Towards Better Generative Models of Language

Gábor Melis

A thesis presented for the degree of
Doctor of Philosophy

at the

Computer Science Department,
University College London,
United Kingdom

April, 2023

UCL

I, Gábor Melis, 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.

ABSTRACT

> A man who could afford fifty dollars had a pair of boots that'd still be keeping his feet dry in ten years' time, while the poor man who could only afford cheap boots would have spent a hundred dollars on boots in the same time and would still have wet feet.
>
> *Capt. Samuel Vimes*

As language modelling has been progressing immensely towards genuinely useful applications driven by the increased availability of data and computational resources, our understanding and tools have not kept pace. On the one hand, it is usual for understanding to lag behind practice; on the other, often a steep price is to be paid for letting practice race too far ahead. A trained language model is the product of the interactions of data, optimization, the training objective, regularization, and the model itself, which not only provide rich veins for research individually but combine to create significant complexity, which hampers progress. This thesis pursues advances along these directions to solidify the footing and our understanding of the model and the process of its training in hopes of guiding research and applications towards better solutions.

IMPACT STATEMENT

> We don't measure our success by results but by activity, and the activity is considerable and productive.
>
> *Sir Humphrey Appleby*

Goals and contributions of this thesis include designing better recurrent neural networks and new methods of optimization as well as inference in latent variable models, with a focus on advancing language models. Recurrent neural networks are state-of-the-art models in small-scale language modelling and various other domains. We improve them even further, which has the most direct impact on machine learning practitioners. Ironically, the practice of language modelling is currently dominated by attention-based models due to their scalability, optimizability and the importance of training on large amounts of data. Can recurrent models match them with enough resources? We take a small step towards answering this question by scaling recurrent models with the help of our proposed optimization algorithm. We demonstrate that the improved recurrent models are capable of matching attention-based models on bigger – but still tiny by today's standards – datasets. Finding that the gap between these models is not nearly as large as previously believed, we turn our attention to other ways of improvement. We hope to inject structure and useful biases into the model with conditional independence assumptions and the shaping of the latent space, but posterior collapse gets in the way. To address this, we introduce a new inference method for latent variable models, opening the door for this class of generative models to be used successfully in practice. While the focus of this thesis is very specific, all of our contributions are applicable beyond language modelling.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisors, Phil Blunsom, Chris Dyer, and Thore Graepel, for instilling a coherent, deeply probabilistic worldview in me, for letting me find my own way in research but guiding me through the PhD process. I thank DeepMind for being a great working environment for so long and for sponsoring my studies at UCL. I am especially grateful to my collaborators and colleagues over the years, who furnished me with insight, valuable feedback, and the Oxford comma.

Végül, de nem utolsósorban, köszönöm szűkebb családomnak a bíztatást, hogy a doktorit kezdjem el, majd hogy fejezzem be. Legvégül talán köszönöttel tartozom azoknak, akik értékeiket belém plántálták, és egész biztosan hálával azoknak, akik megértésükkel, támogatásukkal, tudásukkal segítettek: szüleimnek, Hernádi Zsuzsa tanárnőnek, és a Toldy Ferenc gimnázium tanárainak.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 CONCERNING LANGUAGE

> So, to you, language is more than just a means of communication? Oh, of course it is, of course it is, of course it is, of course it is.
>
> *Fry & Laurie*

Language may not be fundamental for intelligence, but it is at the very least an expression of the higher-level cognitive abilities of humans and forms the communication substrate of our society. We capture thought (e.g. knowledge, abstractions, plans) in words, creating projections of our inner realities, which in turn are affected by sensory input. These projections retain only the interesting bits – those worth saying – thus we end up with a highly compressed representation of thought. To decode and understand this compressed form, humans rely on shared syntax and semantics. While the demarcation between syntax and semantics is fuzzy, they roughly correspond to rules of forming grammatical sentences and to grounding meaning in sensory experiences.

For machines to assist and interact with humans, it is natural to assume that they must understand and produce natural language, which after all evolved for this purpose.[1] To address differences between sensory experiences of machines and people, grounding semantics in real or simulated environments has been pursued for many years [Winograd, 1973, Hermann et al., 2017]. Clearly, this is a necessary step towards humanlike intelligence.

On the other hand, addressing the issue of grounding is unlikely to achieve this goal in itself as there are indications that child language acquisition is much more efficient [Cristia et al., 2017, Pullum and Scholz, 2002, Shneidman and Goldin-Meadow, 2012] and robust to sensory deprivation. Furthermore, machines generalize with low sample efficiency [Hoffmann et al., 2022, Wei et al., 2022, Belinkov and Bisk, 2017, Jia and Liang, 2017, Iyyer et al., 2018, Moosavi and Strube, 2017, Agrawal et al., 2016] due to the lack of systematicity [Dziri et al., 2023, Kuncoro et al., 2018] and their limited ability to chunk input sequences

[1] This is nonetheless an assumption because other means of human–machine communication may be possible.

into meaningful units [Cao and Rimell, 2021, Bostrom and Durrett, 2020, Wang et al., 2017].

Sandwiched between the proof of existence of better models in the form of children's language acquisition [Cristia et al., 2017, Pullum and Scholz, 2002] and the evidence for severe shortcomings of our current crop of models is the reason for focussing on pure language modelling before turning to semantics and the grounding problem. In this work, we simply ask how generalization can be improved and what is necessary to create better models of language. For the most part, our methodology will be highly quantitative: we evaluate on held-out data in standard language modelling datasets, while providing theoretical underpinnings where possible.

1.1 LANGUAGE MODELLING

Modelling language means having an approximate idea of how plausible a given piece of text is and being able to generate plausible looking text from scratch or continue from a prompt. Language models (LMs, for short) work primarily on the symbolic level, and the ones we are considering have no other perceptual modality.

More formally, statistical language models approximate a possibly unknown true distribution $p^\star(x)$ by assigning probabilities $p(x)$ to possible utterances $x \in \mathcal{X}$. The utterances (e.g. sentences or documents) are sequences of tokens $x_1, \ldots, x_{n_x}$ from a finite set of symbols (e.g. words or characters) called the vocabulary. These models are *generative* in the sense that they define a data generating process, from which new data points can be sampled.

In autoregressive LMs, $p(x)$ is factorized as $\prod_{i=1}^{n_x} p(x_i | x_{<i})$. These models can directly support typeahead prediction and also spell checking when combined with search. In other tasks, such as image labelling or machine translation, where additional information is available, the probabilities of utterances are modelled conditioned on their context (e.g. the image or the source document).

Note that the evaluation of $p(x)$ need not be tractable for learning and for some applications. In fact, it is defined only implicitly in GANs [Goodfellow et al., 2014], which instead rely on samples from $p(x)$ and approximate expectations based on them [Huszár, 2017b]. Whether masked language models such as BERT [Devlin et al., 2018] implicitly define a probability distribution over utterances is still an open question although Goyal et al. [2021] strongly suggest that they do.

Instead of modelling the probability of utterances $p(x)$ directly, one may introduce unobserved variables into the model. The reason for doing so is to

gain an explicit representation of some properties of the data – exactly which properties depends on the choice of prior, the conditional independence assumptions made, the biases of the the likelihood, and the details of optimization. With latent variables z, the marginal likelihood is $p(x) = E_{z \sim p(z)} p(x|z)$, and learning typically involves an approximation to this expectation, which makes the optimization problem harder. On the other hand, the explicit representation of the data provided by latent variables may be useful for learning about the underlying generative process, discovering structure in the data, principled representation learning, and improving generalization. Unfortunately, in the domain of language models, latent variable models that extract useful representations in the latents have trouble modelling the data. Conversely, latent variable models that model the data well capture very little information about data in the latents.

In the following, we introduce some examples of autoregressive models with tractable likelihoods and no latent variables.

1.2 N-GRAM MODELS

Recall that autoregressive models decompose the probability of an utterance into the product of the probabilities of next token predictions given the preceding ones: $p(x) = \prod_{i=1}^{n_x} p(x_i|x_{<i})$. A simple choice for $p(x_i|x_{<i})$ is the proportion of cases when the prefix $x_{<i}$ is followed by x_i:

$$p(x_i|x_{<i}) = \frac{C(x_1, \ldots, x_{i-1}, x_i)}{C(x_1, \ldots, x_{i-1})},$$

where the counts C are computed over the training corpus. This corresponds to taking a maximum likelihood estimate, which is overconfident in general.

More concretely, one problem with the above definition is that in many cases one or both counts may be zero. The denominator is zero when no documents in the training corpus start with the prefix $x_1, \ldots, x_i$. To somewhat alleviate this problem of sparsity, N-gram models assume that a token depends on only the preceding N tokens, that is $p(x_i|x_{<i}) = p(x_i|x_{i-N}, \ldots, x_{i-1})$. Even with this modification, the sparsity of observations in the training corpus can easily lead to zero counts especially with large vocabularies. To combat this, various smoothing techniques may be applied to the raw counts, some of which have interpretations as introducing a prior and performing MAP inference.

The simplest smoothing method hallucinates some observations by adding 1 to each count in the enumerator and V (the vocabulary size) to the denominator to keep the probabilities normalized:

$$p(x_i \mid x_{<i}) = \frac{C(x_1, \ldots, x_{i-1}, x_i) + 1}{C(x_1, \ldots, x_{i-1}) + V}.$$

In practice, this simplistic method is outperformed by more complex alternatives, among which the Kneser–Ney method [Ney et al., 1994] is perhaps the most well-known, and whose Interpolated Kneser–Ney smoothing variant can be interpreted as performing approximate inference in a hierarchical Bayesian model consisting of Pitman–Yor processes [Teh, 2006].

Two significant drawbacks of N-gram models are poor generalization and very high storage requirements for the counts. There are at most V^N counts to store, and while this upper bound is overly pessimistic due to the effective branching factor being much smaller than V, it is still high enough that storage becomes a serious concern, hence N cannot be very high. For poor generalization the reason is twofold. One is that the very independence assumption that allows us to disregard prefixes longer than N throws away lots of useful information. Another reason is simply sparsity: for N-grams to be considered the same, all tokens in them must match exactly. N-gram models do not leverage similarity of words; *dog returns*, *collie returns*, and *Lassie returns* are very different bigrams, never mind more complicated rephrasings such as *rough collie comes home*.

1.3 RECURRENT NEURAL NETWORKS

Recurrent nets address both of these issues: the length of the context is unbounded in theory, and they not only learn word similarities from data but also smooth over different phrasings and grammatical constructions.

Elman networks [Elman, 1990] are an early example of recurrent networks. Given an input $\mathbf{x}_t$ at time step t, they maintain and update the hidden state $\mathbf{h}_t$ over time, from which the output $\mathbf{y}_t$ is computed:

$$\mathbf{h}_t = \sigma^h\left(\mathbf{W}^{hx}\mathbf{x}_t + \mathbf{W}^{hh}\mathbf{h}_{t-1} + \mathbf{b}^h\right)$$
$$\mathbf{y}_t = \sigma^y\left(\mathbf{W}^{yh}\mathbf{h}_t + \mathbf{b}^y\right).$$

$\mathbf{W}^*$ are weight matrices, $\mathbf{b}^*$ are bias vectors, and σ^* are activation functions such as tanh. [2] In their groundbreaking work, Mikolov et al. [2010] apply Elman networks to language modelling. They assign a fixed-size, learnable vector to each word in the vocabulary and present this vector as the input $\mathbf{x}_t$. The output is a categorical distribution over the vocabulary whose parameters are computed by σ^y, which is the softmax function, and the network is trained with

[2] We denote vectors and matrices with lower- and uppercase bold letters, standard and other sets with blackboard and calligraphic uppercase letters such as $\mathbb{R}$ and $\mathcal{X}$, respectively.

backpropagation on the cross-entropy loss between the predicted categorical distribution and the data distribution for each time step.

Elman networks suffer from exploding and vanishing gradients [Philipp et al., 2017], where the magnitude of the partial derivative of the hidden-to-hidden transition matrix $\mathbf{W}^{hh}$ either tends to infinity or to zero with more time steps taken. Exploding and vanishing gradients make training unstable and slow, respectively. When the spectral radius of $\mathbf{W}^{hh}$ is greater than 1, Elman networks tend to exhibit exploding gradients, while radiuses less than 1 lead to vanishing gradients, which make long-range dependencies very difficult to learn. LSTMs [Hochreiter and Schmidhuber, 1997b] make the vanishing gradients problem less severe by updating the cell state additively, as in a residual network [He et al., 2016]. LSTMs also introduce a forget gate [Hochreiter and Schmidhuber, 1997c], which controls the retention policy of the state. This more elaborate mechanism makes it easier to learn longer-term dependencies.

Many other recurrent networks have been invented. GRUs [Chung et al., 2015] simplify the gating mechanism of LSTMs, Unitary Evolution RNNs [Arjovsky et al., 2016] fix the eigenvalues of the hidden-to-hidden transition matrix at 1 to deal with exploding and vanishing gradients. To improve performance on parallel hardware such as GPUs, quasi-recurrent neural networks [Bradbury et al., 2016] apply convolutional layers in parallel across time steps and alternate them with minimalist recurrent pooling layers, making these models almost entirely feed-forward.

In summary, RNNs promised long-range dependencies, better generalization, and reduced storage requirements. Despite the clear advancement in all three over N-gram models, there remains room for improvement. In particular, dependencies of over several hundred time steps are still very difficult to learn due to vanishing gradients along the long paths the gradient has to travel, large amounts of data are necessary for learning, and recurrent architectures are a bad match for contemporary parallel hardware.

1.4 FEED-FORWARD NEURAL NETWORKS

The gated convolutional network of Dauphin et al. [2017] was the first feed-forward model competitive with RNNs. Since then, Transformers [Vaswani et al., 2017] forwent both recurrence and convolutions in favour of self-attention [Bahdanau et al., 2014] over a fixed-size window of tokens. In its simplest form, self-attention is simply a reweighting of a sequence of input vectors based on their similarity to the most recent one:

$$\mathbf{a}_t = \sum_{i=t-N}^{t} \alpha_i \mathbf{h}_i.$$

In the above, the input $\mathbf{h}$ is a sequence of vectors, and its most recent $N+1$ values are reweighted to produce $\mathbf{a}_t$ according to α, which may computed as

$$\alpha_i = \frac{e^{\mathbf{h}_t^T \mathbf{h}_i}}{\sum_{i=t-N}^{t} e^{\mathbf{h}_t^T \mathbf{h}_i}},$$

where the dot products $\mathbf{h}_t^T \mathbf{h}_i$ measure the similarity between $\mathbf{h}_t$ and $\mathbf{h}_i$.

In a sense, self-attention routes information over data-dependent pathways. While naive self-attention has a cost that is quadratic in length of the input, its computation is a better fit for parallel hardware. Ironically, by giving up the theoretical possibility of unbounded receptive fields, the shortened gradient paths allow attention-based models to learn dependencies over thousands of time steps.

1.5 GENERATING TEXT

Although this thesis is not directly concerned with generating text from language models, we provide a short overview of the topic. Language models assign a probability $p(x)$ to utterances $x \in \mathcal{X}$. These probabilities can be used, for example, to rerank candidate answers provided by another system according to their probabilities under the model. In addition to evaluating probabilities, most language models can generate text, and they do it by sampling from p. For an auto-regressive model, ancestral sampling is both natural and easy: the first token is sampled as $x_1 \sim p(X_1)$, the second as $x_2 \sim p(X_2|x_1)$, the third as $x_3 \sim p(X_3|x_1, x_2)$, and so on. While this method samples from p, our model, that may not always be desirable. To compensate for the shortcomings of the model, to control the diversity of a set of samples, or to adapt generation to a particular task, one may wish to sample from a related distribution.

