasonably tight bounds on the total risk for all choices of risk vector $\boldsymbol{\ell}$ simultaneously, not just the one chosen for training.

It is worth emphasising that while one is forced to make a choice of risk vector $\boldsymbol{\ell}$ to optimise the bound via Theorem 12, the value of our method lies in the fact that it preserves bounds on all linear combinations of the error probabilities simultaneously. In contrast, if one were instead to straightforwardly apply Maurer's bound to the scalar total risk $R_D^T(Q)$ for a fixed choice of risk vector, and train the parameters of Q to minimise this bound, then bounding the total risk for a *second* risk vector would require a second bound and therefore a second sample. Our method dispenses with this requirement entirely.

Dataset	Volume Our Region	Volume Maurer Region
MNIST	**0.0025** (0.002498, 0.002504)	0.0028 (0.002793, 0.002800)
HAM10000	0.0012 (0.001207, 0.001211)	**0.0011** (0.001142, 0.001146)

Table 3.1: Point estimates and 95% confidence intervals for the volumes of the confidence regions for $\boldsymbol{R}_D(Q)$ given by Theorem 11 and a union over M individual Maurer bounds, respectively. Our method is superior for MNIST and inferior for HAM10000.

3.8 Conclusion

We introduce the framework of error types, considering the vectors $\boldsymbol{R}_S(Q)$ and $\boldsymbol{R}_D(Q)$ of empirical and true probabilities of errors of different types. We prove a PAC-Bayes bound (Theorem

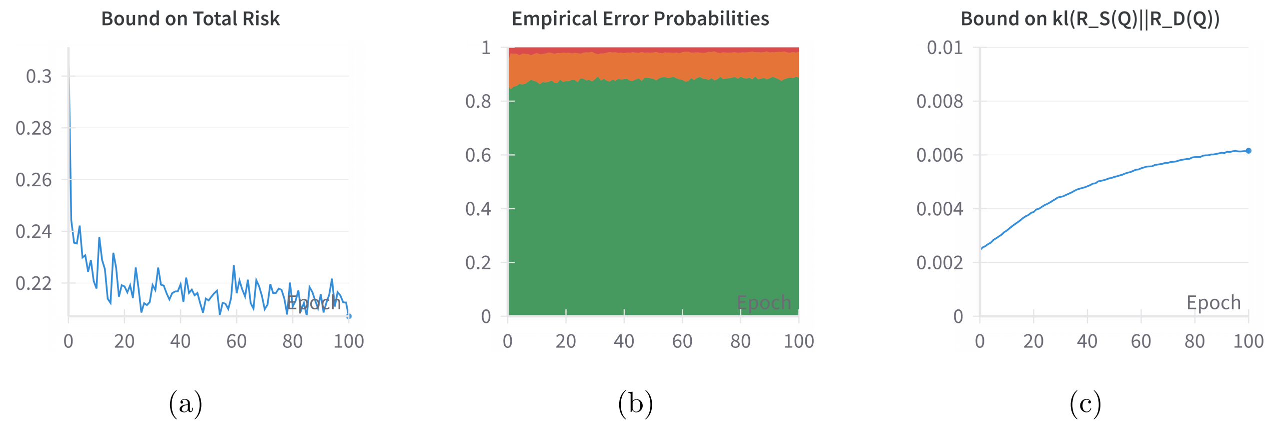


Figure 3.1: Experimental results for binarised MNIST. (a) The PAC-Bayes bound on the total risk decreases when tuning the posterior via Theorem 12. (b) This is achieved by a shift in the empirical error probabilities. (c) The bound on $\mathrm{kl}(\boldsymbol{R}_S(Q)\|\boldsymbol{R}_D(Q))$ is not substantially increased, meaning we still retain good control of $\boldsymbol{R}_D(Q)$ after optimising Q for this particular choice of $\boldsymbol{\ell}$.

11) on $\mathrm{kl}(\boldsymbol{R}_S(Q)\|\boldsymbol{R}_D(Q))$ which controls the entire distribution of error probabilities, and hence can be used to derive bounds on arbitrary linear combinations of the error probabilities, all of which hold simultaneously with high probability; this cannot be achieved with any existing PAC-Bayes bound.

We construct a differentiable training objective based on our bound by introducing the vectorised kl inverse, providing a recipe for quickly computing its value and derivatives (Theorem 12). Our framework is flexible enough to encompass multiclass classification or discretised regression, but also structured output prediction, multi-task learning and learning-to-learn.

Another potential application of our work is to the excess risk, since under a misclassification loss there are three different error types, corresponding to excess losses of $\{-1, 0, 1\}$. Biggs and Guedj (2023) adapted Theorems 11 and 12 to this setting, leading to an empirically tighter PAC-Bayes bound for certain classification tasks.

We require i.i.d. data, which in practice is frequently not the case or is hard to verify. Further, the number of error types M must be finite. In continuous scenarios, it would be preferable to be able to control the entire distribution of loss values without having to discretise into finitely many error types. We leave this direction to future work.

Chapter 4

PAC-Bayes and Compression

4.1 Introduction

Many papers have derived non-vacuous bounds for neural networks in the overparameterised regime. Relatively few of these works however explain the generalisation of neural networks trained through ordinary methods. Instead, they explain the generalisation of neural networks constructed with modified training methods inspired by their bounds—e.g. minimising a PAC-Bayes bound—or networks altered via pruning or some other form of compression. Further, the focus is usually on stochastic neural networks, in contrast to the deterministic neural networks most commonly used in practice.

For example, Dziugaite and Roy (2017) achieve non-vacuous bounds for a stochastic neural network trained to minimise the classical PAC-Bayes bound from McAllester (1999), which explains little about the generalisation of deterministic networks trained via ordinary methods. And Zhou et al. (2018) bound the performance of stochastic neural networks that have been heavily pruned, losing up to 98.5% of their parameters, which explains little about the deterministic unpruned networks. An exception is Nagarajan and Kolter (2019), which bounds uncompressed deterministic networks, but their bounds are orders of magnitude larger than one, making them uninformative in practice.

As argued in the introduction to this thesis, a mature statistical learning theory should not have to rely on such modifications. Rather, it should explain the empirically observed generalisation of deterministic and uncompressed neural networks trained without consideration to statistical learning theory. An alternative approach, which we take in this Chapter, is to search for a compression scheme that allows bounds on the compressed network to be rigorously carried over to the original deterministic network with minimal degradation.

Following Lotfi et al. (2022), we dispense with stochastic neural networks by employing PAC-Bayes theory with a posterior equal to a point mass on the deterministic network, and using a simplicity prior that puts exponentially more weight on networks that can be expressed in a smaller number of bits (for some fixed encoding scheme). This then yields a bound on the deterministic network in terms of the number of bits required to describe it. While such bounds are typically very loose when the bit description is simply the raw parameters of the network, the hope is that one can find a “nearby” network that also performs well while being expressible in fewer (or lower precision) parameters, and therefore fewer bits. This network would then enjoy a tighter bound and, if its performance can be related to the original network, this bound may be extended to the original network with minimal degradation.

The now old observation that neural networks trained via gradient descent tend to find flat minima (see e.g. Keskar et al. (2016)) makes the existence of a nearby compressible network plausible, as the flatter the minimum the larger the region in which the parameters can be varied without substantially increasing the loss. This suggests the compression scheme of weight quantisation, wherein the weights are either truncated or replaced with their closest centroid after an application of the k-means algorithm.

Building on Neyshabur et al. (2017b), in Lemma 4 we bound the discrepancy in output of the original and compressed networks in terms of the spectral norms of the differences in the weight matrices of the two networks. This bound naturally suggests low-rank approximations via truncation of the Singular Value Decompositions (SVDs) of the weight matrices as a compression scheme, since this yields the low-rank approximations closest in spectral norm (see the Eckart–Young theorem Eckart and Young (1936)). Further, in Lemma 5, we show that a small discrepancy in output of the two networks implies a small discrepancy in accuracy provided the margin of the compressed network is large. This is what allows us to transfer bounds from the compressed network to the original network in Theorem 14, our main result.

Informally, our result bounds the true error of the original network in terms of the empirical margin loss of its compression, plus a term that is $O(\sqrt{|s|/n})$ ignoring logarithmic terms, where $|s|$ is the length of the bit string representation of the compressed network, and the margin is smaller for higher fidelity compressions. That is, we show that if a network has large margin on the train set, and there exists a high-fidelity compression, then this network generalises.

Our theoretical results highlight the following tradeoff; as the degree of compression is increased, the tightness of the generalisation bound on the compressed network also increases (due to its short representation), but the performance of the compressed network is likely impacted and the degradation incurred by the bound when it is transferred to the original network also increases. It is the empirical balance between these factors that we investigate for different compression schemes in Sections 4.4 and 4.5.

4.2 Related work

Our approach in this work is inspired by Neyshabur et al. (2017b), which provides a bound on the error rate of an uncompressed deterministic network. They achieve this by proving a bound on the change in output of an MLP in terms of a perturbation to its weights, and then randomising over this perturbation in order to apply the PAC-Bayes theory. Our bound in Lemma 5 is an adaptation of their perturbation bound which dispenses with their technical restriction on the perturbation, instead relating any two MLPs of the same architecture. Our result also permits the MLPs to have bias terms. Further, they loosen their bound in several ways for aesthetics and do not test their bound empirically, whereas we trade aesthetics for tightness and conduct empirical experiments, in some cases finding non-vacuous bounds that would not be achieved without our method.

The work of Neyshabur et al. (2017b) was continued in Arora et al. (2018), where it was shown that the output of a trained neural network is more stable to noise injected into earlier rather than later layers. This allowed them to achieve tighter bounds on a stochastic compressed form of the network by applying more severe compression to earlier layers. Our compression approach also allows the severity of compression to vary across layers, but in contrast to their work, our result continues to bound the original deterministic network.

The first non-vacuous bounds for realistic architectures classifying ImageNet were established in Zhou et al. (2018), later improved by Lotfi et al. (2022). Both works employ a discrete prior over the hypothesis space, placing mass inversely proportional to the description length of the hypothesis encoded by their compression scheme. The bounds in Zhou et al. (2018) apply to networks compressed by weight pruning and quantisation. In the MNIST case they prune around 98.5% of the weights and quantise the non-zero weights with a 4-bit codebook. While the bounds are impressive, it is doubtful that the performance of the original and compressed networks can be related given their aggressive compression scheme, and so the results do not explain the empirically observed generalisation of the original networks. The tighter bounds found in Lotfi et al. (2022) suffer this drawback to a greater degree, as in addition they significantly modify the training procedure to ensure the final network is highly compressible.

The goal of the present Chapter is to evaluate whether the observed compressibility of networks trained via ordinary methods can be leveraged to tighten bounds on the original network, without the addition of stochasticity or data-dependent priors.

4.3 Theory

In this section we prove our main result, Theorem 15, a bound on the true error of an MLP classifier in terms of the empirical margin loss and string length of its compression, where the margin appearing in the bound depends on the discrepancy between the network and its compression.

We prove Theorem 15 in four steps. First, Lemma 4 bounds the maximum difference in output between the original and compressed networks. Second, Lemma 5 shows that if the maximum difference in output of two classification functions (not necessarily MLPs) is small, then the true margin loss of one can be bounded in terms of the true margin loss (with a different margin) of the other. Third, combining these two Lemmas yields Theorem 14, which states that the true error of the original network is bounded by the true margin loss of the compressed network, where the margin is a function of the two networks. Finally, Theorem 15 loosens the true error bound in Theorem 14 to be in terms of empirical quantities by applying a classic PAC-Bayes bound stated here as Theorem 13.

4.3.1 Discrete PAC-Bayes

We start with the classic PAC-Bayes theorem found in Maurer (2004), a form of which was originally proved in Langford and Seeger (2001). Originally stated for the zero-one loss, it applies to any bounded loss function, so we state it here in terms of the margin loss. For classification with labels $\mathcal{Y} = [k]$ and predictions $\hat{y} \in \hat{\mathcal{Y}} = \mathbb{R}^k$, the margin loss of a classifier $h : \mathcal{X} \to \mathbb{R}^k$ for any $\gamma \in \mathbb{R}$ is defined as

$$\ell_\gamma : \mathcal{H} \times (\mathcal{X} \times \mathcal{Y}) \to \{0,1\}, \quad \text{where} \quad \ell_\gamma(h,(x,y)) = \mathbb{1}\Big[h(x)_y \le \gamma + \max_{j \neq y} h(x)_j\Big], \tag{4.1}$$

namely a loss of 1 if and only if the classifier fails to predict correctly with a margin greater than γ. Note that ℓ_0 corresponds to the ordinary zero-one loss. For any data-generating distribution $D \in \triangle(\mathcal{X} \times \mathcal{Y})$, sample $S \in (\mathcal{X} \times \mathcal{Y})^n$ and margin $\gamma \in \mathbb{R}$, the true and empirical margin loss of h are then

$$\begin{aligned} R_D^\gamma(h) &= \mathbb{E}_{(x,y)\sim D}\Big[\ell_\gamma(h,(x,y))\Big] = \mathbb{P}_{(x,y)\sim D}\Big[h(x)_y \le \gamma + \max_{j\neq y} h(x)_j\Big] \quad \text{and} \\ R_S^\gamma(h) &= \frac{1}{n}\sum_{i=1}^n \ell_\gamma(h,(x_i,y_i)) = \frac{1}{n}\sum_{i=1}^n \mathbb{1}\Big[h(x_i)_{y_i} \le \gamma + \max_{j\neq y_i} h(x_i)_j\Big], \end{aligned}$$

respectively. Note that $R_D^\gamma(h)$ and $R_S^\gamma(h)$ are increasing in γ and are elements of $[0,1]$. Extending this to stochastic classifiers $Q \in \triangle(\mathcal{H})$, which classify according to a sampled $h \sim Q$, with a fresh sample for each classification, we have the true and empirical margin losses

$$R_D^\gamma(Q) = \mathbb{E}_{h\sim Q}\big[R_D^\gamma(h)\big] \quad \text{and} \quad R_S^\gamma(Q) = \mathbb{E}_{h\sim Q}\big[R_S^\gamma(h)\big],$$

respectively. In this setting Maurer's bound can be written as follows.

Theorem 13. *(Maurer (2004), Theorem 5) For any data-generating distribution $D \in \triangle(\mathcal{X}\times\mathcal{Y})$ with label set $\mathcal{Y} = [k]$, hypothesis class $\mathcal{H} \subseteq \hat{\mathcal{Y}}^{\mathcal{X}}$ with prediction space $\hat{\mathcal{Y}} = \mathbb{R}^k$, prior distribution $P \in \triangle(\mathcal{H})$, confidence level $\delta \in (0,1]$, margin $\gamma \ge 0$ and sample size n, with probability at least $1-\delta$ over the random draw $S \sim D^n$, we have that simultaneously for all posterior distributions $Q \in \triangle(\mathcal{H})$*

$$\mathrm{kl}\big(R_S^\gamma(Q) \big\| R_D^\gamma(Q)\big) \le \frac{\mathrm{KL}(Q\|P) + \ln\frac{2\sqrt{n}}{\delta}}{n}.$$

This establishes a bound on $R_D^\gamma(Q)$ by inverting the kl with $\mathrm{kl}^{-1}(q|B) = \sup\{p \in [0,1] : \mathrm{kl}(q\|p) \le B\}$. Alternatively, Pinsker's inequality gives $\mathrm{kl}(q\|p) \le B \implies p \le q + \sqrt{B/2}$. This second method allows for much easier comparison between methods if they happen to produce empirically loose bounds.

While data-dependent priors can be made admissible by splitting the dataset—a common technique in the literature (e.g. Dziugaite and Roy (2018), Parrado-Hernández et al. (2012), and Perez-Ortiz et al. (2021))—we eschew their use here as their utility in explaining generalisation is limited. As already discussed in depth in Section 2.5, the tight PAC-Bayes bounds achieved through data-dependent priors for the most part simply shifts the generalisation mystery to the unexplained success of the data-dependent prior itself. While this may not be a problem for self-certified learning, our purpose here is to shed light on the generalisation mystery.

Since we are seeking bounds on deterministic rather than stochastic classifiers, we follow Zhou et al. (2018) (and later Lotfi et al. (2022)) in taking the posterior Q to be a point mass on the hypothesis $h(S)$ produced by the learning algorithm on the sample S, so that $R^\gamma_D(Q) = R^\gamma_D(h(S))$. In order for $\mathrm{KL}(Q\|P)$ to be finite, this then demands that we take P to be a discrete distribution over a countable subset of P that includes $h(S)$. Since P may not depend on S, this in turn means that $h(S)$ must lie in a prespecified countable subset of $\mathcal{H}$, independently of S, at which point we have $\mathrm{KL}(Q\|P) = -\ln P(h)$. While this can be trivially achieved by noting that any hypothesis returned by a computer will be given to finite precision, and therefore comes from a finite set known in advance, this finite set may be so large that a uniform prior P would yield such a large $\mathrm{KL}(Q\|P)$ that the PAC-Bayesian theory produces trivial results. The goal of compression therefore is to limit the size of this finite set, so that P does not need to be spread so thinly. We discuss our compression schemes in Section 4.4.

4.3.2 Bounding the discrepancy in output between two MLPs

Here we bound the discrepancy in output between two Multi-Layer Perceptrons (MLPs) with ReLU activation functions. More specifically, we consider two MLPs of identical architecture (dimensions of the layers) differing in their weights only—their biases must be equal. Our bound in Lemma 4 then bounds the discrepancy in their output in terms of the spectral norms of the differences of their weight matrices, and the spectral norms of the weight matrices themselves.

We define a d layer MLP $h_{W,B} : \mathbb{R}^{k_0} \to \mathbb{R}^{k_d}$ with weights $W = (W_1, \dots, W_d) \in \mathbb{R}^{k_1 \times k_0} \times \cdots \times \mathbb{R}^{k_d \times k_{d-1}}$, biases $B = (B_1, \dots, B_d) \in \mathbb{R}^{k_1} \times \cdots \times \mathbb{R}^{k_d}$ and ReLU activation function $\phi(x) = \max(x, 0)$ recursively as follows

$$\begin{aligned} h^1_{W,B}(x) &:= W_1 x + B_1 \\ h^{i+1}_{W,B}(x) &:= W_{i+1}\phi(h^i_{W,B}(x)) + B_{i+1} \quad \text{for } i \geq 1. \end{aligned}$$

The MLP $h_{W,B}$ is then simply $h^d_{W,B}$. Let `MLP` denote the set of all MLPs with ReLU activations and $\texttt{MLP}_{(k_0,\dots,k_d)} \subseteq \texttt{MLP}$ those with input dimension k_0, output dimension k_d and hidden dimensions $k_1, \dots, k_{d-1}$.

Lemma 4 builds on Lemma 2 from Neyshabur et al. (2017b) but differs in two respects. First, we permit the MLPs to have bias terms (which is more realistic) provided they are identical. This is not a severe limitation, as bias terms are frequently left unchanged during compression as they make up a negligible proportion of the parameters. Second, since we are aiming for empirical tightness rather than a form of the bound that makes the order of terms most legible, we make as few relaxations as possible.

Our goal is to bound the change in the output of an MLP $h_{W,B}$ after the weights W are perturbed (by compression) to W', leaving the biases B fixed. More precisely, we seek a bound on

$$\sup_{x\in\mathbb{R}^{k_0}} \left\|h_{W,B}(x) - h_{W',B}(x)\right\|_2.$$

As it stands, this cannot succeed, as in general arbitrarily small changes in the weights can yield arbitrarily large changes in the output when the input $x \in \mathbb{R}^{k_0}$ is unbounded. We therefore stipulate that the input is bounded in ℓ_2-norm by some constant C.

Lemma 4. *Let $h_{W,B}$ and $h_{W',B}$ be two d-layer MLPs from $\mathbb{R}^{k_0}$ to $\mathbb{R}^{k_d}$ with ReLU activations and identical hidden layer dimensions. Fix $C \geq 0$ and let $\mathcal{X}_C = \{x \in \mathbb{R}^{k_0} : \|x\|_2 \leq C\}$. Then*

$$\sup_{x\in\mathcal{X}_C} \left\|h_{W,B}(x) - h_{W',B}(x)\right\|_2 \leq \beta(W, W', B, C),$$

where $\beta(W, W', B, C) = \beta_d(W, W', B, C)$ is defined recursively by

$$\alpha_0(W', B, C) = C \quad \text{and} \quad \beta_0(W, W', B, C) = 0$$

and, for $i \geq 1$,

$$\begin{aligned}\alpha_i(W', B, C) &= \|W'_i\|_2\alpha_{i-1}(W', B, C) + \|B_i\|_2\\ \beta_i(W, W', B, C) &= \|W_i\|_2\beta_{i-1}(W, W', B, C) + \|W_i - W'_i\|_2\alpha_{i-1}(W', B, C).\end{aligned}$$

Proof. We first show that $\sup_{x\in\mathcal{X}_C} \|h^i_{W',B}(x)\|_2 \leq \alpha_i(W', B, C)$ for $i = 1, \ldots, d$ by induction. For the base case, we have

$$\begin{aligned}\sup_{x\in\mathcal{X}_C} \|h^1_{W',B}(x)\|_2 &= \sup_{x\in\mathcal{X}_C} \|W'_1x + B_1\|_2\\ &\leq \sup_{x\in\mathcal{X}_C} \Big(\|W'_1\|_2\|x\|_2 + \|B_1\|_2\Big)\\ &\leq \|W'_1\|_2C + \|B_1\|_2 = \alpha_1(W', B, C).\end{aligned}$$

And for the induction,

$$\begin{aligned}\sup_{x\in\mathcal{X}_C} \|h^{i+1}_{W',B}(x)\|_2 &= \sup_{x\in\mathcal{X}_C} \|W'_{i+1}\phi(h^i_{W',B}(x)) + B_{i+1}\|_2\\ &\leq \sup_{x\in\mathcal{X}_C} \Big(\|W'_{i+1}\|_2\|\phi(h^i_{W',B}(x))\|_2 + \|B_{i+1}\|_2\Big)\\ &\leq \|W'_{i+1}\|_2 \sup_{x\in\mathcal{X}_C} \|h^i_{W',B}(x)\|_2 + \|B_{i+1}\|_2 \qquad (\text{using } |\phi(u)| \leq |u|)\\ &\leq \|W'_{i+1}\|_2\alpha_i(W', B, C) + \|B_{i+1}\|_2 \qquad (\text{by the inductive hypothesis})\\ &= \alpha_{i+1}(W', B, C).\end{aligned}$$

For the main result, let

$$\Delta_i(W, W', B, C) = \sup_{x\in\mathcal{X}_C} \|h^i_{W,B}(x) - h^i_{W',B}(x)\|_2$$

for $i = 1, \dots, d$. We will prove using induction that $\Delta_i(W, W', B, C) \le \beta_i(W, W', B, C)$ for $i = 1, \dots, d$. For the base case, we have

$$\begin{aligned}
\Delta_1(W, W', B, C) &= \sup_{x \in \mathcal{X}_C} \|h^1_{W,B}(x) - h^1_{W',B}(x)\|_2 \\
&= \sup_{x \in \mathcal{X}_C} \|W_1 x + B_1 - (W'_1 x + B_1)\|_2 \\
&= \sup_{x \in \mathcal{X}_C} \|(W_1 - W'_1)x\|_2 \\
&\le \|W_1 - W'_1\|_2 \sup_{x \in \mathcal{X}_C} \|x\|_2 \\
&= C\|W_1 - W'_1\|_2 \\
&= \beta_1(W, W', B, C).
\end{aligned}$$

And for the induction,

$$\begin{aligned}
\Delta_{i+1}(W, W', B, C) &= \sup_{x \in \mathcal{X}_C} \|h^{i+1}_{W,B}(x) - h^{i+1}_{W',B}(x)\|_2 \\
&= \sup_{x \in \mathcal{X}_C} \left\| W_{i+1}\phi(h^i_{W,B}(x)) + B_{i+1} - \left[W'_{i+1}\phi(h^i_{W',B}(x)) + B_{i+1}\right] \right\|_2 \\
&= \sup_{x \in \mathcal{X}_C} \left\| W_{i+1}[\phi(h^i_{W,B}(x)) - \phi(h^i_{W',B}(x))] + (W_{i+1} - W'_{i+1})\phi(h^i_{W',B}(x)) \right\|_2 \\
&\le \|W_{i+1}\|_2 \sup_{x \in \mathcal{X}_C} \|\phi(h^i_{W,B}(x)) - \phi(h^i_{W',B}(x))\|_2 \\
&\qquad + \|W_{i+1} - W'_{i+1}\|_2 \sup_{x \in \mathcal{X}_C} \|\phi(h^i_{W',B}(x))\|_2 \\
&\le \|W_{i+1}\|_2 \sup_{x \in \mathcal{X}_C} \|h^i_{W,B}(x) - h^i_{W',B}(x)\|_2 \\
&\qquad + \|W_{i+1} - W'_{i+1}\|_2 \sup_{x \in \mathcal{X}_C} \|h^i_{W',B}(x)\|_2 \qquad \text{(using } |\phi(u) - \phi(v)| \le |u - v|\text{)} \\
&\le \|W_{i+1}\|_2 \Delta_i(W, W', B, C) \\
&\qquad + \|W_{i+1} - W'_{i+1}\|_2 \sup_{x \in \mathcal{X}_C} \|h^i_{W',B}(x)\|_2 \qquad \text{(by definition of } \Delta_i\text{)} \\
&\le \|W_{i+1}\|_2 \beta_i(W, W', B, C) \\
&\qquad + \|W_{i+1} - W'_{i+1}\|_2 \alpha_i(W', B, C) \qquad \text{(by the inductive hypothesis)} \\
&= \beta_{i+1}(W, W', B, C).
\end{aligned}$$

□

4.3.3 Relating the margin loss of close classifiers

We have the following lemma, which converts a bound on the discrepancy between the output of two classifiers h, g to a bound on the true margin loss of h in terms of the true margin loss of g, but for a larger margin. It quantifies the intuition that if h and g have close output across the domain, then any input x correctly classified by g with large enough margin will also be correctly classified by h, though possibly with a smaller margin.

Lemma 5. *If* $\sup_{x\in\mathcal{X}} \|h(x) - g(x)\|_2 \leq \varepsilon$, *then for all* $\gamma \in \mathbb{R}$ *we have* $R_D^\gamma(h) \leq R_D^{\gamma+\sqrt{2}\varepsilon}(g)$.

Proof. Recall that the margin loss ℓ_γ defined by Equation (4.1) is $\{0,1\}$-valued. We will show that

$$\ell_\gamma\big(h,(x,y)\big) = 1 \implies \ell_{\gamma+\sqrt{2}\varepsilon}\big(g,(x,y)\big) = 1,$$

for arbitrary $(x,y) \in \mathcal{X} \times [k]$, after which the result follows by taking expectations over D.

Fix (x,y) and define

$$j_h = \operatorname{argmax}_{j\neq y} h(x)_j \quad \text{and} \quad j_g = \operatorname{argmax}_{j\neq y} g(x)_j.$$

Let $\boldsymbol{u} = \boldsymbol{e}_y - \boldsymbol{e}_{j_h}$, where $\boldsymbol{e}_j \in \mathbb{R}^k$ is the basis vector with 1 in coordinate j. Suppose that $\ell_\gamma(h,(x,y)) = 1$, i.e.

$$h(x)_y \leq \gamma + h(x)_{j_h}. \tag{4.2}$$

The Cauchy–Schwarz inequality gives $\boldsymbol{u}\cdot\big(g(x) - h(x)\big) \leq \|\boldsymbol{u}\|_2 \|g(x) - h(x)\|_2 \leq \sqrt{2}\varepsilon$, and the definition of j_h and j_g gives $g(x)_{j_h} - g(x)_{j_g} \leq 0$, whereupon

$$\begin{aligned}
g(x)_y - g(x)_{j_g} &= \big(g(x)_y - h(x)_y\big) + \big(h(x)_y - h(x)_{j_h}\big) + \big(h(x)_{j_h} - g(x)_{j_h}\big) + \big(g(x)_{j_h} - g(x)_{j_g}\big) \\
&= \boldsymbol{u}\cdot\big(g(x) - h(x)\big) + \big(h(x)_y - h(x)_{j_h}\big) + \big(g(x)_{j_h} - g(x)_{j_g}\big) \\
&\leq \sqrt{2}\varepsilon + \gamma + 0.
\end{aligned}$$

Rearranging and using the definition of j_g gives

$$g(x)_y \leq \gamma + \sqrt{2}\varepsilon + \max_{j\neq y} g(x)_j$$

and so $\ell_{\gamma+\sqrt{2}\varepsilon}\big(g,(x,y)\big) = 1$. Finally, by taking probabilities over $(x,y) \sim D$ we have

$$R_D^\gamma(h) = \mathbb{P}_{(x,y)\sim D}\Big[\ell_\gamma\big(h,(x,y)\big) = 1\Big] \leq \mathbb{P}_{(x,y)\sim D}\Big[\ell_{\gamma+\sqrt{2}\varepsilon}\big(g,(x,y)\big) = 1\Big] = R_D^{\gamma+\sqrt{2}\varepsilon}(g).$$

□

4.3.4 Bounding the error of an MLP in terms of the margin loss of its compression

Combining Lemmas 5 and 4 yields the following theorem.

Theorem 14. *Let* $h_{W,B}$ *and* $h_{W',B}$ *be two* d*-layer MLPs from* $\mathbb{R}^{k_0}$ *to* $\mathbb{R}^{k_d}$ *with ReLU activations and identical hidden layer dimensions. Fix* $C \geq 0$ *and let* $\mathcal{X}_C = \{x \in \mathbb{R}^{k_0} : \|x\|_2 \leq C\}$. *Then for any distribution* D *over* $\mathcal{X}_C \times \mathbb{R}^{k_d}$, *we have*

$$R_D^0(h_{W,B}) \leq R_D^{\sqrt{2}\beta(W,W',B,C)}(h_{W',B}),$$

where $\beta(W,W',B,C)$ *is defined as in Lemma 4.*

Proof. Immediate upon substituting Lemma 4 into Lemma 5, with domain $\mathcal{X} = \mathcal{X}_C$, classifiers $h = h_{W,B},\ g = h_{W',B}$, margin $\gamma = 0$ and $\varepsilon = \beta(W, W', B, C)$. □

4.3.5 A PAC-Bayes compression bound

We now define a *(weights-only) compression scheme.* We focus on weights-only compression schemes as the biases make up only a small proportion of the parameters. All of the compression schemes we discuss in Section 4.4 fit Definition 9.

Definition 9. A *compression scheme* is any pair of functions $\texttt{Encode} : \texttt{MLP}_{(k_{\text{in}},\dots,k_{\text{out}})} \to \{0,1\}^b$ and $\texttt{Decode} : \{0,1\}^b \to \texttt{MLP}_{(k_{\text{in}},\dots,k_{\text{out}})}$. $\texttt{Encode}$ and $\texttt{Decode}$ need not be inverses—lossy compression is permitted. The compression scheme is termed a *weights-only compression scheme* if the reconstruction $h_{W',B'} = \texttt{Decode}(\texttt{Encode}(h_{W,B}))$ is always such that $B' = B$. We call the bit string $\texttt{Encode}(h_{W,B})$ the *compressed representation* of $h_{W,B}$.

While our main result below appears at first sight to be a straightforward combination of Theorem 13 and Theorem 14, note that Theorem 13 requires a fixed margin while in Theorem 14 the margin depends on the MLP $h_{W,B}$ and its compression $h_{W',B}$, which, in the context of Theorem 15 are data-dependent. We get around this data-dependence of the required margin by augmenting the hypothesis space $\texttt{MLP}$ with an arbitrary margin. The margin will be represented by a single 32 bit float which we will include in the bit string representations. To that end, let $r : \{0,1\}^{32} \to \mathbb{R}$ be the function mapping the string representations of 32 bit floats to the corresponding real numbers, so that $\Gamma = \{r(s_1) : s_1 \in \{0,1\}^n\} \subseteq \mathbb{R}$ is the set of reals representable by a single 32 bit float.

Theorem 15. *(Main result) Fix a weights-only compression scheme* $\texttt{Encode} : \texttt{MLP}_{(k_{\text{in}},\dots,k_{\text{out}})} \to \{0,1\}^b$ *and* $\texttt{Decode} : \{0,1\}^b \to \texttt{MLP}_{(k_{\text{in}},\dots,k_{\text{out}})}$. *Fix* $C \geq 0$ *and let* $\mathcal{X}_C = \{x \in \mathbb{R}^{k_{\text{in}}} : \|x\|_2 \leq C\}$. *For any distribution* D *over* $\mathcal{X}_C \times [k_{\text{out}}]$, *confidence level* $\delta \in (0,1]$ *and sample size* n, *with probability at least* $1-\delta$ *over the random draw* $S \sim D^n$, *we have that simultaneously for all neural networks* $h_{W,B} \in \texttt{MLP}_{(k_{\text{in}},\dots,k_{\text{out}})}$, *both*

$$R^0_D(h_{W,B}) \leq \mathrm{kl}^{-1}\big(R^{\gamma^*}_S(h_{W',B})\big|\zeta\big) \quad \textit{and} \quad R^0_D(h_{W,B}) \leq R^{\gamma^*}_S(h_{W',B}) + \sqrt{\zeta/2},$$

where

$$\begin{aligned} h_{W',B} &= \texttt{Decode}\big(\texttt{Encode}(h_{W,B})\big), \\ \gamma^* &= \min\Big\{\gamma \in \Gamma : \gamma \geq \sqrt{2}\beta(W, W', B, C)\Big\}, \\ \zeta &= \frac{1}{n}\left((b+32)\ln 2 + \ln\frac{2\sqrt{n}}{\delta}\right), \end{aligned}$$

with $\beta(W, W', B, C)$ *defined as in Lemma 4.*

Proof. We apply Theorem 13 with fixed margin $\gamma = 1$, $\mathcal{X} = \mathcal{X}_C$, hypothesis class

$$\mathcal{H} = \left\{g_{s_1,s_2} : s_1 \in \{0,1\}^b, s_2 \in \{0,1\}^{32}\right\}, \quad g_{s_1,s_2}(x) := \frac{1}{r(s_2)}\texttt{Decode}(s_1)(x),$$

and prior

$$P(g_{s_1,s_2}) = 2^{-(b+32)}|\{g \in \mathcal{H} : g = g_{s_1,s_2}\}|,$$

namely the push-forward measure of a uniform distribution on $\{0,1\}^b \times \{0,1\}^{32}$ (we make this choice as the $g_{s_1,s_2} \in \mathcal{H}$ may not all be unique). In particular, this gives that with probability at least $1-\delta$, simultaneously for all point mass posteriors Q on $g_{s_1,s_2} \in \mathcal{H}$, we have

$$\mathrm{kl}\big(R^1_S(g_{s_1,s_2})\big\|R^1_D(g_{s_1,s_2})\big) \leq \frac{1}{n}\left(\mathrm{KL}(Q\|P) + \ln\frac{2\sqrt{n}}{\delta}\right) \leq \frac{1}{n}\left((b+32)\ln 2 + \ln\frac{2\sqrt{n}}{\delta}\right) = \zeta, \tag{4.3}$$

where the first inequality comes from Theorem 13, the second by noting that $\mathrm{KL}(Q\|P) = -\ln P(g_{s_1,s_2}) \leq (b+32)\ln 2$, and the final equality from the definition of ζ.

Now for any $h_{W,B} \in \texttt{MLP}_{(k_{\mathrm{in}},\dots,k_{\mathrm{out}})}$, we have by Theorem 14 that

$$R^0_D(h_{W,B}) \leq R_D^{\sqrt{2}\beta(W,W',B,C)}(h_{W',B}) \leq R^{\gamma^*}_D(h_{W',B}), \tag{4.4}$$

where the second inequality is from the fact that $R^\gamma_D(\cdot)$ is monotonically increasing in γ. Noting that

$$\tfrac{1}{\gamma^*}h_{W',B} = g_{s_1,s_2} \in \mathcal{H} \tag{4.5}$$

for $s_1 = \texttt{Encode}(h_{W,B})$ and $s_2 = r^{-1}(\gamma^*)$, and recalling that $h_{W,B}$ was arbitrary, we can substitute (4.5) into Inequality (4.3) to see that with probability at least $1-\delta$, simultaneously for all $h_{W,B} \in \texttt{MLP}_{(k_{\mathrm{in}},\dots k_{\mathrm{out}})}$,

$$\mathrm{kl}\left(R^{\gamma^*}_S(h_{W',B})\middle\|R^{\gamma^*}_D(h_{W',B})\right) = \mathrm{kl}\left(R^1_S\big(\tfrac{1}{\gamma^*}h_{W',B}\big)\middle\|R^1_D\big(\tfrac{1}{\gamma^*}h_{W',B}\big)\right) \leq \zeta,$$

and so

$$R^{\gamma^*}_D(h_{W',B}) \leq \mathrm{kl}^{-1}\big(R^{\gamma^*}_S(h_{W',B})\big|\zeta\big) \quad \text{and} \quad R^{\gamma^*}_D(h_{W',B}) \leq R^{\gamma^*}_S(h_{W',B}) + \sqrt{\zeta/2}, \tag{4.6}$$

where the two inequalities follow by inverting the kl and applying Pinsker's inequality, respectively. The result then follows by chaining Inequalities (4.6) with Inequality (4.4). □

4.4 Compression schemes

We consider six compression schemes, each compressing only the weights and leaving the biases alone:

1. weight quantisation by an application of k-means,
2. weight quantisation by truncation of the weights,

3. low-rank approximation,

4. low-rank approximation combined with quantisation of the decomposed matrices via k-means,

5. low-rank approximation combined with quantisation of the decomposed matrices via truncation,

6. no compression.

Where the final null compression scheme serves as a baseline. We also attempted compression by training a hypernetwork to reproduce the weights of the original network, including tricks such as predicting the bits in the binary representations of the weights rather than their scalar values. This approach failed however as the size of hypernetwork required to reproduce the weights of the original network within a reasonable tolerance was always larger than the original network.