A basic method to control diversity is tuning the temperature $T > 0$ of the predictive distribution over the next token:

$$p_i^{(T)} = \frac{p_i^{1/T}}{\sum_{i=1}^{V} p_i^{1/T}},$$

where p_i denotes the predicted probability of token i in the vocabulary. Here, $T = 1$ leaves the predictions p_i unchanged. Choosing a large temperature smooths their distribution towards uniform, while going with a low temperature hardens it: more probability mass is assigned to the most likely token type at

the cost of the rest. In practice, the temperature is often lowered below 1 to discourage generating from the tails of the distribution. On the surface, this just trades off diversity for better looking samples, but if model quality is poorer on the tails of the ground truth distribution, and it often is, then it can be considered a form of filtering.

When the vocabulary size is big, the temperature may need to be decreased substantially to reduce the likelihood of generating from the tail, but that may upset the relative probabilities of non-tail events, which are more likely to be well-calibrated. A number of methods address this issue by truncating the predictive distribution over the next token by assigning zero probability to the least likely events and renormalizing.

- *Top-k sampling*: only the most likely k token types are kept [Fan et al., 2018].
- *Nucleus sampling*: only the most likely token types with a given total probability are kept. [Holtzman et al., 2019].
- *η-sampling*: a generalization of nucleus sampling, where the total probability threshold is adjusted based on the entropy of the predictive distribution [Hewitt et al., 2022].

These methods enjoy empirical success, but they can also be seen to compensate for model deficiencies. Braverman et al. [2020] take another approach: they argue that the maximum likelihood objective used to train language models is equivalent to minimizing the KL divergence of the predicted and empirical one-step distributions, but multi-step generations from the model can still be bad because this one-step KL leads to a rather loose bound on the expected risk over generations. They propose a calibration method to bring the entropy of generations closer to the entropy of data, and their solution is superficially reminiscent of temperature tuning but also guaranteed not to make model perplexity worse. This can be achieved by other means; motivated by observations from psycholinguistics, locally typical sampling [Meister et al., 2022] also controls the entropy of generations.

1.6 THE CASE FOR SMALL-SCALE

A recurring theme in the history of sequence models is that the problem of model design is intermingled with optimizability and scalability. Elman Networks are notoriously difficult to optimize, a property that not only gave birth to the idea of the LSTM but also to more recent models such as the Unitary Evolution RNN [Arjovsky et al., 2016] and fixes like gradient clipping [Pascanu et al., 2013]. Still, it is far from clear – if we could optimize these models well – how different their

biases would turn out to be. The non-separability of model and optimization is fairly evident in these cases.

Scalability, on the other hand, is often optimized for indirectly. Given the limited ability of current models to generalize, we often compensate by throwing more data at the problem. To fit a larger dataset, model size must be increased. Thus the best performing models are evaluated based on their scalability. Today, scaling up still yields tangible gains on down-stream tasks, and language modelling data is abundant. However, we believe that simply scaling up will not solve the generalization problem and better models will be needed. Our hope is that by choosing small enough datasets, so that model size is no longer the limiting factor, we get a number of practical advantages:

- Generalization ability will be more clearly reflected in evaluations. [3]
- Turnaround time in experiments will be reduced, and the freed up computational budget can be put to good use by controlling for nuisance factors.
- The transient effects of changing hardware performance characteristics are somewhat lessened [Hooker, 2020].

On the flipside, we give up the opportunity of studying interesting phenomena such as emergence [Wei et al., 2022] and in-context learning [Lampinen et al., 2022] that seem to occur only beyond a certain scale. Despite the obvious risk this represents, we find the tradeoff worthwhile and complementary to mainstream contemporary research. Thus, we develop, analyze and evaluate models primarily on small datasets. Evaluation on larger datasets is included to learn more about the models' scaling behaviour and because of its relevance for applications, but it is to be understood that these evaluations come with larger error bars and provide less guidance for further research on better models.

[3] Utterances are generated by a complex, non-stationary process (the world), but empirical risk minimization requires i.i.d. samples. Disregarding the time component for a minute, we have only non-independent samples even when considering all the text available on the internet. Thus, a natural approach to measure generalization ability is in a domain adaptation setting where we assume i.i.d. samples within a domain and domains being sampled i.i.d. themselves, but that requires multiple delimited datasets. Time aggravates the problems caused by the i.i.d. assumption.

2 TIMID TRANSFORMATION

> I'm disappointed too, but keep in mind transmogrification is a new technology.
>
> *Calvin & Hobbes*

THE DOMINATION of natural language processing by neural models is hampered only by their questionable sample complexity [Hoffmann et al., 2022, Wei et al., 2022, Belinkov and Bisk, 2017, Jia and Liang, 2017, Iyyer et al., 2018, Moosavi and Strube, 2017, Agrawal et al., 2016], their lack of systematicity [Dziri et al., 2023, Linzen et al., 2016, Kuncoro et al., 2018] and their limited ability to chunk input sequences into meaningful units [Cao and Rimell, 2021, Bostrom and Durrett, 2020, Wang et al., 2017].

While direct attacks on the latter are possible, in this work, we take a language-agnostic approach to improving Recurrent Neural Networks (RNN, Rumelhart et al. [1988]), which brought about many advances in tasks such as language modelling, semantic parsing, machine translation, with no shortage of non-NLP applications either [Bakker, 2002, Mayer et al., 2008]. Many neural models are built from RNNs including the sequence-to-sequence family [Sutskever et al., 2014] and its attention-based branch [Bahdanau et al., 2014]. Thus, innovations in RNN architecture tend to have a trickle-down effect from language modelling, where evaluation is often the easiest and data the most readily available, to many other tasks, a trend greatly strengthened by ULMFiT [Howard and Ruder, 2018], ELMo [Peters et al., 2018] and BERT [Devlin et al., 2018], which promote language models from architectural blueprints to pretrained building blocks.

To improve the generalization ability of language models, we propose an extension to the LSTM [Hochreiter and Schmidhuber, 1997c], where the LSTM's input $\mathbf{x}$ is gated conditioned on the output of the previous step $\mathbf{h}_{prev}$. Next, the gated input is used in a similar manner to gate the output of the previous time step. After a couple of rounds of this mutual gating, the last updated $\mathbf{x}$ and $\mathbf{h}_{prev}$ are fed to an LSTM. With the addition of more gating, in one sense, our model joins the long list of recurrent architectures with gating structures of varying complexity that followed the invention of Elman Networks [Elman,

1990]. Examples include the LSTM, the GRU [Chung et al., 2015], and even designs by Neural Architecture Search [Zoph and Le, 2016].

Intuitively, in the lowermost layer, the first gating step scales the input embedding (itself a representation of the *average* context in which the token occurs) depending on the *actual* context, resulting in a contextualized representation of the input. While intuitive, as §2.3 shows, this interpretation cannot account for all the observed phenomena.

In a more encompassing view, our model can be seen as enriching the mostly additive dynamics of recurrent transitions placing it in the company of the Input Switched Affine Network [Foerster et al., 2017] with a separate transition matrix for each possible input, and the Multiplicative RNN [Sutskever et al., 2011], which factorizes the three-way tensor of stacked transition matrices. Also following this line of research are the Multiplicative Integration LSTM [Wu et al., 2016] and – closest to our model in the literature – the Multiplicative LSTM [Krause et al., 2016]. The results in §2.2.3 demonstrate the utility of our approach, which consistently improves on the LSTM and establishes a new state of the art on all but the largest dataset, Enwik8, where we match similarly sized transformer models.

2.1 MODEL

To allow for ease of subsequent extension, we present the standard LSTM update [Sak et al., 2014] with input and state of size m and n respectively as the following function:

$$\begin{aligned} \mathrm{LSTM}\colon \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m &\to \mathbb{R}^n \times \mathbb{R}^n \\ \mathrm{LSTM}\big(\mathbf{c}_{prev}, \mathbf{h}_{prev}, \mathbf{x}\big) &= \big(\mathbf{c}, \mathbf{h}\big). \end{aligned}$$

The updated state $\mathbf{c}$ and the output $\mathbf{h}$ are computed as follows:

$$\begin{aligned} \mathbf{f} &= \sigma\big(\mathbf{W}^{fx}\mathbf{x} + \mathbf{W}^{fh}\mathbf{h}_{prev} + \mathbf{b}^{f}\big) \\ \mathbf{i} &= \sigma\big(\mathbf{W}^{ix}\mathbf{x} + \mathbf{W}^{ih}\mathbf{h}_{prev} + \mathbf{b}^{i}\big) \\ \mathbf{j} &= \tanh\big(\mathbf{W}^{jx}\mathbf{x} + \mathbf{W}^{jh}\mathbf{h}_{prev} + \mathbf{b}^{j}\big) \\ \mathbf{o} &= \sigma\big(\mathbf{W}^{ox}\mathbf{x} + \mathbf{W}^{oh}\mathbf{h}_{prev} + \mathbf{b}^{o}\big) \\ \mathbf{c} &= \mathbf{f} \odot \mathbf{c}_{prev} + \mathbf{i} \odot \mathbf{j} \\ \mathbf{h} &= \mathbf{o} \odot \tanh\big(\mathbf{c}\big), \end{aligned} \tag{2.1}$$

where σ is the logistic sigmoid function, $\odot$ is the elementwise product, $\mathbf{W}^{**}$ and $\mathbf{b}^{*}$ are weight matrices and biases.

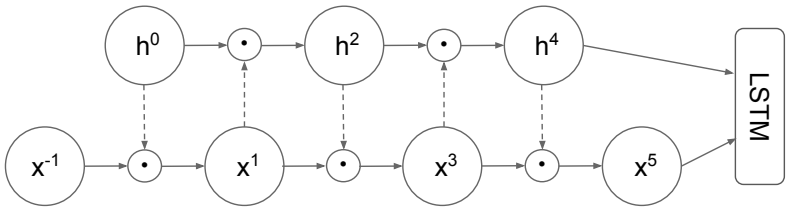


Figure 2.1: Mogrifier with 5 rounds of updates. The previous state $\mathbf{h}^0 = \mathbf{h}_{prev}$ is transformed linearly (dashed arrows), fed through a sigmoid and gates $\mathbf{x}^{-1} = \mathbf{x}$ in an elementwise manner producing $\mathbf{x}^1$. Conversely, the linearly transformed $\mathbf{x}^1$ gates $\mathbf{h}^0$ and produces $\mathbf{h}^2$. After a number of repetitions of this mutual gating cycle, the last values of $\mathbf{h}^*$ and $\mathbf{x}^*$ sequences are fed to an LSTM cell. The *prev* subscript of $\mathbf{h}$ is omitted to reduce clutter.

While the LSTM is typically presented as a solution to the vanishing gradients problem, its gate i can also be interpreted as scaling the rows of weight matrices $\mathbf{W}^{j*}$ (ignoring the non-linearity in j). In this sense, the LSTM nudges Elman Networks towards context-dependent transitions and the extreme case of Input Switched Affine Networks. If we took another, larger step towards that extreme, we could end up with Hypernetworks [Ha et al., 2017]. Here, instead, we take a more cautious step and equip the LSTM with gates that scale the *columns* of all its weight matrices $\mathbf{W}^{**}$ in a context-dependent manner. The scaling of the matrices $\mathbf{W}^{*x}$ (those that transform the cell input) makes the input embeddings dependent on the cell state, while the scaling of $\mathbf{W}^{*h}$ does the reverse.

As Figure 2.1 illustrates, the Mogrifier[1] LSTM is an LSTM where two inputs $\mathbf{h}_{prev}$ and $\mathbf{x}$ modulate one another in an alternating fashion before the usual LSTM computation takes place. That is,

$$\begin{aligned} \text{MogrifierLSTM}\colon \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m &\to \mathbb{R}^n \times \mathbb{R}^n \\ \text{LSTM}\big(\mathbf{c}_{prev}, \text{mogrify}(\mathbf{h}_{prev}, \mathbf{x})\big) &= \big(\mathbf{c}, \mathbf{h}\big), \end{aligned}$$

where the mogrification operation is the following $\mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^m$ function:

$$\begin{aligned} \text{mogrify}(\mathbf{h}, \mathbf{x}) &= \mathbf{h}^{2\lfloor r/2 \rfloor}, \mathbf{x}^{2\lfloor (r+1)/2 \rfloor - 1} \\ \mathbf{x}^{-1}, \mathbf{h}^0 &= \mathbf{x}, \mathbf{h} \\ \mathbf{x}^i &= 2\sigma\big(\mathbf{Q}^i \mathbf{h}^{i-1}\big) \odot \mathbf{x}^{i-2} \qquad \text{for odd i} \in [1..r], \\ \mathbf{h}^i &= 2\sigma\big(\mathbf{R}^i \mathbf{x}^{i-1}\big) \odot \mathbf{h}^{i-2} \qquad \text{for even i} \in [1..r]. \end{aligned} \tag{2.2}$$

Here, the number of rounds, $r \in \mathbb{N}$, is a hyperparameter; $r = 0$ recovers the identity function. Multiplication with the constant 2 ensures that random-

[1]It's like a transmogrifier[2] without the magic: it can only shrink or expand objects.

[2]Transmogrify (verb, 1650s): to completely alter the form of something in a surprising or magical manner.

ly initialized $\mathbf{Q}^i, \mathbf{R}^i$ matrices result in transformations close to identity. To reduce the number of additional model parameters, we typically factorize the $\mathbf{Q}^i, \mathbf{R}^i$ matrices as products of low-rank matrices: $\mathbf{Q}^i = \mathbf{Q}^i_{\text{left}}\mathbf{Q}^i_{\text{right}}$ with $\mathbf{Q}^i \in \mathbb{R}^{m\times n}, \mathbf{Q}^i_{\text{left}} \in \mathbb{R}^{m\times k}, \mathbf{Q}^i_{\text{right}} \in \mathbb{R}^{k\times n}$, where $k < min(m, n)$ is the rank.

2.2 EXPERIMENTS

2.2.1 *Datasets*

We compare models on both word and character-level language modelling datasets. The two word-level datasets we picked are the Penn Treebank (PTB) corpus by Marcus et al. [1993] with preprocessing from Mikolov et al. [2010] and Wikitext-2 by Merity et al. [2016], which is about twice the size of PTB with a larger vocabulary and lighter preprocessing. These datasets are definitely on the small side, but – and *because* of this – they are suitable for exploring different model biases. Their main shortcoming is the small vocabulary size, only in the tens of thousands, which makes them inappropriate for exploring the behaviour of the long tail. For that, open vocabulary language modelling and byte pair encoding [Sennrich et al., 2015] would be an obvious choice. Still, our primary goal here is the comparison of the LSTM and Mogrifier architectures, thus we instead opt for character-based language modelling tasks, where vocabulary size is not an issue, the long tail is not truncated, and there are no additional hyperparameters as in byte pair encoding that make fair comparison harder. The first character-based corpus is Enwik8 from the Hutter Prize dataset [Hutter, 2012]. Following common practice, we use the first 90 million characters for training and the remaining 10 million evenly split between validation and test. The character-level task on the Mikolov preprocessed PTB corpus [Merity et al., 2018] is unique in that it has the disadvantages of closed vocabulary without the advantages of word-level modelling, but we include it for comparison to previous work. The final character-level dataset is the Multilingual Wikipedia Corpus (MWC, Kawakami et al. [2017]), from which we focus on the English and Finnish language subdatasets in the single text, large setting.

2.2.2 *Setup*

We tune hyperparameters following the experimental setup of Melis et al. [2018] using a black-box hyperparameter tuner based on batched Gaussian Process Bandits [Golovin et al., 2017]. For the LSTM, the tuned hyperparameters are

Table 2.1: Word-level perplexities of near state-of-the-art models, our LSTM baseline and the Mogrifier on PTB and Wikitext-2. Models with Mixture of Softmaxes [Yang et al., 2017a] are denoted with *MoS*, depth N with *dN*. *MC* stands for Monte Carlo dropout evaluation. Previous state-of-the-art results in italics. Note the comfortable margin of 2.8–4.3 perplexity points the Mogrifier enjoys over the LSTM.

			No Dyneval		Dyneval	
			Val.	Test	Val.	Test
PTB	FRAGE (d3, MoS15) [Gong et al., 2018]	22M	*54.1*	*52.4*	*47.4*	*46.5*
	AWD-LSTM (d3, MoS15) [Yang et al., 2017a]	22M	56.5	54.4	48.3	47.7
	Transformer-XL [Dai et al., 2019]	24M	56.7	54.5		
	LSTM (d2)	24M	55.8	54.6	48.9	48.4
	Mogrifier (d2)	24M	52.1	51.0	45.1	45.0
	LSTM (d2, MC)	24M	55.5	54.1	48.6	48.4
	Mogrifier (d2, MC)	24M	**51.4**	**50.1**	**44.9**	**44.8**
WT2	FRAGE (d3, MoS15) [Gong et al., 2018]	35M	*60.3*	*58.0*	*40.8*	*39.1*
	AWD-LSTM (d3, MoS15) [Yang et al., 2017a]	35M	63.9	61.2	42.4	40.7
	LSTM (d2, MoS2)	35M	62.6	60.1	43.2	41.5
	Mogrifier (d2, MoS2)	35M	58.7	56.6	40.6	39.0
	LSTM (d2, MoS2, MC)	35M	61.9	59.4	43.2	41.4
	Mogrifier (d2, MoS2, MC)	35M	**57.3**	**55.1**	**40.2**	**38.6**

the same: *input_embedding_ratio*, *learning_rate*, *l2_penalty*, *input_dropout*, *inter_layer_dropout*, *state_dropout*, *output_dropout*. For the Mogrifier, the number of rounds r and the rank k of the low-rank approximation is also tuned (allowing for full rank, too). For word-level tasks, BPTT [Werbos, 1990] window size is set to 70 and batch size to 64. For character-level tasks, BPTT window size is set to 150 and batch size to 128 except for Enwik8 where the window size is 500. Input and output embeddings are tied for word-level tasks following Inan et al. [2016] and Press and Wolf [2016]. Optimization is performed with Adam [Kingma and Ba, 2014] with $\beta_1 = 0$, a setting that resembles RMSProp without momentum. Gradients are clipped [Pascanu et al., 2013] to norm 10. We switch to averaging weights similarly to Merity et al. [2017] after a certain number of checkpoints with no improvement in validation cross-entropy or at 80% of the training time at the latest. We found no benefit to using two-step finetuning.

Model evaluation is performed with the standard, deterministic dropout approximation or Monte Carlo averaging [Gal and Ghahramani, 2016] where explicitly noted (MC). In standard dropout evaluation, dropout is turned off while in MC dropout predictions are averaged over randomly sampled dropout masks (200 in our experiments). Optimal softmax temperature is determined on the validation set, and in the MC case, dropout rates are scaled [Melis et al.,

2018]. Finally, we report results with and without dynamic evaluation [Krause et al., 2017]. Hyperparameters for dynamic evaluation are tuned using the same method (see §2.A for details).

2.2.3 *Results*

Table 2.1 lists our results on word-level datasets. On the PTB and Wikitext-2 datasets, the Mogrifier has lower perplexity than the LSTM by 3–4 perplexity points regardless of whether or not dynamic evaluation [Krause et al., 2017] and Monte Carlo averaging are used. On both datasets, the state of the art is held by the AWD LSTM [Merity et al., 2017] extended with Mixture of Softmaxes [Yang et al., 2017a] and FRAGE [Gong et al., 2018]. The Mogrifier improves the state of the art without either of these methods on PTB, and without FRAGE on Wikitext-2.

Table 2.2 lists the character-level modelling results. On all datasets, our baseline LSTM results are much better than those previously reported for LSTMs, highlighting the issue of scalability and experimental controls. In some cases, these unexpectedly large gaps may be down to lack of hyperparameter tuning as in the case of Merity et al. [2017], or in others, to using a BPTT window size (50) that is too small for character-level modelling [Melis et al., 2017] in order to fit the model into memory. The Mogrifier further improves on these baselines by a considerable margin. Even the smallest improvement of 0.012 bpc on the highly idiosyncratic, character-based, Mikolov preprocessed PTB task is equivalent to gaining about 3 perplexity points on word-level PTB. MWC, which was built for open-vocabulary language modelling, is a much better smaller-scale character-level dataset. On the English and the Finnish corpora in MWC, the Mogrifier enjoys a gap of 0.033–0.046 bpc. Finally, on the Enwik8 dataset, the gap is 0.029–0.039 bpc in favour of the Mogrifier.

Of particular note is the comparison to Transformer-XL [Dai et al., 2019], a state-of-the-art model on larger datasets such as Wikitext-103 and Enwik8. On PTB, without dynamic evaluation, the Transformer-XL is on par with our LSTM baseline which puts it about 3.5 perplexity points behind the Mogrifier. On Enwik8, also without dynamic evaluation, the Transformer-XL has a large, 0.09 bpc advantage at similar parameter budgets, but with dynamic evaluation this gap disappears. However, we did not test the Transformer-XL ourselves, so fair comparison is not possible due to differing experimental setups and the rather sparse result matrix for the Transformer-XL.

Table 2.2: Bits per character on character-based datasets of near state-of-the-art models, our LSTM baseline and the Mogrifier. Previous state-of-the-art results in italics. Depth N is denoted with *dN*. MC stands for Monte Carlo dropout evaluation. Once again the Mogrifier strictly dominates the LSTM and sets a new state of the art on all but the Enwik8 dataset where with dynamic evaluation it closes the gap to the Transformer-XL of similar size († Krause et al. [2019], ‡ Ben Krause, personal communications, May 17, 2019). On most datasets, model size was set large enough for underfitting not to be an issue. This was very much not the case with Enwik8, so we grouped models of similar sizes together for ease of comparison. Unfortunately, a couple of dynamic evaluation test runs diverged (NaN) on the test set and some were just too expensive to run (Enwik8, MC).

			No Dyneval		Dyneval	
			Val.	Test	Val.	Test
PTB	Trellis Networks [Bai et al., 2018]	13.8M		*1.159*		
	AWD-LSTM (d3) [Merity et al., 2017]	13.8M		1.175		
	LSTM (d2)	24M	1.163	1.143	1.116	1.103
	Mogrifier (d2)	24M	1.149	1.131	1.098	1.088
	LSTM (d2, MC)	24M	1.159	1.139	1.115	1.101
	Mogrifier (d2, MC)	24M	**1.137**	**1.120**	**1.094**	**1.083**
MWC EN	HCLM+Cache [Kawakami et al., 2017]	8M	*1.591*	*1.538*		
	LSTM (d1) [Kawakami et al., 2017]	8M	1.793	1.736		
	LSTM (d2)	24M	1.353	1.338	1.239	1.225
	Mogrifier (d2)	24M	1.319	1.305	1.202	1.188
	LSTM (d2, MC)	24M	1.346	1.332	1.238	NaN
	Mogrifier (d2, MC)	24M	**1.312**	**1.298**	**1.200**	**1.187**
MWC FI	HCLM+Cache [Kawakami et al., 2017]	8M	*1.754*	*1.711*		
	LSTM (d1) [Kawakami et al., 2017]	8M	1.943	1.913		
	LSTM (d2)	24M	1.382	1.367	1.249	1.237
	Mogrifier (d2)	24M	1.338	1.326	1.202	**1.191**
	LSTM (d2, MC)	24M	1.377	1.361	1.247	1.234
	Mogrifier (d2, MC)	24M	**1.327**	**1.313**	**1.198**	NaN
Enwik8	Transformer-XL (d24) [Dai et al., 2019]	277M		**0.993**		**0.940**†
	Transformer-XL (d18) [Dai et al., 2019]	88M		1.03		
	LSTM (d4)	96M	1.145	1.155	1.041	1.020
	Mogrifier (d4)	96M	1.110	1.122	1.009	0.988
	LSTM (d4, MC)	96M	1.139	1.147		
	Mogrifier (d4, MC)	96M	1.104	1.116		
	Transformer-XL (d12) [Dai et al., 2019]	41M		1.06		1.01‡
	AWD-LSTM (d3) [Merity et al., 2017]	47M		1.232		
	mLSTM (d1) [Krause et al., 2016]	46M		1.24		1.08
	LSTM (d4)	48M	1.182	1.195	1.073	1.051
	Mogrifier (d4)	48M	1.135	1.146	1.035	1.012
	LSTM (d4, MC)	48M	1.176	1.188		
	Mogrifier (d4, MC)	48M	1.130	1.140		

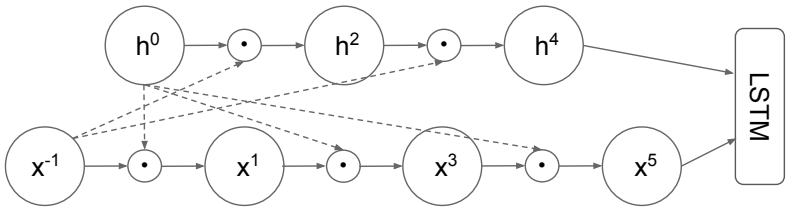


Figure 2.2: No-zigzag Mogrifier for the ablation study, whose gating is always based on the original inputs.

2.3 ANALYSIS

2.3.1 *Ablation Study*

The Mogrifier consistently outperformed the LSTM in our experiments. The optimal settings were similar across all datasets, with $r \in \{5, 6\}$ and $k \in [40..90]$ (see §2.B for a discussion of hyperparameter sensitivity). In this section, we explore the effect of these hyperparameters and show that the proposed model is not unnecessarily complicated. To save computation, we tune all models using a shortened schedule with only 145 epochs instead of 964 and a truncated BPTT window size of 35 on the word-level PTB dataset, and evaluate using the standard, deterministic dropout approximation with a tuned softmax temperature.

Figure 2.3 shows that the number of rounds r greatly influences the results. Second, we found the low-rank factorization of $\mathbf{Q}^i$ and $\mathbf{R}^i$ to help a bit, but the full-rank variant is close behind, which is what we observed on other datasets, as well. Finally, to verify that the alternating gating scheme is not overly complicated, we condition *all* newly introduced gates on the original inputs $\mathbf{x}$ and $\mathbf{h}_{prev}$ (see Figure 2.2). That is, in (2.2) the updates are changed to these no-zigzag variants:

$$\begin{aligned} \mathbf{x}^i &= 2\sigma\big(\mathbf{Q}^i \mathbf{h}_{prev}\big) \odot \mathbf{x}^{i-2} && \text{for odd i} \in [1..r], \\ \mathbf{h}^i_{prev} &= 2\sigma\big(\mathbf{R}^i \mathbf{x}\big) \odot \mathbf{h}^{i-2}_{prev} && \text{for even i} \in [1..r]. \end{aligned}$$

In our experiments, the no-zigzag variant underperformed the baseline Mogrifier by a small but significant margin, and was on par with the $r = 2$ model in Figure 2.3 suggesting that the Mogrifier's iterative refinement scheme does more than simply widen the range of possible gating values of $\mathbf{x}$ and $\mathbf{h}_{prev}$ to $(0, 2^{\lceil r/2 \rceil})$ and $(0, 2^{\lfloor r/2 \rfloor})$, respectively.

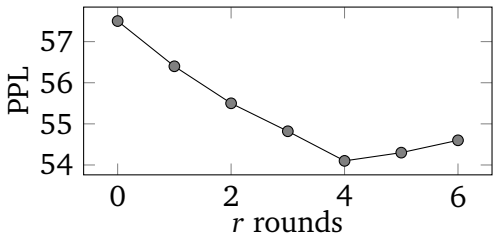


Figure 2.3: Perplexity vs the number of rounds r in the PTB ablation study.

Mogrifier	54.1
Full rank Q^i, P^i	54.6
No zigzag	55.0
LSTM	57.5
mLSTM	57.8

Table 2.3: PTB ablation study validation perplexities with 24M parameters.

2.3.2 *Comparison to the mLSTM*

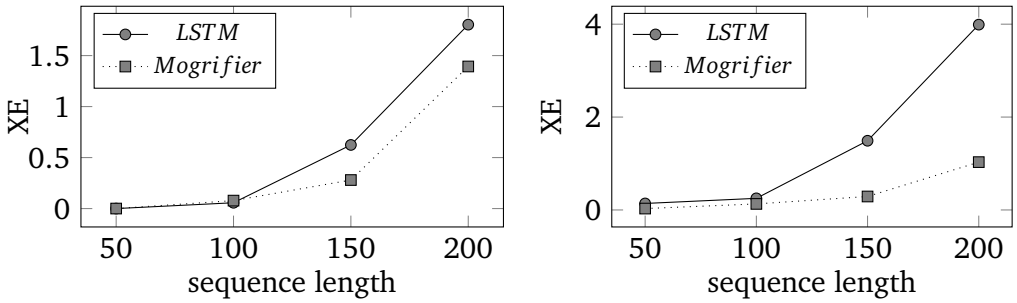
The Multiplicative LSTM [Krause et al., 2016], or mLSTM for short, is closest to our model in the literature. Formulating it as

$$\text{mLSTM}\big(\mathbf{x}, \mathbf{c}_{prev}, \mathbf{h}_{prev}\big) = \text{LSTM}\big(\mathbf{x}, \mathbf{c}_{prev}, \mathbf{h}^m_{prev}\big)$$
$$\mathbf{h}^m_{prev} = \big(\mathbf{W}^{mx}\mathbf{x}\big) \odot \big(\mathbf{W}^{mh}\mathbf{h}_{prev}\big),$$

the differences are readily apparent. First, the mLSTM allows for multiplicative interaction between $\mathbf{x}$ and $\mathbf{h}_{prev}$, but it only overrides $\mathbf{h}_{prev}$, while in the Mogrifier the interaction is two-way, which – as the ablation study showed – is important. Second, the mLSTM can change not only the magnitude but also the sign of values in $\mathbf{h}_{prev}$, something with which we experimented in the Mogrifier, but could not get to work. Furthermore, in the definition of $\mathbf{h}^m_{prev}$, the unsquashed linearities and their elementwise product make the mLSTM more sensitive to initialization and unstable during optimization.

On the Enwik8 dataset, we greatly improved on the published results of the mLSTM [Krause et al., 2016]. In fact, even our LSTM baseline outperformed the mLSTM by 0.03 bpc. We also conducted experiments on PTB based on our reimplementation of the mLSTM following the same methodology as the ablation study and found that the mLSTM did not improve on the LSTM (see Table 2.3).