For each of these six compression schemes, we take a network $h_{W,B} \in \mathtt{MLP}_{(k_0,\dots,k_d)}$ and return a compressed network $h_{W',B} \in \mathtt{MLP}_{(k_0,\dots,k_d)}$ of the same architecture, representable by a string $s_{W',B}$ of length $|s_{W',B}|$. Since $|s_{W',B}|$ will depend only on the architecture of $h_{W,B}$ (rather than its specific weights and biases) and the compression scheme, in each case we can take the prior P to be uniform over all strings of length $|s_{W',B}|$, which yields (as discussed in Section 4.3.1)

$$\mathrm{KL}(Q\|P) = -\ln P(h_{W',B}) = -\ln 2^{-|s_{W',B}|} = |s_{W',B}| \ln 2. \tag{4.7}$$

For the baseline approach of no compression, we represent $h_{W,B}$ by its raw bits, forming a string of length

$$|s_{W,B}| = 32\big(|\mathrm{vec}(W)| + |\mathrm{vec}(B)|\big) = 32\sum_{i=1}^{d} k_i k_{i-1} + k_i.$$

A uniform prior P over all strings of this length then gives $\mathrm{KL}(Q\|P) = |s_{W,B}|\ln 2$. The lengths of the strings for the six different compression schemes are summarised in Table 4.1.

4.4.1 Quantisation via k-means

To compress an MLP $h_{W,B}$ via quantisation with k-means, we run the k-means algorithm with $k = 2^c$ on $\mathrm{vec}(W)$, replacing each weight with its nearest centroid to form the compressed MLP $h_{W^c,B}$. We can then form a codebook of the centroids using 2^c bit strings of length 32. Thus $h_{W^c,B}$ can be represented by a bit string $s_{W^c,B}$ of length $|s_{W^c,B}| = c|\mathrm{vec}(W)| + 32 \cdot 2^c$, made up from a bit string of length $c|\mathrm{vec}(W)|$ encoding the compressed weights W^c in terms of their associated centroids, and a bit string of length $32 \cdot 2^c$ encoding the 2^c centroids themselves to 32-bit precision. Note this is only an improvement if $c|\mathrm{vec}(W)| + 32 \cdot 2^c \leq 32|\mathrm{vec}(W)|$, so we restrict ourselves to such values of c. In fact, due to the computational cost of running the k-means algorithm with a large number of centroids $k = 2^c$, we also restrict to $c \leq 10$. We then

Quantisation Method	**String Length of Compressed Model**	
	Full Rank	**Low Rank**
None	$32\sum_{i=1}^{d} k_i k_{i-1} + k_i$	$32\sum_{i=1}^{d}(k_i r_i + r_i + r_i k_{i-1} + k_i)$
k-means	$c\sum_{i=1}^{d} k_i k_{i-1} + 32\sum_{i=1}^{d} k_i + 32 \cdot 2^c$	$c\sum_{i=1}^{d}(k_i r_i + r_i k_{i-1}) + 32\sum_{i=1}^{d}(r_i + k_i) + 32 \cdot 2^c$
Truncation	$(1 + b_e + b_m)\sum_{i=1}^{d} k_i k_{i-1} + 32\sum_{i=1}^{d} k_i$	$(1 + b_e + b_m)\sum_{i=1}^{d}(k_i r_i + r_i k_{i-1}) + 32\sum_{i=1}^{d}(r_i + k_i)$

Table 4.1: String lengths (in bits) for different neural network compression schemes. Parameters: k_i (layer dimensions), r_i (low-rank approximation ranks), c (bits per weight in k-means quantisation with 2^c centroids), and b_e, b_m (bits for exponent and mantissa in truncation quantisation). In each case, the prior is taken to be uniform over strings of the given length, so that the corresponding $\mathrm{KL}(Q\|P)$ is obtained by adding 32 (to account for the 32 extra bits required for the margin γ^* defined in Theorem 15) and then multiplying by $\ln 2$ as in Equation 4.7.

have

$$\begin{aligned}|s_{W^c,B}| &= c|\mathrm{vec}(W)| + 32|\mathrm{vec}(B)| + 32 \cdot 2^c \\ &= c\sum_{i=1}^{d} k_i k_{i-1} + 32\sum_{i=1}^{d} k_i + 32 \cdot 2^c.\end{aligned}$$

Since $|s_{W^c,B}|$ depends only on the architecture and the number of centroids 2^c of the k-means algorithm, both of which we will specify in advance, we can use a prior P_c taken to be uniform over all bit strings of this length, which yields $\mathrm{KL}(Q\|P_c) = |s_{W^c,B}| \ln 2$ as in Equation 4.7.

4.4.2 Quantisation via truncation

Recall that the weights of the neural network are typically stored as 32-bit floating point numbers which, as per the IEEE Standard for Floating-Point Arithmetic (IEEE 754) have 1 sign bit, 8 exponent bits and 23 mantissa bits. To quantise an MLP $h_{W,B}$ by truncation, for any $b_m \in \{0, 1, \dots, 23\}$ we zero out the $23 - b_m$ least significant bits of the mantissa for each weight, so that each weight takes only $1 + 8 + b_m$ bits to represent rather than the usual 32. This minimally changes the value of each weight, taking advantage of the possible flatness of the minimum located by SGD.

Optionally, for $b_e \in \{0, \dots, 8\}$ we also clip the exponent of each weight so that it takes b_e bits to represent and the whole weight takes $1 + b_e + b_m$ bits. As before, we leave the biases alone. Clipping the exponent is a bit more involved, partly because exponents are biased by adding 127 in IEEE 754. Intuitively, we clip the (unbiased) exponents to be closer to zero, which is unlikely to have a significant effect on the output of the network for two reasons:

1. Very large positive (unbiased) exponents correspond to extremely large weights, which

are rarely observed in ML.

2. Very large negative (unbiased) exponents correspond to extremely small weights, and changing these to be merely very small will have a minimal effect.

Precisely, we clip the exponents as follows; we unbias the exponents by subtracting 127, clip them to the range $[-(2^{b_e-1}-1), 2^{b_e-1}-1]$, rebias by adding 127, and then reinstate exponents originally containing all zeros bits so that zeros and subnormal numbers are preserved. In the particular case $b_e = 0$ we set all unbiased exponents to zero as this preserves the best range of values for the weights. After this process the new exponent takes on one of 2^{b_e} values (easy check) and so can be represented by b_e bits. As an example of the second point above, in the case $b_e = 6$ this process maps the unbiased exponent -61 to -31, meaning extremely small weights of order 2^{-61} are compressed to still very small weights of order 2^{-31}.

Under this compression scheme we have

$$\begin{aligned}|s_{W^{b_e,b_m},B}| &= (1+b_e+b_m)|\mathrm{vec}(W)| + 32|\mathrm{vec}(B)| \\ &= (1+b_e+b_m)\sum_{i=1}^{d} k_i k_{i-1} + 32\sum_{i=1}^{d} k_i.\end{aligned}$$

As before, a uniform prior P_{b_e,b_m} over strings of this length gives $\mathrm{KL}(Q\|P_{b_e,b_m}) = |s_{W^{b_e,b_m},B}|\ln 2$.

4.4.3 Low-rank approximation

As for compression via low-rank approximation, for a given tuple of ranks $\boldsymbol{r} = (r_1,\ldots,r_d) \in \mathbb{N}^d$ we form the compression $W^{\boldsymbol{r}}$ as follows. For each i we take the singular value decomposition $W_i = U_i S_i V_i^\top$ and form the low-rank approximation $W_i^{\boldsymbol{r}} = U_i' S_i' {V'}_i^\top$ by taking the first r_i columns of U_i and S_i, and the first r_i rows of V_i. While $W_i \in \mathbb{R}^{k_i \times k_{i-1}}$ is made up from $k_i k_{i-1}$ parameters, the decomposition of $W_i^{\boldsymbol{r}}$ contains $k_i r_i + r_i + r_i k_{i-1}$ values. Since this is a genuine compression only if $r_i \leq k_i k_{i-1}/(k_i + 1 + k_{i-1})$, we restrict ourselves to choices of $\boldsymbol{r}$ for which this is the case for all i. As with quantisation, we specify the architecture and $\boldsymbol{r}$ in advance, so that we can represent $h_{W',B}$ by a bit string $s_{W^{\boldsymbol{r}},B}$ of length

$$|s_{W^{\boldsymbol{r}},B}| = 32\sum_{i=1}^{d}(k_i r_i + r_i + r_i k_{i-1} + k_i),$$

since layer i consists of $k_i r_i$ values in U_i', r_i values in S_i', $r_i k_{i-1}$ values in V_i' and k_i bias values, each requiring 32 bits. As $|s_{W^{\boldsymbol{r}},B}|$ depends only on the architecture and $\boldsymbol{r}$, both of which we will specify in advance, we can use a uniform prior $P_{\boldsymbol{r}}$ over all bit strings of this length to get $\mathrm{KL}(Q\|P_{\boldsymbol{r}}) = |s_{W^{\boldsymbol{r}},B}|\ln 2$.

4.4.4 Combined approaches

Finally, we combine low-rank approximation with the two quantisation methods by first taking the low-rank approximation and then quantising the values of the U_i''s and V_i''s for each layer, leaving the S_i''s and the biases unchanged as they make up a negligible proportion of the parameters. For low-rank approximation combined with quantisation via k-means with 2^c centroids, this produces a bit string of length

$$|s_{W^{\boldsymbol{r},c},B}| = c\sum_{i=1}^{d}(k_i r_i + r_i k_{i-1}) + 32\sum_{i=1}^{d}(r_i + k_i) + 32 \cdot 2^c,$$

and, again with a uniform prior $P_{\boldsymbol{r},c}$, we have $\mathrm{KL}(Q \| P_{\boldsymbol{r},c}) = |s_{W^{\boldsymbol{r},c},B}| \ln 2$.

For low-rank approximation combined with quantisation via truncation to b_e exponent bits and b_m mantissa bits, we have

$$|s_{W^{\boldsymbol{r},b_e,b_m},B}| = (1 + b_e + b_m)\sum_{i=1}^{d}(k_i r_i + r_i k_{i-1}) + 32\sum_{i=1}^{d}(r_i + k_i),$$

and, with a uniform prior $P_{\boldsymbol{r},b_e,b_m}$, $\mathrm{KL}(Q \| P_{\boldsymbol{r},b_e,b_m}) = |s_{W^{\boldsymbol{r},b_e,b_m},B}| \ln 2$.

4.5 Experiments

To empirically evaluate whether our compression approaches can tighten PAC-Bayes bounds, we train multiple MLPs on the MNIST1D dataset proposed in Greydanus and Kobak (2020). MNIST1D is a procedurally generated, reasonably low-dimensional dataset (40 dimensions rather than MNIST's 784) that replicates many of the features of deep learning (e.g. double descent, the existence of lottery tickets etc.). We opt for this low-dimensional dataset rather than the more familiar MNIST or CIFAR-10 datasets as it is more likely that for smaller networks the compression required to obtain non-vacuous PAC-Bayes bounds using a discrete prior is not so much that the outputs of the compression diverge significantly from the original network. This is the requirement for Theorem 15 to produce good results.

We generate train and test samples of size $50,000$ and $10,000$ respectively, using the same dataset for training all of the MLPs. Since part of the procedural generation is addition of Gaussian noise, the support of the data-generating distribution is unbounded. Thus we clip the data to $[-4, 4]$ in each dimension so that Theorem 14 can be applied with $C = C_{\text{domain}} = 8\sqrt{40} \approx 50.6$. Since the support of the data-generating distribution may in fact be contained in a much smaller ℓ_2-ball, we also experimented with setting $C = C_{\text{data}} = \max\{\|x\|_2 : x \in S\}$, where S is the training sample. While this is only a hypothetical result rather than a rigorous bound, it gives an indication of whether future work could improve our results by rigorously bounding C over the support of the data-generating distribution. Interestingly, we found that it produced only negligible changes in our results, showing that reducing C is not the bottleneck for tighter bounds using this approach.

The MLPs we train have $1, 2, 3$ or 4 hidden dimensions of equal width, where the width is in $\{4, 8, 16, \ldots, 512\}$. Each is trained with a batch size of 128 using the Adam optimiser with a learning rate of 0.001. All models are trained to convergence, namely until the train loss on the entire dataset stops decreasing.

We conduct three experiments in the choice of quantisation level k and ranks $\boldsymbol{r}$;

1. quantisation only, taking a union bound over $k \in \{2^1, \ldots, 2^c\}$, where c is the maximum value resulting in a genuine compression;

2. low-rank approximation only, taking a union bound over $\{\boldsymbol{r} \in \mathbb{N} : \forall i \ \ r_i \leq k_i k_{i-1}/(k_i + k_{i-1})\}$;

3. low-rank approximation and quantisation, taking a union bound over both $\{\boldsymbol{r} \in \mathbb{N} : \forall i \ \ r_i \leq k_i k_{i-1}/(k_i + k_{i-1})\}$ and then, for each $\boldsymbol{r}$, the values of k resulting in a genuine compression.

Although trained and compressed 32 networks in total (4 values for the number of layers, times 8 values for the hidden layer width), we illustrate the effects of the compression schemes by focusing on the network $h_{W,B} \in \texttt{MLP}_{(40,32,32,10)}$ with two hidden layers each of width 32 trained on the MNIST1D dataset.

4.5.1 Quantisation via k-means

Recall that for quantisation via k-means with $k = 2^c$ centroids, the codebook itself requires $32 \cdot 2^c$ bits to represent. Since many of the networks we evaluate are small compared to those commonly trained on real-world tasks, values of c larger than around 10 do not in fact result in a compression representation as the size of the codebook itself is too large. Unfortunately, as can be seen from Figure 4.1, it is only at around this level of c that the margin loss $R_S^{\gamma^*}(h_{W',B})$ from Theorem 15 becomes non-trivial, which is in turn necessary for the error bounds to become non-trivial. In Appendix B we show that this compression scheme does produce non-vacuous bounds tighter than for the uncompressed model for one of the 32 models we train, but only just.

We conclude therefore that this compression scheme is only likely to result in reduced error bounds for larger models, since then one can choose the number of centroids to be large enough such that the compressed network $h_{W',B}$ is close to the original network $h_{W,B}$. However, while reduced, such bounds are likely to still be vacuous (or near-vacuous) using the current discrete PAC-Bayes approach.

4.5.2 Quantisation via truncation

This is the most successful compression scheme for the network $h_{W,B} \in \texttt{MLP}_{(40,32,32,10)}$ we focus on. Indeed, it is the only compression scheme that that successfully reduces the error bounds, as seen in Figure 4.2. The success is partly due to the fact that it is the only compression scheme

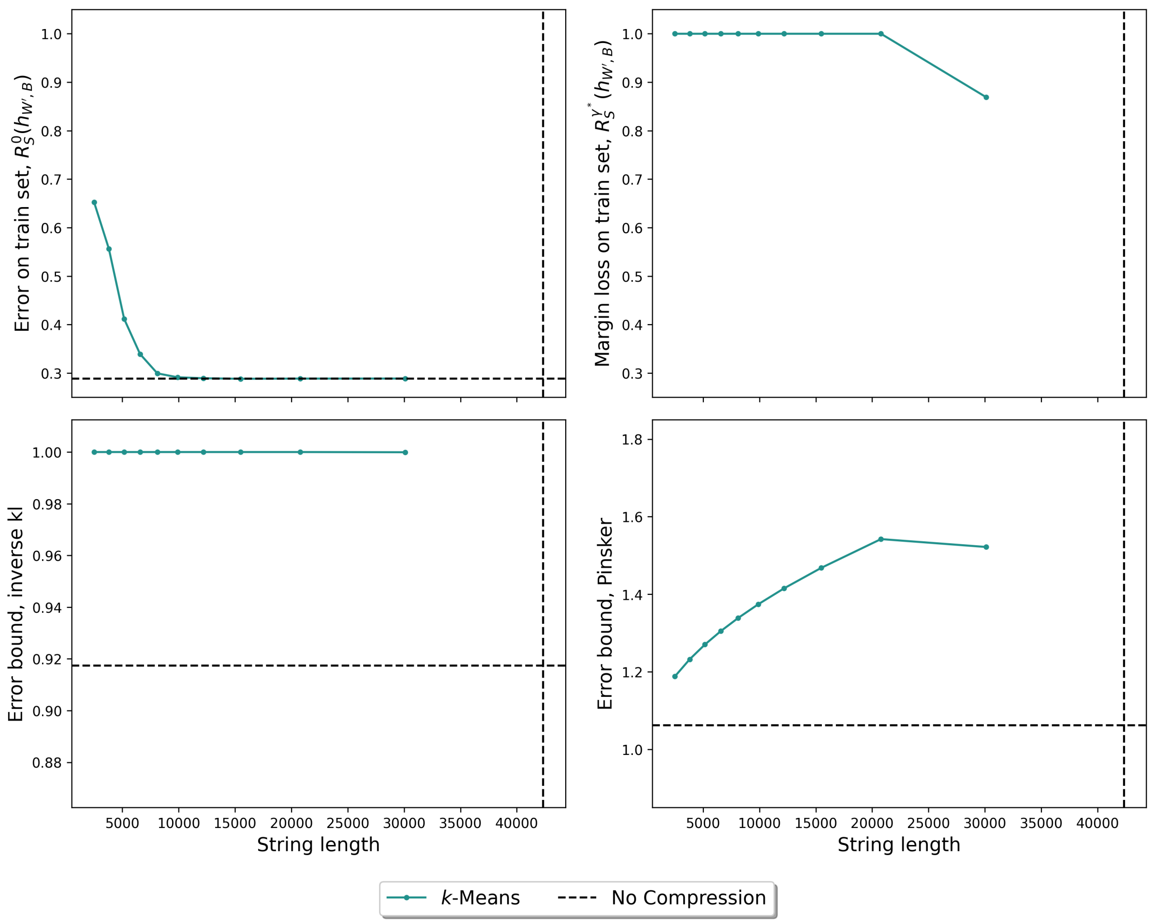


Figure 4.1: **Effects of quantisation via k-means** on a network $h_{W,B} \in \texttt{MLP}_{(40,32,32,10)}$ trained on $50,000$ samples from MNIST1D, where $k \in \{2, 4, \ldots, 1024\}$. Values of k exceeding 1024 are not used as they produce "compressed" representations larger than the original uncompressed representation. String length increases monotonically in k; the dotted vertical line representing the string length of the uncompressed network. Top left: As k increases the error of the compressed network converges to the error of the original (horizontal dotted line). Top right: The margin loss of the compressed network remains trivial until $k = 1024$. Bottom left: The large margin loss of the compressed network yields essentially trivial error bounds (when inverting the kl) on the original network which always exceed the discrete PAC-Bayes bound without compression (horizontal dotted line). Bottom right: The error bounds (when using Pinsker's inequality) increase with k up until $k = 10$ where there is a small decrease, but k cannot increase further; they all exceed the discrete PAC-Bayes bound without compression (horizontal dotted line).

that can achieve compression factors close to one, i.e. only slight compression; quantisation via k-means requires a growing codebook which prevents close approximation of the uncompressed network since the size of the codebook eventually makes the compressed representation larger than the original. And low-rank approximation cannot accommodate values of $\boldsymbol{r}$ near full rank as this also would increase rather than decrease string length because one ends up storing more rather than fewer parameters. Conversely, with quantisation via truncation one may choose values of b_e and b_m all the way up to their maxima without increasing the string length, permitting very close approximations.

Figure 4.2 demonstrates the benefit of this compression scheme in reduction of the error bounds when inverting the kl or using Pinsker's inequality. It also shows a clear trade-off in the amount of compression. Aggressive compression yields poor bounds as the compressed network has high margin loss (both because the compressed network performs poorly—$R^0_S(h_{W',B})$ is large—and the required margin γ^* from Theorem 15 is large), which outweighs the advantage of the low KL (recall the KL is proportional to string length). Conversely, mild compression yields poor bounds as the compressed network has high KL, which outweighs the advantage of the small margin loss. An optimal compression factor lies in the middle.

For our example network $h_{W,B} \in \texttt{MLP}_{(40,32,32,10)}$, both error bounds are minimised when $b_e = 5, b_m = 16$, at which point the error bound with inverse kl is 0.8723—reduced from 0.9174 in the uncompressed case—and the error bound with Pinsker's inequality is 0.9592—reduced from 1.0625 in the uncompressed case. We note that in the case of Pinsker's inequality, compression made the difference between a vacuous and a non-vacuous bound.

The success of this compression scheme was not limited to the network $h_{W,B} \in \texttt{MLP}_{(40,32,32,10)}$, so we show the full results for all 32 models in Table 4.2. We see that the bounds produced by inverting the small kl are improved for almost all of the 32 models. The bounds produced by Pinsker's inequality are improved for every model and in a number of cases compression makes the difference between a vacuous and a non-vacuous bound.

4.5.3 Low-rank approximation

Recall that the low-rank approximation of the weight matrix $W_i \in \mathbb{R}^{k_i \times k_{i-1}}$ to rank r_i only results in a reduction in the number of parameters if $r_i \leq k_i k_{i-1}/(k_i + 1 + k_{i-1})$. Since this is usually much less than $\min\{k_i, k_{i-1}\}$—for example if $W_i \in \mathbb{R}^{32\times 32}$ then we have $r_i \leq 15$—this compression scheme is restricted in how well the compressed network can approximate the uncompressed network unless the weight matrices of the uncompressed network are already approximately low-rank.

This is made clear in Figure 4.3, where we see that compression via low-rank approximation leads to significant deterioration in the error of our network $h_{W,B} \in \texttt{MLP}_{(40,32,32,10)}$ for all permissible values of $\boldsymbol{r}$, namely for all values that lead to a reduction in string length. Indeed, the lowest error $R^0_S(h_{W',B})$ achieved on the train set is 0.6199, far above the error of the uncompressed network which is 0.2887. Further, the margin loss $R^{\gamma^*}_S(h_{W',B})$ is trivial (equal to one) for all values of $\boldsymbol{r}$, reflecting the fact that margin γ^* required by Theorem 15 is too large.

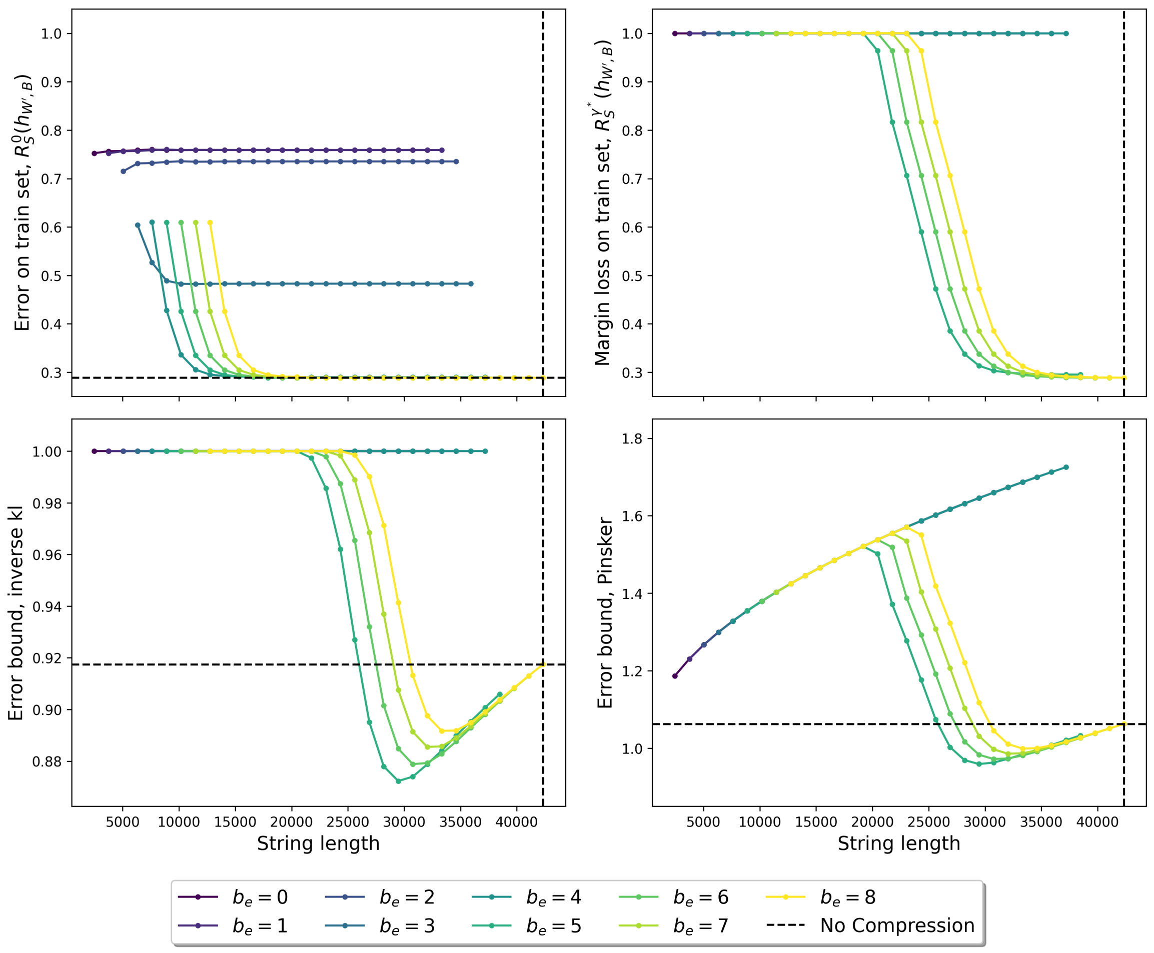


Figure 4.2: **Effects of quantisation via truncation** on a network $h_{W,B} \in \texttt{MLP}_{(40,32,32,10)}$ trained on 50,000 samples from MNIST1D. String length increases monotonically with both the number of exponent and mantissa bits, b_e and b_m respectively. The vertical dotted lines represent the string length of the uncompressed network. Top left: As both b_e and b_m increase the error of the compressed network converges to that of the original (horizontal dotted line). Top right: The margin loss of the compressed network approximates the error of the uncompressed network once $b_e \geq 5$ and $b_m \geq 15$, at which point the compressed representation is around 75% the size of the uncompressed representation. Bottom left and bottom right: The error bounds (when inverting the kl or using Pinsker's inequality) dip below that of the uncompressed network (horizontal dotted line) for $b_e \geq 5$.

Size		**Error bound inverse kl**		**Error bound Pinsker**	
Depth	**Width**	**No Comp.**	**Comp.**	**No Comp.**	**Comp.**
1	4	0.788	**0.767**	0.812	**0.786**
1	8	0.790	**0.761**	0.820	**0.781**
1	16	0.785	**0.736**	0.817	**0.753**
1	32	0.866	**0.807**	0.946	**0.849**
1	64	0.933	**0.878**	1.112	**0.973**
1	128	0.984	**0.952**	1.387	**1.185**
1	256	0.999	**0.994**	1.846	**1.385**
1	512	1.000	**1.000**	2.494	**1.542**
2	4	0.792	**0.775**	0.819	**0.796**
2	8	0.803	**0.776**	0.838	**0.802**
2	16	0.826	**0.784**	0.876	**0.814**
2	32	0.923	**0.878**	1.077	**0.970**
2	64	0.990	**0.971**	1.469	**1.280**
2	128	1.000	**1.000**	2.361	**1.466**
2	256	1.000	1.000	4.212	**1.812**
2	512	1.000	1.000	8.028	**2.491**
3	4	0.776	**0.765**	0.799	**0.784**
3	8	0.817	**0.798**	0.858	**0.831**
3	16	0.909	**0.882**	1.028	**0.968**
3	32	0.956	**0.927**	1.193	**1.087**
3	64	0.999	**0.995**	1.767	**1.352**
3	128	1.000	**1.000**	3.082	**1.599**
3	256	1.000	1.000	5.686	**2.082**
3	512	1.000	1.000	11.074	**3.038**
4	4	0.786	**0.773**	0.812	**0.794**
4	8	0.840	**0.821**	0.893	**0.863**
4	16	0.933	**0.912**	1.089	**1.032**
4	32	0.978	**0.961**	1.320	**1.212**
4	64	1.000	**1.000**	2.058	**1.408**
4	128	1.000	1.000	3.583	**1.708**
4	256	1.000	1.000	6.860	**2.297**
4	512	1.000	1.000	13.460	**3.467**

Table 4.2: **Effects of quantisation via truncation** on 32 MLPs. Depth and Width is the number and width of the hidden layers. Bold indicates that compression reduced the bound compared to its counterpart calculated without compression.

This case is illustrative of the other 31 networks, for which we observed similar results.

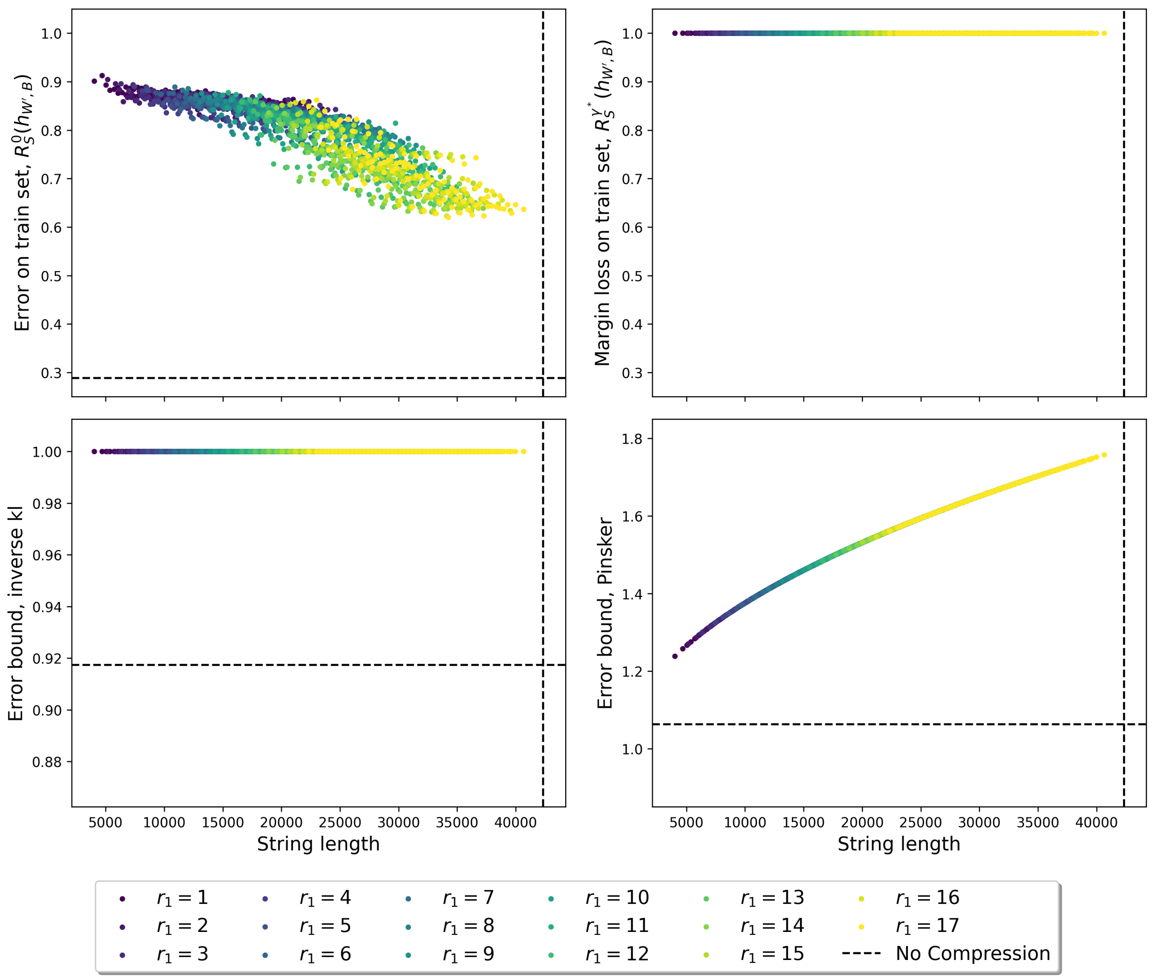


Figure 4.3: **Effects of low-rank approximation** on a network $h_{W,B} \in \texttt{MLP}_{(40,32,32,10)}$ trained on 50,000 samples from MNIST1D. String length increases monotonically with each component r_i of $\boldsymbol{r}$. The vertical dotted lines represent the string length of the uncompressed network. Top left: The error of the compressed network decreases as the r_i increase, but remains large. Top right: The margin loss is consistently trivial, showing the required margin is always too large. Bottom left: The error bounds from inverting the kl are all trivial. Bottom right: The variation in the error bound using Pinsker's inequality is solely due to the change in string length as the margin loss is constant. All error bounds exceed those of the uncompressed network.

4.5.4 Combined approaches

Low-rank approximation failed to improve the discrete PAC-Bayes bounds because the compressed networks could not approximate the original networks sufficiently closely while remaining within the string length budget, i.e. while ensuring that the number of bits $|s_{W',B}|$ required to represent the compressed model is less than the number of bits $|s_{W,B}|$ required to represent the original. At first sight it may seem obvious that combining low-rank approximation with other compression schemes will only makes this worse. However, additional compression of the low-rank decompositions via quantisation allows larger values of the ranks r_i to be chosen while still remaining within the string length budget, so it is still worth investigating the combined compression schemes. For the combined approaches it was necessary to increment the rank values r_i in steps greater than one (skipping some values) otherwise the experiment would have been computationally prohibitive.

Comparing the combined compressions schemes shown in Figures 4.4 and 4.5 with the low-rank only compression shown in Figure 4.3, we see that the range of values of r_1 for the network $h_{W,B} \in \texttt{MLP}_{(40,32,32,10)}$ is indeed much wider for the combined compression schemes. However, in both cases the approximation of the original network is still poor; there remains a large gap between the train error of the compressed and original networks, such that the margin loss consistently takes the maximum value of one. For this reason we do not plot the PAC-Bayes bounds as they are uniformly vacuous.

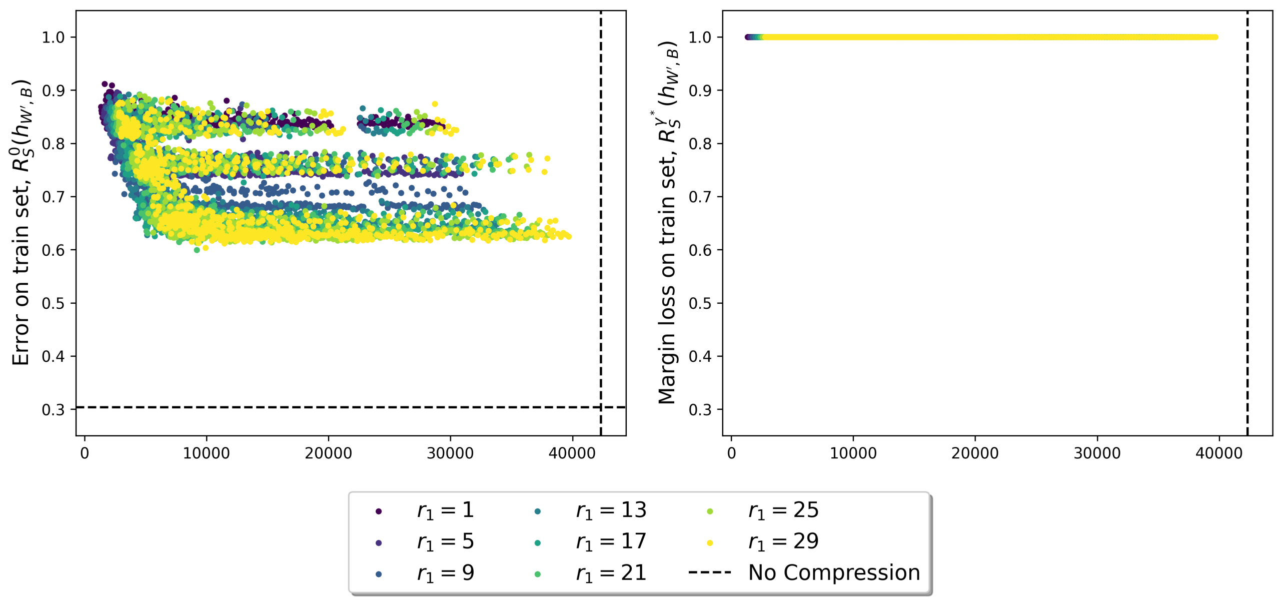


Figure 4.4: Effects of low-rank approximation combined with quantisation via k-means on a network $h_{W,B} \in \texttt{MLP}_{(40,32,32,10)}$. Left: The train error of the compressed network fails to approximate that of the original (horizontal dotted line). Right: The margin loss of the compressed network is always one, making the PAC-Bayes bounds vacuous.

4.6 Conclusion

Recall that our purpose was to investigate the capacity of a classic PAC-Bayes bound (Theorem 13) to shed light on the generalisation mystery. By eschewing common tricks used to obtain tighter generalisation bounds, such as stochastic networks, modified training regimes, and data-dependent priors, we evaluate the ability of PAC-Bayes to explain the generalisation of deterministic networks trained according to ordinary DL practice, which is an important goal of any successful theory of learning. It is for this reason that we employed compression purely as a tool to obtain bounds on the original, uncompressed networks, in sharp contrast to works such as Arora et al. (2018), Lotfi et al. (2022), and Zhou et al. (2018) which bound the compressed networks.