Krause et al. [2016] posit and verify the recovery hypothesis that says that having just suffered a large loss, the loss on the next time step will be smaller on average for the mLSTM than for the LSTM. This was found not to be the case for the Mogrifier. Neither did we observe a significant change in the gap between the LSTM and the Mogrifier in the tied and untied embeddings settings, which would be expected if recovery was affected by $\mathbf{x}$ and $\mathbf{h}_{prev}$ being in different domains.



(a) 10M parameters with vocabulary size 1k. (b) 24M parameters with vocabulary size 10k.

Figure 2.4: Cross-entropy vs sequence length in the reverse copy task with i.i.d. tokens. Lower is better. The Mogrifier is better than the LSTM even in this synthetic task with no resemblance to natural language.

2.3.3 *The Reverse Copy Task*

Our original motivation for the Mogrifier was to allow the context to amplify salient and attenuate nuisance features in the input embeddings. We conduct a simple experiment to support this point of view. Consider the reverse copy task where the network reads an input sequence of tokens and a marker token after which it has to repeat the input in reverse order. In this simple sequence-to-sequence learning [Sutskever et al., 2014] setup, the reversal is intended to avoid the minimal time lag problem [Hochreiter and Schmidhuber, 1997c], which is not our focus here.

The experimental setup is as follows. For the training set, we generate 500 000 examples by uniformly sampling a given number of tokens from a vocabulary of size 1000. The validation and test sets are constructed similarly, and contain 10 000 examples. The model consists of an independent, unidirectional encoder and a decoder, whose total number of parameters is 10 million. The decoder is initialized from the last state of the encoder. Since overfitting is not an issue here, no dropout is necessary, and we only tune the learning rate, the l2 penalty, and the embedding size for the LSTM. For the Mogrifier, the number of rounds r and the rank k of the low-rank approximation are also tuned.

We compare the case where both the encoder and decoder are LSTMs to where both are Mogrifiers. Figure 2.4a shows that, for sequences of length 50 and 100, both models can solve the task perfectly. At higher lengths though, the Mogrifier has a considerable advantage. Examining the best hyperparameter settings found, the embedding/hidden sizes for the LSTM and Mogrifier are 498/787 vs 41/1054 at 150 steps, and 493/790 vs 181/961 at 200 steps. Clearly, the Mogrifier was able to work with a much smaller embedding size than the LSTM,

which is in line with our expectations for a model with a more flexible interaction between the input and recurrent state. We also conducted experiments with a larger model and vocabulary size, and found the effect even more pronounced (see Figure 2.4b).

2.3.4 *What the Mogrifier is Not*

The results on the reverse copy task support our hypothesis that input embeddings are enriched by the Mogrifier architecture, but that cannot be the full explanation as the results of the ablation study indicate. In the following, we consider a number of hypotheses about where the advantage of the Mogrifier lies and the experiments that provide evidence *against* them.

- *Hypothesis: the benefit is in scaling* $\mathbf{x}$ *and* $\mathbf{h}_{prev}$*.* We verified that data dependency is a crucial feature by adding a learnable scaling factor to the LSTM inputs. We observed no improvement. Also, at extremely low-rank (less than 5) settings where the amount of information in its gating is small, the Mogrifier loses its advantage.
- *Hypothesis: the benefit is in making optimization easier.* We performed experiments with different optimizers (SGD, RMSProp), with intra-layer batch normalization and layer normalization on the LSTM gates. While we cannot rule out an effect on optimization difficulty, in all of these experiments the gap between the LSTM and the Mogrifier was the same.
- *Hypothesis: exact tying of embeddings is too constraining, the benefit is in making this relationship less strict.* Experiments conducted with untied embeddings and character-based models demonstrate improvements of similar magnitude.
- *Hypothesis: the benefit is in the low-rank factorization of* $\mathbf{Q}^i, \mathbf{R}^i$ *implicitly imposing structure on the LSTM weight matrices.* We observed that the full-rank Mogrifier also performed better than the plain LSTM. We conducted additional experiments where the LSTM's gate matrices were factorized and observed no improvement.
- *Hypothesis: the benefit comes from better performance on rare words.* The observed advantage on character-based modelling is harder to explain based on frequency. Also, in the reverse copy experiments, a large number of tokens were sampled uniformly, so there were no rare words to speak of.
- *Hypothesis: the benefit is specific to the English language.* The improvements observed on the Finnish dataset and the reverse copy experiments, which did not feature natural language at all, directly contradict this hypothesis.

- *Hypothesis: the benefit is in handling long-range dependencies better.* Experiments in the episodic setting (i.e. sentence-level language modelling) exhibited the same gap as the non-episodic ones.
- *Hypothesis: the scaling up of inputs saturates the downstream LSTM gates.* The idea here is that saturated gates may make states more stable over time. We observed the opposite: the means of the standard LSTM gates in the Mogrifier were very close between the two models, but their variance was smaller in the Mogrifier.

2.4 CONCLUSIONS

§ Many advances in natural language processing have been based upon more expressive models for how inputs interact with the context in which they occur. Recurrent networks, which have enjoyed a modicum of success, still lack the generalization and systematicity ultimately required for modelling language. In this work, we proposed an extension to the venerable Long Short-Term Memory in the form of mutual gating of the current input and the previous output. This mechanism affords the modelling of a richer space of interactions between inputs and their context. Equivalently, our model can be viewed as making the transition function given by the LSTM context-dependent. Experiments demonstrate markedly improved generalization on language modelling in the range of 3–4 perplexity points on Penn Treebank and Wikitext-2, and 0.01–0.05 bpc on four character-based datasets. With the Mogrifier LSTM, we establish a new state of the art on all datasets with the exception of Enwik8, where we close a large gap between the LSTM and Transformer models.

Our original motivation for this work was that the context-free representation of input tokens may be a bottleneck in language models, and by conditioning the input embedding on the recurrent state some benefit was indeed derived. While it may be part of the explanation, this interpretation clearly does not account for the improvements brought by conditioning the recurrent state on the input and especially the benefits of mogrification on character-level datasets. Positioning our work on the Multiplicative RNN line of research offers a more compelling perspective.

To give more credence to this interpretation, in the analysis we highlighted a number of possible alternative explanations and ruled them all out to varying degrees. In particular, the connection to the mLSTM was found to be weaker than expected as the Mogrifier does not exhibit improved recovery (see §2.3.2), and on PTB the mLSTM works only as well as the LSTM. At the same time, the

evidence against easier optimization is weak, and the Mogrifier establishing some kind of sharing between otherwise independent LSTM weight matrices is a distinct possibility.

Finally, note that as shown by Figure 2.1 and (2.2), the Mogrifier is a series of preprocessing steps composed with the LSTM function, but other architectures, such as Mogrifier GRU or Mogrifier Elman Network are possible. We also leave investigations into other forms of parameterization of context-dependent transitions for future work.

APPENDICES

2.A HYPERPARAMETER TUNING RANGES

In all experiments, we tuned hyperparameters using Google Vizier [Golovin et al., 2017]. The tuning ranges are listed in Table 2.4. Obviously, *mogrifier_rounds* and *mogrifier_rank* are tuned only for the Mogrifier. If $input\_embedding\_ratio \geqslant 1$, then the input/output embedding sizes and the hidden sizes are set to equal and the linear projection from the cell output into the output embeddings space is omitted. Similarly, $mogrifier\_rank \leqslant 0$ is taken to mean full rank $\mathbf{Q}^*$, $\mathbf{R}^*$ without factorization. Since Enwik8 is a much larger dataset, we do not tune *input_embedding_ratio* and specify tighter tuning ranges for dropout based on preliminary experiments (see Table 2.5).

Dynamic evaluation hyperparameters were tuned according to Table 2.6. The highest possible value for *max_time_steps*, the BPTT window size, was 20 for word, and 50 for character-level tasks. The batch size for estimating the mean squared gradients over the training data was set to 1024, gradient clipping was turned off, and the l2 penalty was set to zero.

2.B HYPERPARAMETER SENSITIVITY

The parallel coordinate plots in Figure 2.5 and 2.6 give a rough idea about hyperparameter sensitivity. The red lines correspond to hyperparameter combinations closest to the best solution found. To find the closest combinations, we restricted the range for each hyperparameter separately to about 15% of its entire tuning range. For both the LSTM and the Mogrifier, the results are at most 1.2 perplexity points off the best result, so our results are somewhat insensitive to jitter in the hyperparameters. Still, in this setup, grid search would require orders of magnitude more trials to find comparable solutions.

On the other hand, the tuner does take advantage of the stochasticity of training, and repeated runs with the same parameters may give slightly worse results. To gauge the extent of this effect, on PTB we estimated the standard deviation in reruns of the LSTM with the best hyperparameters to be about 0.2 perplexity points, but the mean was about 0.7 perplexity points off the result produced with the weights saved in best tuning run.

Table 2.4: Hyperparameter tuning ranges for all tasks except Enwik8.

	Low	High	Spacing
learning_rate	0.001	0.004	log
input_embedding_ratio	0.0	2.0	
l2_penalty	5e-6	1e-3	log
input_dropout	0.0	0.9	
inter_layer_dropout	0.0	0.95	
state_dropout	0.0	0.8	
output_dropout	0.0	0.95	
mogrifier_rounds (r)	0	6	
mogrifier_rank (k)	-20	100	

Table 2.5: Hyperparameter tuning ranges for Enwik8.

	Low	High	Spacing
learning_rate	0.001	0.004	log
l2_penalty	5e-6	1e-3	log
input_dropout	0.0	0.2	
inter_layer_dropout	0.0	0.2	
state_dropout	0.0	0.25	
output_dropout	0.0	0.25	
mogrifier_rounds (r)	0	6	
mogrifier_rank (k)	-20	100	

Table 2.6: Hyperparameter tuning ranges for dynamic evaluation.

	Low	High	Spacing
max_time_steps	1	20/50	
dyneval_learning_rate	1e-6	1e-3	log
dyneval_decay_rate	1e-6	1e-2	log
dyneval_epsilon	1e-8	1e-2	log

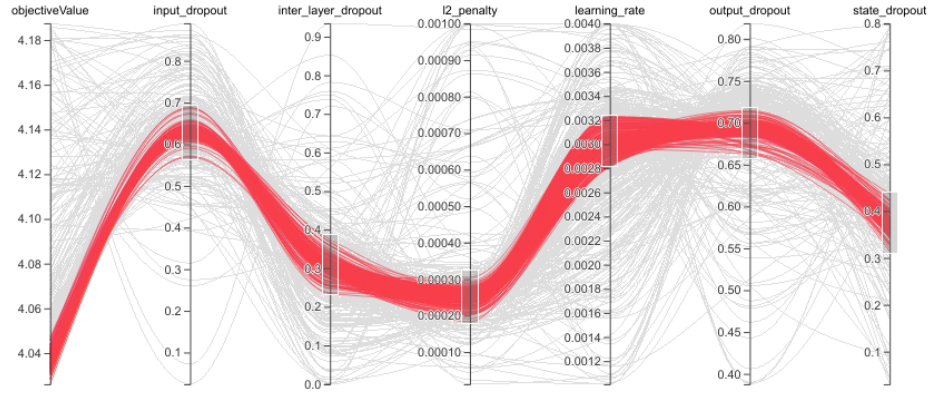


Figure 2.5: Average per-word validation cross-entropies for hyperparameter combinations in the neighbourhood of the best solution for a 2-layer LSTM with 24M weights on the Penn Treebank dataset.

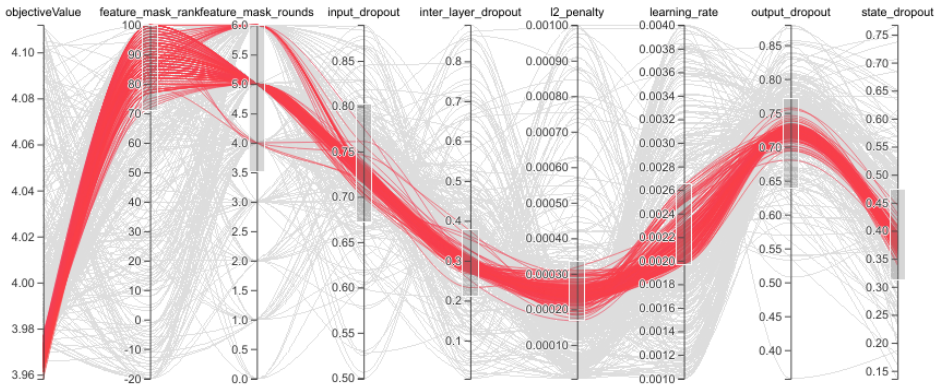


Figure 2.6: Average per-word validation cross-entropies for hyperparameter combinations in the neighbourhood of the best solution for a 2-layer Mogrifier LSTM with 24M weights on the Penn Treebank dataset. *feature_mask_rank* and *feature_mask_rounds* are aliases for *mogrifier_rank* and *mogrifier_rounds*

.

3 OPTIMIZATION OOMPH

> It's not much of a tail, but
> I'm sort of attached to it.
>
> *A.A. Milne*

Mogrification proved surprisingly powerful, but it comes with a performance penalty. With thorough hyperparemeter tuning (a prerequisite of reliable model evaluation), this penalty and the already high base cost of the LSTM must be paid hundreds of times. Scaling to larger datasets makes this process even more resource intensive.

More generally, when training a single model is expensive, there are fewer resources left for exploring the design and hyperparameter space, which compromises our ability to develop with speed and evaluate with accuracy. The exploration is difficult if the space is large and wrinkled, thus the number of hyperparameters and the sensitivity of the results to them is of high practical importance (see §2.A and §2.B). One particularly important hyperparameter is the learning rate. Since it often varies during training, the learning rate is not a single but a bunch of hyperparameters that define its schedule. Tuning these hyperparameters well is time-consuming yet essential. An alternative to scheduling the learning rate is averaging iterates of a stochastic optimizer. This technique was already used to train the Mogrifier, and it worked well but not without drawbacks that would get into the way of scaling. In the following, we develop an algorithm that approximates the optimal iterate average and which can be used with any stochastic optimizer. This will allow us to scale our models and methodology (including the costly hyperparameter tuning) to larger datasets in §4.

3.1 BACKGROUND

For the series of iterates produced by Stochastic Gradient Descent (SGD) [Robbins and Monro, 1985] to converge to a local minimum point of the training loss, the learning rate must be annealed to zero. Polyak averaging [Polyak and Juditsky, 1992, Ruppert, 1988] improves on SGD and achieves a statistically optimal convergence rate by averaging all iterates to produce the final solution.

Tail or suffix averaging [Jain et al., 2018, Rakhlin et al., 2011] takes this further and improves the non-asymptotic behaviour by dropping a number of leading iterates from the average, speeding up the decay of the effect of the initial state while allowing the learning rate to stay constant. Both of these properties are advantageous in practice, where a finite number of optimization steps are taken, and because large learning rates may bias optimization towards flatter and wider minima, which improves generalization [Hochreiter and Schmidhuber, 1997a, Keskar et al., 2016]. Focussing on large learning rates, flat minima, and generalization, Izmailov et al. [2018] propose Stochastic Weight Averaging (SWA), which takes the same form as Tail Averaging but is motivated from an ensembling point of view.

Tail Averaging starts after a given number of optimization steps. Setting this hyperparameter to minimize the training loss already poses some difficulties, which only become more pronounced and numerous in the context of generalization, our primary focus in this work.

- Triggering averaging too early is inefficient as the average must grow long to forget early weights.
- Triggering averaging too late is inefficient as it does not use valuable information.
- Tuning dependent hyperparameters becomes harder.
- Early stopping is unreliable due to learning curves having a sudden drop at the onset of averaging.

Motivated by these problems, we propose the Two-Tailed Averaging algorithm with the following features:

- *Anytime*: An estimate of the optimal tail is available at all optimization steps.
- *Adaptive*: It has no hyperparameters. The number of weights averaged (the *length* of the tail) is determined adaptively based on the evolution of generalization performance.
- *Optimal once in a while*: The tail length achieves near optimality regularly.

The algorithm is very easy to implement. Its principal cost is the storage for a second running average, and it also performs more evaluations of generalization performance (e.g. the validation loss). The main idea, sketched in Figure 3.1, is to maintain two running averages of optimization iterates: a *short* and a *long* one, with the long average being our estimate of the optimal weights.

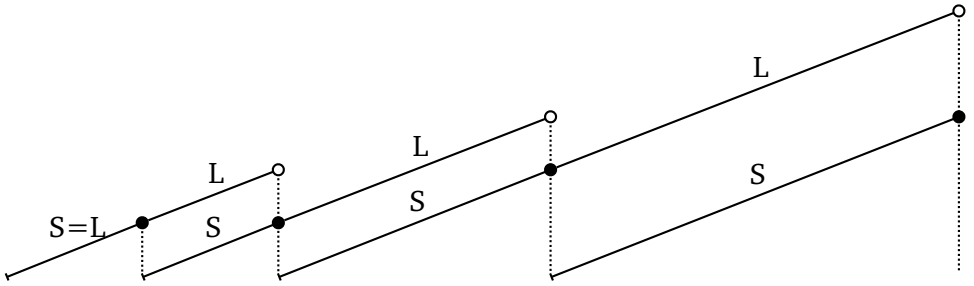


Figure 3.1: Example evolution of the two running averages of weights over optimization steps. Line segments indicate running averages whose length (the number of optimization iterates averaged) is represented on the y axis. The two averages start out the same, but after the first evaluation, there is always one short (S) and one long (L) average, where the long one has more iterates averaged and a better loss. When the loss with the short one is not worse than with the long average, the long one is reset, and the short average becomes the long one. These switch points are marked by dotted lines. In any interval labelled with L, there is at least one point where the length of the long average is near optimal.

3.2 RELATED WORKS

3.2.1 *Averaging in Pure Optimization*

Polyak averaging as originally proposed [Ruppert, 1988, Polyak and Juditsky, 1992] computes the equally weighted average

$$\bar{\theta}_t = \frac{1}{t+1} \sum_{i=0}^{t} \theta_i$$

of all iterates θ_i from the optimizer up to the current time step t. The convergence rate of $\bar{\theta}_t$ was analyzed in the convex case with an appropriately decaying learning rate. Beyond this strictest interpretation, Polyak (or Polyak–Ruppert) averaging may refer to using $\bar{\theta}_t$ without the convexity assumption, without a decaying learning rate, or with another optimizer such as Adam [Kingma and Ba, 2014].

In practice, where finite budget considerations override the asymptotic optimality guarantees offered by theory, Polyak averaging may refer to an exponential moving average (EMA) of the form

$$\begin{aligned} \bar{\theta}_0 &= \theta_0, \\ \bar{\theta}_t &= (1-\beta_t)\theta_t + \beta_t \bar{\theta}_{t-1} \quad (t \geqslant 1), \end{aligned} \tag{3.1}$$

where $\beta_t < 1$ may be a constant near 1 or it may be scheduled as in Martens

[2020]. The idea here is to improve the rate of decay of the effect of the initial error by downweighting early iterates.

Tail Averaging (TA) [Jain et al., 2018], also known as Suffix Averaging [Rakhlin et al., 2011], considers a finite optimization budget of n steps with a constant learning rate. At the cost of introducing a hyperparameter s to control the start of averaging, it improves the rate of decay of the effect of the initial error while obtaining near-minimax rates on the variance. Tail Averaging is defined as

$$\begin{aligned} \bar{\theta}_t &= \theta_t && (t < s), \\ \bar{\theta}_t &= \frac{1}{t-s+1} \sum_{i=s}^{t} \theta_i && (t \geqslant s). \end{aligned} \tag{3.2}$$

Alternatively, the number of iterates to average may change in proportion to the current time step:

$$\bar{\theta}_t = \frac{1}{\lceil ct \rceil} \sum_{i=t+1-\lceil ct \rceil}^{t} \theta_i \qquad (t \geqslant s),$$

where $c \in (0, 1)$ is a hyperparameter. Roux [2019] discusses how to approximate averages of this form without excessive storage needs but do not consider how to automatically adjust the length. [1]

All in all, we have discussed a few representative averaging methods intended for pure optimization but often repurposed for improving generalization by tuning their hyperparameters. Although there are interesting developments in this area [Shamir and Zhang, 2013, Lacoste-Julien et al., 2012], we now move on to the main focus of this work, averaging for improving generalization.

3.2.2 *Averaging for Generalization*

In work parallel to Tail Averaging, Izmailov et al. [2018] propose Stochastic Weight Averaging (SWA), an additional stage of optimization with a constant or cyclical learning rate, which computes an equally weighted average of iterates. SWA can be motivated heuristically in the following way: with the high learning rate, it seeks out wider and flatter basins in the training loss surface to improve generalization, but the high learning rate also prevents it from reaching the bottom of the basin, so the weights bounce around it, thus taking their average

[1] Moreover, they frame the problem as finding a weighting of iterates to minimize the variance of their average while favouring more recent iterates as much as possible to minimize staleness, but the variance is estimated by implicitly assuming that iterates are i.i.d., precluding any notion of staleness.

should land closer to the minimum point. The SWA algorithm is almost identical to Tail Averaging (except for a possibly cyclical learning rate and a periodic subsampling of iterates), but it is motivated from the angle of ensembling and generalization not of optimization.

If our goal is to improve generalization, the decision of when to start averaging the weights should depend on generalization performance. Indeed, Merity et al. [2017] propose an algorithm much like SWA, where averaging is triggered when the validation loss does not improve for a fixed number of optimization steps, which trades one hyperparameter for another and is sensitive to noise in the evaluation of the generalization loss. In other related work, Guo et al. [2022] investigate the repeated application of SWA. Their method is not informed by the validation loss and requires the schedule of multiple SWA stages to be specified. Finally, taking the exponential moving average of iterates is also sensitive to its hyperparameter, the decay rate.

In summary, existing averaging methods for generalization that behave well in practice all have one or more hyperparameters to govern the weighting of early iterates. Tuning these hyperparameters can be costly, particularly in the presence of other hyperparameters and when training runs take a long time. Furthermore, even with their hyperparameters, these methods are not flexible enough to estimate the optimal average at multiple optimization steps in general. We address these issues in the present work. The rest of this chapter is structured as follows. In §3.3, we formally define the problem to solve. In §3.4, we provide a description of the algorithm, whose properties are analyzed in §3.5. We verify our analysis experimentally in §3.6 and discuss the validity of our assumptions in §3.5.1.

3.3 PROBLEM STATEMENT

Let Θ be the parameter (or weight) space, $\theta_t \in \Theta$ ($t \in \mathbb{N}_0$) a sequence of iterates produced by stochastic optimization with θ_0 being the initial value, and $f\colon \Theta \to \mathbb{R}$ the generalization loss function. We may choose the generalization loss to simply be the validation loss, or it may measure performance on a down-stream task.

We assume the generalization loss is evaluated periodically, every $E \in \mathbb{N}$ optimization steps, and it is at these points $n \in [0, E, 2E, 3E, \dots]$ where we would like to know how many of the most recent iterates to average to minimize it. Here and in the following, t and n (with or without subscripts) are assumed to be from $\mathbb{N}_0$ and $[0, E, 2E, 3E, \dots]$, respectively, and *loss* always refers to f.

Denoting the average of most recent δ iterates up to time step t with $\text{avg}(t, \delta) = \frac{1}{\delta} \sum_{i=t+1-\delta}^{t} \theta_i$, we define the optimal averaging length as

$$\Delta(t) = \arg\min_{\delta \in [1..t]} f(\text{avg}(t, \delta)).$$

Our task is to approximate $\Delta(n)$ and $\text{avg}(n, \Delta(n))$ at all evaluation steps n during optimization.

The trivial algorithm to find $\Delta(n)$, which saves all θ_i and performs a search over $\delta \in [1..n]$ to minimize $f(\text{avg}(n, \delta))$, has prohibitive storage and evaluation cost, proportional to n. Even assuming that f improves monotonically in δ up to its optimum, the cost is still proportional to $\Delta(n)$. Our proposed algorithm approximates $\Delta(n)$ and $\text{avg}(n, \Delta(n))$ with a constant cost.

3.4 THE ALGORITHM

Algorithm 1 specifies the core of Two-Tailed Averaging (2TA) in pseudocode, which works as follows. The training loop iterates over weights θ_t produced by a stochastic optimizer, incorporating them into the short and long running averages θ^S, θ^L with lengths S and L. Then, every E steps, the loss is evaluated with the short average θ^S and with the long average θ^L, giving F^S and F^L. If F^S is at least as good as F^L, then we switch: the long average is reset, and since that makes it the shorter of the two averages, we must switch their labels. In other words, on a switch, the long average continues from the current short average and the short average is restarted (see Figure 3.1). For time step t, the estimate of the optimal averaging length is L, and θ^L is the corresponding average.

In Algorithm 2, we present two heuristic extensions to the core algorithm. First, the long and short averages are reset if they have not improved for a few evaluations. This reset heuristic is intended to handle cases where the averages become too long, perhaps due to optimization escaping from one basin of attraction to a better one or due to the loss surface changing in a non-stationary environment. Second, we defer to the non-averaged weights very early in training, where $f(\theta_t)$ is still improving rapidly enough that averaging the minimum E iterates is worse than not averaging at all.

This algorithm is online as it only accesses the current weights. We analyze its other properties in the next section.

Algorithm 1: The core Two-Tailed Averaging algorithm, without extensions. It has 2 running averages, a short one θ^S and a long one θ^L with $S \leqslant L$ number of optimization iterates averaged. If the loss with θ^S becomes lower or equal to the loss with θ^L, then we empty the long average, which becomes the short one.

Require: generalization loss function f, opt. iterates θ_t, evaluation period E

$S, \theta^S, L, \theta^L \leftarrow 0, \mathbf{0}, 0, \mathbf{0}$ ▷ The first evaluation will cause a switch.

procedure $add\_weights(\theta)$
    $S, \theta^S \leftarrow S + 1, \theta^S + (\theta - \theta^S)/(S + 1)$ ▷ Add θ to the short average
    $L, \theta^L \leftarrow L + 1, \theta^L + (\theta - \theta^L)/(L + 1)$ ▷ Add θ to the long average

procedure $switch$ ▷ Reset the long average
    $S, L, \theta^L \leftarrow 0, S, \theta^S$ ▷ Must switch short and long to maintain $S \leqslant L$

function $evaluate\_and\_adapt(\theta, f)$
    $F^S, F^L \leftarrow f(\theta^S), f(\theta^L)$
    if $F^S \leqslant F^L$ **then** ▷ Is the short average better?
        $F^L \leftarrow F^S$
        $switch()$
    return F^L, θ^L, L

for $t \leftarrow 1, 2, \ldots$ **do** ▷ Training loop
    $add\_weights(\theta_t)$ ▷ θ_t comes from the ongoing optimization
    if $t \bmod E = 0$ **then** ▷ Evaluate f every E iterates
        $f_{\bar{\theta}}, \bar{\theta}, len \leftarrow evaluate\_and\_adapt(\theta_t, f)$
        $report(t, f_{\bar{\theta}}, \bar{\theta}, len)$

3.5 ANALYSIS OF THE ALGORITHM

Our analysis hinges on simplifying assumptions, which follow from, for example, a monotonically decreasing loss and averaging producing diminishing returns as the length increases. They represent idealized circumstances; we discuss their validity and failures in §3.5.1.

Assumption 1. *For all n, as a function of $\delta \in [0, E, 2E, \ldots, \Delta_E(n)]$, $f(\text{avg}(n, \delta))$ is monotonically decreasing, where $\Delta_E(n) = \lfloor \Delta(n)/E \rfloor E$. That is, for any given evaluation step n, averaging more iterates from the past monotonically improves f until about the optimum length.*

Assumption 2. *For all n and n_+, such that $n_+ \geqslant \Delta^E(n)$, $f(\text{avg}(n, \Delta^E(n))) \leqslant$*

Algorithm 2: Extensions to Two-Tailed Averaging. This is the version recommended for use in practice. The parts unchanged from Algorithm 1 are greyed out. There are two extensions. First, each average is reset if it is stagnating (i.e. it has not improved for a few evaluations). This reset heuristic makes the algorithm quicker to adapt when Assumption 4 is violated (see §3.5.1). Second, we defer to non-averaged weights very early in training.

```
1:  function evaluate_and_adapt(θ, f)
2:      F^1, F^S, F^L ← f(θ), f(θ^S), f(θ^L)
3:      if F^S ⩽ F^L or F^L is stagnating then
4:          F^L ← F^S
5:          switch()
6:      else if F^S is stagnating then
7:          S ← 0
8:      if L > 1 and F^1 ⩽ F^L then          ▷ Use the non-averaged weights if better
9:          if L = E then                    ▷ If they have always been better so far,
10:             S, L ← 0, 0                  ▷ ... then reinitialize
11:         return F^1, θ, 1
12:     return F^L, θ^L, L
```

$f(\text{avg}(n, n_+))$*, where* $\Delta^E(n) = \lceil \Delta(n)/E \rceil E$*. That is, averaging slightly more than optimal is better than averaging a lot more.*

Assumption 3. $\forall n \colon \exists n_s \colon \Delta(n + n_s) - \Delta(n) < n_s$*, that is, the optimal average forgets over a sufficiently long interval.*

Assumption 4. $\Delta(n) \leqslant \Delta(n+E)$*, that is, the optimal number of weights to average is monotonically increasing from one evaluation to the next.*

Let $S(t)$, $\theta^S(t)$, $L(t)$, and $\theta^L(t)$ stand for the values of variables S, θ^S, L, and θ^L in Algorithm 1, respectively, after t times through the loop. Similarly, let $S'(t)$, $\theta^{S'}(t)$, $L'(t)$, and $\theta^{L'}(t)$ stand for the values of the same variables at the same iteration but after Line 18 in Algorithm 1 (i.e. before the possible switch of the short and long averages). Furthermore, we introduce the shorthands $f^X(t) = f(\text{avg}(t, X(t)))$ for $X \in \{S, S', L, L', \Delta\}$ with $f^X(t) = +\infty$ if $X(t) = 0$.