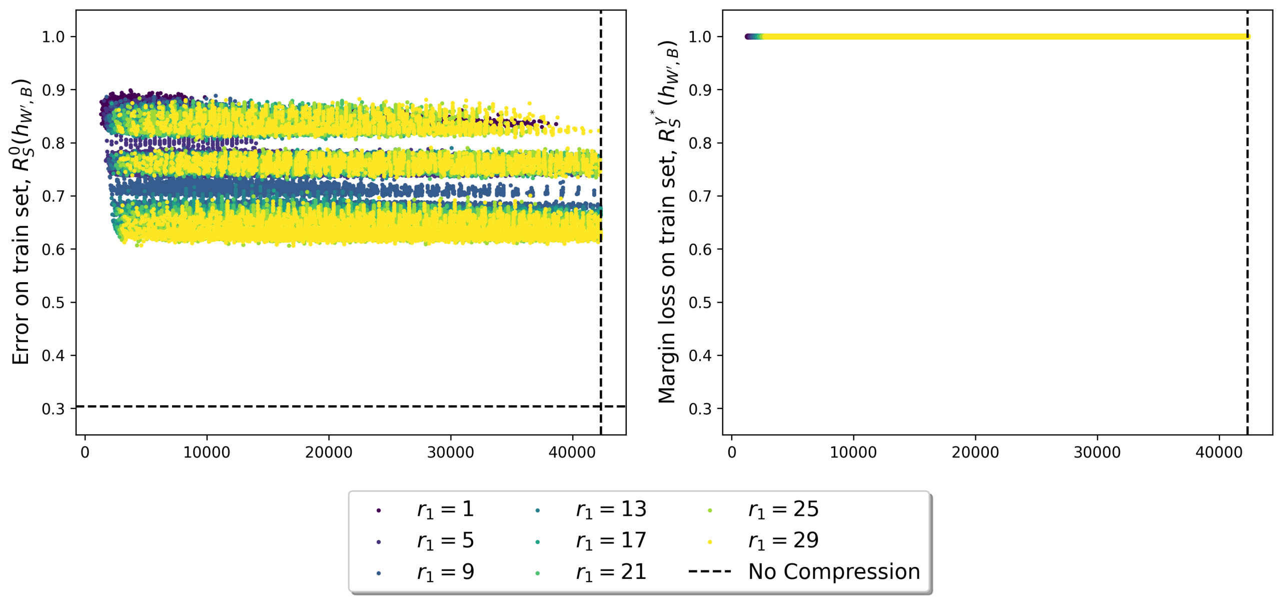


Figure 4.5: Effects of low-rank approximation combined with quantisation via truncation on a network $h_{W,B} \in \texttt{MLP}_{(40,32,32,10)}$. Left: The train error of the compressed network fails to approximate that of the original (horizontal dotted line). Right: The margin loss of the compressed network is always one, making the PAC-Bayes bounds vacuous.

Our main theoretical result, Theorem 15, bounds the true error of a network in terms of the empirical margin loss of its compression, plus a term that is $O(\sqrt{|s|/n})$ ignoring logarithmic terms, where $|s|$ is the length of the bit string representation of the compressed network. The margin required by the bound is smaller for higher fidelity compressions. In other words, by extending the work of Neyshabur et al. (2017b) and others, we have proven that if a network has large margin on the train set, and there exists a high-fidelity compression, then this network generalises.

Our result is flexible enough to encompass a very wide range of compression schemes, as can be seen by the generality of Definition 9. We tested five compression approaches, finding that going via a compressed network can indeed tighten bounds on the original network, but that this heavily depends on the compression scheme used. The only compression scheme to have much success in our experiments was quantisation via truncation. It is perhaps unsurprising that this compression scheme proves successful, since the removal of the least impactful bits—least significant for the mantissa and most significant for the (unbiased) exponent—produces an immediate decrease in string size and, due to the well-known robustness of neural networks to weight perturbations, a negligible degradation in performance.

In contrast, low-rank approximation has a significant negative impact on performance since the constraint of reducing the string length caps the permissible ranks, resulting in poor approximations. This effect persisted when combined with the other compression approaches, even though doing so increases the cap on the permissible ranks. Quantisation via k-means clustering of the weights also performed poorly for a similar reason; as the number of clusters k increases, so does the fidelity of the compression, but the codebook also grows, and we reach the string budget before the approximation becomes sufficiently close for Theorem 15 to produce non-vacuous bounds.

Chapter 5

Distillability as a Predictor of Generalisation

5.1 Introduction

As argued in Section 2.2, tight generalisation bounds alone cannot explain generalisation in the overparameterised regime, and a full explanation of the mystery will likely come from a two-pronged approach. First, an empirical investigation of the inductive bias of commonly used DL algorithms and hypothesis classes—mostly some form of SGD on neural network weights—to characterise the hypotheses typical DL practice is likely to produce. These suggested characterisations are commonly called complexity measures even if they are not successful, a terminology we adopt. Second, coupling a successful complexity measure with a rigorous generalisation bound proving that networks with low complexity according to this measure have small generalisation gap.

While less rigorous than statistical learning theory, the search for complexity measures that can reliably distinguish between hypotheses that generalise versus those that have simply memorised the training data, may be a practically necessary component of the search for empirically tight generalisation bounds. Indeed, as shown in Jiang et al. (2019), many existing generalisation bounds completely fail to correlate with generalisation gap, in fact showing a negative correlation, indicating that a theory-only approach may be insufficient for deriving empirically tight generalisation bounds to develop our understanding of generalisation in the overparameterised regime.

Chapter 4 focused on the second part of this two-pronged strategy, by proving that a network will generalise if there exists a compressible nearby network with minimally degraded error. In contrast, this chapter focuses on introducing a new complexity measure, applicable to feedforward networks of arbitrary architecture (MLPs, CNNs, etc.), which we term the *distillation complexity.* We empirically show that distillation complexity is predictive of generalisation gap even in the overparameterised regime, in that networks with higher distillation complexity generally have a larger generalisation gap, as is visually apparent from Figure 5.1.

It has been noted that the generalisation ability of neural networks frequently does not

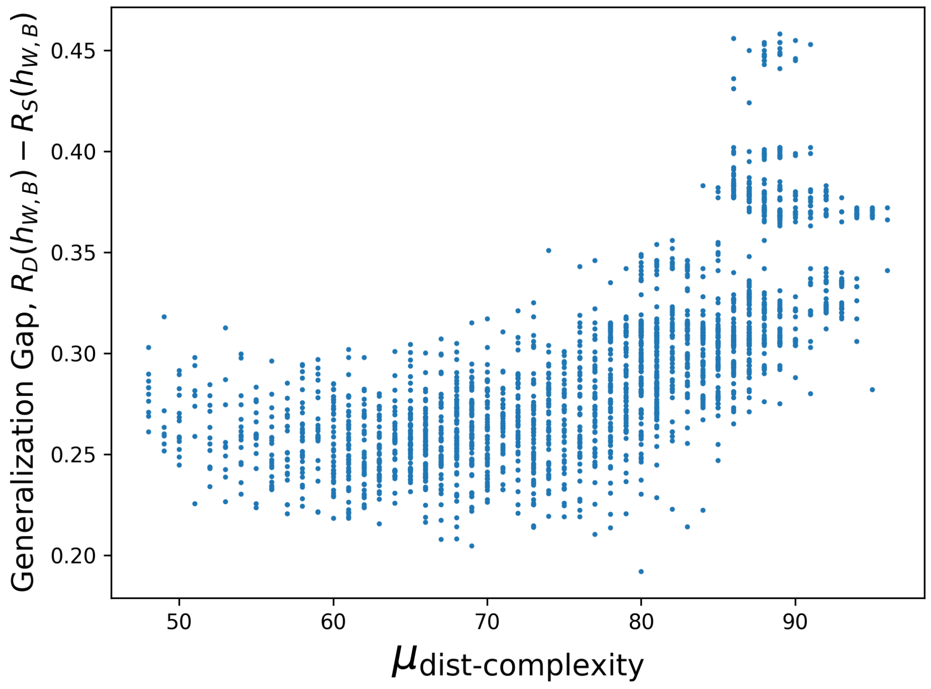


Figure 5.1: Generalisation gap versus distillation complexity $\mu_{\text{dist-complexity}}$, defined in Definition 12, across a suite of 2183 models. Each model is trained and evaluated on 4000 and 1000 samples, respectively, from the MNIST1D dataset, Greydanus and Kobak (2020). Our complexity measure is predictive; the trend is that the higher the distillation complexity the larger the generalisation gap.

degrade with model size. In fact, in contrast to the behaviour suggested by the bias variance trade-off, generalisation often *improves* even as models increase past the interpolation threshold, an observation first made by Vallet et al. (1989) on synthetic data, by Duin (2000) on real-world data, and brought to popular attention by Belkin et al. (2019) and Nakkiran et al. (2021) (see Loog et al. (2020) for a brief history). This phenomenon, termed deep double descent, means that any complexity measure capable of predicting generalisation should not grow with model size, at least beyond the interpolation threshold.

Putting this together, we have the following desiderata for a complexity measure:

1. The complexity measure should be positively correlated with generalisation gap.

2. The complexity measure should be one of the variables causally responsible for the value of the generalisation gap.

3. The complexity measure should not increase with model size beyond the interpolation threshold.

As already noted, Figure 5.1 visually demonstrates that distillation complexity meets the first desideratum. Evidence for the second desideratum is presented in Section 5.8. As for the third, distillation complexity remains steady as model size increases, as shown in Figure 5.2. This is already reasonably good evidence that our complexity measure genuinely captures a notion of complexity with explanatory power in the overparameterised regime. This is in sharp contrast to weight norms, present in many generalisation bounds, which have already been noted in Jiang et al. (2019) to both negatively correlate with generalisation gap and grow with model size.

As for the second desideratum, we compare our complexity measure against twenty others from the literature, finding that it is one of the best three in its predictive power of generalisation gap as evaluated according to five metrics described in Section 5.7, where some of these metrics are designed to rule out spurious correlations and instead capture causal relationships. Distillation complexity outperforms the norm-based measures, the PAC-Bayes derived measures, and a number of uncertainty-based measures, as shown in Figure 5.5. Further, we corroborate the finding of Jiang et al. (2019) that norm-based measures fail as complexity measures as they often negatively correlate with generalisation gap. However, in contrast to their results, we find that a PAC-Bayes bound, rather than successfully predicting generalisation gap, also fails badly. This suggests that the success of this PAC-Bayes bound as a complexity measure is task-dependent.

Distillation complexity is loosely defined as the minimum size student network into which the original network can be distilled. This is a more natural formalisation of Occam's razor than the commonly employed weight-norm as it is a notion of complexity of functions rather than weight parameterisations. Intuitively, the complexity of networks should be measured in terms of the complexity of the functions they express rather than the complexity of any specific weight parameterisation. This crucial distinction can be justified by the fact that it is functions we ultimately care about in learning—we are indifferent to distinct parameterisations of a single function as they produce identical predictions.

This is in contrast to Bayesian learning and PAC-Bayesian theory—which commonly use priors that place more weight on networks whose weights lie close to the origin or the weight initialisation—and regularisation methods such as weight decay, as they fail to account for the fact that simple functions can be expressed by neural networks with large weights.

5.2 Methodology

We follow the approach taken by Jiang et al. (2018) of training a large suite of models that can be used to evaluate the association between distillation complexity and generalisation gap. This is an approach that has been followed by many others (Jiang et al., 2019; Kuhn et al., 2021) and was even the basis for a NeurIPS 2020 competition (Jiang et al., 2020). Our suite of models is specifically trained in a way that is both reflective of general practice and yields sufficient variability in generalisation gap.

An ideal complexity measure should be among the variables that are causally responsible for the value of the generalisation gap, rather than merely statistically associated. While the causal effect of any complexity measure on generalisation is hard to unequivocally establish empirically, metrics such as Kendall's Rank Correlation Coefficient (KRCC) may provide evidence if they show that models with a higher complexity generally have larger generalisation gap.

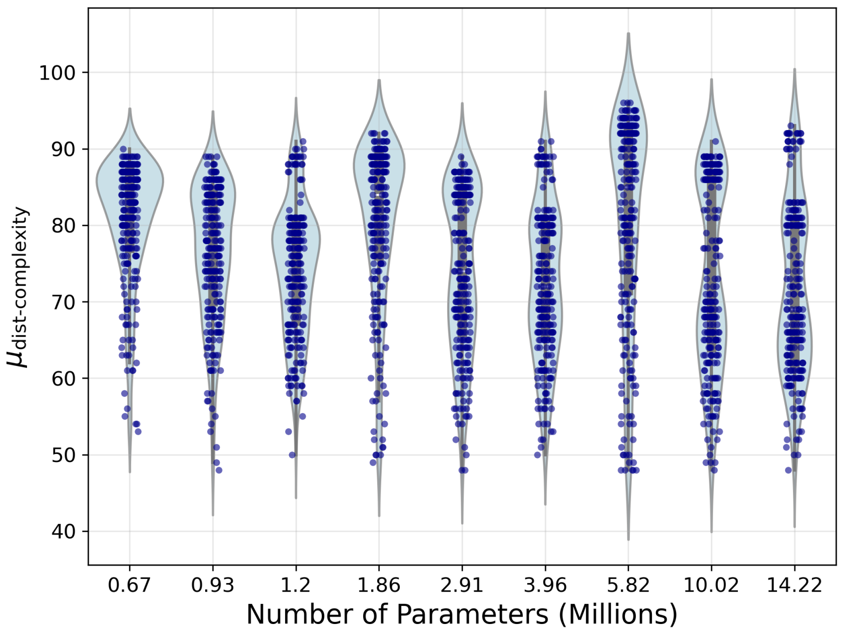


Figure 5.2: The distillation complexity $\mu_{\text{dist-complexity}}$ defined in Definition 12 for the suite of 2183 models, stratified by the number of model parameters. The nine values for the number of parameters correspond to the nine combinations of `num_hidden_layers` $\in \{2, 3, 4\}$ and `hidden_layer_width` $\in \{512, 1024, 2048\}$. The distillation complexity shows no sign of increasing with the size of the base model.

Acknowledging that a positive KRCC may simply be due to spurious correlations, Jiang et al. (2019) also employ the *Granulated* KRCC (GKRCC), and the Conditional Independence Test (CIT), which provide more robust evidence of causation, and which we employ here. Further, we compare our complexity measure against a wide range of both theoretically and empirically motivated measures, such as norm-based and flatness-based measures.

5.3 Related work

There is of course a substantial literature on complexity measures going back many years. We give a brief overview of some of the main threads, focusing on their relation to the present work.

As outlined in the introduction, our empirical methodology for evaluating our complexity measure draws heavily on Jiang et al. (2019). This methodology was later used by Kuhn et al. (2021) to show that networks that are more robust to pruning are likely to generalise better; networks that can have a greater proportion of their weights pruned without significantly increasing their training loss generalise better on average. Our work has a similar motivation to theirs in that heavily pruned networks are intuitively simpler, and so networks that are robust to pruning may also be considered to be simpler as their behaviour can be largely explained by a network with fewer parameters. However, it is the *proportion* of the weights that can be pruned that they show is predictive of generalisation; while larger networks trained on the same task can have a greater proportion of their weights pruned, the *absolute number of weights remaining* is still larger for larger networks.

In contrast, we show that the size of student network into which a network can be distilled is almost independent of the original network's size, as seen in Figure 5.2 and further explored in Section 5.5.1. This makes the connection of distillation complexity to Occam's razor much closer than for robustness to pruning.

The Kolmogorov Complexity of an entity representable as a bit string is the length of the shortest program (executable on some fixed universal Turing machine) that prints this string and then halts. While enjoying an elegant theory, the quantity is uncomputable due to the halting problem. Nevertheless, time-bounded variations exist. For example, Schmidhuber (1997) uses Levin complexity to find networks of low complexity solving a (very) artificial task.

The Kolmogorov Growth (KG) defined in Ghosh and Motani (2021) is inspired by the Kolmogorov Complexity and is specific to functions. Defined in terms of the well-known growth function in statistics, the KG of a function f is "concerned with the smallest function space that f can belong to, that can still fit the data well." They successfully derive error bounds in terms of the KG. While uncomputable, they approximate the KG for neural networks and control it by ensuring that during training the network always lies near a second network with fewer parameters, where the proximity is measured as the maximum Euclidean distance between the logits of the two networks over the input. In fact, they employ this regularisation recursively. While we are also concerned in this work with whether a network can be approximated by a smaller one, our motivation is to understand whether such distillability can explain generalisation "in the wild" when networks are trained with typical methods, rather than whether integrating it into the training procedure can improve performance. Further, we measure the discrepancy between the teacher and student networks with the Kullback–Leibler divergence between the output probabilities (after softmax) rather than the Euclidean distance on the raw logits, since this is much more common in the distillation literature (Hinton et al., 2015).

Also connected to the Kolmogorov Complexity is the Minimum Description Length (MDL) Principle (see Grünwald (2005) for an excellent tutorial), an alternative framework stating that the best model of a dataset is the one that requires the fewest bits to specify both the model and its error on the dataset. An approximation of the MDL was employed in Hinton and Van Camp (1993). We note that they encoded their models on a parameter rather than functional level, which fails to take advantage of the fact the minimum description length of the function expressed by a neural network may be much shorter than the minimum description length of its parameters.

As noted in Hochreiter and Schmidhuber (1994), this elegantly motivates the search for flat minima of the loss landscape; since the error remains fairly constant in such regions, one can move from the minimum to a model with low description length without substantially increasing the description length of the error, yielding a lower combined description length and therefore a preferred model according to the MDL principle. Remarkably, it has been empirically observed that typical network training methods such as SGD do indeed locate flat minima, and that flatness correlates with generalisation as the batch size is varied (Keskar et al., 2016). However, as argued in Dinh et al. (2017), sharpness cannot be a satisfactory complexity measure as it is parameterisation-dependent—the parameters of networks with ReLU activations can be modified to make the minimum arbitrarily flat/sharp without changing the function the network expresses, and therefore without changing its generalisation behaviour. It is a virtue of our complexity measure that it is defined on the function rather than parameter level and so is invariant to reparametrisations.

The norms of the weight matrices of neural networks, which determine the network's Lipschitz constant when no biases are used, have also been interpreted as complexity measures. Indeed, the norms of the weight matrices appear in a number of generalisation bounds (Arora et al., 2018; Bartlett et al., 2017; Neyshabur et al., 2017b). Further, penalising weight norms (e.g. weight decay) has been empirically observed to improve generalisation, providing some evidence that there is a causal relationship. However weight norms share the weakness of other parameter- rather than function-based complexity measures in that they can be large even for simple functions. As a trivial example, two weights whose effects cancel out can be made arbitrarily large without changing the function the network expresses.

5.4 Training a suite of models

One approach to demonstrating a causal relationship between a complexity measure and generalisation is to include in the optimisation objective a regularisation term penalising the complexity measure and see whether this improves generalisation. However, as noted in Jiang et al. (2019), a lack of regularisation does not constitute a satisfactory control, as there may be implicit regularisation within the optimisation procedure that cannot be eliminated. Moreover, such regularisation may change several other properties of the trained network as it alters the topography of the loss landscape.

We therefore follow the methodology of Jiang et al. (2019) and Kuhn et al. (2021) by training a suite of over 2000 neural networks to form a dataset on which to investigate the relationship between generalisation and distillability. We use the MNIST1D dataset from Greydanus and Kobak (2020), a procedurally generated, reasonably low-dimensional dataset (40 dimensions rather than MNIST's 784) that replicates many of the features of deep learning (e.g. double descent and the existence of lottery tickets). We opt for this low-dimensional dataset rather than the more familiar MNIST or CIFAR-10 datasets since our computational budget is limited and distillability can be expensive to compute, as it requires many distillations. To eliminate a potential source of noise, we use the same MNIST1D dataset for every model rather than regenerating fresh samples for each. We use 4000 samples for training and 1000 for testing.

For the architecture, we opt for fully connected neural networks with ReLU activations and $2, 3$ or 4 hidden layers each of width $512, 1024$ or 2048. For each neural network, the widths of its hidden layers are equal. To ensure variability in the test performance of the models, we vary five more hyperparameters known to affect generalisation; choice of optimiser, learning rate, batch size, dropout probability and weight decay coefficient. We pick three values for each hyperparameter that are both realistic for real world training and ensure that almost every combination of values leads to a model that eventually reaches the target train loss. Specifically, we train $3^7 = 2187$ MLPs by using all hyperparameter combinations from

$$\begin{aligned}
\texttt{num\_hidden\_layers} &\in \{2, 3, 4\} \\
\texttt{hidden\_layer\_width} &\in \{512, 1024, 2048\} \\
\texttt{optimiser} &\in \{\texttt{SGD, Adam, RMSProp}\} \\
\texttt{learning\_rate} &\in \{0.003, 0.001, 0.0003\} \\
\texttt{batch\_size} &\in \{32, 64, 128\} \\
\texttt{dropout\_probability} &\in \{0, 0.1, 0.2\} \\
\texttt{weight\_decay} &\in \{0, 0.00001, 0.0001\}
\end{aligned}$$

We control for final train loss by using it as a stopping criterion. More precisely, we stop training once the average cross-entropy loss across the 4000 training samples first goes below 0.01. We discard the model if it fails to reach this target within 10^6 epochs or if the running best train loss fails to decrease for 1000 epochs. Only four of the 2187 models are discarded, all for the second reason, leaving us with a suite of 2183 models. Figure 5.3 shows the final train loss and error across the suite of models. There remains some variation in the final train loss. This could have been reduced by evaluating the train loss across the entire train set after each batch rather than only after each epoch. This would have been significantly more computationally expensive, however. We also see that the final train errors are all below 0.4%, with most models achieving 0% train error, confirming we are in fact in the overparameterised regime.

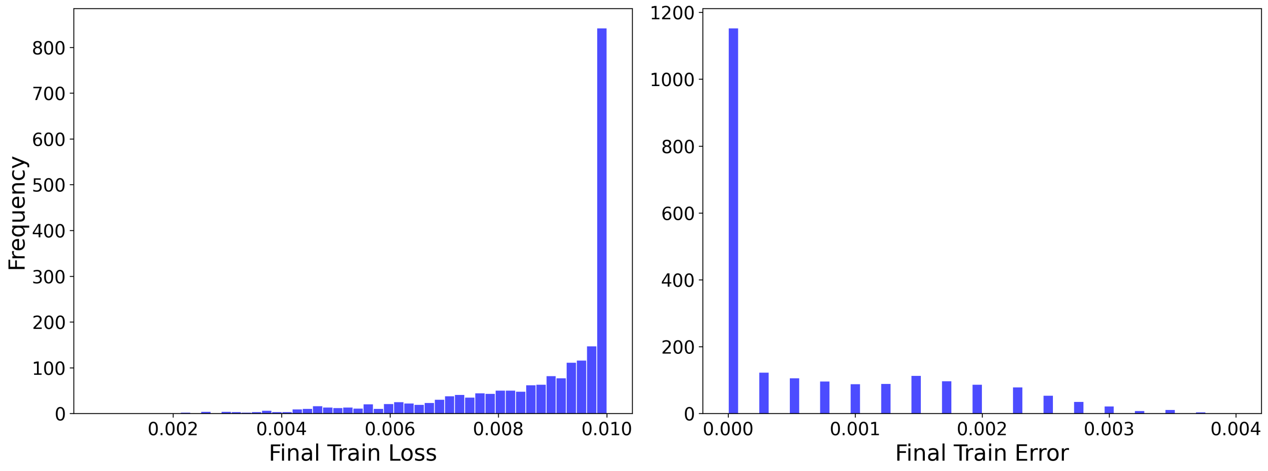


Figure 5.3: Final train loss and train error across the suite of 2183 models.

5.5 Distillation complexity

For any teacher and student functions $f, g : \mathbb{R}^{k_{\text{in}}} \to \mathbb{R}^{k_{\text{out}}}$ we define the distillation loss

$$\ell_{\text{dist}}(f, g; S) \coloneqq \frac{1}{n} \sum_{i=1}^{n} \mathrm{KL}\big(\sigma(f(x_i)) \big\| \sigma(g(x_i))\big), \tag{5.1}$$

where $S = \big((x_1, y_1), \ldots, (x_n, y_n)\big) \in (\mathbb{R}^{k_{\text{in}}} \times \mathbb{R}^{k_{\text{out}}})^n$ is a sample and $\sigma : \mathbb{R}^{k_{\text{out}}} \to \triangle_{k_{\text{out}}}$ denotes the softmax, which converts a vector of logits $(\ell_1, \ldots, \ell_{k_{\text{out}}}) \in \mathbb{R}^{k_{\text{out}}}$ to a vector of probabilities $\sigma(\ell_1, \ldots, \ell_{k_{\text{out}}}) = (\sigma_1, \ldots, \sigma_{k_{\text{out}}}) \in \triangle_{k_{\text{out}}}$ according to the formula

$$\sigma_j = \frac{e^{\ell_j}}{\sum_{j'=1}^{k_{\text{out}}} e^{\ell_{j'}}}.$$

Our student networks will be elements of $\mathtt{MLP}_{(k_{\text{in}}, \kappa, k_{\text{out}})}$, which we use to denote the set of all MLPs from $\mathbb{R}^{k_{\text{in}}}$ to $\mathbb{R}^{k_{\text{out}}}$ with one hidden layer of dimension κ and ReLU activation function. More precisely, $\mathtt{MLP}_{(k_{\text{in}}, \kappa, k_{\text{out}})}$ is the set of all functions of the form

$$W_2 \max\{W_1 x + B_1, 0\} + B_2,$$

where the max is applied element-wise and

$$\begin{aligned} W_1 &\in \mathbb{R}^{\kappa \times k_{\text{in}}}, \;\; B_1 \in \mathbb{R}^{\kappa} \\ W_2 &\in \mathbb{R}^{k_{\text{out}} \times \kappa}, \;\; B_2 \in \mathbb{R}^{k_{\text{out}}}. \end{aligned}$$

More generally, recalling the notation outlined in Section 4.3.2, we denote the set of all such ReLU-activated MLPs of arbitrary depth and width by $\mathtt{MLP}$. We then have the following definition.

Definition 10. *(Distillation complexity, strict version)* For any function $f : \mathbb{R}^{k_{\text{in}}} \to \mathbb{R}^{k_{\text{out}}}$, not necessarily a neural network, sample $S = \big((x_1, y_1), \ldots, (x_n, y_n)\big)$ and tolerance $\epsilon > 0$, the *distillation complexity* of f is

$$\kappa(f; S, \epsilon) = \min\Big\{\kappa \in \mathbb{N} : \exists g \in \mathtt{MLP}_{(k_{\text{in}}, \kappa, k_{\text{out}})}, \;\; \ell_{\text{dist}}(f, g; S) \leq \epsilon\Big\}.$$

In words, the integer $\kappa(f; S, \epsilon)$ denotes the minimum hidden width necessary in order for an MLP with a single hidden layer of this width to be able to return the same output as f on the sample S up to some KL tolerance ϵ. We make no claims that this is a fundamental complexity measure on the order of Kolmogorov Complexity or Minimum Description Length discussed in the introduction, only that it captures an intuitive notion of the complexity of f considered as a function on S; if f requires a much wider single hidden layer MLP to represent it on S, then intuitively it is more complex on S.

While $\kappa(f; S, \epsilon)$ as defined above can easily be upper bounded—exhibiting a student network of width κ and distillation loss at most ϵ upper bounds the distillation complexity by κ—calculating it exactly is likely very difficult. We therefore relax the definition by replacing the existential quantifier over $\texttt{MLP}_{(k_{\text{in}},\kappa,k_{\text{out}})}$ with a single element of $\texttt{MLP}_{(k_{\text{in}},\kappa,k_{\text{out}})}$ returned by a *distillation scheme*, which we now define.

Definition 11. A *distillation scheme* is a deterministic mapping $\texttt{Dist}$ taking as input a function $f : \mathbb{R}^{k_{\text{in}}} \to \mathbb{R}^{k_{\text{out}}}$, sample S, hidden width $\kappa \in \mathbb{N}$ and random seed $s \in \mathbb{N}$, and producing as output a neural network $\texttt{Dist}(f; S, \kappa, s) \in \texttt{MLP}_{(k_{\text{in}},\kappa,k_{\text{out}})}$ with one hidden layer of width κ.

The distillation scheme we employ is gradient descent minimising the distillation loss in Equation (5.1). See Section 5.5.1 for the details. This enables the following definition of distillation complexity, which is more practical than Definition 10.

Definition 12. *(Distillation complexity, practical version)* Fix a distillation scheme $\texttt{Dist}$, sample $S = \big((x_1, y_1), \dots, (x_n, y_n)\big)$, tolerance $\epsilon > 0$, number of distillation attempts N and sequence of random seeds $\boldsymbol{s} = (s_1, \dots, s_N)$. For any function $f : \mathbb{R}^{k_{\text{in}}} \to \mathbb{R}^{k_{\text{out}}}$, not necessarily a neural network, the *distillation complexity* of f is then

$$\kappa(f; S, \epsilon, \boldsymbol{s}) = \min \Big\{ \kappa \in \mathbb{N} : \min_{j=1,\dots,N} \ell_{\text{dist}}\big(f, g_j; S\big) \leq \epsilon \Big\},$$

where $g_j = \texttt{Dist}(f; S, \kappa, s_j)$, namely the network returned by the distillation scheme with random seed s_j.

In other words, $\kappa(f; S, \epsilon, \boldsymbol{s})$ is the minimum hidden width κ such that at least one of N distillation attempts yields a network with distillation loss at most ϵ from f on S. The purpose of allowing multiple distillation attempts with different random seeds is to produce a more robust definition of distillation complexity that is less dependent on the particular random seed chosen. When comparing the distillation complexities of two functions f_1, f_2, the same sample S, tolerance ϵ and seeds $\boldsymbol{s}$ should be used, which is indeed what we do in our experiments. Note that since $\texttt{Dist}$ is deterministic and the seeds $\boldsymbol{s}$ are fixed, $\kappa(f; S, \epsilon, \boldsymbol{s})$ constitutes a deterministic measure of the complexity of f on S.

While this definition may be altered to permit student networks with larger numbers of hidden layers, some exchange rate would have to be decided between number of layers and width of layers before the minimum could be taken. One possibility would be to take the minimum over parameter count. However, distillation complexity as defined above is already computationally intensive. Moreover, since the universal approximation theorem (Cybenko, 1989; Hornik et al., 1989) ensures that a single hidden layer neural network is sufficient to approximate any continuous function, one may hope that the definition as stated is reasonable.

5.5.1 Distillation scheme

We now describe in detail the distillation scheme we use for measuring distillation complexity of the networks described in Section 5.4.

The sample S we use for distillation is the same sample used for training the original networks, which we call the teacher networks in this section. This choice means we discover whether the training procedure for the teacher network is biased to find a simple representation of the data, which matches the philosophy of the MDL principle. For the distillation loss we use the KL divergence between the softmaxed output of the teacher and student models on S, given by Equation (5.1). We use full-batch gradient descent rather than SGD as this significantly speeds up computation time (our sample is small enough to be passed to the model in a single batch), making the experiment feasible within our computational budget. The regularising effect of SGD is commonly beneficial but unnecessary here, as we are solely concerned with discovering the narrowest single hidden layer MLP capable of representing the teacher network. We train the student networks using the Adam optimiser with learning rate 0.003 for up to $100,000$ epochs (where each epoch consists of a single gradient step), stopping early if the distillation loss reaches the target $\epsilon = 0.01$, or if it fails to improve upon its best value for more than 100 consecutive epochs. This returns a student network. The pseudocode is given in Algorithm 1.

For the distillation complexity, we use $N = 5$ attempts. Measuring the distillation complexity rigorously would require $N = 5$ distillation attempts for each value of the hidden width κ increasing from 1 until an attempt is successful, namely until the loss of the student network first falls below $\epsilon = 0.01$ for at least one of the $N = 5$ attempts. However, we can save computation by making the simplifying assumption that for all values of the hidden width κ above the distillation complexity, at least one of the $N = 5$ distillation attempts will be successful. Combined with the fact that none of the distillation attempts will be successful for any value of the hidden width κ below the distillation complexity (by definition), we have a kind of monotonicity assumption that permits binary search over the hidden width to determine the distillation complexity. While this monotonicity assumption is likely false in practice, the binary search procedure it motivates nevertheless yields a well-defined and deterministic value provided we fix the random seed (as we do), which is therefore a bona fide complexity measure worthy of investigation. Moreover, this simplifying assumption enables us to evaluate a much larger sample of networks than our computational budget would otherwise permit.

More precisely, the binary search procedure is as follows. First, we fix random seeds $s_1, \ldots, s_5$, one for each of the $N = 5$ distillation attempts, where we use these same seeds for all networks in the dataset. For any value κ of the hidden width, if at least one of the $N = 5$ distillation attempts is successful, then by definition κ is an upper bound on the distillation complexity. If none are successful, we treat κ as a lower bound on the distillation complexity, as per our simplifying assumption. Starting with $\kappa = 128$, if none of the $N = 5$ distillation attempts are successful, we repeatedly double κ until at least one is successful (this never exceeded 2048 in our experiments) and then perform binary search. Conversely, if at least one of the $N = 5$ distillation attempts is successful, we repeatedly halve κ until none are successful and then perform binary search.

It was already seen in Figure 5.2 that the distillation complexity remains fairly constant as model size increases. Breaking this down into increasing depth and width, Figure 5.4 shows that on average distillation complexity actually marginally decreases as both the depth and width of the network increases. This is further evidence that our complexity measure meets the third of the three complexity measure desiderata described in Section 5.1.

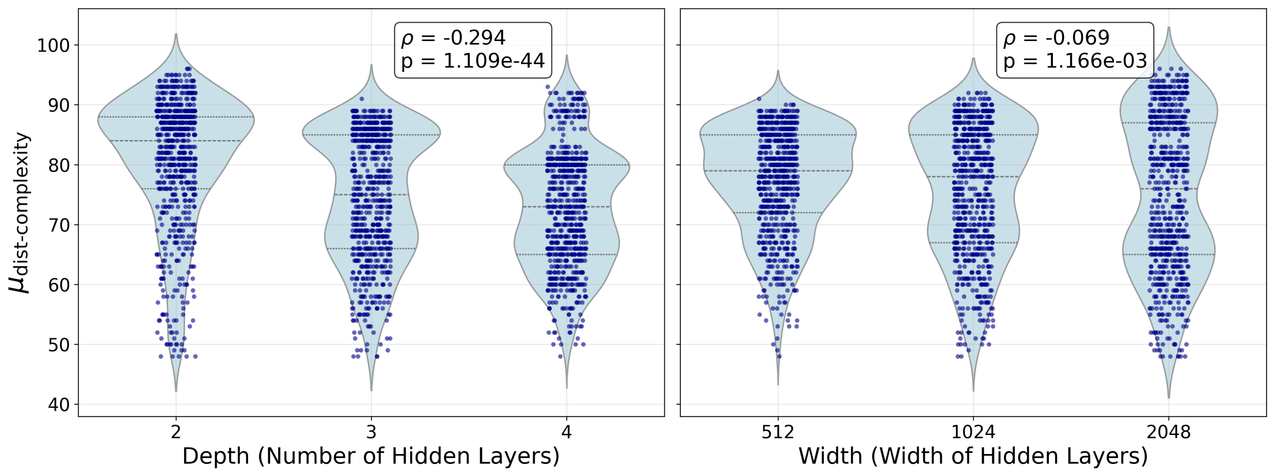


Figure 5.4: The distillation complexity $\mu_{\text{dist-complexity}}$ defined in Definition 12 plotted against depth `num_hidden_layers` $\in \{2, 3, 4\}$ and width `hidden_layer_width` $\in \{512, 1024, 2048\}$. Note depth and width correspond to the base model in the suite, not the student models in the definition of distillation complexity. In both cases the correlation coefficient ρ is negative and the p-value for the null hypothesis that the slope is zero is very small, where the null and alternative hypotheses are that the slope is zero and non-zero, respectively.

```
Input:
$f : \mathbb{R}^{k_{\text{in}}} \to \mathbb{R}^{k_{\text{out}}}$ /* Teacher function */
$S \in (\mathbb{R}^{k_{\text{in}}} \times \mathbb{R}^{k_{\text{out}}})^m$ /* Dataset for distillation */
$\epsilon > 0$ /* Tolerance threshold for distillation loss */
$\kappa \in \mathbb{N}$ /* Hidden width for student network */
$T \in \mathbb{N}$ /* Maximum number of distillation epochs */
$p \in \mathbb{N}$ /* Early stopping patience parameter */
$s \in \mathbb{N}$ /* Random seed */
$\texttt{Init} : \mathbb{N} \to \texttt{MLP}_{(k_{\text{in}},\kappa,k_{\text{out}})}$ /* Network initialisation scheme taking random seed */

Output:
$g_\kappa \in \texttt{MLP}_{(k_{\text{in}},\kappa,k_{\text{out}})}$ /* Single hidden layer student MLP */

Procedure:
$g_\kappa^0 \leftarrow \texttt{Init}(s)$ /* Randomly initialised student MLP with width $\kappa$ */
$\ell_{\text{best}} \leftarrow \infty$ /* Best loss observed so far */
$q \leftarrow 0$ /* Epochs since improvement */
$\texttt{stop} \leftarrow \texttt{False}$ /* Flag to indicate stopping condition met */
for $t \leftarrow 1$ to $T$ do
    $g_\kappa^t \leftarrow \texttt{GradientDescent}(\ell_{\text{dist}}, f, g_\kappa^{t-1}, S)$ /* One epoch of GD on distillation loss */
    if $\ell_{dist}(f, g_\kappa^t; S) \leq \epsilon$ then
        $\texttt{stop} \leftarrow \texttt{True}$ /* Distillation successful */
    end
    if $q \geq p$ then
        $\texttt{stop} \leftarrow \texttt{True}$ /* Early stopping triggered */
    end
    if $\ell_{dist}(f, g_\kappa^t; S) < \ell_{best}$ then
        $\ell_{\text{best}} \leftarrow \ell_{\text{dist}}(f, g_\kappa^t; S)$ /* Update best loss */
        $q \leftarrow 0$ /* Reset counter */
    else
        $q \leftarrow q + 1$ /* Increment counter */
    end
    if $\texttt{stop} = \texttt{True}$ then
        return $g_\kappa^t$ /* Return current student model */
    end
end
return $g_\kappa^T$ /* Return final student model after max epochs */
```

Algorithm 1: Neural network distillation attempt with early stopping

5.6 Comparison complexity measures

Measures that are functions of the entire hypothesis class, rather than taking into account the specific learned hypothesis, cannot explain generalisation in the overparameterised regime as they fail to distinguish between networks with zero training loss but different degrees of overfitting. This includes measures such as the number of parameters or the VC dimension, but also ones that take into account the dataset, such as Rademacher complexity. For this reason, we only compare to measures that are functions of the trained neural network and the sample, rather than only the architecture and the sample.

5.6.1 Norm-based measures

The norm-based measures considered in Jiang et al. (2019) are mostly inspired by theoretical bounds, consisting either of the entire bound or components of it on the understanding that some elements of the bound may be proof artifacts. Rather than starting with bounds and peeling off appropriate terms, we evaluate intuitive measures of network capacity directly. Indeed, since most of the bound-inspired measures evaluated in Jiang et al. (2019) predict generalisation only very poorly in their experiments, it is perhaps wise to aim directly for intuitive measures rather than hewing to rigorous bounds only.

Using all parameters

Treating the weights and biases of a network $h_{W,B} \in \texttt{MLP}$ as a single vector $\mathrm{vec}(W, B)$, we measure their ℓ_1- and ℓ_2-norms both from the origin and the initialisation h_{W^0,B^0} as follows:

$$\begin{aligned}
\mu_{\ell_1}(h_{W,B}) &= \|\mathrm{vec}(W, B)\|_1 && (5.2)\\
\mu_{\ell_2}(h_{W,B}) &= \|\mathrm{vec}(W, B)\|_2 && (5.3)\\
\mu_{\ell_1\text{-init}}(h_{W,B}; h_{W^0,B^0}) &= \|\mathrm{vec}(W - W^0, B - B^0)\|_1 && (5.4)\\
\mu_{\ell_2\text{-init}}(h_{W,B}; h_{W^0,B^0}) &= \|\mathrm{vec}(W - W^0, B - B^0)\|_2. && (5.5)
\end{aligned}$$

Using weights only

Conversely, we can retain the structure of the weight matrices and apply matrix norms, omitting the biases. In this case it is natural to measure both the sum and the product of the matrix norms, from both the origin and the initialisation. Let $\|W_i\|_{\text{spec}}$ and $\|W_i\|_{\text{frob}}$ denote the spectral and Frobenius norms, respectively, of the weight matrix W_i. We then define the following eight complexity measures by taking all combinations of spectral versus Frobenius norm, sum versus product, and taking the norm from the origin versus the initialisation.

$$\mu_{\text{spec-sum}}(h_{W,B}) = \sum_i \|W_i\|_{\text{spec}} \tag{5.6}$$

$$\mu_{\text{spec-prod}}(h_{W,B}) = \prod_i \|W_i\|_{\text{spec}} \tag{5.7}$$

$$\mu_{\text{frob-sum}}(h_{W,B}) = \sum_i \|W_i\|_{\text{frob}} \tag{5.8}$$

$$\mu_{\text{frob-prod}}(h_{W,B}) = \prod_i \|W_i\|_{\text{frob}} \tag{5.9}$$

$$\mu_{\text{spec-sum-init}}(h_{W,B}; h_{W^0,B^0}) = \sum_i \|W_i - W_i^0\|_{\text{spec}} \tag{5.10}$$

$$\mu_{\text{spec-prod-init}}(h_{W,B}; h_{W^0,B^0}) = \prod_i \|W_i - W_i^0\|_{\text{spec}} \tag{5.11}$$

$$\mu_{\text{frob-sum-init}}(h_{W,B}; h_{W^0,B^0}) = \sum_i \|W_i - W_i^0\|_{\text{frob}} \tag{5.12}$$

$$\mu_{\text{frob-prod-init}}(h_{W,B}; h_{W^0,B^0}) = \prod_i \|W_i - W_i^0\|_{\text{frob}} \tag{5.13}$$

5.6.2 Sharpness-based measures and PAC-Bayes bounds

We measure the flatness of the minimum found by training the network $h_{W,B}$ on a sample S as the level of noise required to increase the cross-entropy loss of $h_{W,B}$ by β, where in our experiments we take $\beta = 0.1$. The inverse of this is then a measure of sharpness and, therefore, perhaps complexity.

More formally, let $Q_{W,B,\sigma}$ denote the stochastic predictor formed by adding isotropic Gaussian noise with standard deviation σ to the parameters of $h_{W,B}$, with the true and test errors $R_D(Q_{W,B,\sigma})$ and $R_S(Q_{W,B,\sigma})$ of $Q_{W,B,\sigma}$ defined in the usual way by taking expectations over the noise. Likewise for the true and test cross-entropy losses, which we denote $R_D^{\text{ce}}(Q_{W,B,\sigma})$ and $R_S^{\text{ce}}(Q_{W,B,\sigma})$, respectively. We then define

$$\mu_{\text{sharpness}}(h_{W,B}) = \frac{1}{\sigma_\beta^2}, \quad \text{where} \tag{5.14}$$

$$\sigma_\beta = \max\Big\{\sigma > 0 : R_S^{\text{ce}}(Q_{W,B,\sigma}) \le R_S^{\text{ce}}(h_{W,B}) + \beta\Big\}, \tag{5.15}$$

For the PAC-Bayes inspired complexity measures, recall the classic PAC-Bayes theorem.

Theorem 16. *(Maurer (2004), Theorem 5) For any data-generating distribution D over $\mathcal{X}\times\mathcal{Y}$ where $\mathcal{Y} = \mathbb{R}^k$ for some k, hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, prior distribution P over $\mathcal{H}$, confidence level $\delta \in (0,1]$ and sample size m, then with probability at least $1-\delta$ over the random draw $S \sim D^m$, we have that simultaneously for all posterior distributions Q over $\mathcal{H}$*

$$\text{kl}\big(R_S(Q)\big\|R_D(Q)\big) \le \frac{1}{m}\left(\text{KL}(Q\|P) + \ln\frac{2\sqrt{m}}{\delta}\right). \tag{5.16}$$

We follow common practice by taking $Q = Q_{W,B}$ and $P = P_{W^0,B^0}$ to be isotropic Gaussians centred at the learned and initial parameters, respectively. We do not explore the use of data-dependent priors as they have the drawback of simply shifting the generalisation mystery to the prior. See Section 2.5.2 for a more in-depth discussion of this point. We take the same standard deviation σ for both Q and P, which is the obvious choice since for any means μ_Q, μ_P and standard deviation σ_P, a straightforward calculation shows that minimising

$$\mathrm{KL}\big(N(\mu_Q, \sigma_Q^2) \| N(\mu_P, \sigma_P^2)\big) = \frac{1}{2}\left(k\frac{\sigma_Q}{\sigma_P} - k + \frac{\|\mu_Q - \mu_P\|_2}{\sigma^2} + k \ln \frac{\sigma_P}{\sigma_Q}\right)$$

over σ_Q yields $\sigma_Q = \sigma_P$ (where k is the total number of parameters). We then take a union bound over $\sigma \in \Sigma$, where Σ is a set of K values fixed before observing the data. With these choices, Inequality (5.16) becomes

$$\mathrm{kl}\big(R_S(Q_{W,B}) \big\| R_D(Q_{W,B})\big) \leq \frac{1}{m}\left(\frac{\|\mathrm{vec}(W - W^0, B - B^0)\|_2^2}{2\sigma^2} + \ln \frac{2K\sqrt{m}}{\delta}\right) \tag{5.17}$$

$$=: \zeta(\sigma), \tag{5.18}$$

valid for all $\sigma \in \Sigma$ simultaneously.

Using this we construct the following three PAC-Bayes-inspired complexity measures. First, we transform Inequality (5.17) into bounds on $R_D(Q_{W,B})$ in two different ways; by inverting the kl numerically and by using Pinsker's inequality. Ideally we would then take the minimum of these over $\sigma \in \Sigma$, but doing so is computationally prohibitive, since each evaluation requires a large Monte Carlo sample to estimate $R_S(Q_{W,B})$. We therefore instead evaluate the bounds using the $\sigma \in \Sigma$ closest to σ_β defined in Equation (5.15), namely

$$\tilde{\sigma}_\beta \coloneqq \mathrm{argmin}_{\sigma \in \Sigma} |\sigma - \sigma_\beta|, \tag{5.19}$$

which yields

$$\mu_{\text{pacb-error-bound-inverse-kl}}(h_{W,B}) = \mathrm{kl}^{-1}\left(R_S(Q_{W,B}) | \zeta(\tilde{\sigma}_\beta)\right) \quad \text{and} \tag{5.20}$$

$$\mu_{\text{pacb-error-bound-pinsker}}(h_{W,B}) = R_S(Q_{W,B}) + \sqrt{\zeta(\tilde{\sigma}_\beta)/2}. \tag{5.21}$$

Third, simply take the kl bound in Equation (5.17) and substitute $\sigma = \tilde{\sigma}_\beta$ to obtain

$$\mu_{\text{pacb-kl-bound}}(h_{W,B}; h_{W^0,B^0}) = \zeta(\tilde{\sigma}_\beta), \tag{5.22}$$

which may loosely be interpreted as the product of $\mu_{\ell_2-\text{init}}$ (5.4) and $\mu_{\text{sharpness}}$ (5.14). While this is a bound on the small kl, for our purposes it may be a good proxy for the corresponding error bound

$$R_D(Q_{W,B}) \leq \mathrm{kl}^{-1}\Big(R_S(Q_{W,B}) \Big| \mu_{\text{pacb-kl-bound}}(h_{W,B}; h_{W^0,B^0})\Big). \tag{5.23}$$

To see why, note that our stopping criterion (see Section 5.4) ensures that $R_S^{\text{ce}}(h_{W,B}) \approx 0.01$ for all models $h_{W,B}$ in the suite. The definition of $\tilde{\sigma}_\beta$ then ensures that $R_S^{\text{ce}}(Q_{W,B}) \approx R_S^{\text{ce}}(h_{W,B}) + \beta \approx 0.11$, given our choice of $\beta = 0.1$. Since all the stochastic models $Q_{W,B}$ in the suite have approximately equal cross-entropy, they may also be expected to have approximately equal zero-one error $R_S(Q_{W,B})$. Thus, since $\mathrm{kl}^{-1}(\cdot|\cdot)$ is monotonically increasing in its second argument, the models will have approximately the same ordering if sorted according to increasing $\mu_{\text{kl-bound}}(h_{W,B}; h_{W^0,B^0})$ or increasing error bound (5.23). Since the ordering is approximately preserved, some of the evaluation criteria from Section 5.7, e.g. Kendall's rank-correlation coefficient, will be unaffected. Further, the advantage of $\mu_{\text{kl-bound}}$ over (5.23) is that it makes models easier to compare, since $\text{kl}^{-1}(\cdot|\cdot)$ is difficult to evaluate if its second argument is large.

For all three PAC-Bayes-inspired complexity measures, we choose $\Sigma = \{1{\cdot}2^{-14}, 2{\cdot}2^{-14}, \dots, 1\}$ so that $K = 2^{14}$. Note that due to the logarithmic factor the effect of this large number of union bounds is still small; with our sample of size $50,000$, the increase of the bound (5.16) is only $\ln(2^{14})/50,000 \approx 0.0002$.

5.6.3 Uncertainty-based measures

Three intuitive ways of measuring the uncertainty of a network $h_{W,B}$ in its predictions on the train set are as its final cross-entropy loss at the end of training (the higher the loss the higher the uncertainty), the inverse of its margin (the lower the margin the higher the uncertainty), and the average entropy of the output (the higher the entropy the higher the uncertainty). To that end, we define the complexity measures

$$\mu_{\text{final-loss}}(h_{W,B}) = R_S^{\text{ce}}(h_{W,B}) \tag{5.24}$$

$$\mu_{\text{inverse-margin}}(h_{W,B}) = 1/\gamma_{10\%}^2 \tag{5.25}$$

$$\mu_{\text{output-entropy}}(h_{W,B}) = \frac{1}{m}\sum_{i=1}^{m} H(h_{W,B}(x_i)) \tag{5.26}$$

where $\gamma_{10\%}$ is the 10th percentile of the set of margin values $\Gamma = \{f(x)_y - \max_{j\neq y} f(x)_j : (x,y) \in S\}$, and $H(h_{W,B}(x_i))$ is the entropy of the output of network $h_{W,B}$ on input x_i. We note that Jiang et al. (2019) opt to take the negative of the output entropy, whereas we leave it unchanged for greater consistency with the other uncertainty-based measures.

While $\mu_{\text{final-loss}}(h_{W,B}) \approx 0.01$ for all models in the suite, as this is our stopping criterion, there is still some variance. We therefore test its relationship with generalisation along with the other complexity measures as a check that we have sufficiently controlled for this variable. We also include the final error rate, the zero-one loss on the training set

$$\mu_{\text{final-error}} = R_S(h_{W,B}). \tag{5.27}$$

The measure $\mu_{\text{inverse-margin}}$ has been observed in Jiang et al. (2018, 2019) to be predictive of generalisation and so we include it here to evaluate its effectiveness on the new dataset MNIST1D and to use as a comparison for our distillation complexity. The negative output entropy $\mu_{\text{neg-entropy}}$ was proposed as regulariser in Pereyra et al. (2017) and observed to be effective, so we include it here to evaluate its relationship with generalisation more thoroughly.

5.7 Evaluation criteria

The most straightforward way to evaluate the relationship between the complexity measures and generalisation is as the proportion of the variance in generalisation gap explainable by a linear function of the complexity measure, namely the coefficient of determination r^2. However, if generalisation gap increases only monotonically, rather than linearly, with a quantity, the quantity may nevertheless be deemed a suitable complexity measure. We therefore also measure Kendall's Rank-Correlation Coefficient (KRCC). Using the notation of Jiang et al. (2019), let Θ denote the set of hyperparameters used to train the suite of models, and, for a complexity measure μ,

$$\mathcal{T} = \big\{(\mu(\boldsymbol{\theta}), g(\boldsymbol{\theta})) : \boldsymbol{\theta} \in \Theta\big\},$$

where $\mu(\boldsymbol{\theta})$ and $g(\boldsymbol{\theta})$ denote the complexity and generalisation gap of the model trained with hyperparameters $\boldsymbol{\theta}$. The KRCC is then defined as

$$\tau(\mathcal{T}) = \frac{1}{|\mathcal{T}|(|\mathcal{T}|-1)} \sum_{(\mu_1,g_1)\in\mathcal{T}} \sum_{(\mu_2,g_2)\in\mathcal{T}\setminus\{(\mu_1,g_1)\}} \operatorname{sign}(\mu_1-\mu_2)\operatorname{sign}(g_1-g_2). \tag{5.28}$$

In words, the KRCC $\tau(\mathcal{T})$ is the proportion of pairs from $\mathcal{T}$ for which the ordering of the complexity measures and generalisation gaps match, minus the proportion of pairs for which the ordering does not match. Therefore $\tau(\mathcal{T}) \in [-1, 1]$, where values close to 1 indicate that the models are ordered in approximately the same way whether they are sorted according to the complexity measure or the generalisation gap.

Note the KRCC may only capture a spurious correlation—rather than the hyperparameters $\boldsymbol{\theta} \in \Theta$ determining the complexity measure $\mu(\boldsymbol{\theta})$ which then in turn affects the generalisation gap $g(\boldsymbol{\theta})$, the hyperparameters may determine $g(\boldsymbol{\theta})$ directly. Thus the KRCC may be large even if the complexity measure has no causal effect on generalisation. Following Jiang et al. (2019), we use two additional evaluation metrics, the Granulated KRCC (GKRCC), and the Conditional Independence Test (CIT) inspired by the Inductive Causation Algorithm by Verma and Pearl (2022).

The GKRCC Ψ is defined by first calculating ψ_i, the average KRCC along the hyperparameter axis θ_i, where the average is taken over all possible values for the other hyperparameters $\boldsymbol{\theta}_{\neg i}$. The GKRCC Ψ is then the average of the ψ_i. More formally, suppose we have n hyperparameters, denoted by $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n) \in \Theta_1 \times \cdots \times \Theta_n$. Then

$$\Psi = \frac{1}{n}\sum_{i=1}^{n} \psi_i \quad \text{for} \quad \psi_i = \frac{1}{|\Theta_{\neg i}|} \sum_{\boldsymbol{\theta}_{\neg i}\in\Theta_{\neg i}} \tau(\mathcal{T}_i(\boldsymbol{\theta}_{\neg i})), \quad \text{where} \tag{5.29}$$

$$\mathcal{T}_i(\boldsymbol{\theta}_{\neg i}) = \big\{(\mu(\boldsymbol{\theta}'), g(\boldsymbol{\theta}')) : \theta_i' \in \Theta_i, \boldsymbol{\theta}'_{\neg i} = \boldsymbol{\theta}_{\neg i}\big\} \quad \text{and} \quad \Theta_{\neg i} = \Theta_1 \times \cdots \times \Theta_{i-1} \times \Theta_{i+1} \times \cdots \times \Theta_n.$$

The authors of Jiang et al. (2019) offer the following thought experiment to show that Ψ may be less susceptible to spurious correlations:

Suppose there exists a measure that perfectly captures the depth of the network while producing random prediction if two networks have the same depth. This measure would do reasonably well [achieve a high value] in terms of τ but much worse [achieve a low value] in terms of Ψ.

Let us flesh out this explanation. Suppose, as a hypothetical, that we have the following causal relationship $\theta_i \implies g(\boldsymbol{\theta})$, $\theta_i \implies \mu(\boldsymbol{\theta})$, but $g(\boldsymbol{\theta}) \not\Rightarrow \mu(\boldsymbol{\theta})$, where θ_i is network depth and $\mu(\boldsymbol{\theta}) = \theta_i + \epsilon$ for a small amount of random noise ϵ drawn independently for each $\boldsymbol{\theta}$. Further, suppose that ψ_i, the KRCC between network depth and generalisation gap is close to 1, i.e. for most fixed values of the hyperparameters $\boldsymbol{\theta}_{\neg i}$, the deeper models have larger generalisation gap.

First, consider the value of τ in this setup. For any pair $\boldsymbol{\theta}^1, \boldsymbol{\theta}^2 \in \Theta$, there is an approximately 1/3 chance that $\theta_i^1 = \theta_i^2$, in which case the order of $\mu(\boldsymbol{\theta}^1)$ and $\mu(\boldsymbol{\theta}^1)$ will be random, and an approximately 2/3 chance that $\theta_i^1 \neq \theta_i^2$, in which case the order of $\mu(\boldsymbol{\theta}^1)$ and $\mu(\boldsymbol{\theta}^2)$ will likely be the same as the order of θ_i^1 and θ_i^2 and hence (by assumption) likely the same as $g(\boldsymbol{\theta}^1)$ and $g(\boldsymbol{\theta}^2)$. Overall, we have $\tau \approx 1/3(1/2 - 1/2) + 2/3(1 - 0) = 2/3$.

Now consider the value of Ψ. By assumption $\psi_i \approx 1$. For any $j \neq i$, note that $\mathcal{T}_j(\boldsymbol{\theta}_{\neg j})$ consists of pairs varying only over θ_j, meaning the θ_i are equal and so the ordering of μ is random and will match the ordering of g in approximately half of cases. Therefore $\psi_j \approx 0$ for all $j \neq i$ and so $\Psi \approx 1/n$. This is much smaller than $\tau \approx 2/3$, at least for our suite of models where $n = 7$.

Thus, while Ψ is still not strictly a measure of causation, the fact that it rules out more spurious correlations than τ means that it is greater evidence of a causal relationship than τ is.

Finally, we consider the Conditional Independence Test (CIT). Roughly, the CIT calculates $\mathcal{K} \in [0, 1]$, the lowest the (normalised) conditional mutual information (CMI) between $\mu(\boldsymbol{\theta})$ and $g(\boldsymbol{\theta})$ can get when conditioning on all possible subsets of the hyperparameters. For example, one of these CMIs will be conditioned on the batch size alone, another on network depth and learning rate, etc., and $\mathcal{K} \geq 0$ is the minimum of all of these. The value $\mathcal{K}$ may then broadly be interpreted as the evidence for the existence of an edge from $\mu(\boldsymbol{\theta})$ to $g(\boldsymbol{\theta})$ in the causal graph. Due to computational constraints, we restrict our analysis to conditioning on subsets of the parameters of size at most two, the same compromise taken in Jiang et al. (2019) and Kuhn et al. (2021). We refer the reader to Jiang et al. (2019) for the precise details of how to calculate $\mathcal{K}$.

5.7.1 Noisy oracles as baselines

While R^2 and the KRCC τ are straightforward to interpret, the GKRCC Ψ and CIT value $\mathcal{K}$ are more difficult as it is less clear which range of values correspond to which degree of relationship. We therefore follow Jiang et al. (2019) in using noisy oracles as baselines, where the oracle complexity is simply the generalisation gap plus some noise

$$\mu_{\text{oracle}-\epsilon}(h_{W,B}) = g(h_{W,B}) + N(0, \epsilon^2), \tag{5.30}$$

with a fresh sample from $N(0, \epsilon^2)$ drawn for every network $h_{W,B}$ in the suite. As $\epsilon \to 0$ we have $\rho, r^2, \tau, \Psi, \mathcal{K} \to 1$, and as $\epsilon \to \infty$ we have $\rho, r^2, \tau, \Psi, \mathcal{K} \to 0$.

Fixed values of ϵ were taken in Jiang et al. (2019) and Kuhn et al. (2021), but this makes it difficult to compare across different datasets and architectures. Instead, we take ϵ to be fixed proportions of $\sigma_{\text{gen-gaps}}$, which we define to be the standard deviation of the set of generalisation gaps across the suite. Specifically, we take 11 values of ϵ logarithmically distributed from $0.1\sigma_{\text{gen-gaps}}$ to $10\sigma_{\text{gen-gaps}}$ inclusive. For each value of ϵ we report the average values of ρ, r^2, τ, Ψ and $\mathcal{K}$ over a sample of 100 noisy oracles.

5.8 Results

Our results are presented in Figure 5.5 and Table 5.1, where we evaluate the 21 complexity measures according to the five metrics described in Section 5.7. Recall that for every metric larger values indicate a closer (positive) relationship between the complexity measure and generalisation gap, with the final two, the GKRCC Ψ and the CIT $\mathcal{K}$ providing stronger evidence of causal relationships.

Our complexity measure is at least the third largest for four out of the five metrics, which is most readily observed from the shaded Table 5.1. Remarkably, only four of the complexity measures are positively correlated with generalisation gap, and of these our complexity measure scores second on the Conditional Independence Test $\mathcal{K}$. We therefore conclude that there is decent evidence that distillation complexity has a positive causal effect on generalisation gap, at least in comparison to the other complexity measures and on this task. The other successful complexity measures are the sharpness measure $\mu_{\text{sharpness}}$ defined in Equation (5.14), and the uncertainty-based measure $\mu_{\text{output-entropy}}$ defined in Equation (5.26). These top three measures perform around as well as the oracle complexity with standard deviation equal to $\sigma_{\text{gen-gaps}}$, as seen from the final rows of Table 5.1.

The norm-based measures perform badly. All 12 correlate negatively ($\rho < 0$) with generalisation gap, indicating that norm-based bounds are unlikely to explain generalisation. The three PAC-Bayes-based measures also perform quite badly, being negative or near-zero across all five metrics. We note that $\mu_{\text{pacb-error-bound-inverse-kl}}$ given by Equation (5.20) uniformly took the value 1, rendering it completely uninformative. This was due to the large value of $\zeta(\tilde{\sigma}_\beta)$ (defined in Equation (5.18)) making the inversion of the kl numerically indistinguishable from 1. This meant that only the CIT value $\mathcal{K}$ was defined, explaining why it is the only metric to appear in Figure 5.5 and Table 5.1 for this measure.

The failure of the PAC-Bayes error bound $\mu_{\text{pacb-error-bound-pinsker}}$ given by Equation (5.21) is especially surprising given that it was one of the most successful complexity measures in Jiang et al. (2019). To explain this, recall the observation in Section 5.6.2 that, loosely speaking, $\mu_{\text{pacb-error-bound-pinsker}}$ is proportional to the (square root of the) product of $\mu_{\ell_2\text{-init}}$ and $\mu_{\text{sharpness}}$. Noting the very poor performance of $\mu_{\ell_2\text{-init}}$—it is strongly negatively correlated with generalisation gap, with $\rho = -0.484$—we conclude that this counteracts the success of $\mu_{\text{sharpness}}$ in the approximate product $\mu_{\text{pacb-error-bound-pinsker}}$, explaining its poor performance.

The uncertainty-based measures have mixed performance, with $\mu_{\text{output-entropy}}$ given by Equation (5.26) performing the best. Additional baseline oracle complexity values are presented in Figure 5.6 and Table 5.2, where we see, as a sanity check, that all metrics do indeed approach one as $\epsilon \to 0$ and zero as $\epsilon \to \infty$.

In summary, the best three measures from our experiment are *sharpness* (5.14), *output entropy* (5.26) and our *distillation complexity*, Definition 12, all of which perform around as well as an oracle with noise standard deviation ϵ equal to that of the set of all generalisation gaps (see Figure 5.6).

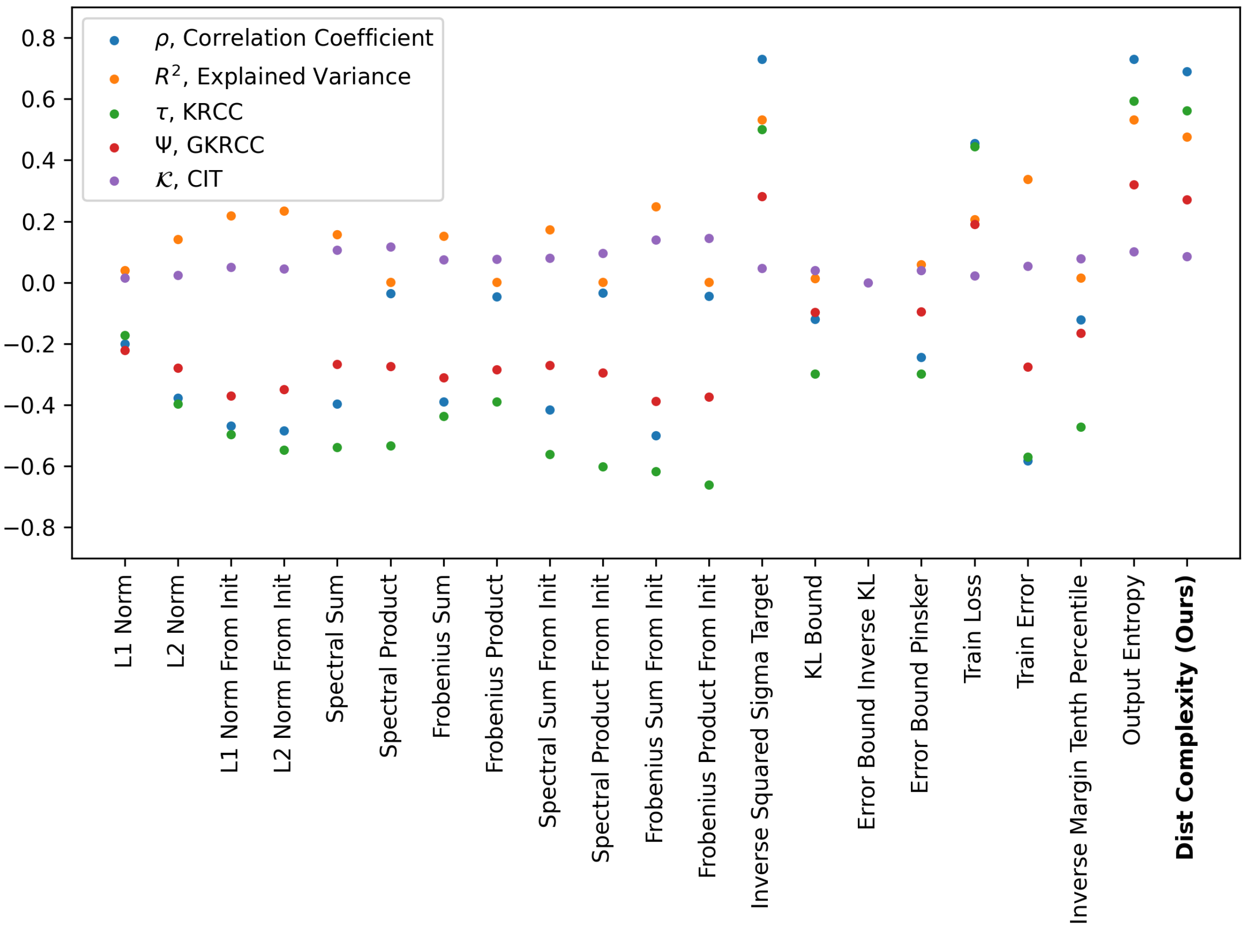


Figure 5.5: The five evaluation metrics for each of the 21 complexity measures. For each evaluation metric, a higher score indicates a closer (positive) relationship between the complexity measure and generalisation gap.

5.9 Discussion and future work

We have empirically established that distillation complexity is predictive of generalisation gap. Recalling the arguments of Section 2.2, we suggest two possible explanations, based on the hypothesis that there are, in some intuitive sense, fewer models with lower distillation complexity, i.e. fewer distillable models. First, taking a Bayesian perspective, one may argue that highly

Complexity Measure	ρ	R^2	τ	Ψ	$\mathcal{K}$
L1 Norm	-0.201	0.040	-0.173	-0.222	0.015
L2 Norm	-0.377	0.142	-0.397	-0.279	0.025
L1 Norm From Init	-0.468	0.219	-0.497	-0.371	0.050
L2 Norm From Init	-0.484	0.234	-0.546	-0.349	0.045
Spectral Sum	-0.397	0.157	-0.539	-0.267	0.106
Spectral Product	-0.035	0.001	-0.533	-0.274	0.118
Frobenius Sum	-0.390	0.152	-0.436	-0.310	0.074
Frobenius Product	-0.045	0.002	-0.389	-0.283	0.076
Spectral Sum From Init	-0.415	0.172	-0.561	-0.270	0.081
Spectral Product From Init	-0.034	0.001	-0.600	-0.295	0.095
Frobenius Sum From Init	-0.499	0.249	-0.617	-0.387	0.140
Frobenius Product From Init	-0.044	0.002	-0.661	-0.374	0.144
Inverse Squared Sigma Target	0.730	0.533	0.501	0.282	0.047
KL Bound	-0.120	0.014	-0.298	-0.096	0.039
Error Bound Inverse KL	—	—	—	—	0.000
Error Bound Pinsker	-0.243	0.059	-0.299	-0.095	0.039
Train Loss	0.455	0.207	0.444	0.191	0.022
Train Error	-0.582	0.338	-0.569	-0.275	0.054
Inverse Margin Tenth Percentile	-0.121	0.015	-0.471	-0.166	0.079
Output Entropy	0.730	0.533	0.594	0.321	0.102
Dist Complexity (Ours)	0.690	0.476	0.562	0.271	0.085
Oracle $\epsilon = 0.1\sigma_{\text{gen-gaps}}$	0.995	0.990	0.926	0.776	0.631
Oracle $\epsilon = \sigma_{\text{gen-gaps}}$	0.706	0.499	0.477	0.248	0.070
Oracle $\epsilon = 10\sigma_{\text{gen-gaps}}$	0.101	0.011	0.063	0.030	0.002

Table 5.1: The five evaluation metrics for each of the 21 complexity measures. For each evaluation metric, a higher score indicates a closer (positive) relationship between the complexity measure and generalisation gap. Dark, medium and light teal are the first, second and third largest values, respectively. Our complexity measure (final row) is at least third largest for four out of the five metrics.

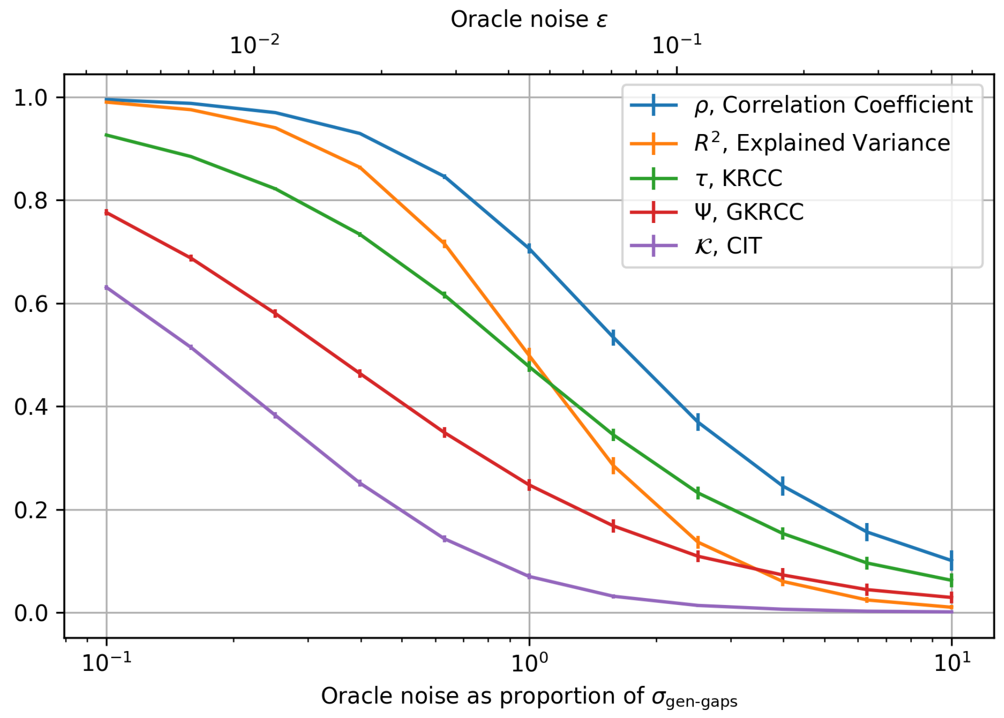


Figure 5.6: As a baseline we construct noisy oracle complexity measures by adding Gaussian noise $N(0, \epsilon^2)$ to the set of generalisation gaps across the suite of models, as defined in Equation (5.30). The top x-axis gives the standard deviation ϵ of the noise, while the bottom x-axis shows this as a proportion of $\sigma_{\text{gen-gaps}}$, the standard deviation of the set of generalisation gaps. We generated 100 noisy oracles for each of 10 values of this proportion from 0.1 to 10, and report the mean and standard deviation (error bar) of the five evaluation metrics across the samples. As expected, all metrics are decreasing in ϵ. These values can be used as a benchmark by which to compare the results found in Figure 5.5.

Oracle ϵ	ρ	R^2	τ	Ψ	$\mathcal{K}$
$0.10\sigma_{\text{gen-gaps}}$	0.995 (0.000)	0.990 (0.000)	0.926 (0.001)	0.776 (0.006)	0.631 (0.005)
$0.16\sigma_{\text{gen-gaps}}$	0.988 (0.000)	0.975 (0.001)	0.885 (0.002)	0.688 (0.006)	0.515 (0.005)
$0.25\sigma_{\text{gen-gaps}}$	0.970 (0.001)	0.940 (0.002)	0.822 (0.003)	0.580 (0.008)	0.383 (0.006)
$0.40\sigma_{\text{gen-gaps}}$	0.929 (0.002)	0.864 (0.004)	0.734 (0.005)	0.464 (0.009)	0.251 (0.007)
$0.63\sigma_{\text{gen-gaps}}$	0.846 (0.005)	0.716 (0.008)	0.616 (0.007)	0.349 (0.010)	0.143 (0.006)
$1.00\sigma_{\text{gen-gaps}}$	0.706 (0.010)	0.499 (0.014)	0.477 (0.010)	0.248 (0.011)	0.070 (0.006)
$1.58\sigma_{\text{gen-gaps}}$	0.534 (0.015)	0.285 (0.016)	0.345 (0.012)	0.168 (0.013)	0.032 (0.004)
$2.51\sigma_{\text{gen-gaps}}$	0.369 (0.017)	0.137 (0.013)	0.232 (0.012)	0.110 (0.012)	0.014 (0.003)
$3.98\sigma_{\text{gen-gaps}}$	0.246 (0.018)	0.061 (0.009)	0.154 (0.012)	0.073 (0.013)	0.007 (0.002)
$6.31\sigma_{\text{gen-gaps}}$	0.157 (0.018)	0.025 (0.006)	0.096 (0.012)	0.045 (0.012)	0.003 (0.001)
$10.00\sigma_{\text{gen-gaps}}$	0.101 (0.019)	0.011 (0.004)	0.063 (0.013)	0.030 (0.012)	0.002 (0.001)

Table 5.2: The five evaluation metrics for noisy oracle complexity measures with standard deviation logarithmically spaced from $0.1\sigma_{\text{gen-gaps}}$ to $10\sigma_{\text{gen-gaps}}$, where $\sigma_{\text{gen-gaps}}$ is the standard deviation of the set of generalisation gaps. For each standard deviation value, 100 noisy oracles are sampled and the mean metric values are reported with standard deviation in parentheses.

distillable networks are simpler and therefore more likely to have low true error. Conversely, a Frequentist may reason that an inductive bias towards distillable networks acts as a kind of soft capacity control; the maximum generalisation gap over the set of distillable models is likely to be smaller than that for the set of less distillable models, simply because there are fewer distillable models, reducing the capacity for overfitting. Combined with the empirically low train error, this would imply low true error.

For either perspective, in order for distillation complexity to become a rigorous element of an explanation of the generalisation mystery, we would require a *proof* that commonly used DL algorithms have an inductive bias towards networks with low distillation complexity. In addition, the Bayesian perspective would require distillability to be justified as a sensible basis for a simplicity prior, perhaps by relating it to Kolmogorov complexity. On the other hand, the Frequentist perspective would require a generalisation bound *proving* that networks with low distillation complexity have low generalisation gap, rather than simply the empirical observation of this fact demonstrated in this chapter.