Definition 1 (Switch point)**.** *We say that n is a switch point if at $t = n$ the* switch *procedure is called at Line 14 in Algorithm 1. We denote the most recent switch point before iteration t with* $\text{SP}(t)$*, where* $\text{SP}(t) < t$*. If there is no such switch point, then* $\text{SP}(t) = -1$*.*

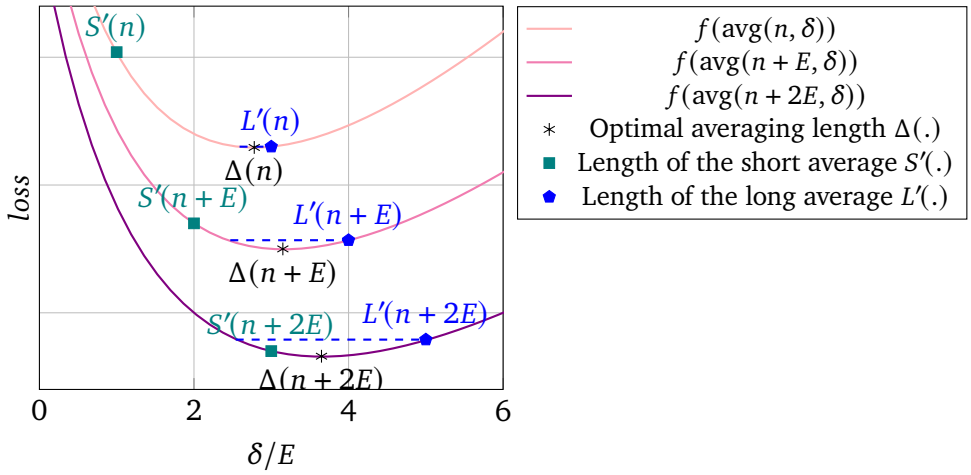


Figure 3.2: Idealized illustration of switching. The three curves show the loss as a function of averaging length at three subsequent evaluations (at optimization steps n, $n+E$, and $n+2E$). The raw loss keeps improving, hence later evaluations have lower loss curves. The optimal averaging length increases, $\Delta(n) \leqslant \Delta(n+E) \leqslant \Delta(n+2E)$, as per Assumption 4. At $t = n+2E$, where the loss of the short average dips below the loss of the long average, the long average is reset, and the short average becomes the long average, so we have $S(n+2E) = 0$ and $L(n+2E) = S'(n+2E)$.

Assumption 4 states that the optimal averaging length monotonically increases, so to simplify the analysis, without loss of generality, we assume throughout that the raw loss F^1 has already been eclipsed by F^L at the first evaluation. We also assume that the reset heuristic cannot trigger. In effect, we ignore the extensions in Algorithm 2 and analyze the core logic in Algorithm 1.

Proposition 1 (Basic properties). $\forall t \geqslant E\colon$ *and* $\forall n \geqslant E\colon$

(i) $E \mid S(n)$, $E \mid L(n)$

(ii) $S(n) \leqslant L(n) \leqslant n$

(iii) $f^S(n) > f^L(n)$

(iv) $L(t) = S(t) + S'(\text{SP}(t))$ *if* $\text{SP}(t) \neq -1$ *else* $L(t) = S(t)$

Item (i) states that the averaging lengths are multiples of the evaluation period (because $n \in [0, E, 2E, 3E, \ldots]$); (ii) follows from that the lengths increase by 1 at every iteration except at switches, where S is reset to 0; (iii) is because we switch if it is not true; and (iv) expresses that all long averages except the first are continuations of the previous short average.

Proposition 2 (Bounds for the averaging lengths). *The lengths of the short and long averages are bounded as* $S(n) < \Delta(n)$ *and* $L(n) < 2\Delta(n) + E$.

Proof. We prove $S(n) < \Delta(n)$ by contradiction. Suppose $\Delta(n_0) \leqslant S(n_0)$ for some n_0. As $\Delta(n)$ is monotonically increasing and $S(n)$ increases by E, there exists $n \leqslant n_0$ such that $\Delta^E(n) = S(n)$. Since $S(n) \leqslant L(n)$ (by (ii) of Proposition 1), from Assumption 2 and $\Delta^E(n) = S(n) \leqslant L(n)$, we have that $f^S(n) \leqslant f^L(n)$, which contradicts (iii) of Proposition 1.

Next, we prove $L(n) < 2\Delta(n) + E$. From (iv) of Proposition 1, we have that at the beginning, when there has not yet been a switch, $L(t) = S(t)$, else $L(t) = S(t) + S'(\mathrm{SP}(t))$ for all t. In the first case, $L(n) = S(n) < \Delta(n) < 2\Delta(n) + E$, and we are done.

In the second, usual case, $L(n) = S(n) + S'(\mathrm{SP}(n))$. That is, the length of the current long average is the sum of the lengths of the current and the previously finished short average. Since $S(n) < \Delta(n)$ and $S'(\mathrm{SP}(n)) = S(\mathrm{SP}(n) - E) + E$, so $L(n) < \Delta(n) + S(\mathrm{SP}(n) - E) + E$, from which $L(n) < \Delta(n) + \Delta(\mathrm{SP}(n) - E) + E$. Finally, from the monotonicity of O in Assumption 4, $\Delta(\mathrm{SP}(n) - E) \leqslant \Delta(n)$, we get $L(n) < 2\Delta(n) + E$. □

Proposition 3 (Infinite number of switch points)**.** *Switch points keep coming, that is,* $\forall n \colon \exists n_s \geqslant n \colon \mathrm{SP}(n_s) \neq -1$.

Proof. Because $S(n) < \Delta(n)$ and $S(n)$ increases by E between switch points, it must catch $\Delta(n)$ at some step n_s because $\Delta(n)$ grows more slowly by Assumption 3. At the point where $S(n_s) = \Delta^E(n_s)$, $f^S(n_s) \leqslant f^L(n_s)$ by Assumption 2, thus there must be a switch. □

Proposition 4 (Once-in-a-while optimality)**.** *Between any two subsequent switch points* n_1 *and* n_2*, the long average is nearly optimal at least once. Formally,* $\exists n \in [n_1, n_2 - E] \colon L(n) = \Delta^E(n) \vee L(n) = \Delta_E(n)$.

Proof. Since $S(n) < \Delta(n)$ and $E \mid S(n)$ for all n, so $S'(n_1) \leqslant \Delta^E(n_1)$. Then it is either that $S'(n_1) = \Delta^E(n_1)$ or $S'(n_1) < \Delta(n_1)$. Since at switch points $L(n) = S'(n)$, in the former case, we conclude the proof with $L(n_1) = \Delta^E(n_1)$. Considering the latter case, $L(n_1) = S'(n_1) < \Delta(n_1)$, so $L(n_1) \leqslant \Delta_E(n_1)$. Also, switches happen when $f^{S'}(n) \leqslant f^{L'}(n)$, but as per Assumption 1 this can happen only if $\Delta(n) < L'(n)$. Thus for n_2 to be a switch point, it must be that $\Delta(n_2) < L'(n_2) = L(n_2 - E) + E$, hence $\Delta_E(n_2) \leqslant L(n_2 - E)$. Combining it with $L(n_1) \leqslant \Delta_E(n_1)$, we get $L(n_1) \leqslant \Delta_E(n_1) \leqslant \Delta_E(n_2) \leqslant L(n_2 - E)$. Therefore, since $L(n)$ and $\Delta_E(n)$ are monotonically increasing over $[n_1, n_2 - E]$, both take values that are multiples

of E, and L overtakes Δ_E while not skipping any such value, there must be a point n where L is equal to Δ_E. □

In short, we have shown that the long average is at most twice as long as optimal, there are infinitely many switch points, and between any two switch points the long average gets as close to the optimal length as possible given the periodic evaluation scheme. Our results are in terms of lengths of averages, and relating the actual loss with the long average f^L to the loss with the optimal length f^Δ would be desirable. Here, we informally point out that, all things being equal, the worse f^L gets relative to f^Δ, the quicker f^S is to catch up, making long periods of highly suboptimal solution less likely. Formalizing this notion requires making further assumptions about the loss-vs-averaging-length function (of the kind plotted in Figure 3.2) and would make analysis considerably more cumbersome.

3.5.1 *When Assumptions Fail*

To augment the theoretical analysis, which is based on idealized assumptions, we make the following observations. The strongest assumption by far is Assumption 1. It says that increasing the averaging length monotonically improves f until the optimum. Since stochastic optimization produces noisy iterates, this does not hold exactly in practice. However, the length of the shortest average is one evaluation period, and its variance is inversely proportional to E. Thus, the likelihood of noise posing a problem can be very small. In terms of the loss, the algorithm is fairly robust to when the assumption holds only approximately because small deviations of $f(\text{avg}(n, \delta))$ from monotonicity can change switch times only when f^S and f^L are close.

Assumption 2 says that averaging slightly more iterates than optimal (i.e. rounded up to the evaluation period) is better than averaging a lot more. This is a weak assumption due to subsequent iterates being highly correlated. If it is violated sporadically, the algorithm can fail to detect when the short average becomes longer than optimal, which delays the switch.

Assumption 3 failing means that the optimal average incorporates all new iterates without ever dropping old ones. In this case, the short average, which is always shorter than optimal, will be a constant number of iterates behind and its loss will converge to the loss of the optimal average. If the long average is shorter than optimal, then the same argument applies to it. If the long average

is longer than optimal, then eventually a switch will happen. In either case, the loss of the long average converges to that of the optimal average.

Regarding Assumption 4, $\Delta(n) \leqslant \Delta(n+E)$ can fail if the improvement of the raw loss accelerates, but that is a rather uncommon and temporary occurrence. It may also fail if the raw loss has started to worsen due to overfitting or optimization has escaped from one basin to the next and the average is slowly climbing the ridge separating them or when the loss landscape changes during learning in a non-stationary environment. With the exception of accelerating improvement, these cases are likely to be caught by the reset heuristic, wherein the long average is reset if its loss does not improve for a few evaluations (see Algorithm 2).

The reset heuristic can trigger when it should not, i.e. when Assumption 4 holds. Such a spurious reset makes the estimate of the long average worse either directly or indirectly by delaying the next switch. Either way, without further violations of this assumption, the algorithm recovers by the next switch. Note that 2TA cannot in general correct overfitting, although the reset heuristic may help in the unlikely case that overfitting proves to be transitory.

All in all, we can expect the algorithm to display some degree of robustness to minor violations of the assumptions. In practice, we recommend choosing a reasonably large E to reduce the noise originating from the stochasticity of optimization.

3.5.2 *A Note on Pure Optimization*

Applying 2TA to pure optimization is unlikely to bring about practical benefits because of the evaluation cost. For example, if f computes the loss over the entire training set and evaluation is performed every epoch, then the cost of optimization is effectively doubled. Furthermore, as we have pointed out above, Assumption 1 does not hold exactly with stochastic optimization: the short average can get lucky and become better than the long one (causing a switch) but then quickly succumb to variance and become worse as new iterates are added to it. Thus, the averaged weights of Algorithms 1 and 2 do not converge in the strict sense, although this point is somewhat moot because in empirical risk minimization – due to the mismatch between the true and the training losses – convergence in the training loss is almost never desirable when optimizing for generalization.

Nevertheless, it is instructive to consider how the algorithm behaves when f is the training loss as the limit of the common case where f is the validation

loss, both the training and the validation sets consist of i.i.d. samples from the same distribution, and their sizes tend to infinity. Focussing on the setting where convergence results are available for Polyak and Tail Averaging, we show in the following that 2TA converges in probability to the optimum in ordinary least squares regression.

Definition 2 (Nth switch point)**.** *For all $N \in \mathbb{N}$, we define three random variables:*

- $Q_N \in [E, 2E, \dots]$ *is the time step corresponding to the Nth switch point.*
- $F_N = f(\theta^{S'}(Q_N))$ *is the loss with the Nth short average just before it becomes the long average.*
- $S_N = S'(Q_N)$ *is the final length of the Nth short average.*

From now on, we use N to index switch points or to refer to short averages that end at that switch point.

Proposition 5. *If the generalization loss function f is convex, then $(F_N)_{N\in\mathbb{N}}$ is monotonically decreasing.*

Proof. The long-averaged weights are a convex combination of the weights of the current and the previous short averages:

$$\theta^{L'}(n) = \alpha\theta^{S'}(n) + (1-\alpha)\theta^{S'}(\mathrm{SP}(n))$$
$$\alpha = \frac{S'(n)}{S'(n) + S'(\mathrm{SP}(n))} \in (0,1).$$

Switching happens when $f(\theta^{S'}(n)) \leqslant f(\theta^{L'}(n))$. Expanding $\theta^{L'}(n)$ and using that f is convex, we get

$$\begin{aligned} f(\theta^{S'}(n)) &\leqslant f(\theta^{L'}(n)) \\ &= f(\alpha\theta^{S'}(n) + (1-\alpha)\theta^{S'}(\mathrm{SP}(n))) \\ &\leqslant \alpha f(\theta^{S'}(n)) + (1-\alpha) f(\theta^{S'}(\mathrm{SP}(n))), \end{aligned}$$

from which, $(1-\alpha)f(\theta^{S'}(n)) \leqslant (1-\alpha)f(\theta^{S'}(\mathrm{SP}(n)))$. Using $\alpha < 1$, we get $f(\theta^{S'}(n)) \leqslant f(\theta^{S'}(\mathrm{SP}(n)))$. This is true at all switch points, hence $F_{N+1} \leqslant F_N$ for all N. □

Note that in non-convex settings, the above monotonicity property could be enforced also by changing the switching condition to $f(\theta^{S'}(n)) \leqslant \min(f(\theta^{L'}(n)), f(\theta^{S'}(\mathrm{SP}(n))))$. However, this would make the algorithm less able to adapt to violations of Assumption 4.

Proposition 6. *Assume that the loss function is bounded from below, the* $(F_N)_{N\in\mathbb{N}}$ *sequence monotonically decreases, and that* $(\theta_t)_{t\in\mathbb{N}_0}$ *approaches a stationary distribution with a density. Then,* $L(t) \xrightarrow{p} \infty$.

Proof. First, we prove $S_N \xrightarrow{p} \infty$ by contradiction. Assume that $S_N \not\xrightarrow{p} \infty$.

(i) *For some* $l_0 \in \mathbb{N}$ *and* $\epsilon_0 > 0$*, there are infinitely many* $N^+ \in \mathbb{N}$ *such that* $P(S_{N^+} = l_0) > \epsilon_0$.

Proof. By the definition of convergence in probability, $S_n \xrightarrow{p} \infty$ is equivalent to $\forall l \in \mathbb{N}\colon \lim_{N\to\infty} P(S_N \leqslant l) = 0$. Suppose that is false, hence $\exists l \in \mathbb{N}, \epsilon > 0\colon \forall N \in \mathbb{N}\colon \exists N^+ > N\colon P(S_{N^+} \leqslant l) > \epsilon$. Then, we have an infinite number of short averages N^+ that are at most length l with at least ϵ probability: $P(S_{N^+} \leqslant l) > \epsilon$. Since l is finite, for all such N^+, there exists $l_{N^+} \in \mathbb{N}$ such that $P(S_{N^+} = l_{N^+}) > \epsilon/l$. Hence, there must be at least one $l_0 \leqslant l$ and $\epsilon_0 > 0$ such that $P(S_{N^+} = l_0) > \epsilon_0$ for infinitely many N^+.

(ii) *The final losses of the short averages converge in probability:* $F_N \xrightarrow{p} F^*$.

Proof. From the assumption that the loss function is bounded from below and that all realizations of the F_N sequence decrease monotonically, all realizations must converge, which implies almost sure convergence hence convergence in probability.

(iii) *Let* $F_N^{l_0+}$ *denote what the loss of* N*th short average at length* l_0 *would be if the algorithm were modified to perform no switching for this short average only. Then,* $P(F^* \leqslant F_N^{l_0+} \leqslant F_{N-1}) \to 0$.

Proof. We have assumed that iterates converge to a stationary distribution with a density. Note that this rules out convergence in the strict sense, which would require a zero-variance stationary distribution. For any random variable X with a density, $lim_{\delta\to 0} P(a \leqslant X \leqslant a + \delta) = 0$ for all $a \in \mathbb{R}$. By (ii), $F_N \xrightarrow{p} F^*$, so for all $\epsilon > 0$, $P(F_N - F^* < \epsilon)$ is close to 1 for all large enough N. With $F_{N-1} - F^*$, the size of the interval into which $F_N^{l_0+}$ must fit, thus bounded uniformly in probability, we get $P(F^* \leqslant F_N^{l_0+} \leqslant F_{N-1}) \to 0$.