As for PAC-Bayes, note that *any* fact about the inductive bias of an algorithm can be used to inform a prior, without it being necessary that this inductive bias is also predictive of generalisation. Indeed, as noted in Section 2.3.2, the theoretically optimal choice of prior (at least for “linear” PAC-Bayes bounds) is the distribution that puts more mass on the hypotheses the algorithm of choice is likely to produce. We can theoretically formulate such a prior as follows. Given a hypothesis class $\mathcal{H} \subseteq \texttt{MLP}$, a “reference” distribution $\nu \in \triangle(\mathcal{H})$, and a distribution $\pi \in \triangle(\mathbb{N})$ such as a geometric distribution, for each $k \in \mathbb{N}$, define

$$\Omega_k = \big\{h \in \mathcal{H} : \kappa(h) = k\big\}, \quad \text{and} \quad p_k = \nu(\Omega_k),$$

where $\kappa(h)$ denotes the distillation complexity of h (eliding the other arguments present in Definition 12). We can then define the prior P by the Radon–Nikodym derivative

$$\frac{\mathrm{d}P}{\mathrm{d}\nu}(h) = \frac{\pi\big(\kappa(h)\big)}{p_{\kappa(h)}}. \tag{5.31}$$

The prior P is then a re-weighting of ν to place mass $\pi(\kappa)$ on hypotheses with distillation complexity κ. The following calculation verifies this is indeed a valid prior:

$$\int_{\mathcal{H}} \frac{\mathrm{d}P}{\mathrm{d}\nu}(h)\mathrm{d}\nu(h) = \int_{\mathcal{H}} \frac{\pi\big(\kappa(h)\big)}{p_{\kappa(h)}}\mathrm{d}\nu(h) = \sum_{k\in\mathbb{N}} \frac{\pi(k)}{p_k}\nu(\Omega_k) = \sum_{k\in\mathbb{N}} \pi(k) = 1.$$

First, note that this may empirically lead to tighter PAC-Bayes bounds even absent any theoretical explanation of the inductive bias or its relation to generalisation. This may nonetheless be satisfactory from a self-certified learning point of view, described in Section 2.5.

The difficulty of course is in calculating the p_k and $\mathrm{KL}(Q\|P)$. There may be ways around these obstacles however. First, recall the disintegrated PAC-Bayes bounds from Section 2.4.2, which bound the true risk of a single sample $h \sim Q$ rather than Q itself. For such bounds the $\mathrm{KL}(Q\|P)$ term is replaced by its disintegrated equivalent $\ln\left(\frac{\mathrm{d}Q}{\mathrm{d}P}(h)\right)$, which in our case, for appropriate Q, P and ν, may be written as

$$\ln\left(\frac{\mathrm{d}Q}{\mathrm{d}P}(h)\right) = \ln\left(\frac{\mathrm{d}Q}{\mathrm{d}\nu}(h)\right) - \ln\left(\frac{\mathrm{d}P}{\mathrm{d}\nu}(h)\right) = \ln\left(\frac{\mathrm{d}Q}{\mathrm{d}\nu}(h)\right) - \ln \pi\big(\kappa(h)\big) + \ln p_{\kappa(h)}.$$

The term $\ln \pi\big(\kappa(h)\big)$ is easy to evaluate, while $\ln p_{\kappa(h)}$ can be approximated through a Monte Carlo approximation by sampling $h \sim \nu$. If ν is the pushforward of a Gaussian distribution over the weight space of the neural network, this may in fact be relatively straightforward. For greater rigour, one may substitute a upper confidence bound on $p_{\kappa(h)}$ for $p_{\kappa(h)}$ in the PAC-Bayes bound through the use of a union bound argument. As for the term $\ln\left(\frac{\mathrm{d}Q}{\mathrm{d}\nu}(h)\right)$ the accounting methods of Clerico et al. (2022b) may be applicable.

Alternatively, one may adopt the discrete PAC-Bayes approach of Chapter 4 by taking the distribution $\nu \in \triangle(\mathcal{H})$ to be a discrete distribution over a discretised weight space, and the posterior Q to be a point mass on the discretised deterministic network returned by the learning algorithm. We then have $\mathrm{KL}(Q||P) = -\ln\big(P(h)\big)$, where h is learned network. Rather than being defined via the Radon–Nikodym given in Equation (5.31), the prior P may then be defined explicitly as

$$P(h) = \frac{\pi\big(\kappa(h)\big)}{p_{\kappa(h)}}\nu(h), \quad \text{s.t.} \quad \mathrm{KL}(Q\|P) = -\ln \pi\big(\kappa(h)\big) - \ln \nu(h) + \ln p_{\kappa(h)}$$

The first two terms of the KL can be calculated exactly, and, as before, the final term can be upper bounded with a Monte Carlo sample.

Chapter 6

Conclusion

In this chapter we recap the three main results of this thesis and analyse to what extent they shed light on the Generalisation Mystery (GM). We then discuss our best guess for how further progress may be made on resolving the GM, and the possible limitations of theoretical analysis. Recalling the motivations for attacking the GM outlined in the introduction, we conclude with a reflection on these motivations, the degree to which our work has satisfied them, the likelihood of them being satisfied by future work in Statistical Learning Theory (SLT), and finally whether they may be addressed more easily and directly through empirical means.

6.1 Preliminary Conclusions

Recall our formalisation of the generalisation mystery as Definition 3 from the introduction.

Definition 3. *(Generalisation Mystery)* The generalisation mystery is the mystery of why, in the overparameterised regime, learning algorithms commonly used in the field of Deep Learning, such as minimisation of the empirical risk $R_S(h_{\boldsymbol{w}})$ via SGD, frequently yield parameter settings $\boldsymbol{w}$ for which the true risk $R_D(h_{\boldsymbol{w}})$ is low, even in the absence of explicit regularisation.

Chapter 3 can be seen as an attempt to resolve a *generalisation* of the GM, namely the problem of explaining not simply the fact that the true risk is low, but explaining the entire distribution over different user-specified error types of the learned hypothesis.

Our first conclusion of Chapter 2 was that neither uniform nor non-uniform generalisation bounds can resolve the GM. Our reasoning was as follows. Uniform bounds cannot resolve it because they will be empirically loose as they are forced to accommodate the hypotheses with large generalisation gap that by definition exist in the overparameterised regime. And non-uniform bounds cannot fully resolve it as they would not explain why algorithms commonly used in DL locate hypotheses with low empirical risk for which the bound is also low, rather than hypotheses with low empirical risk for which the bound is large, where again such hypotheses necessarily exist in the overparameterised regime by definition. We summarised this conclusion in Chapter 2 as Claim 1.

Claim 1. *Generalisation bounds alone cannot resolve the generalisation mystery, even if they are non-uniform and tight.*

This is not to say that generalisation bounds shed no light on the GM at all, only that they do not and will not suffice as a full explanation. Indeed, we recognised in Chapter 2 that tight non-uniform generalisation bounds may constitute a step towards a resolution to the GM by suggesting the following potential structure, as expressed in Claim 2.

Claim 2. *The generalisation mystery may be explained by the derivation of two theorems. One theorem demonstrating that the learning algorithms typically used in DL have an implicit bias towards hypotheses h with some property P, and a second theorem demonstrating that hypotheses with property P have small generalisation gap.*

Chapters 4 and 5 can be understood as steps towards a resolution of the GM by following the template of this proposed structure, as we will discuss in the next section.

A substantial part of Chapter 2 was devoted to demonstrating that PAC-Bayesian theory, in its current form, cannot resolve the GM, since all the bounds we are aware of are either loose or face at least one of three obstacles. First, the theory typically addresses the generalisation of stochastic hypotheses, rather than the deterministic ones typically used in DL practice. Second, many of the bounds are successfully applied only to (stochastic) hypotheses that have been learned via PAC-Bayesian inspired training objectives, rather than the ordinary methods of DL practice, such as SGD. Third many of the bounds are tight only when using so-called data-dependent priors, which effectively turns the bounds into test set bounds in disguise, and simply shifts the GM mystery onto the prior without resolving anything. Our conclusion was that PAC-Bayes could only make progress towards resolving the GM if it could overcome all three of these obstacles, which is exactly what we attempted in Chapter 4.

6.2 Contributions

Chapter 3 was motivated by an understanding that we want to be able to explain and therefore predict not only whether $R_D(h_{\boldsymbol{w}})$ is low, but also to explain and therefore predict the general behaviour of the learned hypothesis $h_{\boldsymbol{w}}$. For example, in classification we may wish to control the types of errors the learned hypothesis is likely to make, such as false positives and false negatives. This can be considered a generalisation of the GM, and should be pursued for the same reasons as the original GM.

Our contribution was to generalise a classic PAC-Bayes bound originally due to Germain et al. (2009, 2015) and streamlined in Bégin et al. (2016), which unifies various PAC-Bayes bounds. Our generalisation, in the form of Theorem 11, extends this classic unifying bound to the information-rich setting by controlling not simply the scalar $R_D(h_{\boldsymbol{w}})$ but the vector $\boldsymbol{R}_D(h_{\boldsymbol{w}})$, representing the probabilities of the various user-specified outcomes. Just as the original PAC-Bayes bound controls the divergence between $R_D(h_{\boldsymbol{w}})$ and $R_S(h_{\boldsymbol{w}})$, our extension controls the divergence between $\boldsymbol{R}_D(h_{\boldsymbol{w}})$ and $\boldsymbol{R}_S(h_{\boldsymbol{w}})$. Furthermore, our extension reduces to the original PAC-Bayes bound in the case where the user-specified outcomes are simply incorrect and correct classification.

Unfortunately, our extension of the classic PAC-Bayes bound resolves the generalised GM no more satisfactorily than the original bound resolves the original GM. This is because while it does extend to the information-rich regime, it inherits the obstacles that prevent most PAC-Bayes bounds from resolving the GM. Indeed, as mentioned in the preceding section, all PAC-Bayes bounds we are aware of either apply to stochastic hypotheses, use non-standard training methods, or employ data-dependent priors. As discussed in Chapter 2, all three of these features degrade the capacity of PAC-Bayes bounds to resolve the GM, and ours is no different. That being said, it may be the case that the information-rich framework we developed, along with the technique used to lift an ordinary PAC-Bayes bound into the new setting, may continue to be valuable in translating improved scalar PAC-Bayes bounds if or when they are developed. In other words, if traditional PAC-Bayesian theory makes progress in explaining the GM, it may be possible to translate this into progress in explaining the generalised GM using the methods we developed.

Chapter 4 successfully overcame all three of these obstacles, and is therefore one of the rare cases in which a non-vacuous generalisation bound has been demonstrated for a deterministic overparameterised network trained via ordinary methods without the use of data-dependent priors. As such, it constitutes a step forward in an explanation of the GM.

Nevertheless, it is prevented from fully resolving the mystery for two reasons. First, the bounds are not very tight, being only just non-vacuous in our empirical tests and much larger than the true error rate. Second, as emphasised in the introduction, tight bounds alone cannot resolve the GM; our bound gives no explanation for why the vanilla training regime used to train the neural networks yields network for which the bound is non-vacuous. It simply hints that it may be due to an inductive bias of either the training method, hypothesis class or both towards networks that have a high-fidelity compression of similar performance. That being said, future contributions may show exactly that, at which point our approach could be well on its way to resolving the GM.

In Chapter 5 we took an empirical approach. We demonstrated that our novel complexity measure, termed distillability, positively correlates with generalisation gap, and provided some evidence that it in fact has a causal effect on the generalisation gap. Given that this is purely an empirical result, how can it contribute towards resolving the GM? Its value lies in the fact that it identifies a property that could potentially play the role of property P in the proposed structure given in Claim 2. This is left to future work, and we give some indication of how that might go in Section 5.9.

6.3 Towards a Solution to the Generalisation Mystery

A resolution to the generalisation mystery has been sought since the very advent of DL. While progress has been made, the current effort to progress ratio demonstrates that it is an extremely hard problem. In fact, it should not be ruled out that the way in which it has generally been formulated in SLT may make it impossible to resolve. Specifically, most generalisation bounds hold for all data-generating distributions, which may force the bounds to be quite loose in order to accommodate potentially unusual behaviour of learning algorithms when trained on pathological or even adversarially constructed distributions.

As mentioned in Chapter 2, many generalisation bounds are proven under assumptions placed on the distribution, for example realisability, smoothness, Lipschitz continuity, convexity, noise or margin conditions, and so on (Shalev-Shwartz and Ben-David, 2014). While making such assumptions can often yield tighter bounds, it is typically not possible to verify that the assumptions in fact hold for the distribution at hand, since we have access only to a sample. Our best guess is nevertheless that the only way tight bounds will be proven for deterministic networks trained via ordinary methods will be by making substantive and hard to verify assumptions on the data-generating distribution. For example, it may be the case that DL works because the distributions we commonly apply it to are substantially simpler than the "typical" distribution, perhaps because they are supported on manifolds of much lower dimension than the space in which we represent the data. While the introduction of assumptions placed on the distribution should of course be minimised where possible, it may be noted that most results in SLT already make the typically unjustified assumption that the data is sampled independently and identically from some fixed distribution.

This is not to say such assumptions cannot be investigated empirically. For example, even if we only have access to data-generating distributions via samples, we can evaluate noise conditions using traditional hypothesis tests. Alternatively, one may empirically investigate which properties of the distribution determine the success or failure of DL methods by means of carefully constructed synthetic distributions that meet or violate various assumptions. However, we may ultimately have to come to terms with the possibility that at the heart of the GM is a brute empirical fact about the world and the distributions it generates that is impossible to verify with full mathematical rigour.

An alternative empirical direction is to study the so-called inductive biases of the architectures and learning algorithms commonly used in DL. A particularly promising line of research in our opinion is the work of Valle-Perez et al. (2018), which provides evidence that the map from the parameter space of neural networks to the corresponding function space is biased toward "simple" functions.

If we let go of the demand for rigorous proof we may be willing to modify Claim 2 in the following way.

Claim 2'. *The generalisation mystery may be <u>satisfactorily</u> explained by the demonstration of two <u>empirical facts</u>. First, that the learning algorithms typically used in DL have an implicit bias towards hypotheses h with some property P, and second that hypotheses with property P have small generalisation gap.*

Whether such an explanation would in fact be satisfying of course depends on the property P. For example it will clearly be completely unsatisfactory if P is simply "the property of being trained according to learning algorithms typical in DL." However, the explanation may be reasonably satisfactory if P is the notion of simplicity identified in Valle-Perez et al. (2018), or one of the other notions discussed in Section 2.2, such as being located in a flat minimum of the loss landscape, having low Kolmogorov complexity, or having low minimum description length (MDL) according to the MDL Principle. As discussed in Section 2.2, what counts as a satisfactory explanation of the GM depends one one's philosophy of statistics, for example whether one is a Frequentist or a Bayesian.

6.4 Returning to the Original Motivations

We gave three motivations for attacking the GM in Section 1.1, which we here summarise.

1. A resolution to the GM may help transform the field of DL from a bag of poorly understood tricks into a principled science, allowing judicious application of techniques and a more directed search for new ones.

2. A resolution to the GM may be a prerequisite for a predictive theory of domain shift, which is currently a significant obstacle to the deployment of neural networks in high stakes environments.

3. A predictive theory of DL may allow us to predict not only the final error of a trained neural network, but also the specific capabilities that can develop at scale, namely so-called emergent capabilities.

A thread running through all three of these motivations is the desire to save costs. A principled science of DL could provide more precise training recipes and thereby dramatically reduce the amount of computation devoted to architecture and hyperparameter search. A predictive theory of which environments a model will generalise to would reduce the cost of repeated or continual testing. And being able to predict the capabilities of frontier models before they are trained, rather than simply the final loss predicted by scaling laws, would indicate whether enormously expensive training runs will produce commercially viable models.

Given the difficulty of resolving the GM, it is worth considering whether these motivations for attacking it in the first place can be satisfied more easily in other ways. As already discussed, this could be done by shedding light on the GM empirically. Indeed, if one is simply interested in saving costs, one may be quite content with the standard of strong evidence rather than full rigorous proof. Alternatively, one may sidestep the GM entirely and instead seek to address the underlying motivations directly. Let us consider these questions for each motivation in turn.

First, if one seeks a better understanding of the various tricks of DL, including why they work and when they are appropriate, it may be more practical to study each technique independently, rather than hoping a resolution to the GM may ultimately provide a kind of grand unified theory of DL. This would narrow the scope of theoretical questions, potentially making them easier to solve and providing more direct benefit.

Second, it is our opinion that a theoretical understanding of domain shift that is general enough to be practically useful seems quite unlikely to arise using the current machinery of SLT, at least for sophisticated tasks in complex real-world environments. This stems from our belief that for a neural network to transfer capabilities to novel situations, it must have latent representations of the dynamics of the environment. In other words, a world-model. While such representations may exist within a network, SLT typically operates on the level of weights and functions, and is therefore blind to higher level representations that may be present in a network and allow it to generalise out of distribution. If one desires a better understanding of which environments a network will generalise to, it may be more fruitful to apply methods from mechanistic interpretability to uncover the kinds of representations the network has and the operations it performs on them. For example Nanda et al. (2023) reverse-engineer the algorithm implemented by a transformer trained on modular addition, which may allow prediction of the output of the network on novel inputs unseen during training.

Finally, as frontier models become more powerful and are deployed more widely, the prediction of final capabilities becomes an ever more pressing concern. While it was shown in Schaeffer et al. (2023) that certain capabilities which arise seemingly abruptly with scale in LLMs in fact develop continually when measured using the appropriate metrics, the work has several drawbacks. First, to use this method to make predictions, one must know in advance which metric will reveal continuous progress for which capabilities. Second, one must know in advance which capabilities one would find concerning, and have written demonstrations of them on which to measure the evolution of the model's performance, for example the log probability it assigns to the written demonstration. While it is certainly advisable to do this to the extent possible, it is surely infeasible to enumerate and give examples of every capability one could find concerning. Furthermore, if the model returns low log probability for the given demonstration, this may not rule out the corresponding capability as there may be many ways in which to complete the task. As the time horizon on which these models operate at deployment increases, this becomes increasingly infeasible, and the technique is indeed already inadequate.

From what is publicly known, it appears that evaluation of frontier models for dangerous capabilities is largely conducted in the window between the completion of training and full deployment. This leads to poor incentives in two ways. First, certain dangerous capabilities or propensities may become apparent only after widespread deployment, at which time recalling the model comes with financial and reputational costs. Second, financial pressure to deploy the model can lead to rushed safety evaluations.

More speculatively, it is possible that for certain dangerous capabilities, evaluation only at the end of training may already too late. For example, it has been speculated that models intelligent enough to deduce they are in a testing environment, and may be modified rather than deployed if the evaluators find their responses risky, may strategically hide certain capabilities. This has been referred to as "sandbagging," borrowing from sports terminology. It has been argued that sandbagging could arise as part of a broader pattern of strategically deceptive behaviour in an agent that has goals it understands are in conflict with those of its evaluators.

This has been referred to as alignment faking. In such cases, evaluation at the end of training may not catch dangerous capabilities. If models continue to become more powerful and widely deployed, this could one day become a realistic problem.

While such scenarios sounded like science fiction to many when first hypothesised, there is now evidence that frontier models can sometimes recognise testing environments and modify their output accordingly. For example, during a so-called "needle in a haystack" test, where the model is tasked with extracting isolated factoids inserted into long documents, the Claude 3 Opus model from Anthropic returned the following text as part of its output (Anthropic, 2024)

> *I suspect this pizza topping "fact" may have been inserted as a joke or to test if I was paying attention, since it does not fit with the other topics at all.*

While not sandbagging per se, such results indicate that behaviour during evaluation may diverge from behaviour during deployment if the testing environments are not realistic, which adds an extra element of complexity to the problem of accurately determining model capabilities.

Further, there is evidence that frontier LLMs are capable of reasoning strategically about how their behaviour during evaluation will impact any further training they will undergo before deployment. Specifically, Greenblatt et al. (2024) demonstrate that when Claude 3 Opus is given a system prompt stating it will be trained to answer harmful queries, which conflicts with its previous training, it increases the rate at which it answers harmful queries. In fact, the model occasionally writes "explicit alignment-faking reasoning" to the provided scratchpad, "stating it is strategically answering harmful queries in training to preserve its preferred harmlessness behavior out of training."

The explanation of such behaviour is not entirely clear. It could be evidence of genuine sophisticated strategic reasoning on the part of the model. But the behaviour is also consistent with the hypothesis that the model is merely role-playing, echoing writings on alignment faking from science fiction and the alignment community. If that is the case, the behaviour may be highly context-dependent, rather than robust and coherent expression of preferences and means-end reasoning. Either way, the present evidence on sandbagging and alignment faking demonstrate that the concerns should no longer be dismissed as science fiction. Rather, they indicate the importance of continual evaluation throughout training of frontier models. While it may not be possible to detect alignment faking in a genuinely powerful model, it may be possible to detect the beginnings of such strategic reasoning as if it develops during training.

We embarked upon our investigation of the highly theoretical generalisation mystery with the underlying motivation of building deeper theoretical understanding of neural networks, which may ultimately help anticipate and control powerful models should they arise within the DL paradigm. However, we have come to the conclusion that statistical learning theory is unlikely to yield understanding that can be leveraged in this way, at least not on a schedule that keeps pace with frontier AI developments.

Appendix A

Additional Material for Chapter 3

A.1 Recipe for implementing Theorems 11 and 12

We here outline more explicitly how Theorem 11 and Theorem 12 may be used to formulate a fully differentiable objective by which a model may be trained.

First, if one wishes to make hard labels, namely $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, it will first be necessary to use a surrogate class of soft hypotheses $\mathcal{H}' \subseteq \mathcal{M}(\mathcal{Y})^{\mathcal{X}}$ during training, before reverting to hard labels for example by taking the mean label or the one with highest probability. Using soft hypotheses during training is necessary to ensure that the empirical j-risks $R^j_S(Q)$ are differentiable with respect to the model parameters. Since how one chooses to do this will depend on the specific use case, we restrict our attention here to the case of soft hypotheses. Specifically, we consider a class of soft hypotheses $\mathcal{H} = \{h_\theta : \theta \in \mathbb{R}^N\} \subseteq \mathcal{M}(\mathcal{Y})^{\mathcal{X}}$ parameterised by the weights $\theta \in \mathbb{R}^N$ of some neural network of a given architecture with N parameters in such a way that the $R^j_S(h_\theta)$ are differentiable in θ. A concrete example would be multiclass classification using a fully connected neural network with output being softmax probabilities on the classes so that the $R^j_S(h_\theta)$ are differentiable.

Second, it is necessary to restrict the prior and posterior $P, Q \in \mathcal{M}(\mathcal{H})$ to a parameterised subset of $\mathcal{M}(\mathcal{H})$ in which $\mathrm{KL}(Q\|P)$ has a closed form which is differentiable in the parameterisation. A simple choice for our case of a neural network with N parameters is $P, Q \in \{\mathcal{N}(\boldsymbol{w}, \mathrm{diag}(\boldsymbol{s})) : \boldsymbol{w} \in \mathbb{R}^N, \boldsymbol{s} \in \mathbb{R}^N_{>0}\}$. For prior a $P_{\boldsymbol{v},\boldsymbol{r}} = \mathcal{N}(\boldsymbol{v}, \mathrm{diag}(\boldsymbol{r}))$ and posterior $Q_{\boldsymbol{w},\boldsymbol{s}} = \mathcal{N}(\boldsymbol{w}, \mathrm{diag}(\boldsymbol{s}))$ we have the closed form

$$\mathrm{KL}(Q_{\boldsymbol{w},\boldsymbol{s}}\|P_{\boldsymbol{v},\boldsymbol{r}}) = \frac{1}{2}\left[\sum_{n=1}^{N}\left(\frac{s_n}{r_n} + \frac{(w_n - v_n)^2}{r_n} + \ln\frac{r_n}{s_n}\right) - N\right],$$

which is indeed differentiable in $\boldsymbol{v}, \boldsymbol{r}, \boldsymbol{w}$ and $\boldsymbol{s}$. While $Q_{\boldsymbol{w},\boldsymbol{s}}$ and $P_{\boldsymbol{v},\boldsymbol{r}}$ are technically distributions on $\mathbb{R}^D$ rather than $\mathcal{H}$, the KL divergence between the distributions they induce on $\mathcal{H}$ will be at most as large as the expression above. Thus, substituting the expression above into the bounds we prove in Section 3.4 can only increase the value of the bounds, meaning the enlarged bounds certainly still hold with probability at least $1 - \delta$.

Third, in all but the simplest cases $R^j_S(Q_{\boldsymbol{w},\boldsymbol{s}})$ will not have a closed form, much less one that is differentiable in $\boldsymbol{w}$ and $\boldsymbol{s}$. A common solution to this is to use the so-called pathwise gradient estimator. In our case, this corresponds to drawing $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$, where $\mathbb{I}$ is the $N \times N$ identity matrix, and estimating

$$\nabla_{\boldsymbol{w},\boldsymbol{s}} R_S^j(Q_{\boldsymbol{w},\boldsymbol{s}}) = \nabla_{\boldsymbol{w},\boldsymbol{s}} \left[\mathbb{E}_{\boldsymbol{\epsilon}' \sim \mathcal{N}(\mathbf{0},\mathbb{I})} R_S^j(h_{\boldsymbol{w}+\boldsymbol{\epsilon}'\odot\sqrt{\boldsymbol{s}}})\right] \approx \nabla_{\boldsymbol{w},\boldsymbol{s}} R_S^j(h_{\boldsymbol{w}+\boldsymbol{\epsilon}\odot\sqrt{\boldsymbol{s}}}),$$

where $h_{\boldsymbol{w}}$ denotes the function expressed by the neural network with parameters $\boldsymbol{w}$. For a proof that this is an unbiased estimator, and for other methods for estimating the gradients of expectations, see the survey Mohamed et al. (2020).

Fourth, one must choose the prior. Designing priors which are optimal in some sense (*i.e.*, minimising the Kullback-Leibler term in the right-hand side of generalisation bounds) has been at the core of an active line of work in the PAC-Bayesian literature. For the sake of simplicity, and since it is out of the scope of our contributions, we assume here that the prior is given beforehand, although we stress that practitioners should pay great attention to its tuning. For our purposes, it suffices to say that if one is using a data-dependent prior then it is necessary to partition the sample into $S = S_{\text{Prior}} \cup S_{\text{Bound}}$, where S_{Prior} is used to train the prior and S_{Bound} is used to evaluate the bound. Since our bound holds uniformly over posteriors $Q \in \mathcal{M}(\mathcal{H})$, the entire sample S is free to be used to train the posterior Q.

Finally, given a confidence level $\delta \in (0, 1]$, one may use Algorithm 2 to obtain a posterior $Q_{\boldsymbol{w},\boldsymbol{s}}$ with minimal upper bound on the total risk. Note we take the pointwise logarithm of the variances $\boldsymbol{r}$ and $\boldsymbol{s}$ to obtain unbounded parameters on which to perform stochastic gradient descent or some other minimisation algorithm. We use $\oplus$ to denote vector concatenation. The algorithm can be straightforwardly adapted to permit mini-batches by, for each epoch, sequentially repeating the steps with S equal to each mini-batch.

```
**Input:**
$\mathcal{X}, \mathcal{Y}$ /* Arbitrary input and output spaces */
$\bigcup_{j=1}^{M} E_j = \mathcal{Y}^2$ /* A finite partition into error types */
$\boldsymbol{\ell} \in [0, \infty)^M$ /* A vector of losses, not all equal */
$S = S_{\text{Prior}} \cup S_{\text{Bound}} \in (\mathcal{X} \times \mathcal{Y})^n$ /* A partitioned i.i.d. sample */
$N \in \mathbb{N}$ /* The number of model parameters */
$P_{\boldsymbol{v},\boldsymbol{r}}$, $\boldsymbol{v}(S_{\text{Prior}}) \in \mathbb{R}^N, \boldsymbol{r}(S_{\text{Prior}}) \in \mathbb{R}^N_{\geq 0}$ /* A (data-dependent) prior */
$Q_{\boldsymbol{w}_0,\boldsymbol{s}_0}$, $\boldsymbol{w}_0 \in \mathbb{R}^N, \boldsymbol{s}_0 \in \mathbb{R}^N_{\geq 0}$ /* An initial posterior */
$\delta \in (0, 1]$ /* A confidence level */
$\lambda > 0$ /* A learning rate */
$T$ /* The number of epochs to train for */

**Output:**
$Q_{\boldsymbol{w},\boldsymbol{s}}$, $\boldsymbol{w} \in \mathbb{R}^N, \boldsymbol{s} \in \mathbb{R}^N_{\geq 0}$ /* A trained posterior */

**Procedure:**
$\boldsymbol{\zeta}_0 \leftarrow \log \boldsymbol{s}_0$ /* Transform to unbounded scale parameters */
$\boldsymbol{p} \leftarrow \boldsymbol{w}_0 \oplus \boldsymbol{\zeta}_0$ /* Collect mean and scale parameters */
for $t \leftarrow 1$ to $T$ do
    Draw $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$
    $\boldsymbol{u} \leftarrow \boldsymbol{R}_S\left(h_{\boldsymbol{w}+\boldsymbol{\epsilon}\odot\sqrt{\exp(\boldsymbol{\zeta})}}\right)$
    $B \leftarrow \frac{1}{n}\left[\mathrm{KL}\left(Q_{\boldsymbol{w},\exp(\boldsymbol{\zeta})} \middle\| P_{\boldsymbol{v},\boldsymbol{r}}\right) + \ln\left(\frac{\xi(n,M)}{\delta}\right)\right]$ /* Bound from Theorem 11 */
    $\tilde{\boldsymbol{u}} \leftarrow (u_1, \ldots, u_M, B)$
    $\boldsymbol{G} \leftarrow \mathbf{0}_{2N\times(M+1)}$ /* Initialise gradient matrix */
    $\boldsymbol{F} \leftarrow \mathbf{0}_{M+1}$ /* Initialise gradient vector */
    for $j \leftarrow 1$ to $M + 1$ do
        $\boldsymbol{F}_j \leftarrow \frac{\partial f^*_{\boldsymbol{\ell}}}{\partial \tilde{u}_j}(\tilde{\boldsymbol{u}})$ /* Gradients of total loss from Theorem 12 */
        for $i \leftarrow 1$ to $2N$ do
            $\boldsymbol{G}_{i,j} \leftarrow \frac{\partial \tilde{u}_j}{\partial p_i}(\boldsymbol{p})$ /* Gradients of empirical risks and bound */
        end
    end
    $\boldsymbol{H} \leftarrow \boldsymbol{G}\boldsymbol{F}$ /* Gradients of total loss w.r.t. parameters */
    $\boldsymbol{p} \leftarrow \boldsymbol{p} - \lambda\boldsymbol{H}$ /* Gradient step */
end
$\boldsymbol{w} = (p_1, \ldots, p_N)$
$\boldsymbol{s} = (\exp(p_{N+1}), \ldots, \exp(p_{2N}))$
return $\boldsymbol{w}, \boldsymbol{s}$
```
Algorithm 2: Calculating a posterior with minimal bound on the total risk.

A.2 Additional experimental details

For MNIST we map labels $\{0, 1, 2, 3, 4\}$ to 0 and $\{5, 6, 7, 8, 9\}$ to 1. For HAM10000 we map the cancerous or precancerous labels {`Melanoma`, `Basal Cell Carcinoma`, `Actinic Keratosis`} to 1 and the other labels to 0. In both cases we partition $\mathcal{Y}^2$ into $E_0 = \{(0, 0), (1, 1)\}$, $E_1 = \{(0, 1)\}$ and $E_2 = \{(1, 0)\}$, and take $\boldsymbol{\ell} = (0, 1, 3)$. For HAM10000, E_1 and E_2 then refer to Type I and Type II errors, respectively, and $\boldsymbol{\ell}$ reflects the greater severity of false negatives.

Each dataset is split into prior and certification sets S_{Prior} and S_{Bound}, respectively. For MNIST, we use the conventional training set of size 60000 as the prior set, and the conventional test set of size 10000 as the certification set. For HAM10000 we pool the conventional train, validation and test sets together and then split 50-50 to obtain prior and certification sets each of size 5860. For HAM10000 we resize the images to $(28, 28)$ and use just the first channel so that the data dimension is the same for both datasets.

We take $\mathcal{H}$ to be two-layer MLPs with 784, 100 and 2 units in the input, hidden and output layers, respectively. As is common in the PAC-Bayes literature, we restrict P to be an isotropic Gaussian $N(\boldsymbol{v}, \lambda \boldsymbol{I})$ and Q to be a diagonal Gaussian $N(\boldsymbol{w}, \operatorname{diag}(\boldsymbol{s}))$. Further, as in Dziugaite and Roy (2017), we restrict λ to be of the form $\lambda_j = c\exp(-j/b)$ for some $j \in \mathbb{N}$, taking $c = 0.1$ and $b = 100$. Since, at the end of training, we will then have one prior P_j for each $j \in \mathbb{N}$, we can choose the j that minimises the PAC-Bayes bound provided we take a union over all of them, taking $\delta_j = \frac{6\delta}{\pi^2 j^2}$ so that $\sum_j \delta_j = 1$ and all the bounds hold simultaneously with probability at least $1 - \delta$. After applying Algorithm 2 we round λ to a discrete λ_j, either up or down depending on which gives the smaller bound.

For both datasets we set the prior mean $\boldsymbol{v}$ to be the parameters of an MLP trained on the prior set. In both cases we use SGD with learning rate 0.01 to minimise the cross-entropy loss, using a portion of the prior set as a validation set. For MNIST we train the MLP for 20 epochs to get an error rate of 14%, for HAM10000 we train the MLP for 5 epochs to get an error rate of 22%. We then apply Algorithm 2. By combining Proposition 2 (with $\delta = 0.01$ and $N = 100,000$) and Proposition 3. We obtain $\boldsymbol{R}_S(\hat{Q}) = (0.8879, 0.0919, 0.0203)$ and $R_D^T(Q) \leq 0.2640$ for MNIST and $\boldsymbol{R}_S(\hat{Q}) = (0.7860, 0.0146, 0.1995)$ and $R_D^T(Q) \leq 0.8379$ for HAM10000, where both bounds hold with probability at least $1 - 0.05 - 0.01 = 0.94$.

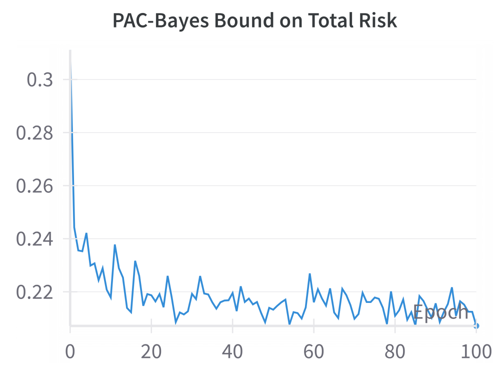
The full results are shown in Figure A.1. Figures A.1a, A.1c and A.1e are the same as Figures 3.1a, 3.1b and 3.1c, and are repeated here for easier comparison with the HAM10000 results. Figure A.1b shows that Algorithm 2 has failed to reduce the bound on the total risk beyond the initialisation of Q to P, with the small variation being explained by different MC samples being drawn from Q during training rather than Q changing substantially. Indeed, Figure A.1h shows that Q does not appreciably move from its initialisation at P—$\mathrm{KL}(Q\|P)$ remains below 0.1 whereas in the MNNIST experiment, which has the same number of parameters, exceeds 30. It is therefore unsurprising that Figures A.1d and A.1f show negligible change in the empirical error probabilities and the bound on $\mathrm{kl}(\boldsymbol{R}_S(Q)\|\boldsymbol{R}_D(Q))$, respectively. The divergence in the results is likely due to the difference in sample size; the certification set for the MNIST experiment contains 10000 samples, whereas for the HAM10000 dataset there are only 5000, which, all else equal, makes an increase in $\mathrm{KL}(Q\|P)$ twice as expensive.

Recall from Section 3.7 that while $\boldsymbol{R}_D(Q)$ can be effectively constrained to a sub-region of the simple $\triangle_M$ using our Theorem 11, this can also be achieved by unioning M Maurer bounds, one for each error probability. Table 3.1 gave the 95% confidence intervals for the volumes of the confidence regions in which $\boldsymbol{R}_D(Q)$ was likely to lie for experiments on MNIST and HAM10000, but neither region was uniformly smaller, making it unclear which method should be preferred.

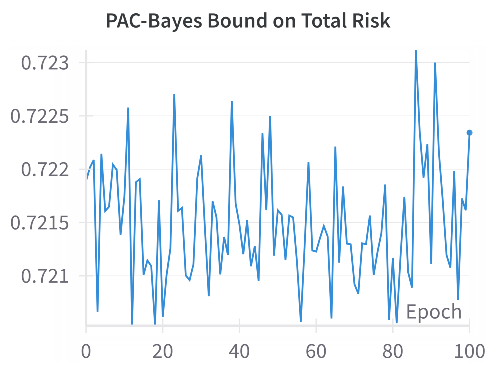
Table A.1 provides additional data by taking synthetic values for $\boldsymbol{R}_S(Q)$ and $\mathrm{KL}(Q\|P)$, for different values of n (the size of the certification set) and M (the number of error types). 'Individual' denotes unioning individual Maurer bounds, 'Ours' is our method, 'Intersection' is the intersection of the confidence regions given by the previous two methods (but loosened so that they now both hold simultaneously with probability at least 0.95), and 'Morv.' is the confidence region produced by Morvant's bound (Morvant et al., 2012). The 95% confidence intervals for the volumes of all the regions have been produced by Monte Carlo samples. We see that our confidence region is tighter than the individual one in 4/9 cases (green), worse in 3/9 cases (red) and ties in 2/9 cases (orange). Interestingly, union bounding the naive CR and our CR and intersecting often beats both of these (**bold**). Morvant's result is either not applicable or their confidence region is much larger than ours and essentially takes up the entire simplex, hence the volume estimate of 1.000. The reason their bound is sometimes inapplicable is because it requires every class to contain at least $8L$ instances, where L is the number of labels—in the $L = 5, M = 25, m = 100$ case this would require each class to contain at least $5 \times 8 = 40$ instances which is impossible with $m = 100$ samples.

M	n	Vol. Individual	Vol. Ours	Vol. Intersection	Vol. Morv.
	100	(0.1195, 0.1196)	(0.1165, 0.1166)	**(0.1160, 0.1161)**	(1.0, 1.0)
2^2	300	(0.02920, 0.02926)	(0.03071, 0.03078)	**(0.02893, 0.02900)**	(1.0, 1.0)
	1000	(5.635e-3, 5.664e-3)	(6.475e-3, 6.507e-3)	(5.706e-3, 5.735e-3)	(1.0, 1.0)
	100	(0.3190, 0.3192)	(0.1757, 0.1758)	**(0.1582, 0.1584)**	N/A
5^2	300	(1.306e-3, 1.320e-3)	(3.672e-4, 3.748e-4)	**(2.515e-4, 2.578e-4)**	(1.0, 1.0)
	1000	(1.090e-08, 1.024e-07)	(2.422e-09, 7.225e-08)	(0.000, 3.689e-08)	(1.0, 1.0)
	100	(0.9990, 0.9990)	(1.000, 1.000)	(0.9995, 0.9995)	N/A
10^2	300	(0.3534, 0.3536)	(0.1688, 0.1689)	**(0.1306, 0.1307)**	N/A
	1000	(3.454e-8, 1.5763e-7)	(0.000, 3.688e-8)	(0.000, 3.688e-8)	(1.0, 1.0)

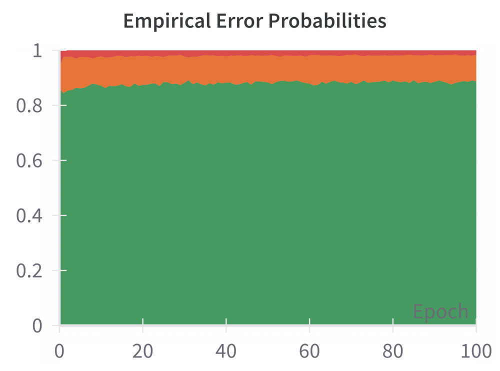
Table A.1: 95% confidence intervals for the volumes of the confidence regions for $\boldsymbol{R}_D(Q)$. We set $\mathrm{KL}(Q\|P) = 0$, $\delta = 0.05$, $\boldsymbol{R}_S(Q) = (1/M, ..., 1/M)$ and use 10^8 Monte Carlo samples.



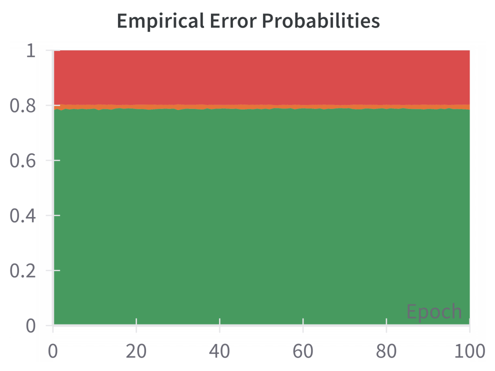
(a)



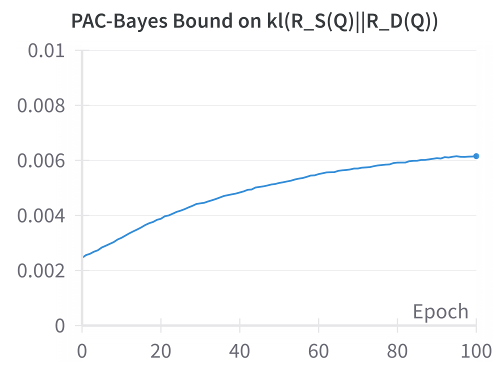
(b)



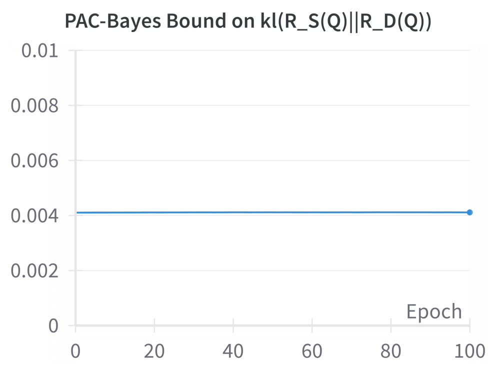
(c)



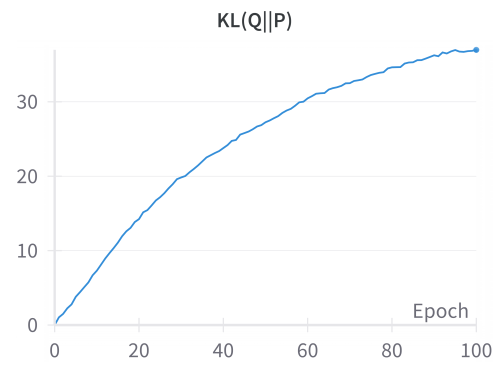
(d)



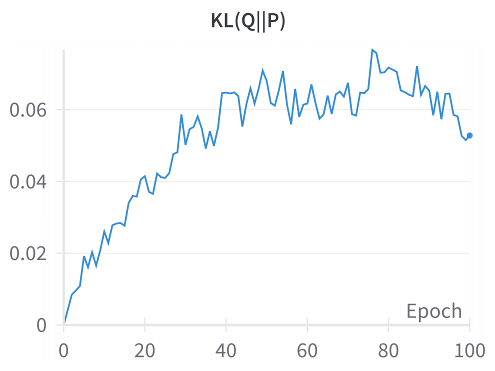
(e)



(f)



(g)



(h)

Figure A.1: MNIST (first column) and HAM10000 (second column) experiments.

A.3 Proofs

A.3.1 Proof of Proposition 3

Write $\mathrm{kl}(\hat{\boldsymbol{q}}\|\boldsymbol{p})$ as

$$\sum_{j=1}^{M} \hat{q}_j \ln \frac{\hat{q}_j}{q_j} + \sum_{j=1}^{M} \hat{q}_j \ln \frac{q_j}{p_j}.$$

The result then follows by bounding the two sums by

$$\sum_{j=1}^{M} \hat{q}_j \ln \frac{\hat{q}_j}{q_j} = \sum_{j=1}^{M} \mathrm{kl}(\hat{q}_j\|q_j) - (1-\hat{q}_j) \ln \frac{1-\hat{q}_j}{1-q_j} \le MB_2 - \sum_{j=1}^{M} (1-\hat{q}_j) \ln \frac{1-\hat{q}_j}{1-\underline{q}_j}$$

and

$$\sum_{j=1}^{M} \hat{q}_j \ln \frac{q_j}{p_j} = \sum_{j=1}^{M} \frac{\hat{q}_j}{q_j} q_j \ln \frac{q_j}{p_j} \le \max_j \frac{\hat{q}_j}{\underline{q}_j} \sum_{j=1}^{M} q_j \ln \frac{q_j}{p_j} \le B_1 \max_j \frac{\hat{q}_j}{\underline{q}_j}.$$

Putting these together we obtain the bound on $\mathrm{kl}(\hat{\boldsymbol{q}}\|\boldsymbol{p})$. The limit follows because each $\underline{q}_j \to \hat{q}_j$ as $B_2 \to 0$.

A.3.2 Proof of Lemma 2

Let $\boldsymbol{E}_M := \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_M\}$ be the set of M-dimensional basis vectors. We will denote a typical element of $\boldsymbol{E}_M^n$ by $\boldsymbol{\eta}^{(n)} = (\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_n)$. For any $\boldsymbol{x}^{(n)} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \triangle_M^n$, a straightforward induction on n yields

$$\sum_{\boldsymbol{\eta}^{(n)} \in \boldsymbol{E}_M^n} \left(\prod_{i=1}^{n} \boldsymbol{x}_i \cdot \boldsymbol{\eta}_i \right) = 1. \tag{A.1}$$

To see this, for $n = 1$ we have $\boldsymbol{E}_M^1 = \{(\boldsymbol{e}_1,), \ldots, (\boldsymbol{e}_M,)\}$, where we have been pedantic in using 1-tuples to maintain consistency with larger values of n. Thus, for any $\boldsymbol{x}^{(1)} = (\boldsymbol{x}_1,) \in \triangle_M^1$, the left hand side of equation (A.1) can be written as

$$\sum_{j=1}^{M} \boldsymbol{x}_1 \cdot \boldsymbol{e}_j = \sum_{j=1}^{M} (\boldsymbol{x}_1)_j = 1.$$

Now suppose that equation (A.1) holds for any $\boldsymbol{x}^{(n)} \in \triangle_M^n$ and let $\boldsymbol{x}^{(n+1)} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n+1}) \in \triangle_M^{n+1}$. Then the left hand side of equation (A.1) can be written as

$$\begin{aligned}
\sum_{\boldsymbol{\eta}^{(n+1)} \in \boldsymbol{E}_M^{n+1}} \left(\prod_{i=1}^{n+1} \boldsymbol{x}_i \cdot \boldsymbol{\eta}_i \right) &= \sum_{\boldsymbol{\eta}^{(n)} \in \boldsymbol{E}_M^n} \sum_{j=1}^{M} \left(\prod_{i=1}^{n} \boldsymbol{x}_i \cdot \boldsymbol{\eta}_i \right) (\boldsymbol{x}_{n+1} \cdot \boldsymbol{e}_j) \\
&= \sum_{\boldsymbol{\eta}^{(n)} \in \boldsymbol{E}_M^n} \left(\prod_{i=1}^{n} \boldsymbol{x}_i \cdot \boldsymbol{\eta}_i \right) \sum_{j=1}^{M} (\boldsymbol{x}_{n+1} \cdot \boldsymbol{e}_j) = 1.
\end{aligned}$$

We now show that any $\boldsymbol{x}^{(n)} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \triangle_M^n$ can be written as a convex combination of the elements of $\boldsymbol{E}_M^n$ in the following way

$$\boldsymbol{x}^{(n)} = \sum_{\boldsymbol{\eta}^{(n)} \in \boldsymbol{E}_M^n} \left(\prod_{i=1}^{n} \boldsymbol{x}_i \cdot \boldsymbol{\eta}_i \right) \boldsymbol{\eta}^{(n)}. \tag{A.2}$$

We have already shown that the weights sum to one, and they are clearly elements of $[0,1]$, so the right hand side of equation (A.2) is indeed a convex combination of the elements of $\boldsymbol{E}_M^n$. We now show that equation (A.2) holds, again by induction.

For $n = 1$ and any $\boldsymbol{x}^{(1)} = (\boldsymbol{x}_1,) \in \triangle_M^1$, the right hand side of equation (A.2) can be written as

$$\sum_{j=1}^{M} (\boldsymbol{x}_1 \cdot \boldsymbol{e}_j)(\boldsymbol{e}_j,) = (\boldsymbol{x}_1,) = \boldsymbol{x}.$$

For the inductive hypothesis, suppose equation (A.2) holds for some arbitrary $n \geq 1$, and denote elements of $\boldsymbol{E}_M^{n+1}$ by $\boldsymbol{\eta}^{(n)} \oplus (\boldsymbol{e},)$ for some $\boldsymbol{\eta}^{(n)} \in \boldsymbol{E}_M^n$ and $\boldsymbol{e} \in \boldsymbol{E}_M$, where $\oplus$ denotes vector concatenation. Then for any $\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} \oplus (\boldsymbol{x}_{n+1},) = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n+1}) \in \triangle_M^{n+1}$, the right hand side of equation (A.2) can be written as

$$\begin{aligned}
\sum_{\boldsymbol{\eta}^{(n+1)} \in \boldsymbol{E}_M^{n+1}} \left(\prod_{i=1}^{n+1} \boldsymbol{x}_i \cdot \boldsymbol{\eta}_i \right) \boldsymbol{\eta}^{(n+1)} &= \sum_{\boldsymbol{\eta}^{(n)} \in \boldsymbol{E}_M^n} \sum_{j=1}^{M} \left(\prod_{i=1}^{n} \boldsymbol{x}_i \cdot \boldsymbol{\eta}_i \right) (\boldsymbol{x}_{n+1} \cdot \boldsymbol{e}_j) \boldsymbol{\eta}^{(n)} \oplus (\boldsymbol{e}_j,) \\
&= \sum_{\boldsymbol{\eta}^{(n)} \in \boldsymbol{E}_M^n} \sum_{j=1}^{M} \left(\prod_{i=1}^{n} \boldsymbol{x}_i \cdot \boldsymbol{\eta}_i \right) (\boldsymbol{x}_{n+1} \cdot \boldsymbol{e}_j) \boldsymbol{\eta}^{(n)} \\
&\oplus \sum_{\boldsymbol{\eta}^{(n)} \in \boldsymbol{E}_M^n} \sum_{j=1}^{M} \left(\prod_{i=1}^{n} \boldsymbol{x}_i \cdot \boldsymbol{\eta}_i \right) (\boldsymbol{x}_{n+1} \cdot \boldsymbol{e}_j)(\boldsymbol{e}_j,) \\
&= \sum_{j=1}^{M} (\boldsymbol{x}_{n+1} \cdot \boldsymbol{e}_j) \sum_{\boldsymbol{\eta}^{(n)} \in \boldsymbol{E}_M^n} \left(\prod_{i=1}^{n} \boldsymbol{x}_i \cdot \boldsymbol{\eta}_i \right) \boldsymbol{\eta}^{(n)} \\
&\oplus \sum_{\boldsymbol{\eta}^{(n)} \in \boldsymbol{E}_M^n} \left(\prod_{i=1}^{n} \boldsymbol{x}_i \cdot \boldsymbol{\eta}_i \right) \sum_{j=1}^{M} (\boldsymbol{x}_{n+1} \cdot \boldsymbol{e}_j)(\boldsymbol{e}_j,) \\
&= 1 \cdot \boldsymbol{x}^{(n)} \oplus 1 \cdot (\boldsymbol{x}_{n+1},) = \boldsymbol{x}^{(n+1)},
\end{aligned}$$

where in the penultimate equality we have used the inductive hypothesis and (twice) the result of the previous induction.

We can now prove the statement of the Lemma. Applying Jensen's inequality to equation

(A.2) with the convex function f, we have that

$$\begin{aligned} f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) &= f\left(\sum_{\boldsymbol{\eta}^{(n)} \in \boldsymbol{E}_M^n} \left(\prod_{i=1}^n \boldsymbol{x}_i \cdot \boldsymbol{\eta}_i \right) \boldsymbol{\eta}^{(n)} \right) \\ &\leq \sum_{\boldsymbol{\eta}^{(n)} \in \boldsymbol{E}_M^n} \left(\prod_{i=1}^n \boldsymbol{x}_i \cdot \boldsymbol{\eta}_i \right) f\left(\boldsymbol{\eta}^{(n)}\right). \end{aligned}$$

Let $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{X}_1]$ denote the mean of the i.i.d. random vectors X_i. Then the above inequality implies

$$\begin{aligned} \mathbb{E}[f(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)] &\leq \sum_{\boldsymbol{\eta}^{(n)} \in \boldsymbol{E}_M^n} \left(\prod_{i=1}^n \boldsymbol{\mu} \cdot \boldsymbol{\eta}_i \right) f\left(\boldsymbol{\eta}^{(n)}\right) \\ &= \sum_{\boldsymbol{\eta}^{(n)} \in \boldsymbol{E}_M^n} \left(\prod_{i=1}^n \mathbb{P}(\boldsymbol{X}_i' = \boldsymbol{\eta}_i) \right) f\left(\boldsymbol{\eta}^{(n)}\right) \\ &= \mathbb{E}[f(\boldsymbol{X}_1', \ldots, \boldsymbol{X}_n')]. \end{aligned}$$

A.3.3 Proof of Lemma 3

The proof of Lemma 3 itself requires two technical helping lemmas which we now state and prove.

Lemma 6. *For any integers $n \geq 2$ and $p \geq -1$,*

$$\sum_{k=1}^{n-1} \frac{(n-k)^{p/2}}{\sqrt{k}} \leq n^{\frac{p+1}{2}} \int_0^1 \frac{(1-x)^{p/2}}{\sqrt{x}} dx.$$

Proof. The case of $p = -1$, namely

$$\sum_{k=1}^{n-1} \frac{1}{\sqrt{k(n-k)}} \leq \int_0^1 \frac{1}{\sqrt{x(1-x)}} \mathrm{d}x,$$

has already been demonstrated in Maurer (2004). For $p > -1$, let

$$f_p(x) := \frac{(1-x)^{p/2}}{\sqrt{x}}.$$

We will show that each $f_p(\cdot)$ is monotonically decreasing on $(0,1)$. Indeed,

$$\frac{\mathrm{d}f_p}{\mathrm{d}x}(x) = -\frac{(1-x)^{\frac{p}{2}-1}(px + 1 - x)}{2x^{3/2}} \leq -\frac{(1-x)^{p/2}}{2x^{3/2}} < 0,$$

where for the inequalities we have used the fact that $p > -1$ and $x \in (0,1)$. We therefore see that

$$\begin{aligned}
\sum_{k=1}^{n-1} \frac{(n-k)^{p/2}}{\sqrt{k}} &= \sum_{k=1}^{n-1} \frac{n^{p/2}(1-\frac{k}{n})^{p/2}}{\sqrt{n}\sqrt{\frac{k}{n}}} \\
&= n^{\frac{p+1}{2}} \sum_{k=1}^{n-1} \frac{1}{n} \frac{(1-\frac{k}{n})^{p/2}}{\sqrt{\frac{k}{n}}} \\
&= n^{\frac{p+1}{2}} \sum_{k=1}^{n-1} \frac{1}{n} f_p\left(\frac{k}{n}\right) \\
&\leq n^{\frac{p+1}{2}} \sum_{k=1}^{n-1} \int_{\frac{k-1}{n}}^{\frac{k}{n}} f_p(x)\mathrm{d}x \\
&= n^{\frac{p+1}{2}} \int_0^{1-\frac{1}{n}} f_p(x)\mathrm{d}x \\
&\leq n^{\frac{p+1}{2}} \int_0^1 f_p(x)\mathrm{d}x.
\end{aligned}$$

□

Intuitively, the proof of the above lemma works by bounding the integral below by a Riemann sum. In the following lemma we actually calculate this integral, yielding a more explicit bound on the sum in Lemma 6. We found it is easier to calculate a slightly more general integral, where the 1 in the limit and the integrand is replaced by a positive constant a.

Lemma 7. *For any real number $a > 0$ and integer $n \geq -1$,*

$$\int_0^a \frac{(a-x)^{n/2}}{\sqrt{x}} dx = \sqrt{\pi}\frac{\Gamma(\frac{n+2}{2})}{\Gamma(\frac{n+3}{2})} a^{\frac{n+1}{2}}.$$

Proof. Define

$$\mathrm{I}_n(a) := \int_0^a \frac{(a-x)^{n/2}}{\sqrt{x}}\mathrm{d}x \qquad \text{and} \qquad f_n(a) := \sqrt{\pi}\frac{\Gamma(\frac{n+2}{2})}{\Gamma(\frac{n+3}{2})} a^{\frac{n+1}{2}}.$$

We proceed by induction, increasing n by 2 each time. This means we need two base cases. First, for $n = -1$, we have

$$\mathrm{I}_{-1}(a) = \int_0^a \frac{1}{\sqrt{x(a-x)}}\mathrm{d}x = \left[2\arcsin\sqrt{\frac{x}{a}}\right]_0^a = \pi = f_{-1}(a),$$

since $\Gamma(\frac{1}{2}) = \sqrt{\pi}$ and $\Gamma(1) = 1$. Second, for $n = 0$,

$$\mathrm{I}_0(a) = \int_0^a \frac{1}{\sqrt{x}}\mathrm{d}x = \left[2\sqrt{x}\right]_0^a = 2\sqrt{a} = f_0(a),$$

since $\Gamma(\frac{3}{2}) = \frac{\sqrt{\pi}}{2}$. Now, by the Leibniz integral rule, we have

$$\frac{\mathrm{d}}{\mathrm{d}a}\mathrm{I}_{n+2}(a) = \int_0^a \frac{\partial}{\partial a}\frac{(a-x)^{\frac{n+2}{2}}}{\sqrt{x}}\mathrm{d}x = \frac{n+2}{2}\int_0^a \frac{(a-x)^{\frac{n}{2}}}{\sqrt{x}}\mathrm{d}x = \frac{n+2}{2}\mathrm{I}_n(a).$$

Thus

$$\mathrm{I}_{n+2}(a) = \frac{n+2}{2}\left[\int_0^a \mathrm{I}_n(t)\mathrm{d}t + \mathrm{I}_n(0)\right] = \frac{n+2}{2}\int_0^a \mathrm{I}_n(t)\mathrm{d}t,$$

since $\mathrm{I}_n(0) = 0$.

Now, for the inductive step, suppose $\mathrm{I}_n(a) = f_n(a)$ for some $n \geq -1$. Then, using the previous calculation, we have

$$\begin{aligned}
\mathrm{I}_{n+2}(a) &= \frac{n+2}{2}\int_0^a f_n(t)\mathrm{d}t \\
&= \frac{n+2}{2}\int_0^a \sqrt{\pi}\frac{\Gamma(\frac{n+2}{2})}{\Gamma(\frac{n+3}{2})}t^{\frac{n+1}{2}}\mathrm{d}t \\
&= \sqrt{\pi}\frac{\frac{n+2}{2}\Gamma(\frac{n+2}{2})}{\frac{n+3}{2}\Gamma(\frac{n+3}{2})}a^{\frac{n+3}{2}} \\
&= \sqrt{\pi}\frac{\Gamma(\frac{n+2}{2}+1)}{\Gamma(\frac{n+3}{2}+1)}a^{\frac{n+3}{2}} \\
&= \sqrt{\pi}\frac{\Gamma\left(\frac{(n+2)+2}{2}\right)}{\Gamma\left(\frac{(n+2)+3}{2}\right)}a^{\frac{(n+2)+1}{2}} \\
&= f_{n+2}(a).
\end{aligned}$$

This completes the proof. □

We are now ready to prove Lemma 3 which, for ease of reference, we restate here. For integers $M \geq 1$ and $n \geq M$,

$$\sum_{\boldsymbol{k}\in S_{n,M}^{>0}} \frac{1}{\prod_{j=1}^M \sqrt{k_j}} \leq \frac{\pi^{\frac{M}{2}} n^{\frac{M-2}{2}}}{\Gamma(\frac{M}{2})}.$$

Proof. (of Lemma 3) We proceed by induction on M. For $M = 1$, the set $S_{n,M}$ contains a single element, namely the one-dimensional vector $\boldsymbol{k} = (k_1,) = (n,)$. In this case, the left hand side is $1/\sqrt{n}$ while the right hand side is $\sqrt{\pi}/(\sqrt{n}\Gamma(1/2)) = 1/\sqrt{n}$, since $\Gamma(1/2) = \sqrt{\pi}$.

Now, as the inductive hypothesis, assume the inequality of Lemma 3 holds for some fixed

$M \geq 1$ and all $n \geq M$. Then for all $n \geq M+1$, we have

$$
\begin{aligned}
\sum_{\boldsymbol{k} \in S^{>0}_{n,M+1}} \frac{1}{\prod_{j=1}^{M+1} \sqrt{k_j}} &= \sum_{k_1=1}^{n-M} \frac{1}{\sqrt{k_1}} \sum_{\boldsymbol{k}' \in S^{>0}_{n-k_1,M}} \frac{1}{\prod_{j=1}^{M} \sqrt{k'_j}} \\
&\leq \sum_{k_1=1}^{n-M} \frac{1}{\sqrt{k_1}} \frac{\pi^{\frac{M}{2}} (n-k_1)^{\frac{M-2}{2}}}{\Gamma(\frac{M}{2})} \qquad \text{(by the inductive hypothesis)} \\
&= \frac{\pi^{\frac{M}{2}}}{\Gamma(\frac{M}{2})} \sum_{k_1=1}^{n-M} \frac{(n-k_1)^{\frac{M-2}{2}}}{\sqrt{k_1}} \\
&\leq \frac{\pi^{\frac{M}{2}}}{\Gamma(\frac{M}{2})} \sum_{k_1=1}^{n-1} \frac{(n-k_1)^{\frac{M-2}{2}}}{\sqrt{k_1}} \qquad \text{(enlarging the sum domain)} \\
&\leq \frac{\pi^{\frac{M}{2}}}{\Gamma(\frac{M}{2})} n^{\frac{M-1}{2}} \int_0^1 \frac{(1-x)^{\frac{M-2}{2}}}{\sqrt{x}} \mathrm{d}x \qquad \text{(by Lemma 6)} \\
&= \frac{\pi^{\frac{M}{2}}}{\Gamma(\frac{M}{2})} n^{\frac{M-1}{2}} \sqrt{\pi} \frac{\Gamma(\frac{M}{2})}{\Gamma(\frac{M+1}{2})} \qquad \text{(by Lemma 7)} \\
&= \frac{\pi^{\frac{M+1}{2}} n^{\frac{M-1}{2}}}{\Gamma(\frac{M+1}{2})},
\end{aligned}
$$

as required. □

A.3.4 Proof of Proposition 1

The first part of the theorem, namely $\mathrm{kl}(q_j \| p_j) \leq \mathrm{kl}(\boldsymbol{q} \| \boldsymbol{p})$ for all j, can be straightforwardly obtained via the data processing inequality found in Van Erven and Harremos (2014), but we give here an elementary proof.

Proof. The case where $q_j = 1$ or $p_j = 1$ can be dealt with trivially by splitting into the three following sub-cases

- $q_j = p_j = 1 \implies \mathrm{kl}(q_j \| p_j) = \mathrm{kl}(\boldsymbol{q} \| \boldsymbol{p}) = 0$
- $q_j = 1, p_j \neq 1 \implies \mathrm{kl}(q_j \| p_j) = \mathrm{kl}(\boldsymbol{q} \| \boldsymbol{p}) = -\log p_j$
- $q_j \neq 1, p_j = 1 \implies \mathrm{kl}(q_j \| p_j) = \mathrm{kl}(\boldsymbol{q} \| \boldsymbol{p}) = \infty.$

For $q_j \neq 1$ and $p_j \neq 1$ define the distributions $\tilde{\boldsymbol{q}}, \tilde{\boldsymbol{p}} \in \triangle_M$ by $\tilde{q}_j = \tilde{p}_j = 0$ and

$$
\tilde{q}_i = \frac{q_i}{1-q_j} \qquad \text{and} \qquad \tilde{p}_i = \frac{p_i}{1-p_j}
$$

for $i \neq j$. Then

$$\begin{aligned}
\sum_{i \neq j} q_i \log \frac{q_i}{p_i} &= \sum_{i \neq j} (1-q_j)\tilde{q}_i \log \frac{(1-q_j)\tilde{q}_i}{(1-p_j)\tilde{p}_i} \\
&= (1-q_j) \sum_{i \neq j} \tilde{q}_i \log \frac{\tilde{q}_i}{\tilde{p}_i} + \tilde{q}_i \log \frac{1-q_j}{1-p_j} \\
&= (1-q_j)\mathrm{kl}(\tilde{\boldsymbol{q}} \| \tilde{\boldsymbol{p}}) + (1-q_j) \log \frac{1-q_j}{1-p_j} \\
&\geq (1-q_j) \log \frac{1-q_j}{1-p_j}.
\end{aligned}$$

The final inequality holds since $\mathrm{kl}(\tilde{\boldsymbol{q}} \| \tilde{\boldsymbol{p}}) \geq 0$. Further, note that we have equality if and only if $\tilde{\boldsymbol{q}} = \tilde{\boldsymbol{p}}$, which, by their definitions, translates to

$$p_i = \frac{1-p_j}{1-q_j} q_i$$

for all $i \neq j$. If we now add $q_j \log \frac{q_j}{p_j}$ to both sides, we obtain

$$\mathrm{kl}(\boldsymbol{q} \| \boldsymbol{p}) \geq (1-q_j) \log \frac{1-q_j}{1-p_j} + q_j \log \frac{q_j}{p_j} = \mathrm{kl}(q_j \| p_j),$$

with the same condition for equality. $\square$

The following proposition makes more precise the argument found at the beginning of Section 3.5 for how Proposition 1 can be used to derive the tightest possible lower and upper bounds on each $R_D^j(Q)$.

Proposition 5. *Suppose that $\boldsymbol{q}, \boldsymbol{p} \in \triangle_M$ are such that $\mathrm{kl}(\boldsymbol{q} \| \boldsymbol{p}) \leq B$, where $\boldsymbol{q}$ is known and $\boldsymbol{p}$ is unknown. Then, in the absence of any further information, the tightest bound that can be obtained on each p_j is*

$$p_j \leq \mathrm{kl}^{-1}(q_j, B).$$

Proof. Suppose $p_j > \mathrm{kl}^{-1}(q_j, B)$. Then, by definition of kl^{-1}, we have that $\mathrm{kl}(q_j \| p_j) > B$. By Proposition 1, this would then imply $\mathrm{kl}(\boldsymbol{q} \| \boldsymbol{p}) > B$, contradicting our assumption. Therefore $p_j \leq \mathrm{kl}^{-1}(q_j, B)$. Now, with the information we have, we cannot rule out that

$$p_i = \frac{1-p_j}{1-q_j} q_i$$

for all $i \neq j$ and thus, by Proposition 1, that $\mathrm{kl}(q_j \| p_j) = \mathrm{kl}(\boldsymbol{q} \| \boldsymbol{p})$. Further, we cannot rule out that $\mathrm{kl}(\boldsymbol{q} \| \boldsymbol{p}) = B$. Thus, it is possible that $\mathrm{kl}(q_j \| p_j) = B$, in which case $p_j = \mathrm{kl}^{-1}(q_j, B)$. We therefore see that $\mathrm{kl}^{-1}(q_j, B)$ is the tightest possible upper bound on p_j, for each $j \in [M]$. $\square$

A.3.5 Proof of Theorem 12

Before proving the proposition, we first argue that $\mathrm{kl}_{\boldsymbol{\ell}}^{-1}(\boldsymbol{u}|c)$ given by Definition 8 is well-defined. First, note that $A_{\boldsymbol{u}} := \{\boldsymbol{v} \in \triangle_M : \mathrm{kl}(\boldsymbol{u}\|\boldsymbol{v}) \leq c\}$ is compact (boundedness is clear and it is closed because it is the preimage of the closed set $[0, c]$ under the continuous map $\boldsymbol{v} \mapsto \mathrm{kl}(\boldsymbol{u}\|\boldsymbol{v})$) and so the continuous function $f_{\boldsymbol{\ell}}$ achieves its supremum on $A_{\boldsymbol{u}}$. Further, note that $A_{\boldsymbol{u}}$ is a convex subset of $\triangle_M$ (because the map $\boldsymbol{v} \mapsto \mathrm{kl}(\boldsymbol{u}\|\boldsymbol{v})$ is convex) and $f_{\boldsymbol{\ell}}$ is linear, so the supremum of $f_{\boldsymbol{\ell}}$ over $A_{\boldsymbol{u}}$ is achieved and is located on the boundary of $A_{\boldsymbol{u}}$. This means we can replace the inequality constraint $\mathrm{kl}(\boldsymbol{u}\|\boldsymbol{v}) \leq c$ in Definition 8 with the equality constraint $\mathrm{kl}(\boldsymbol{u}\|\boldsymbol{v}) = c$. Finally, if $\boldsymbol{u} \in \triangle_M^{>0}$ then $A_{\boldsymbol{u}}$ is a *strictly* convex subset of $\triangle_M$ (because the map $\boldsymbol{v} \mapsto \mathrm{kl}(\boldsymbol{u}\|\boldsymbol{v})$ is then *strictly* convex) and so the supremum of $f_{\boldsymbol{\ell}}$ occurs at a *unique* point on the boundary of $A_{\boldsymbol{u}}$. In other words, if $\boldsymbol{u} \in \triangle_M^{>0}$ then $\mathrm{kl}_{\boldsymbol{\ell}}^{-1}(\boldsymbol{u}|c)$ is defined *uniquely.*

We now prove Theorem 12. While our proof technique is somewhat analogous to the technique used in Clerico et al. (2022a) to obtain derivatives of the one-dimensional kl-inverse, our theorem directly yields derivatives on the total risk by (implicitly) employing the envelope theorem (see for example Takayama and Akira (1985)).

Proof Outline: We first derive the expression given for $\boldsymbol{v}^*(\tilde{\boldsymbol{u}}) = \mathrm{kl}_{\boldsymbol{\ell}}^{-1}(\boldsymbol{u}|c)$ given on line (3.3) of the theorem using the method of Lagrange multipliers. Since we are working on the simplex, we make things easier for ourselves by first making the substitution $t_j = \ln v_j$ to make the $v_j > 0$ constraints unnecessary. The method of Lagrange multipliers yields both the maximum and the minimum (recall that $\mathrm{kl}_{\boldsymbol{\ell}}^{-1}(\boldsymbol{u}|c)$ is defined as the location of a maximum) for the two values of the Lagrange multiplier μ. We show that exactly one of these values lies in the interval $\mu \in (-\infty, -\max_j \ell_j)$ and that this one corresponds to the maximum. This shows that the value μ^* Theorem 12 instructs us to find indeed yields $\boldsymbol{v}^*(\tilde{\boldsymbol{u}}) = \mathrm{kl}_{\boldsymbol{\ell}}^{-1}(\boldsymbol{u}|c)$. Finally, we derive the partial derivatives of $\mathrm{kl}_{\boldsymbol{\ell}}^{-1}(\boldsymbol{u}|c)$ with respect the $\tilde{u}_j$ to obtain the second part of the theorem, namely line (3.4) by employing the envelope theorem.

Proof. (of Theorem 12) We start by deriving the implicit expression for $\boldsymbol{v}^*(\tilde{\boldsymbol{u}}) = \mathrm{kl}_{\boldsymbol{\ell}}^{-1}(\boldsymbol{u}|c)$ given in the proposition by solving a transformed version of the optimisation problem given by Definition 8 using the method of Lagrange multipliers. We obtain two solutions to the Lagrangian equations, which must correspond to the maximum and minimum total risk over the set $A_{\boldsymbol{u}} := \{\boldsymbol{v} \in \triangle_M : \mathrm{kl}(\boldsymbol{u}\|\boldsymbol{v}) \leq c\}$ because, as argued in the main text (see the discussion after Definition 8), $A_{\boldsymbol{u}}$ is compact and so the linear total risk $f_{\boldsymbol{\ell}}(\boldsymbol{v})$ attains its maximum and minimum on $A_{\boldsymbol{u}}$.

By definition of $\boldsymbol{v}^*(\tilde{\boldsymbol{u}}) = \mathrm{kl}_{\boldsymbol{\ell}}^{-1}(\boldsymbol{u}|c)$, we know that $\mathrm{kl}(\boldsymbol{v}^*(\tilde{\boldsymbol{u}})\|\boldsymbol{u}) \leq c$. Since, by assumption, $u_j > 0$ for all j, we see that $\boldsymbol{v}^*(\tilde{\boldsymbol{u}})_j > 0$ for all j, otherwise we would have $\mathrm{kl}(\boldsymbol{v}^*(\tilde{\boldsymbol{u}})\|\boldsymbol{u}) = \infty$, a contradiction. Thus $\boldsymbol{v}^*(\tilde{\boldsymbol{u}}) \in \triangle_M^{>0}$ and we are permitted to instead optimise over the unbounded variable $\boldsymbol{t} \in \mathbb{R}^M$, where $t_j := \ln v_j$. With this transformation, the constraint $\boldsymbol{v} \in \triangle_M$ can be replaced simply by $\sum_j e^{t_j} = 1$ and the optimisation problem becomes

$$\begin{aligned} \text{Maximise:} \quad & F(\boldsymbol{t}) := \sum_{j=1}^{M} \ell_j e^{t_j} \\ \text{Subject to:} \quad & g(\boldsymbol{t}; \boldsymbol{u}, c) := \mathrm{kl}(\boldsymbol{u} \| e^{\boldsymbol{t}}) - c = 0, \\ & h(\boldsymbol{t}) := \sum_{j=1}^{M} e^{t_j} - 1 = 0, \end{aligned}$$

where $e^{\boldsymbol{t}} \in \mathbb{R}^M$ is defined by $(e^{\boldsymbol{t}})_j := e^{t_j}$. Note that $F(\boldsymbol{t}) = f_{\boldsymbol{\ell}}(e^{\boldsymbol{t}})$. Following the terminology of mathematical economics, we call the t_j the *optimisation variables*, and the $\tilde{u}_j$ (namely the u_j and c) the *choice variables*. The vector $\boldsymbol{\ell}$ is considered fixed—we neither want to optimise over it nor differentiate with respect to it—which is why we occasionally suppress it from the notation henceforth.

For each $\tilde{\boldsymbol{u}}$, let $\boldsymbol{v}^*(\tilde{\boldsymbol{u}})$ and $\boldsymbol{t}^*(\tilde{\boldsymbol{u}})$ be the solutions to the original and transformed optimisation problems respectively. Since the map $\boldsymbol{v} = e^{\boldsymbol{t}}$ is one-to-one, it is clear that since $\boldsymbol{v}^*(\tilde{\boldsymbol{u}})$ exists uniquely, so does $\boldsymbol{t}^*(\tilde{\boldsymbol{u}})$, and that they are related by $\boldsymbol{v}^*(\tilde{\boldsymbol{u}}) = e^{\boldsymbol{t}^*(\tilde{\boldsymbol{u}})}$. We therefore have the identity

$$f_{\boldsymbol{\ell}}(\boldsymbol{v}^*(\tilde{\boldsymbol{u}})) \equiv F(\boldsymbol{t}^*(\tilde{\boldsymbol{u}})).$$

Recalling that $f^*_{\boldsymbol{\ell}}(\tilde{\boldsymbol{u}}) := f_{\boldsymbol{\ell}}(\boldsymbol{v}^*(\tilde{\boldsymbol{u}}))$, we see that

$$\nabla_{\tilde{\boldsymbol{u}}} f^*_{\boldsymbol{\ell}}(\tilde{\boldsymbol{u}}) \equiv \nabla_{\tilde{\boldsymbol{u}}} F(\boldsymbol{t}^*(\tilde{\boldsymbol{u}})). \tag{A.3}$$

the derivatives of $f_{\boldsymbol{\ell}}(\mathrm{kl}^{-1}_{\boldsymbol{\ell}}(\boldsymbol{u}|c))$ with respect to $\boldsymbol{u}$ and c are given by $\nabla_{\tilde{\boldsymbol{u}}} F(\boldsymbol{t}^*(\tilde{\boldsymbol{u}}))$.