We assumed that $S_N \not\xrightarrow{p} \infty$ and in (i) showed that $P(S_{N^+} = l_0) > \epsilon_0$ for some $l_0 \in \mathbb{N}$, $\epsilon_0 > 0$ and infinitely many N^+. Since $S_N = l_0$ implies $F_N = F_N^{l_0+}$ for any N, we have that $P(F_{N^+} = F_{N^+}^{l_0+}) > \epsilon_0$. However, due to the monotonicity assumption, $F^* \leqslant F_N \leqslant F_{N-1}$ for all N, hence $P(F^* \leqslant F_{N^+}^{l_0+} \leqslant F_{N^+-1}) > \epsilon_0$, which contradicts $P(F^* \leqslant F_N^{l_0+} \leqslant F_{N-1}) \to 0$ from (iii).

Finally, every long average except the first is a continuation of the previous short average, that is, for all $t \geqslant E$, $L(t) \geqslant S'(\mathrm{SP}(t))$ and $S'(\mathrm{SP}(t)) = S_N$ for some N. Therefore, $S_N \xrightarrow{p} \infty$ implies that $L(t) \xrightarrow{p} \infty$. □

Proposition 7. *Consider applying SGD with a constant learning rate to an ordinary least squares problem with unique minimum point $\theta^\star$. Then, for a sufficiently low learning rate, $\theta^L(t) \xrightarrow{p} \theta^\star$.*

Proof. The loss function is convex, so F_N is monotonically decreasing by Proposition 5. It is also bounded from below, and it satisfies Assumptions 2.1-2.3 of Yu et al. [2020], hence – by Proposition 2 therein – SGD iterates admit a unique stationary distribution for an appropriately bounded learning rate. Thus, appealing to Proposition 6, we have that $L(t) \xrightarrow{p} \infty$. In the ordinary least squares regression setting, Jain et al. [2018] prove that Tail Averaging converges to the optimum with an appropriately bounded learning rate. Hence, by choosing a learning rate that satisfies both bounds and leveraging the fact that $\theta^L(t)$ is a tail average, we get $\theta^L(t) \xrightarrow{p} \theta^\star$. □

Paralleling strict convergence results for Polyak and Tail Averaging, we have proved that 2TA converges in probability to the optimum in the ordinary least squares regression setting when f is the training loss. We stress again that 2TA is not intended for pure optimization, and this result is to serve as a characterization of behaviour in the infinite data case.

With pure optimization very much a secondary consideration, we provide only weak, anecdotal support for the rate of convergence: the length of the long average tended to increase exponentially in all experiments described in §3.6 and also on simple synthetic data. Intuitively, this is to be expected when f is locally convex because at stationarity, every time a short average finishes at length l, it halves the probability mass available for subsequent short averages to finish at that length: $P(F^l_{N+1} \leqslant F_N \mid S_{N+1} \geqslant l,\ S_N = l) \to 0.5P(F^l_N \leqslant F_{N-1} \mid S_N \geqslant l)$. This halving effect is strongest at the same length, but in diminished form, it extends to longer averages due to the similarity of their distributions.

3.6 EXPERIMENTS

Tail Averaging or Stochastic Weight Averaging have been shown previously to be beneficial not only in theory and on simulated data [Jain et al., 2018] but also in language modelling [Merity et al., 2017, Melis et al., 2020] and image classification [Izmailov et al., 2018] experiments. Hence, in this work we restrict our attention to experiments in a single domain to corroborate the analysis in §3.5. Our goals are to *i)* verify that 2TA is on par with well-tuned TA and EMA, *ii)* explore the effect of basing the switching logic on the training loss instead

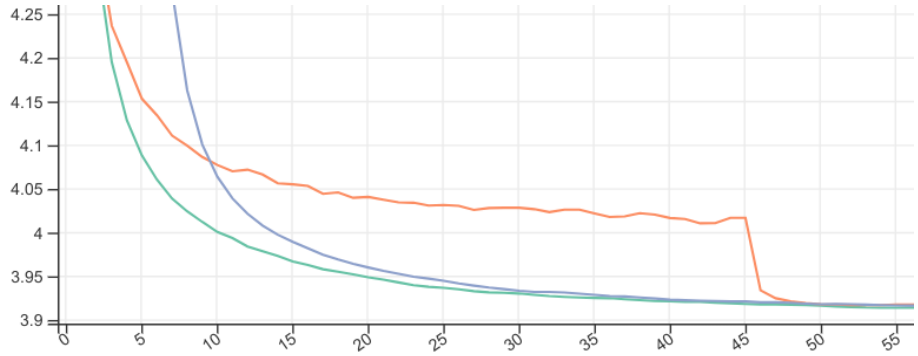


Figure 3.3: Validation loss with Two-Tailed Averaging (2TA, green), Tail Averaging (TA, orange), and an exponential moving average (EMA, blue) of weights on language modelling on Penn Treebank. For both TA and EMA, their hyperparameters (the start time and the decay rate) were tuned to minimize the final loss, so it is not a surprise that all three have similar optima. 2TA has no hyperparameters and produces much better early solutions. These two factors make tuning easier and early stopping much more reliable. Additionally, the noise in the raw loss is effectively smoothed out. Note that while the 2TA loss decreases monotonically, gentler and steeper slopes are manifest before and after switch points, respectively.

of the validation loss, *iii)* demonstrate robustness to the choice of evaluation period, *iv)* and check how well the assumptions made in §3.5 hold in practice.

In particular, we trained a recurrent language model with several hyperparameters on Penn Treebank [Mikolov et al., 2010] using the Rectified Adam optimizer [Liu et al., 2019], evaluating every 1000 optimization steps. The hyperparameters were tuned separately for 2TA, TA eq. (3.2), and EMA eq. (3.1). Figure 3.3 shows that the final validation losses with all methods are very close, but early losses with 2TA are much better. This is expected because TA and EMA are not flexible enough to produce optimal averaging lengths at multiple points along the learning curve despite having an extra hyperparameter. Conversely, 2TA has at least one nearly optimal solution between any two subsequent peaks (i.e. switch points) in Figure 3.4 despite having no hyperparameters.

We also tried the version of the algorithm where the switching logic was based on comparison of the training losses of the short and long averages instead of the validation losses, but the true validation losses were reported. On this particular language modelling task, the best validation loss with the modified algorithm worsened moderately (3.93 vs 3.92) and was well below the raw validation loss (4.02). Results on the test set exhibited the same gap. Since the modified 2TA was minimizing the training loss, the smoothness of the reported validation losses observed in Figure 3.3 were lost in the process. Similar results were obtained by scheduling a learning rate drop without averaging.

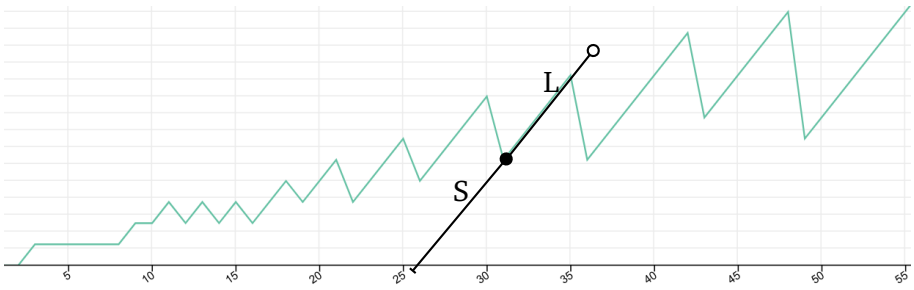


Figure 3.4: Length of the long average (L) vs number of evaluations of 2TA in Figure 3.3. Note how the heights of both peaks and valleys increase almost monotonically. Also, the correspondence between this figure and the schematic in Figure 3.1 is illustrated on one of the S and L intervals.

To explore the effect of the evaluation period E, we tuned models with four times larger and four times smaller E than in our previously discussed experiments. As expected, the best final results were very close to each other, with shorter periods having an advantage early in training as the raw loss F^1 was more quickly eclipsed by F^L.

In addition, we found that the assumptions made in §3.5 held rather well in practice: the resulting loss F^L in Figure 3.3 and the averaging lengths tended to change monotonically (see heights of peaks and valleys in Figure 3.4), making our length-based theoretical results more closely linked to the actual loss. When that was not the case, we found that the raw loss F^1 had started to worsen due to overfitting or, much more rarely, optimization had entered a new basin, violating Assumption 4. Figure 3.5 and Figure 3.6 demonstrate the reset heuristic being triggered in these cases. Finally, TA and 2TA having almost identical final validation losses weakly supports our assumptions, although a conclusive demonstration would need to plot the results obtained with TA tuned separately for each evaluation.

3.7 CONCLUSIONS

§ Tail averaging improves on Polyak averaging's non-asymptotic behaviour by excluding a number of leading iterates of stochastic optimization from its calculations. In practice, with a finite number of optimization steps and a learning rate that cannot be annealed to zero, Tail Averaging can get much closer to a local minimum point of the training loss than either the individual iterates or the Polyak average. However, the number of leading iterates to ignore is an important hyperparameter, and starting averaging too early or too late leads

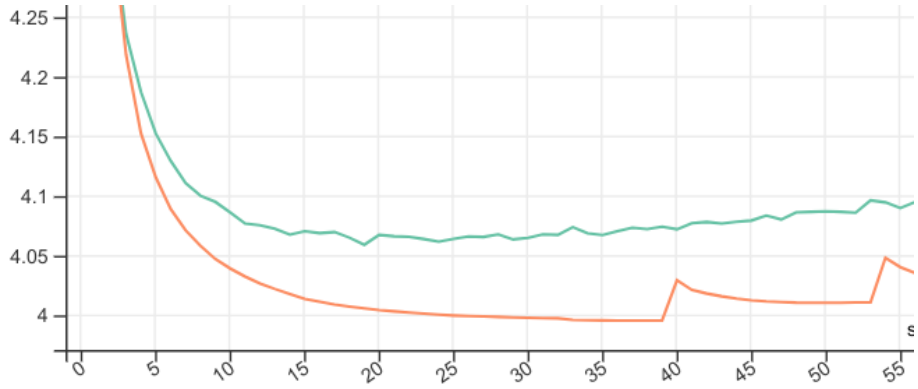
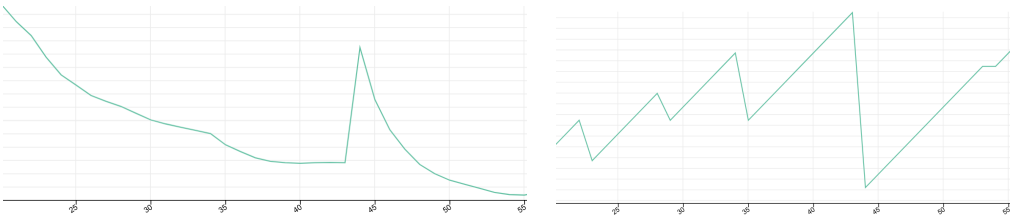


Figure 3.5: Raw (green) and 2TA (orange) validation loss with overfitting. The averaged loss bottoms out due to overfitting. Thus when the averages are reset (twice), the validation loss does not recover. Although the losses reported after the reset are suboptimal, it does not really matter as a better loss was reported already.



(a) Validation loss with the long average (F^L).

(b) Length of the long average (L).

Figure 3.6: Example of optimization entering a new basin. At $t = 40E$ the validation loss bottoms out and starts to slightly worsen. This is detected at $t = 43E$, and both averages are reset. With the averages now way too short, the loss spikes but then recovers.

to inefficient use of resources or suboptimal solutions. Our work focussed on improving generalization, which makes setting this hyperparameter even more difficult, especially in the presence of other hyperparameters and overfitting. Furthermore, before averaging starts, the loss is only weakly informative of the final performance, which makes early stopping unreliable. To alleviate these problems, we propose an anytime variant of Tail Averaging intended for improving generalization not pure optimization that has no hyperparameters and approximates the optimal tail at all optimization steps. Our algorithm is based on two running averages with adaptive lengths bounded in terms of the optimal tail length, one of which achieves approximate optimality with some regularity.

In summary, we presented a variant of Tail Averaging and Stochastic Weight Averaging based on two running averages. Compared to them, Two-Tailed Averaging requires additional storage for the second running average and relies

on periodic evaluation of generalization performance. In return, 2TA removes a hyperparameter and provides an estimate of the optimal tail at all optimization steps. This makes hyperparameter tuning easier and early evaluation more representative of final performance, allowing it to support early and anytime stopping better. Owing to its simplicity, low implementation cost and adaptivity, 2TA is a practical and widely applicable method for improving generalization.

Looking beyond the scope of this work, exploring the relationship between iterate averaging and learning rate schedules is a promising direction as existing [Merity et al., 2017] and our own limited experimental results indicate that dropping the learning rate and Tail Averaging perform comparably. The properties of our algorithm are particularly compelling for continual learning: by allowing the learning rate to remain high and being able to adapt the averaging length to changing circumstances, 2TA lets the model maintain high plasticity while reaping the benefits of averaging.

In addition, averaging weights can be viewed as a cheap approximation to averaging predictions when the averaged weights reside in a region with a suitable geometry. The combination of averaging weights within such regions and averaging predictions over regions (each with its own weight average) could potentially achieve a better loss than weight averaging alone at much lower storage and evaluation cost than pure prediction averaging. We leave these avenues for future work to explore.

4 REFINING RECURRENCES

> The proverbial German phenomenon of the "verb-at-the-end", about which droll tales of absentminded professors who would begin a sentence, ramble on for an entire lecture, and then finish up by rattling off a string of verbs by which their audience, for whom the stack had long since lost its coherence, would be totally nonplussed, are told, is an excellent example of linguistic recursion.
>
> *Douglas Hofstadter*

RELIABLE MODEL comparison is crucial for continued innovation in language modelling. With so much attention on Transformers [Vaswani et al., 2017], the development of purely recurrent models has ebbed away. It might be tempting to claim that purely recurrent models are fundamentally worse models of language than attention-based ones, but they seem to have an edge on small datasets, while on larger datasets their lack of scalability on current hardware [Hooker, 2020] also plays an important role in their evaluation. Due to being unfashionable and slow, their fitness might be underestimated especially on all but the smallest datasets with the exception of S4 [Gu et al., 2021], a highly parallelizable, long-range model. The purpose of this work is simply to apply more resources to advancing and evaluating recurrent models – despite, and to compensate for, their computational inefficiency – to better represent their performance in future model comparisons.

Starting from the Mogrifier LSTM [Melis et al., 2020], a state-of-the-art recurrent language model, we introduce several small changes, which involve not only the design of the recurrent cell but the overall architecture, the training objective, and optimization. On all datasets, these changes combine to a significant, and in some cases large, effect.