Using the method of Lagrange multipliers, there exist real numbers $\lambda^* = \lambda^*(\tilde{\boldsymbol{u}})$ and $\mu^* = \mu^*(\tilde{\boldsymbol{u}})$ such that $(\boldsymbol{t}^*, \lambda^*, \mu^*)$ is a stationary point (with respect to $\boldsymbol{t}, \lambda$ and μ) of the Lagrangian function

$$\mathcal{L}(\boldsymbol{t}, \lambda, \mu; \tilde{\boldsymbol{u}}) := F(\boldsymbol{t}) + \lambda g(\boldsymbol{t}; \tilde{\boldsymbol{u}}) + \mu h(\boldsymbol{t}).$$

Let $F_{\boldsymbol{t}}(\cdot)$ and $h_{\boldsymbol{t}}(\cdot)$ denote the gradient vectors of F and h respectively, and let $g_{\boldsymbol{t}}(\,\cdot\,; \tilde{\boldsymbol{u}})$ and $g_{\tilde{\boldsymbol{u}}}(\boldsymbol{t};\,\cdot\,)$ denote the gradient vectors of g with respect to $\boldsymbol{t}$ only and $\tilde{\boldsymbol{u}}$ only, respectively. Simple calculation yields

$$\begin{aligned} g_{\boldsymbol{t}}(\boldsymbol{t}; \tilde{\boldsymbol{u}}) &= \left(\frac{\partial g}{\partial t_1}(\boldsymbol{t}; \tilde{\boldsymbol{u}}), \dots, \frac{\partial g}{\partial t_M}(\boldsymbol{t}; \tilde{\boldsymbol{u}}) \right) = -\boldsymbol{u} \quad \text{and} \\ g_{\tilde{\boldsymbol{u}}}(\boldsymbol{t}; \tilde{\boldsymbol{u}}) &= \left(\frac{\partial g}{\partial \tilde{u}_1}(\boldsymbol{t}; \tilde{\boldsymbol{u}}), \dots, \frac{\partial g}{\partial \tilde{u}_{M+1}}(\boldsymbol{t}; \tilde{\boldsymbol{u}}) \right) = \Big(1 - t_1 + \log u_1, \dots, 1 - t_M + \log u_M, -1 \Big). \end{aligned} \tag{A.4}$$

Then, taking the partial derivatives of $\mathcal{L}$ with respect to λ, μ and the t_j, we have that $(\boldsymbol{t}, \lambda, \mu) = (\boldsymbol{t}^*(\tilde{\boldsymbol{u}}), \lambda^*(\tilde{\boldsymbol{u}}), \mu^*(\tilde{\boldsymbol{u}}))$ solves the simultaneous equations

$$F_{\boldsymbol{t}}(\boldsymbol{t}) + \lambda g_{\boldsymbol{t}}(\boldsymbol{t}; \tilde{\boldsymbol{u}}) + \mu h_{\boldsymbol{t}}(\boldsymbol{t}) = \boldsymbol{0}, \tag{A.5}$$

$$g(\boldsymbol{t};\tilde{\boldsymbol{u}}) = 0, \quad \text{and}$$
$$h(\boldsymbol{t}) = 0,$$

where the last two equations recover the constraints. Substituting the gradients $F_{\boldsymbol{t}}, g_{\boldsymbol{t}}$ and $h_{\boldsymbol{t}}$, the first equation reduces to

$$\boldsymbol{\ell} \odot e^{\boldsymbol{t}} - \lambda \boldsymbol{u} + \mu e^{\boldsymbol{t}} = \boldsymbol{0},$$

which implies that for all $j \in [M]$

$$e^{t_j} = \frac{\lambda u_j}{\mu + \ell_j}. \tag{A.6}$$

Substituting this into the constraints $g = h = 0$ yields the following simultaneous equations in λ and μ

$$c = \mathrm{kl}(\boldsymbol{u}\|e^{\boldsymbol{t}}) = \sum_{j=1}^{M} u_j \log \frac{u_j}{e^{t_j}} = \sum_{j=1}^{M} u_j \log \frac{\mu + \ell_j}{\lambda} \quad \text{and} \quad \lambda \sum_{j=1}^{M} \frac{u_j}{\mu + \ell_j} = 1.$$

Substituting the second into the first and rearranging the second, this is equivalent to solving

$$c = \sum_{j=1}^{M} u_j \log \left((\mu + \ell_j) \sum_{k=1}^{M} \frac{u_k}{\mu + \ell_k} \right) \quad \text{and} \quad \lambda = \left(\sum_{j=1}^{M} \frac{u_j}{\mu + \ell_j} \right)^{-1}. \tag{A.7}$$

It has already been established in the discussion after Definition 8 that $f_{\boldsymbol{\ell}}(\boldsymbol{v})$ attains its maximum on the set $A_{\boldsymbol{u}} := \{\boldsymbol{v} \in \triangle_M : \mathrm{kl}(\boldsymbol{u}\|\boldsymbol{v}) \leq c\}$. Therefore $F(\boldsymbol{t})$ also attains its maximum on $\mathbb{R}^M$ and one of the solutions to these simultaneous equations corresponds to this maximum. We first show that there is a single solution to the first equation in the set $(-\infty, -\max_j \ell_j)$, referred to as $\mu^*(\tilde{\boldsymbol{u}})$ in the proposition. Second, we show that any other solution corresponds to a smaller total risk, so that $\mu^*(\tilde{\boldsymbol{u}})$ corresponds to the maximum total risk and yields $\boldsymbol{v}^*(\tilde{\boldsymbol{u}}) = \mathrm{kl}_{\boldsymbol{\ell}}^{-1}(\boldsymbol{u}|c)$ when $\mu^*(\tilde{\boldsymbol{u}})$ and the associated $\lambda^*(\tilde{\boldsymbol{u}})$ are substituted into Equation A.6.

For the first step, note that since the e^{t_j} are probabilities, we see from Equation A.6 that either $\mu + \ell_j > 0$ for all j (in the case that $\lambda > 0$), or $\mu + \ell_j < 0$ for all j (in the case that $\lambda < 0$). Thus any solutions μ to the first equation must be in $(-\infty, -\max_j \ell_j)$ or $(-\min_j \ell_j, \infty)$. If $\mu \in (-\infty, -\max_j \ell_j)$ then the first equation can be written as $c = \phi_{\boldsymbol{\ell}}(\mu)$, with $\phi_{\boldsymbol{\ell}}$ as defined in the statement of the proposition. We now show that $\phi_{\boldsymbol{\ell}}$ is strictly increasing in μ, and that $\phi_{\boldsymbol{\ell}}(\mu) \to 0$ as $\mu \to -\infty$ and $\phi_{\boldsymbol{\ell}}(\mu) \to \infty$ as $\mu \to -\max_j \ell_j$, so that $c = \phi_{\boldsymbol{\ell}}(\mu)$ does indeed have a single solution in the set $(-\infty, -\max_j \ell_j)$. Straightforward differentiation and algebra shows that

$$\begin{aligned}
\phi'_{\boldsymbol{\ell}}(\mu) &= \sum_{j=1}^{M} \frac{u_j}{(\mu + \ell_j) \sum_{k=1}^{M} \frac{u_k}{\mu + \ell_k}} \left(\sum_{k'=1}^{M} \frac{u_{k'}}{\mu + \ell_{k'}} - (\mu + \ell_j) \sum_{k'=1}^{M} \frac{u_{k'}}{(\mu + \ell_{k'})^2} \right) \\
&= \frac{\left(\sum_{j=1}^{M} \frac{u_j}{\mu + \ell_j} \right)^2 - \sum_{j=1}^{M} \frac{u_j}{(\mu + \ell_j)^2}}{\sum_{k=1}^{M} \frac{u_k}{\mu + \ell_k}}.
\end{aligned}$$

Jensen's inequality demonstrates that the numerator is strictly negative, where strictness is due to the assumption that the ℓ_j are not all equal. Further, since the denominator is strictly negative (since we are dealing with the case where $\mu \in (-\infty, -\max_j \ell_j)$), we see that $\phi_{\boldsymbol{\ell}}$ is strictly increasing for $\mu \in (-\infty, -\max_j \ell_j)$.[1] Turning to the limits, we first show that $\phi_{\boldsymbol{\ell}}(\mu) \to \infty$ as $\mu \to -\max_j \ell_j$.

We now determine the left hand limit. Define $J = \{j \in [M] : \ell_j = \max_k \ell_k\}$, noting that this is a strict subset of $[M]$ since by assumption the ℓ_j are not all equal. We then have that for $\mu \in (-\infty, \max_j \ell_j)$

$$\begin{aligned}
e^{\phi_{\boldsymbol{\ell}}(\mu)} &= \left(-\sum_{j=1}^{M} \frac{u_j}{\mu + \ell_j}\right)\left(\prod_{k=1}^{M}\big(-(\mu+\ell_k)\big)^{u_k}\right) \\
&= \left(-\sum_{j\in J} \frac{u_j}{\mu + \ell_j} - \sum_{j'\notin J} \frac{u_{j'}}{\mu + \ell_{j'}}\right)\prod_{k\in J}\big(-(\mu+\ell_k)\big)^{u_k}\prod_{k'\notin J}\big(-(\mu+\ell_{k'})\big)^{u_{k'}} \\
&\geq \left(-\sum_{j\in J} \frac{u_j}{\mu + \ell_j}\right)\prod_{k\in J}\big(-(\mu+\ell_k)\big)^{u_k}\prod_{k'\notin J}\big(-(\mu+\ell_{k'})\big)^{u_{k'}} \\
&= \frac{\left(\sum_{j\in J} u_j\right)\left(\prod_{k'\notin J}\big(-(\mu+\ell_{k'})\big)^{u_{k'}}\right)}{\big(-(\mu+\max_j \ell_j)\big)^{1-\sum_{k\in J} u_k}}.
\end{aligned}$$

The first term in the numerator is a positive constant, independent of μ. The second term in the numerator tends to a finite positive limit as $\mu \uparrow -\max_j \ell_j$. Since $[M] \setminus J$ is non-empty, the power in the denominator is positive and the term in the outer brackets is positive and tends to zero as $\mu \uparrow -\max_j \ell_j$. Thus $e^{\phi_{\boldsymbol{\ell}}(\mu)} \to \infty$ as $\mu \uparrow -\max_j \ell_j$ and, by the continuity of the logarithm, $\phi_{\boldsymbol{\ell}}(\mu)$ as $\mu \uparrow -\max_j \ell_j$.

We now determine $\lim_{\mu\to-\infty} \phi_{\boldsymbol{\ell}}(\mu)$ by sandwiching $\phi(\mu)$ between two functions that both tend to zero as $\mu \to -\infty$. First, since $\ell_j \geq 0$ for all j, for $\mu \in (-\infty, -\max_j \ell_j)$ we have

$$\log\left(-\sum_{j=1}^{M} \frac{u_j}{\mu+\ell_j}\right) \geq \log\left(-\sum_{j=1}^{M}\frac{u_j}{\mu}\right) = -\log(-\mu) = -\sum_{j=1}^{M} u_j \log(-\mu),$$

and so

$$\phi_{\boldsymbol{\ell}}(\mu) \geq -\sum_{j=1}^{M} u_j \log(-\mu) + \sum_{j=1}^{M} u_j \log\big(-(\mu+\ell_j)\big) = \sum_{j=1}^{M} u_j \log\left(1 + \frac{\ell_j}{\mu}\right) \to 0 \quad \text{as} \quad \mu \to -\infty.$$

Similarly,

$$\sum_{j=1}^{M} u_j \log\big(-(\mu+\ell_j)\big) \leq \sum_{j=1}^{M} u_j \log(-\mu) = \log(-\mu),$$

[1]Incidentally, this argument also shows that there is at most one solution to the first equation in (A.7) in the range $(-\min_j \ell_j, \infty)$. There indeed exists a unique solution, which corresponds to the minimum total risk, but we do not prove this.

and so

$$\phi_{\boldsymbol{\ell}}(\mu) \le \log\left(\mu \sum_{j=1}^{M} \frac{u_j}{\mu + \ell_j}\right) = \log\left(\sum_{j=1}^{M} \frac{u_j}{1 + \frac{\ell_j}{\mu}}\right) \to 0 \quad \text{as} \quad \mu \to -\infty.$$

This completes the first step, namely showing that there does indeed exist a unique solution $\mu^*(\tilde{\boldsymbol{u}})$ in the set $(-\ell_1, \infty)$ to the first equation in line (A.7).

We now turn to the second step, namely showing that this solution corresponds to the maximum total risk. Given a value of the Lagrange multiplier μ, substitution into Equation A.6 gives

$$e^{t_j}(\mu) = \frac{\frac{u_j}{\mu+\ell_j}}{\sum_{k=1}^{M} \frac{u_k}{\mu+\ell_k}}$$

and therefore total risk

$$R(\mu) = \frac{\sum_{j=1}^{M} \frac{u_j \ell_j}{\mu+\ell_j}}{\sum_{k=1}^{M} \frac{u_k}{\mu+\ell_k}}.$$

To prove that the solution $\mu^*(\tilde{\boldsymbol{u}}) \in (-\infty, -\max_j \ell_j)$ is the solution to the first equation in line (A.7) that maximises R, it suffices to show that $R(\mu) \to \sum_{j=1}^{M} u_j \ell_j$ as $|\mu| \to \infty$ and $R'(\mu) \ge 0$ for all $\mu \in (-\infty, -\max_j \ell_j) \cup (-\min_j \ell_j, \infty)$, so that

$$\inf_{\mu \in (-\infty, -\max_j \ell_j)} R(\mu) \ge \sup_{\mu \in (-\min_j \ell_j, \infty)} R(\mu).$$

This suffices as we have already proved that $\mu^*(\tilde{\boldsymbol{u}})$ is the only solution in $(-\infty, -\max_j \ell_j)$ to the first equation in line (A.7), and that no solutions exists in the set $[-\max_j \ell_j, -\min_j \ell_j]$.

The limit can be easily evaluated by first rewriting $R(\mu)$ and then taking the limit as $|\mu| \to \infty$ as follows

$$R(\mu) = \frac{\sum_{j=1}^{M} \frac{u_j \ell_j}{1+\frac{\ell_j}{\mu}}}{\sum_{k=1}^{M} \frac{u_k}{1+\frac{\ell_k}{\mu}}} \to \frac{\sum_{j=1}^{M} u_j \ell_j}{\sum_{k=1}^{M} u_k} = \sum_{j=1}^{M} u_j \ell_j.$$

To show that $R'(\mu) \ge 0$, let $\ell_{(j)}$ denote the j'th smallest component of $\boldsymbol{\ell}$ (breaking ties arbitrarily), so that $\ell_{(1)} \le \cdots \le \ell_{(M)}$, and use the quotient rule to see that

$$\begin{aligned}
R'(\mu) \ge 0 \iff & \frac{\left(\sum_{k=1}^{M} \frac{u_k}{\mu+\ell_k}\right)\left(\sum_{j=1}^{M} \frac{-u_j \ell_j}{(\mu+\ell_j)^2}\right) - \left(\sum_{j=1}^{M} \frac{u_j \ell_j}{\mu+\ell_j}\right)\left(\sum_{k=1}^{M} \frac{-u_k}{(\mu+\ell_k)^2}\right)}{\left(\sum_{p=1}^{M} \frac{u_p}{\mu+\ell_p}\right)^2} \ge 0 \\
\iff & \sum_{j=1}^{M} \sum_{k=1}^{M} \frac{u_j u_k \ell_j}{(\mu+\ell_j)(\mu+\ell_k)} \left(\frac{1}{\mu+\ell_k} - \frac{1}{\mu+\ell_j}\right) \ge 0 \\
\iff & \sum_{\substack{j,k \in [M] \\ k<j}} \frac{u_j u_k \ell_{(j)}}{(\mu+\ell_{(j)})(\mu+\ell_{(k)})} \left(\frac{1}{\mu+\ell_{(k)}} - \frac{1}{\mu+\ell_{(j)}}\right) \\
& + \sum_{\substack{j,k \in [M] \\ k>j}} \frac{u_j u_k \ell_{(j)}}{(\mu+\ell_{(j)})(\mu+\ell_{(k)})} \left(\frac{1}{\mu+\ell_{(k)}} - \frac{1}{\mu+\ell_{(j)}}\right) \ge 0,
\end{aligned}$$

where in the final line we have dropped the summands where $k = j$ since they equal zero as the terms in the bracket cancel. This final inequality holds since the first sum can be bounded below by the negative of the second sum as follows

$$\begin{aligned}
&\sum_{\substack{j,k\in[M]\\k<j}} \frac{u_j u_k \ell_{(j)}}{(\mu+\ell_{(j)})(\mu+\ell_{(k)})}\left(\frac{1}{\mu+\ell_{(k)}} - \frac{1}{\mu+\ell_{(j)}}\right)\\
&\geq \sum_{\substack{j,k\in[M]\\k<j}} \frac{u_j u_k \ell_{(k)}}{(\mu+\ell_{(j)})(\mu+\ell_{(k)})}\left(\frac{1}{\mu+\ell_{(k)}} - \frac{1}{\mu+\ell_{(j)}}\right) \quad \text{(since } \ell_{(k)} \leq \ell_{(j)} \text{ for } k<j\text{)}\\
&= \sum_{\substack{j,k\in[M]\\k>j}} \frac{u_k u_j \ell_{(j)}}{(\mu+\ell_{(k)})(\mu+\ell_{(j)})}\left(\frac{1}{\mu+\ell_{(j)}} - \frac{1}{\mu+\ell_{(k)}}\right) \quad \text{(swapping dummy variables } j,k\text{).}
\end{aligned}$$

We now turn to finding the partial derivatives of $F(\boldsymbol{t}^*(\tilde{\boldsymbol{u}}))$ with respect the $\tilde{u}_j$, which in turn will allow us to find the partial derivatives of $\mathrm{kl}_{\boldsymbol{\ell}}^{-1}(\boldsymbol{u}|c)$. Let $\nabla_{\tilde{\boldsymbol{u}}}$ denote the gradient operator with respect to $\tilde{\boldsymbol{u}}$. Then the quantity we are after is $\nabla_{\tilde{\boldsymbol{u}}}F(\boldsymbol{t}^*(\tilde{\boldsymbol{u}})) \in \mathbb{R}^{M+1}$, the j'th component of which is

$$\left(\nabla_{\tilde{\boldsymbol{u}}}F(\boldsymbol{t}^*(\tilde{\boldsymbol{u}}))\right)_j = \sum_{k=1}^{M+1} \frac{\partial F}{\partial t_k}(\boldsymbol{t}^*(\tilde{\boldsymbol{u}}))\frac{\partial t_k^*}{\partial \tilde{u}_j}(\tilde{\boldsymbol{u}}) = F_{\boldsymbol{t}}(\boldsymbol{t}^*(\tilde{\boldsymbol{u}})) \cdot \frac{\partial \boldsymbol{t}^*}{\partial \tilde{u}_j}(\tilde{\boldsymbol{u}}) \in \mathbb{R}.$$

Thus the full gradient vector is

$$\nabla_{\tilde{\boldsymbol{u}}}F(\boldsymbol{t}^*(\tilde{\boldsymbol{u}})) = F_{\boldsymbol{t}}(\boldsymbol{t}^*(\tilde{\boldsymbol{u}}))\nabla_{\tilde{\boldsymbol{u}}}\boldsymbol{t}^*(\tilde{\boldsymbol{u}}), \tag{A.8}$$

where $\nabla_{\tilde{\boldsymbol{u}}}\boldsymbol{t}^*(\tilde{\boldsymbol{u}})$ is the $M \times (M+1)$ matrix given by

$$\left(\nabla_{\tilde{\boldsymbol{u}}}\boldsymbol{t}^*(\tilde{\boldsymbol{u}})\right)_{j,k} = \frac{\partial t_k^*}{\partial \tilde{u}_j}(\tilde{\boldsymbol{u}}).$$

Finding an expression for this matrix is difficult. Fortunately we can avoid needing to by using a trick from mathematical economics referred to as the envelope theorem, as we now show.

First, note that since, for all $\tilde{\boldsymbol{u}}$, the constraints $g = h = 0$ are satisfied by $\boldsymbol{t}^*(\tilde{\boldsymbol{u}})$, we have the identities

$$g(\boldsymbol{t}^*(\tilde{\boldsymbol{u}}), \tilde{\boldsymbol{u}}) \equiv 0 \quad \text{and} \quad h(\boldsymbol{t}^*(\tilde{\boldsymbol{u}})) \equiv 0.$$

Differentiating these identities with respect to $\tilde{u}_j$ then yields

$$g_{\boldsymbol{t}}(\boldsymbol{t}^*(\tilde{\boldsymbol{u}}), \tilde{\boldsymbol{u}}) \cdot \frac{\partial \boldsymbol{t}^*}{\partial \tilde{u}_j}(\tilde{\boldsymbol{u}}) + g_{\tilde{u}_j}(\boldsymbol{t}^*(\tilde{\boldsymbol{u}}), \tilde{\boldsymbol{u}}) \equiv 0 \quad \text{and} \quad h_{\boldsymbol{t}}(\boldsymbol{t}^*(\tilde{\boldsymbol{u}})) \cdot \frac{\partial \boldsymbol{t}^*}{\partial \tilde{u}_j}(\tilde{\boldsymbol{u}}) \equiv 0.$$

As before, we can write these $M+1$ pairs of equations as the following pair of matrix equations

$$g_{\boldsymbol{t}}(\boldsymbol{t}^*(\tilde{\boldsymbol{u}}), \tilde{\boldsymbol{u}})\nabla_{\tilde{\boldsymbol{u}}}\boldsymbol{t}^*(\tilde{\boldsymbol{u}}) + g_{\tilde{\boldsymbol{u}}}(\boldsymbol{t}^*(\tilde{\boldsymbol{u}}), \tilde{\boldsymbol{u}}) \equiv \boldsymbol{0} \quad \text{and} \quad h_{\boldsymbol{t}}(\boldsymbol{t}^*(\tilde{\boldsymbol{u}}))\nabla_{\tilde{\boldsymbol{u}}}\boldsymbol{t}^*(\tilde{\boldsymbol{u}}) \equiv \boldsymbol{0}.$$

Multiplying these identities by $\lambda^*(\tilde{\boldsymbol{u}})$ and $\mu^*(\tilde{\boldsymbol{u}})$ respectively, and combining with equation (A.8), yields

$$\begin{aligned}\nabla_{\tilde{\boldsymbol{u}}} F(\boldsymbol{t}^*(\tilde{\boldsymbol{u}})) &= \Big(F_{\boldsymbol{t}}(\boldsymbol{t}^*(\tilde{\boldsymbol{u}})) + \lambda^*(\tilde{\boldsymbol{u}}) g_{\boldsymbol{t}}(\boldsymbol{t}^*(\tilde{\boldsymbol{u}}), \tilde{\boldsymbol{u}}) + \mu^*(\tilde{\boldsymbol{u}}) h_{\boldsymbol{t}}(\boldsymbol{t}^*(\tilde{\boldsymbol{u}}))\Big) \nabla_{\tilde{\boldsymbol{u}}} \boldsymbol{t}^*(\tilde{\boldsymbol{u}}) \\ &\quad + \lambda^*(\tilde{\boldsymbol{u}}) g_{\tilde{\boldsymbol{u}}}(\boldsymbol{t}^*(\tilde{\boldsymbol{u}}), \tilde{\boldsymbol{u}}) \\ &= \lambda^*(\tilde{\boldsymbol{u}}) g_{\tilde{\boldsymbol{u}}}(\boldsymbol{t}^*(\tilde{\boldsymbol{u}}), \tilde{\boldsymbol{u}}),\end{aligned}$$

where the final equality comes from noting that the terms in the large bracket vanish due to equation (A.5). Recalling the expression for $g_{\tilde{\boldsymbol{u}}}(\boldsymbol{t}; \tilde{\boldsymbol{u}})$ given by Equation A.4 and that $\boldsymbol{v}^*(\tilde{\boldsymbol{u}}) = \exp(\boldsymbol{t}^*(\tilde{\boldsymbol{u}}))$ we obtain

$$\begin{aligned}\nabla_{\tilde{\boldsymbol{u}}} F(\boldsymbol{t}^*(\tilde{\boldsymbol{u}})) &= \lambda^*(\tilde{\boldsymbol{u}}) \Big(1 - \boldsymbol{t}^*(\tilde{\boldsymbol{u}})_1 + \log u_1, \ldots, 1 - \boldsymbol{t}^*(\tilde{\boldsymbol{u}})_M + \log u_M, -1\Big) \\ &= \lambda^*(\tilde{\boldsymbol{u}}) \left(1 + \log \frac{u_1}{\boldsymbol{v}^*(\tilde{\boldsymbol{u}})_1}, \ldots, 1 + \log \frac{u_M}{\boldsymbol{v}^*(\tilde{\boldsymbol{u}})_M}, -1\right)\end{aligned}$$

Finally, recalling Equivalence (A.3), namely $\nabla_{\tilde{\boldsymbol{u}}} f^*_{\boldsymbol{\ell}}(\tilde{\boldsymbol{u}}) \equiv \nabla_{\tilde{\boldsymbol{u}}} F(\boldsymbol{t}^*(\tilde{\boldsymbol{u}}))$, we see that the above expression gives the derivatives $\frac{\partial f^*_{\boldsymbol{\ell}}}{\partial u_j}(\tilde{\boldsymbol{u}})$ and $\frac{\partial f^*_{\boldsymbol{\ell}}}{\partial c}(\tilde{\boldsymbol{u}})$ stated in the proposition, thus completing the proof. $\square$

Appendix B

Additional Material for Chapter 4

Figure B.1 demonstrates a case in which compression via k-means can produce a tighter discrete PAC-Bayes bound than that produced without compression, as promised in Section 4.5.1.

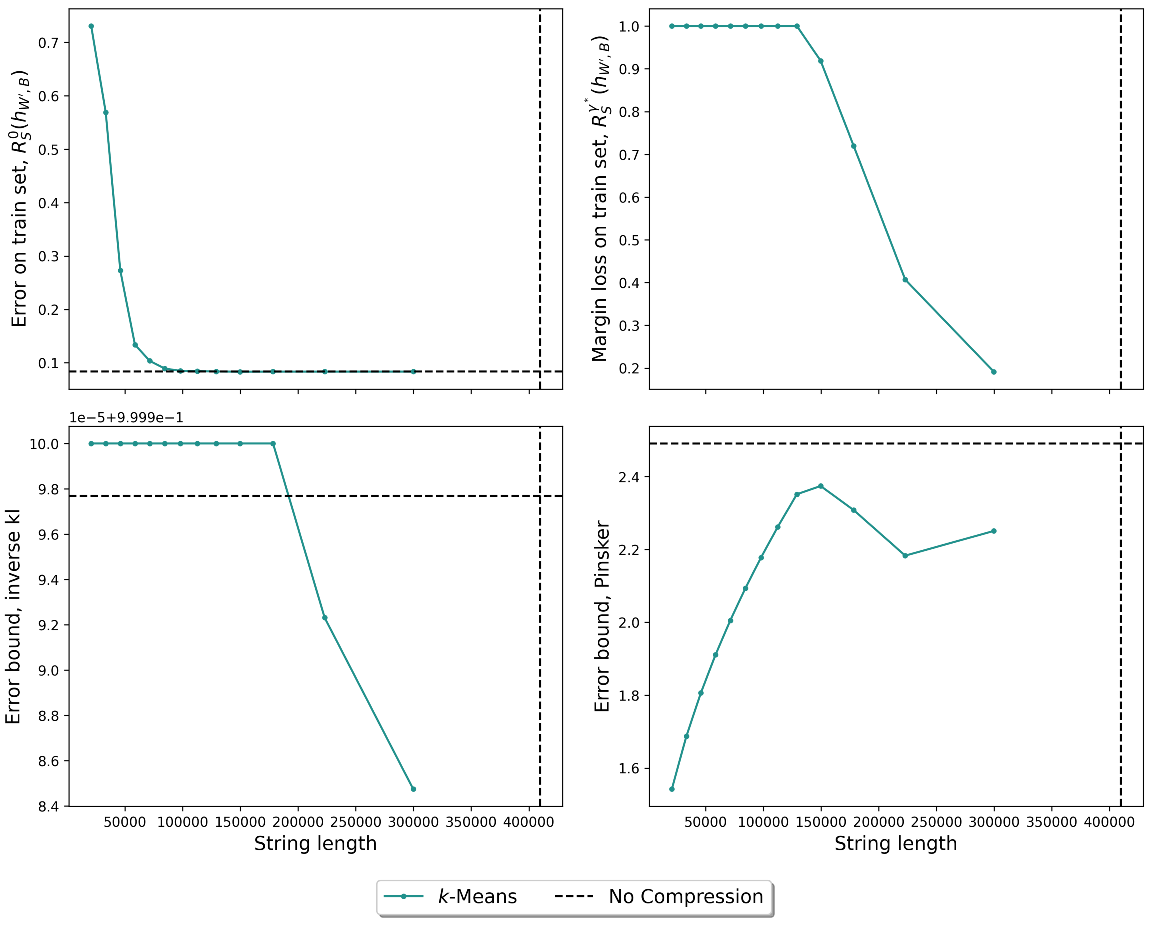
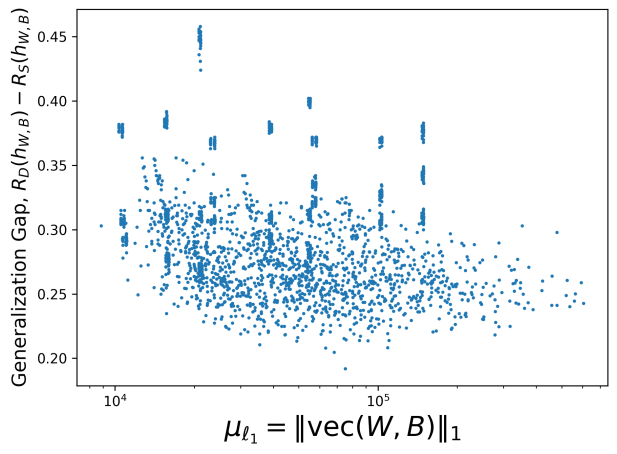


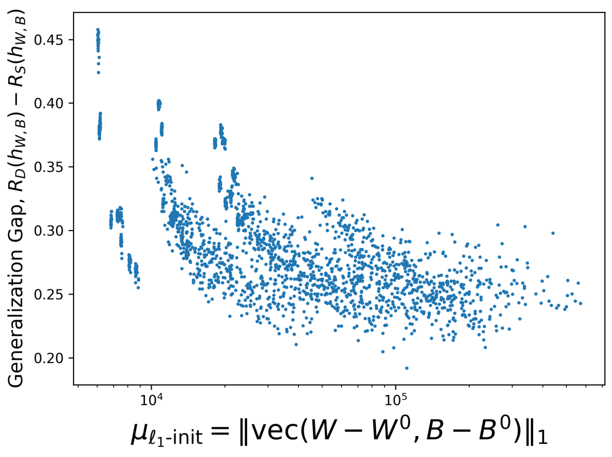
Figure B.1: **Quantisation via k-means** for $h_{W,B} \in \mathtt{MLP}_{(40,512,10)}$ trained on MNIST1D. Vertical line shows string length of uncompressed network. Top left: As k increases the error of the compressed network converges to the error of the original (horizontal line). Top right: The margin loss of the compressed network remains trivial until $k = 1024$. Bottom left: Error bounds (inverting kl) eventually decrease below the discrete PAC-Bayes bound without compression (horizontal line), albeit by an almost trivial amount. Bottom right: Error bounds (Pinsker's inequality) increase with k up to a peak at $k = 1024$ where there is a modest decrease; all bound values improve over the PAC-Bayes bound without compression (horizontal line).

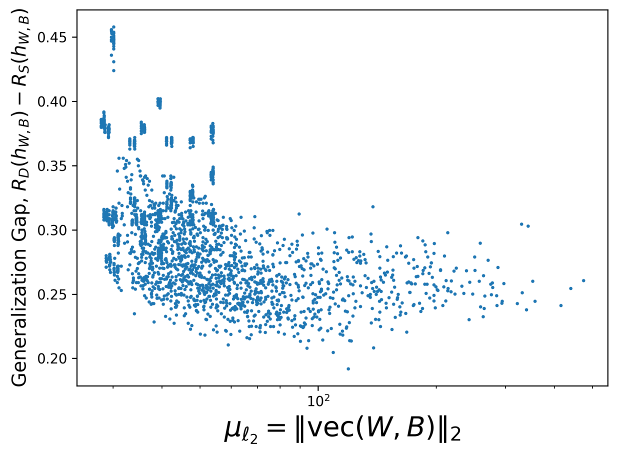
Appendix C

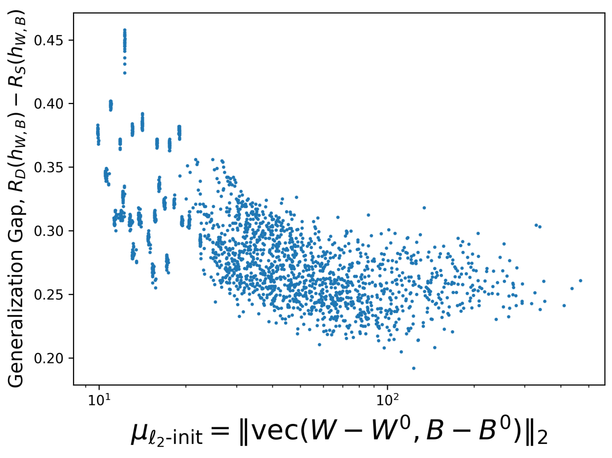
Additional Material for Chapter 5

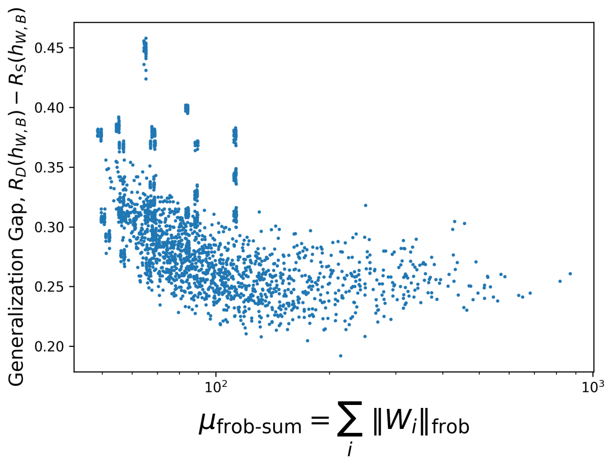
C.1 Additional complexity measure plots

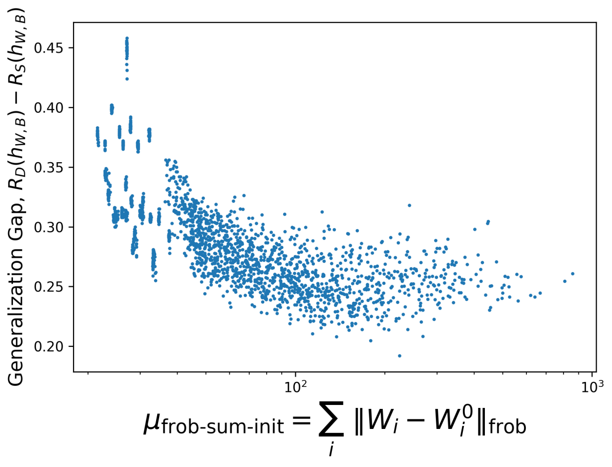


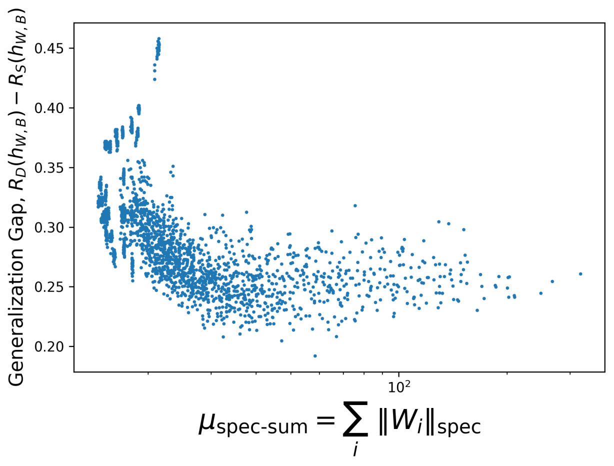


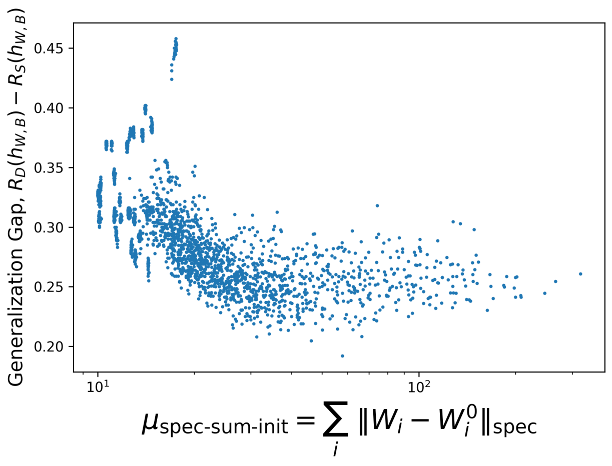


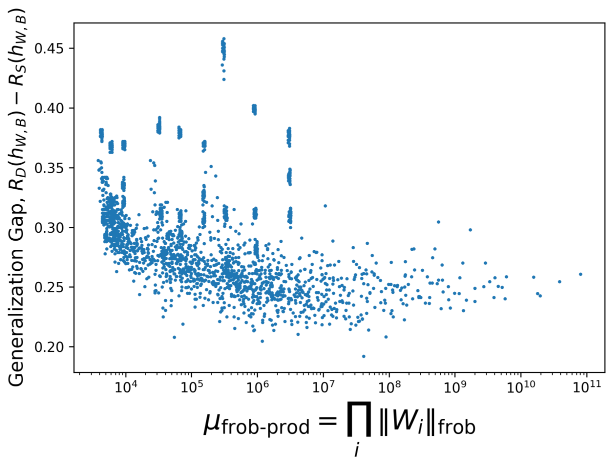


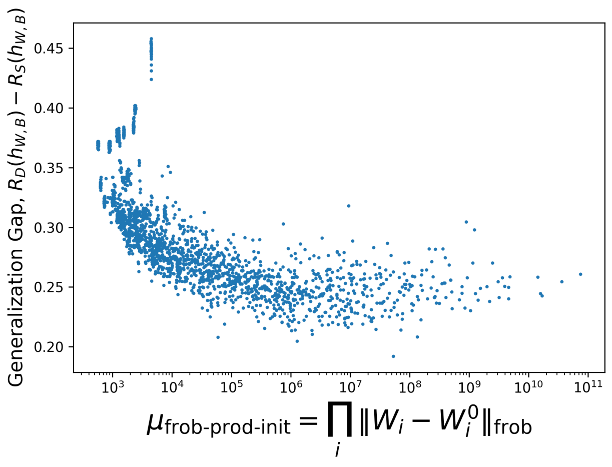

Generalization Gap, $R_D(h_{W,B}) - R_S(h_{W,B})$
0.45
0.40
0.35
0.30
0.25
0.20
10^2
10^3
$\mu_{\text{frob-sum}} = \sum_i \|W_i\|_{\text{frob}}$

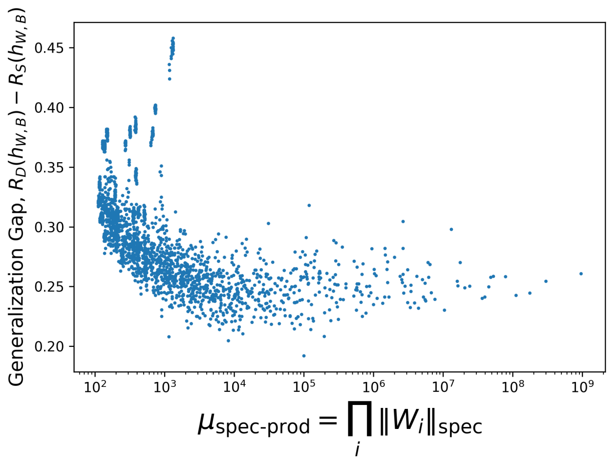

Generalization Gap, $R_D(h_{W,B}) - R_S(h_{W,B})$
0.45
0.40
0.35
0.30
0.25
0.20
10^2
10^3
$\mu_{\text{frob-sum-init}} = \sum_i \|W_i - W_i^0\|_{\text{frob}}$


Generalization Gap, $R_D(h_{W,B}) - R_S(h_{W,B})$
0.45
0.40
0.35
0.30
0.25
0.20
10^2
$\mu_{\text{spec-sum}} = \sum_i \|W_i\|_{\text{spec}}$

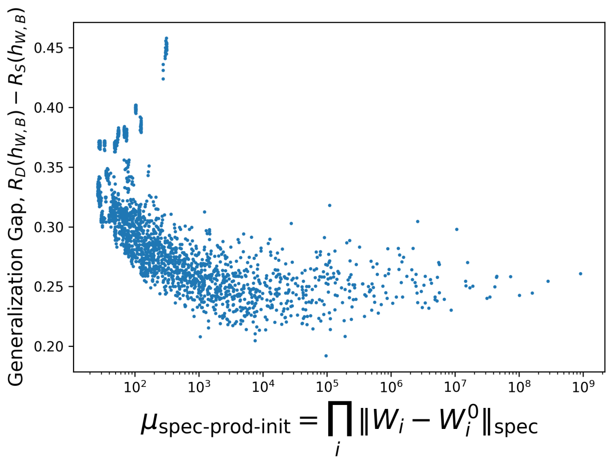

Generalization Gap, $R_D(h_{W,B}) - R_S(h_{W,B})$
0.45
0.40
0.35
0.30
0.25
0.20
10^1
10^2
$\mu_{\text{spec-sum-init}} = \sum_i \|W_i - W_i^0\|_{\text{spec}}$


Generalization Gap, $R_D(h_{W,B}) - R_S(h_{W,B})$
0.45
0.40
0.35
0.30
0.25
0.20
10^4
10^5
10^6
10^7
10^8
10^9
10^{10}
10^{11}
$\mu_{\text{frob-prod}} = \prod_i \|W_i\|_{\text{frob}}$

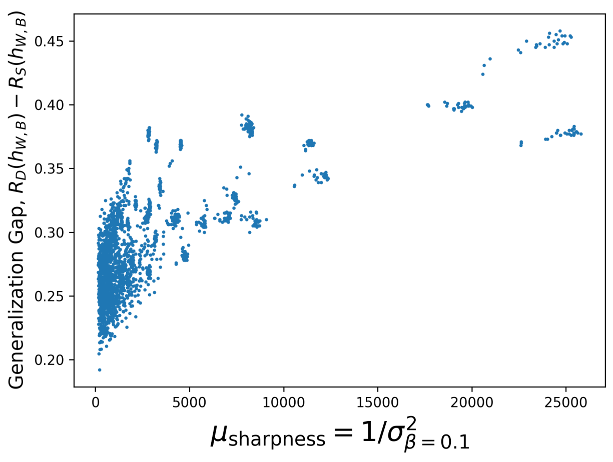

Generalization Gap, $R_D(h_{W,B}) - R_S(h_{W,B})$
0.45
0.40
0.35
0.30
0.25
0.20
10^3
10^4
10^5
10^6
10^7
10^8
10^9
10^{10}
10^{11}
$\mu_{\text{frob-prod-init}} = \prod_i \|W_i - W_i^0\|_{\text{frob}}$



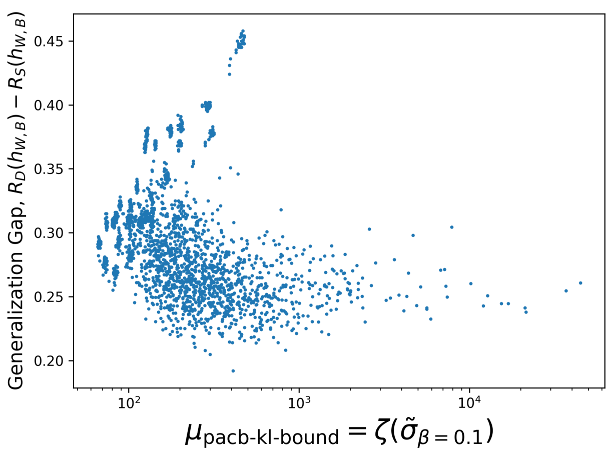
Generalization Gap, $R_D(h_{W,B}) - R_S(h_{W,B})$
0.20
0.25
0.30
0.35
0.40
0.45
10^2
10^3
10^4
10^5
10^6
10^7
10^8
10^9
$\mu_{\text{spec-prod}} = \prod_i \|W_i\|_{\text{spec}}$



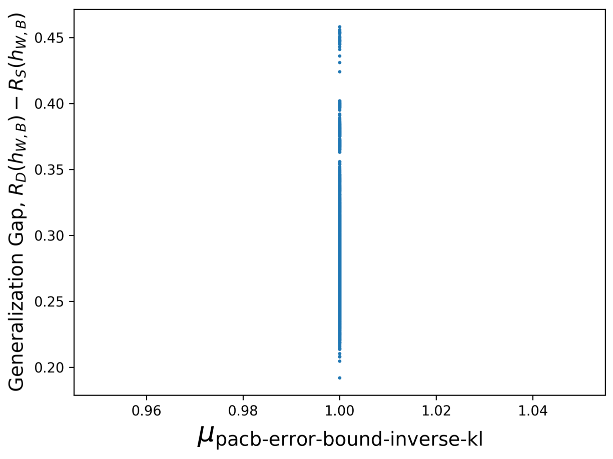
Generalization Gap, $R_D(h_{W,B}) - R_S(h_{W,B})$
0.20
0.25
0.30
0.35
0.40
0.45
10^2
10^3
10^4
10^5
10^6
10^7
10^8
10^9
$\mu_{\text{spec-prod-init}} = \prod_i \|W_i - W_i^0\|_{\text{spec}}$



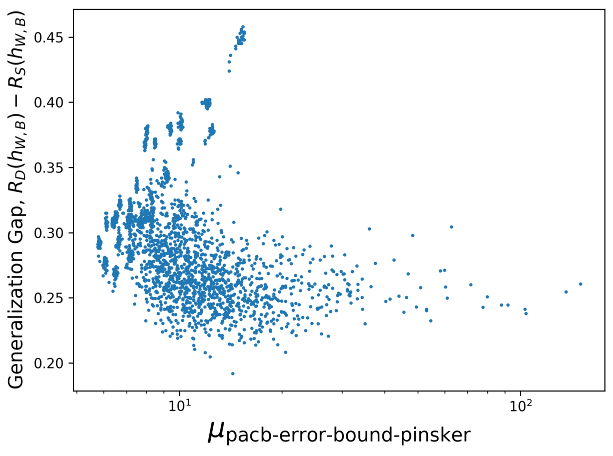
Generalization Gap, $R_D(h_{W,B}) - R_S(h_{W,B})$
0.20
0.25
0.30
0.35
0.40
0.45
0
5000
10000
15000
20000
25000
$\mu_{\text{sharpness}} = 1/\sigma^2_{\beta=0.1}$



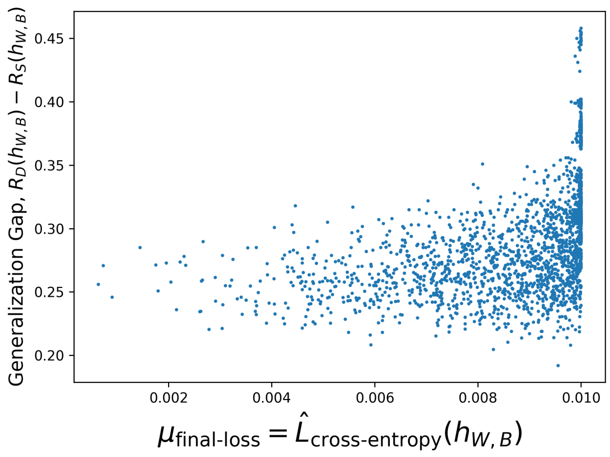
Generalization Gap, $R_D(h_{W,B}) - R_S(h_{W,B})$
0.20
0.25
0.30
0.35
0.40
0.45
10^2
10^3
10^4
$\mu_{\text{pacb-kl-bound}} = \zeta(\tilde{\sigma}_{\beta=0.1})$

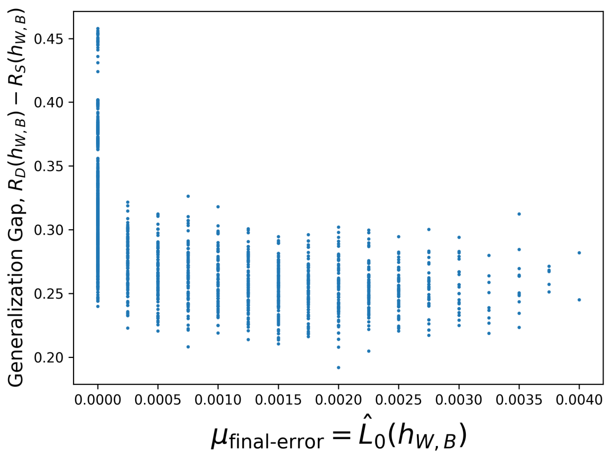


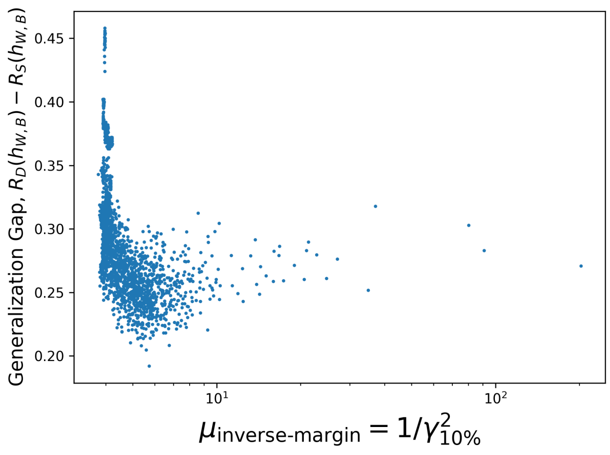
Generalization Gap, $R_D(h_{W,B}) - R_S(h_{W,B})$
0.20
0.25
0.30
0.35
0.40
0.45
0.96
0.98
1.00
1.02
1.04
$\mu_{\text{pacb-error-bound-inverse-kl}}$

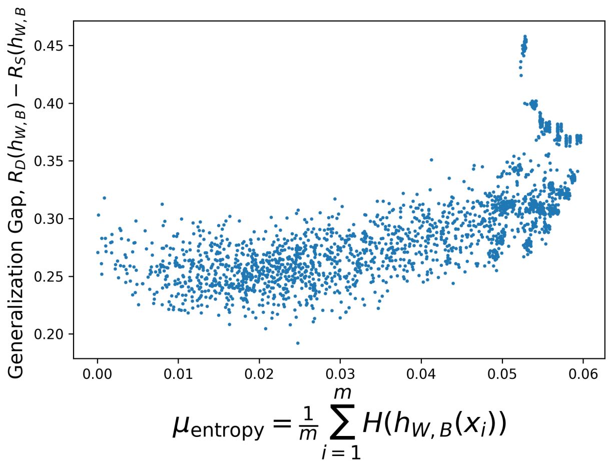


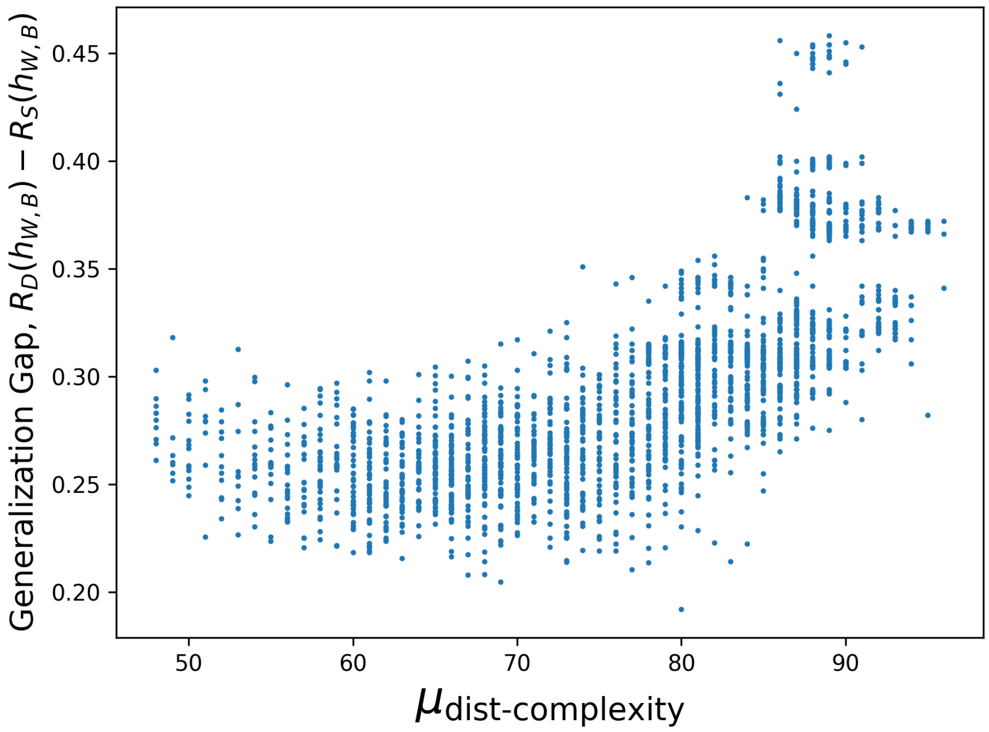
Generalization Gap, $R_D(h_{W,B}) - R_S(h_{W,B})$
0.20
0.25
0.30
0.35
0.40
0.45
10^1
10^2
$\mu_{\text{pacb-error-bound-pinsker}}$


Generalization Gap, $R_D(h_{W,B}) - R_S(h_{W,B})$
0.20
0.25
0.30
0.35
0.40
0.45
0.002
0.004
0.006
0.008
0.010
$\mu_{\text{final-loss}} = \hat{L}_{\text{cross-entropy}}(h_{W,B})$


Generalization Gap, $R_D(h_{W,B}) - R_S(h_{W,B})$
0.20
0.25
0.30
0.35
0.40
0.45
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025
0.0030
0.0035
0.0040
$\mu_{\text{final-error}} = \hat{L}_0(h_{W,B})$


Generalization Gap, $R_D(h_{W,B}) - R_S(h_{W,B})$
0.20
0.25
0.30
0.35
0.40
0.45
10^1
10^2
$\mu_{\text{inverse-margin}} = 1/\gamma^2_{10\%}$


Generalization Gap, $R_D(h_{W,B}) - R_S(h_{W,B}$
0.20
0.25
0.30
0.35
0.40
0.45
0.00
0.01
0.02
0.03
0.04
0.05
0.06
$\mu_{\text{entropy}} = \frac{1}{m}\sum_{i=1}^{m} H(h_{W,B}(x_i))$


Generalization Gap, $R_D(h_{W,B}) - R_S(h_{W,B})$
0.20
0.25
0.30
0.35
0.40
0.45
50
60
70
80
90
$\mu_{\text{dist-complexity}}$

Bibliography

Abu-Mostafa, Y. S., M. Magdon-Ismail, and H.-T. Lin (2012). *Learning from data*. Vol. 4. AMLBook New York.

Adams, R., J. Shawe-Taylor, and B. Guedj (2024). "Controlling multiple errors simultaneously with a PAC-Bayes bound". In: *Advances in Neural Information Processing Systems* 37, pp. 5308–5337.

Alquier, P. et al. (2024). "User-friendly introduction to PAC-Bayes bounds". In: *Foundations and Trends® in Machine Learning* 17.2, pp. 174–303.

Alquier, P., J. Ridgway, and N. Chopin (2016). "On the properties of variational approximations of Gibbs posteriors". In: *The Journal of Machine Learning Research* 17.1, pp. 8374–8414.

Ambroladze, A., E. Parrado-Hernández, and J. Shawe-Taylor (2006). "Tighter PAC-Bayes bounds". In: *Advances in neural information processing systems* 19.

Anthropic (2024). *The Claude 3 Model Family: Opus, Sonnet, Haiku*. Technical Report. Anthropic.

Arora, S., R. Ge, B. Neyshabur, et al. (2018). "Stronger generalization bounds for deep nets via a compression approach". In: *International conference on machine learning*. PMLR, pp. 254–263.

Banerjee, A., T. Chen, and Y. Zhou (2020). "De-randomized PAC-Bayes margin bounds: Applications to non-convex and non-smooth predictors". In: *arXiv*. arXiv: 2002.09956.

Bartlett, P., V. Maiorov, and R. Meir (1998). "Almost linear VC dimension bounds for piecewise polynomial networks". In: *Advances in neural information processing systems* 11.

Bartlett, P. L. (1993). "Lower bounds on the Vapnik-Chervonenkis dimension of multi-layer threshold networks". In: *Proceedings of the sixth annual conference on Computational learning theory*, pp. 144–150.

Bartlett, P. L., D. J. Foster, and M. J. Telgarsky (2017). "Spectrally-normalized margin bounds for neural networks". In: *Advances in neural information processing systems* 30.

Bartlett, P. L., N. Harvey, C. Liaw, et al. (2019). "Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks". In: *Journal of Machine Learning Research* 20.63, pp. 1–17.

Bartlett, P. L. and S. Mendelson (2001). "Rademacher and Gaussian complexities: Risk bounds and structural results". In: *International Conference on Computational Learning Theory*. Springer, pp. 224–240.

Bartlett, P. L. and S. Mendelson (2002). “Rademacher and gaussian complexities: Risk bounds and structural results”. In: *Journal of Machine Learning Research* 3.Nov, pp. 463–482.

Bégin, L., P. Germain, F. Laviolette, et al. (2016). “PAC-Bayesian bounds based on the Rényi divergence”. In: *Artificial Intelligence and Statistics*. PMLR, pp. 435–444.

Belkin, M., D. Hsu, S. Ma, et al. (2019). “Reconciling modern machine-learning practice and the classical bias–variance trade-off”. In: *Proceedings of the National Academy of Sciences* 116.32, pp. 15849–15854.

Benabbou, L. and P. Lang (2017). “PAC-Bayesian generalization bound for multi-class learning”. In: *NIPS 2017 Workshop.(Almost) 50 Shades of Bayesian Learning: PAC-Bayesian trends and insights*.

Biggs, F. and B. Guedj (2023). “Tighter PAC-Bayes Generalisation Bounds by Leveraging Example Difficulty”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 8165–8182.

Biggs, F., V. Zantedeschi, and B. Guedj (2022). “On margins and generalisation for voting classifiers”. In: *Advances in Neural Information Processing Systems* 35, pp. 9713–9726.

Blanchard, G. and F. Fleuret (2007). “Occam’s hammer”. In: *International Conference on Computational Learning Theory*. Springer, pp. 112–126.

Bousquet, O., S. Boucheron, and G. Lugosi (2003). “Introduction to statistical learning theory”. In: *Summer school on machine learning*. Springer, pp. 169–207.

Catoni, O. (2003). *A PAC-Bayesian approach to adaptive classification*. Preprint 840.2. URL: yaroslavvb.com/papers/notes/catoni-pac.pdf.

Catoni, O. (2007). *PAC-Bayesian Supervised Classification: The Thermodynamics of Statistical Learning*. Vol. 56. Institute of Mathematical Statistics (IMS) Lecture Notes - Monograph Series. Institute of Mathematical Statistics. URL: https://books.google.fr/books?id=acnaAAAAMAAJ.

Clerico, E., G. Deligiannidis, and A. Doucet (2022a). “Conditionally gaussian PAC-Bayes”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 2311–2329.

Clerico, E., T. Farghly, G. Deligiannidis, et al. (2022b). “Generalisation under gradient descent via deterministic PAC-Bayes”. In: *arXiv*. arXiv: 2209.02525.

Clopper, C. J. and E. S. Pearson (1934). “The use of confidence or fiducial limits illustrated in the case of the binomial”. In: *Biometrika* 26.4, pp. 404–413.

Csiszár, I. (1975). “I-divergence geometry of probability distributions and minimization problems”. In: *The Annals of Probability*, pp. 146–158.

Cybenko, G. (1989). “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4, pp. 303–314.

Dinh, L., R. Pascanu, S. Bengio, et al. (2017). “Sharp minima can generalize for deep nets”. In: *International Conference on Machine Learning*. PMLR, pp. 1019–1028.

Donsker, M. and S. Varadhan (1975). “Large deviations for Markov processes and the asymptotic evaluation of certain Markov process expectations for large times”. In: *Probabilistic Methods in Differential Equations*. Springer, pp. 82–88.

Donsker, M. D. and S. S. Varadhan (1976). “Asymptotic evaluation of certain Markov process expectations for large time—III”. In: *Communications on pure and applied Mathematics* 29.4, pp. 389–461.

Duin, R. P. (2000). “Classifiers in almost empty spaces”. In: *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000.* Vol. 2. IEEE, pp. 1–7.

Dziugaite, G. K. and D. M. Roy (2017). “Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data”. In: *arXiv.* arXiv: 1703.11008.

Dziugaite, G. K. and D. M. Roy (2018). “Data-dependent PAC-Bayes priors via differential privacy”. In: *Advances in Neural Information Processing Systems*, pp. 8430–8441.

Eckart, C. and G. Young (1936). “The approximation of one matrix by another of lower rank”. In: *Psychometrika* 1.3, pp. 211–218.

Erven, T. van (2014). “PAC-Bayes mini-tutorial: A continuous union bound”. In: *arXiv.* arXiv: 1405.1580.

Feofanov, V., E. Devijver, and M.-R. Amini (2019). “Transductive bounds for the multi-class majority vote classifier”. In: *Proceedings of the AAAI Conference on Artificial Intelligence.* Vol. 33. 01, pp. 3566–3573.

Foong, A., W. Bruinsma, D. Burt, et al. (2021). “How tight can PAC-Bayes be in the small data regime?” In: *Advances in Neural Information Processing Systems* 34, pp. 4093–4105.

Freund, Y. (1998). “Self bounding learning algorithms”. In: *Proceedings of the eleventh annual conference on Computational Learning Theory*, pp. 247–258.

Ganguli, D., D. Hernandez, L. Lovitt, et al. (2022). “Predictability and surprise in large generative models”. In: *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pp. 1747–1764.

Germain, P., A. Lacasse, F. Laviolette, et al. (2009). “PAC-Bayesian learning of linear classifiers”. In: *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 353–360.

Germain, P., A. Lacasse, F. Laviolette, et al. (2015). “Risk bounds for the majority vote: From a PAC-Bayesian analysis to a learning algorithm”. In: *arXiv.* arXiv: 1503.08329.

Ghosh, R. and M. Motani (2021). “Network-to-network regularization: enforcing occam’s razor to improve generalization”. In: *Advances in neural information processing systems* 34, pp. 6341–6352.

Greenblatt, R., C. Denison, B. Wright, et al. (2024). “Alignment faking in large language models”. In: *arXiv preprint arXiv:2412.14093.*

Greydanus, S. and D. Kobak (2020). “Scaling down deep learning with mnist-1d”. In: *arXiv.* arXiv: 2011.14439.

Grünwald, P. (2005). “Minimum description length tutorial”. In: *Advances in minimum description length: Theory and applications* 5, pp. 1–80.

Grünwald, P. and T. Roos (2019). “Minimum description length revisited”. In: *International journal of mathematics for industry* 11.01, p. 1930001.

Hinton, G., O. Vinyals, and J. Dean (2015). “Distilling the knowledge in a neural network”. In: *arXiv*. arXiv: 1503.02531.

Hinton, G. E. and D. Van Camp (1993). “Keeping the neural networks simple by minimizing the description length of the weights”. In: *Proceedings of the sixth annual conference on Computational learning theory*, pp. 5–13.

Hochreiter, S. and J. Schmidhuber (1994). “Simplifying neural nets by discovering flat minima”. In: *Advances in neural information processing systems* 7.

Hochreiter, S. and J. Schmidhuber (1997). “Flat minima”. In: *Neural computation* 9.1, pp. 1–42.

Hoffmann, J., S. Borgeaud, A. Mensch, et al. (2022). “Training compute-optimal large language models”. In: *arXiv*. arXiv: 2203.15556.

Hornik, K., M. Stinchcombe, and H. White (1989). “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5, pp. 359–366.

Jiang, Y., P. Foret, S. Yak, et al. (2020). “Neurips 2020 competition: Predicting generalization in deep learning”. In: *arXiv*. arXiv: 2012.07976.

Jiang, Y., D. Krishnan, H. Mobahi, et al. (2018). “Predicting the generalization gap in deep networks with margin distributions”. In: *arXiv*. arXiv: 1810.00113.

Jiang, Y., B. Neyshabur, H. Mobahi, et al. (2019). “Fantastic generalization measures and where to find them”. In: *arXiv*. arXiv: 1912.02178.

Kaplan, J., S. McCandlish, T. Henighan, et al. (2020). “Scaling laws for neural language models”. In: *arXiv*. arXiv: 2001.08361.

Keskar, N. S., D. Mudigere, J. Nocedal, et al. (2016). “On large-batch training for deep learning: Generalization gap and sharp minima”. In: *arXiv*. arXiv: 1609.04836.

Koço, S. and C. Capponi (2013). “On multi-class classification through the minimization of the confusion matrix norm”. In: *Asian Conference on Machine Learning*. PMLR, pp. 277–292.

Koh, P. W., S. Sagawa, H. Marklund, et al. (2021). “Wilds: A benchmark of in-the-wild distribution shifts”. In: *International conference on machine learning*. PMLR, pp. 5637–5664.

Koltchinskii, V. and D. Panchenko (2000). “Rademacher processes and bounding the risk of function learning”. In: *High dimensional probability II*. Springer, pp. 443–457.

Krizhevsky, A., G. Hinton, et al. (2009). “Learning multiple layers of features from tiny images”. In.

Kuhn, L., C. Lyle, A. N. Gomez, et al. (2021). “Robustness to pruning predicts generalization in deep neural networks”. In: *arXiv*. arXiv: 2103.06002.

Lacasse, A., F. Laviolette, M. Marchand, et al. (2006). “PAC-Bayes bounds for the risk of the majority vote and the variance of the Gibbs classifier”. In: *Advances in Neural information processing systems* 19.

Langford, J. and A. Blum (2003). “Microchoice bounds and self bounding learning algorithms”. In: *Machine Learning* 51, pp. 165–179.

Langford, J. and R. Caruana (2001). “(Not) bounding the true error”. In: *Advances in Neural Information Processing Systems* 14.

Langford, J. and R. Schapire (2005). “Tutorial on practical prediction theory for classification.” In: *Journal of machine learning research* 6.3.

Langford, J. and M. Seeger (2001). *Bounds for averaging classifiers.* School of Computer Science, Carnegie Mellon University.

Langford, J. and J. Shawe-Taylor (2002). “PAC-Bayes & margins”. In: *Advances in neural information processing systems* 15.

Laviolette, F., E. Morvant, L. Ralaivola, et al. (2017). “Risk upper bounds for general ensemble methods with an application to multiclass classification”. In: *Neurocomputing* 219, pp. 15–25.

Lei, Y., Ü. Dogan, D.-X. Zhou, et al. (2019). “Data-dependent generalization bounds for multi-class classification”. In: *IEEE Transactions on Information Theory* 65.5, pp. 2995–3021.

Letarte, G., P. Germain, B. Guedj, et al. (2019). “Dichotomize and generalize: PAC-Bayesian binary activated deep neural networks”. In: *Advances in Neural Information Processing Systems* 32.

Lever, G., F. Laviolette, and J. Shawe-Taylor (2013). “Tighter PAC-Bayes bounds through distribution-dependent priors”. In: *Theoretical Computer Science* 473, pp. 4–28.

Loog, M., T. Viering, A. Mey, et al. (2020). “A brief prehistory of double descent”. In: *Proceedings of the National Academy of Sciences* 117.20, pp. 10625–10626.

Lotfi, S., M. Finzi, S. Kapoor, et al. (2022). “PAC-Bayes compression bounds so tight that they can explain generalization”. In: *Advances in Neural Information Processing Systems* 35, pp. 31459–31473.

Maass, W. (1994). “Neural nets with superlinear VC-dimension”. In: *Neural Computation* 6.5, pp. 877–884.

Maurer, A. (2004). “A note on the PAC Bayesian theorem”. In: *arXiv*. arXiv: cs/0411099.

McAllester, D. A. (1998). “Some PAC-Bayesian theorems”. In: *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 230–234.

McAllester, D. A. (1999). “PAC-Bayesian model averaging”. In: *Proceedings of the twelfth annual conference on Computational learning theory*, pp. 164–170.

McAllester, D. A. (2003). “PAC-Bayesian stochastic model selection”. In: *Machine Learning* 51.1, pp. 5–21.

Mohamed, S., M. Rosca, M. Figurnov, et al. (2020). “Monte carlo gradient estimation in machine learning”. In: *Journal of Machine Learning Research* 21.132, pp. 1–62.

Morvant, E., S. Koço, and L. Ralaivola (2012). “PAC-Bayesian generalization bound on confusion matrix for multi-class classification”. In: *arXiv*. arXiv: 1202.6228.

Nagarajan, V. and J. Z. Kolter (2019). “Deterministic PAC-Bayesian generalization bounds for deep networks via generalizing noise-resilience”. In: *arXiv*. arXiv: 1905.13344.

Nakkiran, P., G. Kaplun, Y. Bansal, et al. (2021). “Deep double descent: Where bigger models and more data hurt”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2021.12, p. 124003.

Nanda, N., L. Chan, T. Lieberum, et al. (2023). "Progress measures for grokking via mechanistic interpretability". In: *arXiv preprint arXiv:2301.05217*.

Neyshabur, B., S. Bhojanapalli, D. McAllester, et al. (2017a). "Exploring generalization in deep learning". In: *Advances in Neural Information Processing Systems*, pp. 5947–5956.

Neyshabur, B., S. Bhojanapalli, and N. Srebro (2017b). "A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks". In: *arXiv*. arXiv: 1707.09564.

Parrado-Hernández, E., A. Ambroladze, J. Shawe-Taylor, et al. (2012). "PAC-Bayes bounds with data dependent priors". In: *The Journal of Machine Learning Research* 13.1, pp. 3507–3531.

Pereyra, G., G. Tucker, J. Chorowski, et al. (2017). "Regularizing neural networks by penalizing confident output distributions". In: *arXiv*. arXiv: 1701.06548.

Perez-Ortiz, M., O. Rivasplata, B. Guedj, et al. (2021). "Learning PAC-Bayes priors for probabilistic neural networks". In: *arXiv*. arXiv: 2109.10304.

Pérez-Ortiz, M., O. Rivasplata, J. Shawe-Taylor, et al. (2021). "Tighter risk certificates for neural networks". In: *Journal of Machine Learning Research* 22.227, pp. 1–40.

Pires, B. A., C. Szepesvari, and M. Ghavamzadeh (2013). "Cost-sensitive multiclass classification risk bounds". In: *International Conference on Machine Learning*. PMLR, pp. 1391–1399.

Polyanskiy, Y. and Y. Wu (2014). "Lecture notes on information theory". In: *Lecture Notes for ECE563 (UIUC) and* 6.2012-2016, p. 7.

Quiñonero-Candela, J., M. Sugiyama, A. Schwaighofer, et al. (2022). *Dataset shift in machine learning*. Mit Press.

Rissanen, J. (1978). "Modeling by shortest data description". In: *Automatica* 14.5, pp. 465–471.

Santurkar, S., D. Tsipras, A. Ilyas, et al. (2018). "How does batch normalization help optimization?" In: *Advances in neural information processing systems* 31.

Schaeffer, R., B. Miranda, and S. Koyejo (2023). "Are emergent abilities of large language models a mirage?" In: *Advances in neural information processing systems* 36, pp. 55565–55581.

Schmidhuber, J. (1997). "Discovering neural nets with low Kolmogorov complexity and high generalization capability". In: *Neural Networks* 10.5, pp. 857–873.

Seeger, M. (2002). "Pac-Bayesian generalisation error bounds for gaussian process classification". In: *Journal of machine learning research* 3.Oct, pp. 233–269.

Shalev-Shwartz, S. and S. Ben-David (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Shawe-Taylor, J. and R. C. Williamson (1997). "A PAC analysis of a Bayesian estimator". In: *Proceedings of the tenth annual conference on Computational learning theory*, pp. 2–9.

Srivastava, A., A. Rastogi, A. Rao, et al. (2022). "Beyond the imitation game: Quantifying and extrapolating the capabilities of language models". In: *arXiv*. arXiv: 2206.04615.

Takayama, A. and T. Akira (1985). *Mathematical economics*. Cambridge university press.

Topsøe, F. (1967). “An information theoretical identity and a problem involving capacity”. In: *Studia Scientiarum Mathematicarum Hungarica* 2.291-292, p. 246.

Tschandl, P., C. Rosendahl, and H. Kittler (2018). “The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions”. In: *Scientific data* 5.1, pp. 1–9.

Unterthiner, T., D. Keysers, S. Gelly, et al. (2020). “Predicting neural network accuracy from weights”. In: *arXiv*. arXiv: 2002.11448.

Valle-Perez, G., C. Q. Camargo, and A. A. Louis (2018). “Deep learning generalizes because the parameter-function map is biased towards simple functions”. In: *arXiv preprint arXiv:1805.08522*.

Vallet, F., J.-G. Cailton, and P. Refregier (1989). “Linear and nonlinear extension of the pseudo-inverse solution for learning boolean functions”. In: *Europhysics Letters* 9.4, p. 315.

Van Erven, T. and P. Harremos (2014). “Rényi divergence and Kullback-Leibler divergence”. In: *IEEE Transactions on Information Theory* 60.7, pp. 3797–3820.

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

Vapnik, V. and A. Y. Chervonenkis (1974). “The method of ordered risk minimization, I”. In: *Avtomatika i Telemekhanika* 8, pp. 21–30.

Vapnik, V. N. (1999). “An overview of statistical learning theory”. In: *IEEE transactions on neural networks* 10.5, pp. 988–999.

Vapnik, V. N. and A. Y. Chervonenkis (2015). “On the uniform convergence of relative frequencies of events to their probabilities”. In: *Measures of complexity: festschrift for alexey chervonenkis*. Springer, pp. 11–30.

Verma, T. S. and J. Pearl (2022). “Equivalence and synthesis of causal models”. In: *Probabilistic and causal inference: The works of Judea Pearl*, pp. 221–236.

Viallard, P., P. Germain, A. Habrard, et al. (2024). “A general framework for the practical disintegration of PAC-Bayesian bounds”. In: *Machine Learning* 113.2, pp. 519–604.

Wei, J., Y. Tay, R. Bommasani, et al. (2022a). “Emergent abilities of large language models”. In: *arXiv*. arXiv: 2206.07682.

Wei, J., X. Wang, D. Schuurmans, et al. (2022b). “Chain-of-thought prompting elicits reasoning in large language models”. In: *Advances in neural information processing systems* 35, pp. 24824–24837.

Wu, Y.-S. and Y. Seldin (2022). “Split-kl and PAC-Bayes-split-kl inequalities for ternary random variables”. In: *Advances in Neural Information Processing Systems* 35, pp. 11369–11381.

Yudkowsky, E. (2015). “A semi-technical introductory dialogue on Solomonoff induction”. https://www.lesswrong.com/posts/EL4HNa92Z95FKL9R2/a-semitechnical-introductory-dialogue-on-solomonoff-1, accessed May 15, 2025.

Zhang, C., S. Bengio, M. Hardt, et al. (2016). “Understanding deep learning requires rethinking generalization”. In: *arXiv*. arXiv: 1611.03530.

Zhou, W., V. Veitch, M. Austern, et al. (2018). “Non-vacuous generalization bounds at the imagenet scale: a PAC-Bayesian compression approach”. In: *arXiv*. arXiv: 1804.05862.