4.1 REWIRED LSTM

We propose a novel recurrent cell, a slightly tweaked version of the LSTM [Hochreiter and Schmidhuber, 1997c], as presented in (2.1). To recap, with m and n being the sizes of the input and the cell state, let the LSTM: $\mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^n$ be a function that computes the updated state $\mathbf{c}$ and output $\mathbf{h}$

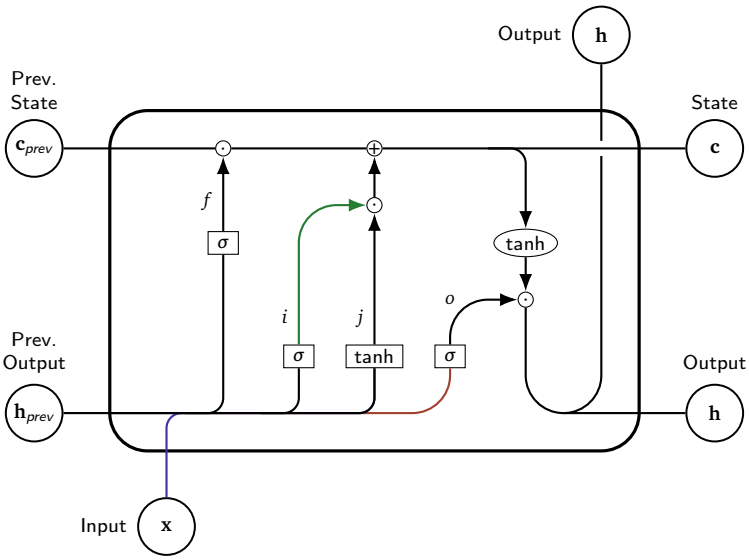


Figure 4.1: Schematic of the LSTM cell in the style of `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`. All gates are computed from $\mathbf{h}_{prev}$ and $\mathbf{x}$. Differences from the RLSTM cell are colour coded to ease comparison to Figure 4.2.

from the previous state $\mathbf{c}_{prev}$, output $\mathbf{h}_{prev}$ and the current input $\mathbf{x}$. Concretely, $\text{LSTM}(\mathbf{c}_{prev}, \mathbf{h}_{prev}, \mathbf{x}) = (\mathbf{c}, \mathbf{h})$, where

$$
\begin{aligned}
\mathbf{i} &= \sigma\left(\mathbf{W}^{ix}\mathbf{x} + \mathbf{W}^{ih}\mathbf{h}_{prev} + \mathbf{b}^{i}\right) \\
\mathbf{j} &= \tanh(\mathbf{W}^{jx}\mathbf{x} + \mathbf{W}^{jh}\mathbf{h}_{prev} + \mathbf{b}^{j}) \\
\mathbf{f} &= \sigma(\mathbf{W}^{fx}\mathbf{x} + \mathbf{W}^{fh}\mathbf{h}_{prev} + \mathbf{b}^{f}) \\
\mathbf{c} &= \mathbf{f} \odot \mathbf{c}_{prev} + \mathbf{i} \odot \mathbf{j} \\
\mathbf{o} &= \sigma(\mathbf{W}^{ox}\mathbf{x} + \mathbf{W}^{oh}\mathbf{h}_{prev} + \mathbf{b}^{o}) \\
\mathbf{h} &= \mathbf{o} \odot \tanh(\mathbf{c}).
\end{aligned}
$$

In the above, σ is the logistic sigmoid function, $\odot$ is the elementwise product, $\mathbf{W}^{**}$ and $\mathbf{b}^{*}$ are weight matrices and biases. As in Figure 4.1, the parts altered by the RLSTM (see Figure 4.2) are colour coded.

To accommodate the performance characteristics of parallel hardware, the activations of $\mathbf{i}$, $\mathbf{j}$, $\mathbf{f}$, $\mathbf{o}$ are in practice often computed by tiling the eight $\mathbf{W}^{**}$ into one large matrix and multiplying it with the concatenated input and recurrent state $[\mathbf{x}, \mathbf{h}_{prev}]$.

Intuitively, in $\mathbf{c} = \mathbf{f} \odot \mathbf{c}_{prev} + \mathbf{i} \odot \mathbf{j}$ the forget gate $\mathbf{f}$ and the proposed update $\mathbf{i} \odot \mathbf{j}$ depend on each other, and computing them in parallel from the same inputs may be partially redundant. On the flipside, the parametrization of the

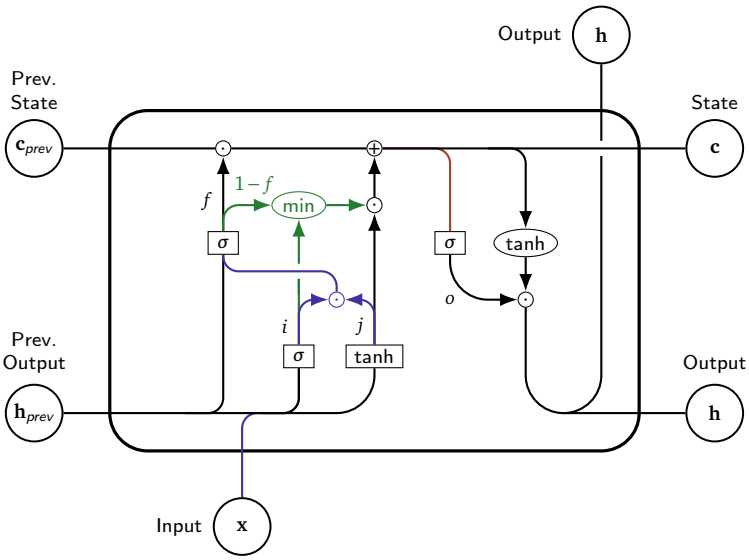


Figure 4.2: Schematic of the RLSTM cell. 1. The forget gate $\mathbf{f}$ is computed from $\mathbf{i} \odot \mathbf{j}$, 2. $\mathbf{i}$ is capped at $1 - \mathbf{f}$, 3. $\mathbf{o}$ is computed from $\mathbf{c}$. There is an apparent loss of parallelization opportunities due to the increased depth of the computation graph.

proposed update as a product is more expressive than that of the forget gate, so the overall cell update may lose some of the extra expressivity. Hence, it makes sense to compute the forget gate from the proposed update instead. Note that this reduces the opportunities for parallelization and makes cell updates slower.

Further disregarding efficiency on today's hardware, to reduce parameter count and to encourage storing more information in the cell state $\mathbf{c}$, we compute $\mathbf{o}$ from $\mathbf{c}$, dropping a potential bypass connection from $\mathbf{x}$ to $\mathbf{h}$. Finally, to make exploding gradients less likely, we cap $\mathbf{i}$ at $\mathbf{1} - \mathbf{f}$ to ensure $|c_u| \leqslant 1$ for all memory units u. The update of our Rewired LSTM (RLSTM) cell takes the form

$$
\begin{aligned}
\mathbf{i} &= \sigma\big(\mathbf{W}^{ix}\mathbf{x} + \mathbf{W}^{ih}\mathbf{h}_{prev} + \mathbf{b}^{i}\big) \\
\mathbf{j} &= \tanh\big(\mathbf{W}^{jx}\mathbf{x} + \mathbf{W}^{jh}\mathbf{h}_{prev} + \mathbf{b}^{j}\big) \\
\mathbf{f} &= \sigma\big(\mathbf{W}^{fu}\mathbf{i} \odot \mathbf{j} + \mathbf{W}^{fh}\mathbf{h}_{prev} + \mathbf{b}^{f}\big) \\
\mathbf{c} &= \mathbf{f} \odot \mathbf{c}_{prev} + \min(\mathbf{i}, 1 - \mathbf{f}) \odot \mathbf{j} \\
\mathbf{o} &= \sigma\big(\mathbf{W}^{oc}\mathbf{c} + \mathbf{b}^{o}\big) \\
\mathbf{h} &= \mathbf{o} \odot \tanh(\mathbf{c}),
\end{aligned}
$$

where the changes from the LSTM are coloured. Based on this altered computation, we define the RLSTM function similarly to the LSTM above.

The way recurrent cells are combined can also be improved. Here, we opt to use residual connections, where previous works have used stacked LSTMs [Merity et al., 2017] or skip connections [Melis et al., 2020], which feed directly into the final output. Along with this change, we apply dropout [Hinton et al., 2012] to the cell output before it is added to the residual branch.

To describe the overall architecture more formally, with time steps $t \in [1, 2, \dots]$ and layers $l \in [1..L]$, from the vector of token indices $\mathbf{w}$ in a fixed vocabulary, the probability distribution of the next token $p(.|\mathbf{w}_{<t}, \mathbf{M}_t)$ is computed as

$$
\begin{aligned}
\mathbf{c}_0^l, \mathbf{h}_0^l &= \mathbf{0}, \mathbf{0} \\
\hat{\mathbf{x}}_t^0 &= \text{onehot}(\mathbf{w}_t)\mathbf{E}^{\text{in}} \odot \mathbf{M}_t^{\text{in}} \\
\hat{\mathbf{x}}_t^l &= \mathbf{h}_t^l \odot \mathbf{M}_t^{\text{cell},l} && (l > 1) \\
\hat{\mathbf{h}}_t^l &= \mathbf{h}_t^l \odot \mathbf{M}^{\text{state},l} \\
\mathbf{c}_t^1, \mathbf{h}_t^1 &= \text{[R]LSTM}\Big(\mathbf{c}_{t-1}^0, \text{mogrify}\Big(\hat{\mathbf{h}}_{t-1}^1, \hat{\mathbf{x}}_t^0\Big)\Big) \\
\mathbf{c}_t^l, \mathbf{h}_t^l &= \text{[R]LSTM}\Big(\mathbf{c}_{t-1}^l, \text{mogrify}\Big(\hat{\mathbf{h}}_{t-1}^l, \textstyle\sum_{i=1}^{l-1} \hat{\mathbf{x}}_t^i\Big)\Big) \\
p(.|\mathbf{w}_{<t}, \mathbf{M}_t) &= \text{softmax}\Big(\Big(\textstyle\sum_{l=1}^{L} \hat{\mathbf{x}}_t^l\Big) \odot \mathbf{M}_t^{\text{out}}\mathbf{E}^{\text{out}} + \mathbf{b}^{\text{out}}\Big),
\end{aligned}
$$

where we assume that $m = n$ to allow for a residual architecture without projections and use the *mogrify* function as defined in (2.2). $\mathbf{E}^{\text{in}}$ and $\mathbf{E}^{\text{out}}$ are the input and output embedding matrices. For word-based language modelling, we set $\mathbf{E}^{\text{out}}$ to the transpose of $\mathbf{E}^{\text{in}}$ [Zoph and Le, 2016, Press and Wolf, 2016]. $\mathbf{M}_t$ is the set of individual input, cell, state and output dropout mask matrices $\mathbf{M}_t^{\text{in}}$, $\mathbf{M}_t^{\text{cell},l}$, $\mathbf{M}^{\text{state},l}$ and $\mathbf{M}_t^{\text{out}}$. Note that for state dropout, we use the variational dropout of Gal and Ghahramani [2016], thus $\mathbf{M}^{\text{state},l}$ does not depend on t. In addition, when an RLSTM is used instead of an LSTM cell, we also apply the state dropout mask to $\mathbf{c}$ in the calculation of $\mathbf{o} = \sigma(\mathbf{W}^{oc}(\mathbf{c} \odot \mathbf{M}^{\text{state},l}) + \mathbf{b}^o)$.

4.3 OBJECTIVE

In the objective, we average model predictions over multiple dropout samples:

$$
\ln p\big(\mathbf{w}_t \;\big|\; \mathbf{w}_{<t}\big) = \ln\Bigg(\frac{1}{D}\sum_{d=1}^{D} p\Big(\mathbf{w}_t \;\Big|\; \mathbf{w}_{<t}, \mathbf{M}_t^d\Big)\Bigg),
$$

where D is the number of samples taken. As pointed out by Noh et al. [2017], when dropout is interpreted as optimizing a variational lower bound on the log likelihood [Gal and Ghahramani, 2016], this procedure is an instantiation of Importance Weighted Autoencoders [Burda et al., 2015], which provide a bound tighter than the single-sample ELBO.

4.4 OPTIMIZATION

To increase the stability of optimization and allow slightly higher learning rates, we use Rectified Adam [Liu et al., 2019]. If training diverges, we reset the weights and the optimization state to the previous best checkpoint and multiply the learning rate by 0.9.

Merity et al. [2017] switch to averaging weights at a late stage of optimization when the validation loss has not decreased for a while. A similar procedure, called Stochastic Weight Averaging (SWA), was also proposed [Izmailov et al., 2018], and corresponding theory was developed in Jain et al. [2018] under the name of Tail Averaging. Here, we employ Two-Tailed Averaging [Melis, 2022], which has no hyperparameters and provides a good approximation to the optimal weight average at every step of optimization. Two-Tailed Averaging (2TA) thus requires no tuning and is a much better fit with early stopping, which is what we do on Enwik8 and Text8 [Hutter, 2012]. While easier to work with, 2TA does not improve the final results over well-tuned Tail Averaging, which is also used by our baseline, the Mogrifier LSTM.

LSTM and RLSTM forget gates are initialized with Chrono init [Tallec and Ollivier, 2018] as $\mathbf{b}^f \sim \ln(\mathcal{U}(1, T_{\max} - 1))$, where $T_{\max}$ is a tuned hyperparameter. Finally, we just train models longer where it is beneficial.

4.5 DYNAMIC EVALUATION

Hinton and Plaut [1987] proposed fast weights, wherein parameters have a slow- and a fast-changing component. Much later, Ba et al. [2016] showed a form of attention to be an instantiation of fast weights. Similarly, some forms of meta learning, e.g. few-shot adaptation with gradient updates, can be interpreted as fast weights. To mimic this setting and gain a particularly general fast weights implementation, it would be desirable to allow the model weights to depend on the context as in $p(x_i|\theta(x_{<i}), x_{<i})$, where the fast weights $\theta(x_{<i})$ are computed with gradient-based updates to the slow weights θ_0. However, due to the practical difficulties involved in training batches with per-example

inner gradient updates, we eschew fast weights at training time but not when performing evaluation, thus we end up with what is called dynamic evaluation [Krause et al., 2017]. Here, we present dynamic evaluation results as a proxy for the gradient-based fast weights model.

As argued heuristically above, attention may be interpreted as a particular way of adapting the weights to the context (outer product memory), like the Mogrifier, which has an even more restricted form of update (scaling columns of weight matrices). Thus, it is not surprising that Melis et al. [2020] found that Transformers are better at adapting without changing their weights, leaving less in-context signal for dynamic evaluation to pick up.

4.6 EXPERIMENTAL SETUP

We follow the experimental setup of our baseline [Melis et al., 2020]. In the following, we list only the most pertinent choices in our experimental setup; everything else is the same as in the baseline. Note that the baseline's and our LSTM implementation already includes the capped input gate ($\min(1 - \mathbf{f}, \mathbf{i})$) of the RLSTM. For the black-box hyperparameter tuner [Golovin et al., 2017], due to the switch to a residual architecture, the baseline's *inter_layer_dropout* hyperparameter is replaced with *cell_output_dropout* (see $\mathbf{M}^{\text{cell},l}$ in §4.2). In addition, the top of the Chrono init range T_{max} is a new hyperparameter from the $[e^2, e^5]$ range.

For word-level language modelling on Penn Treebank [Marcus et al., 1993] with preprocessing by [Mikolov et al., 2010], we trained 2-layer models for about 400 epochs with 8 dropout samples. Batch size was 128, and we trained with a BPTT [Werbos, 1990] window size of 70. Experiments on Wikitext-2 [Merity et al., 2016] were conducted similarly, with the exception of training for 250 epochs and using 4 dropout samples.

For character-based language modelling on Penn Treebank, we trained 2-layer models for 100 epochs with 4 dropout samples, batch size 128, and a BPTT window size of 200. On Enwik8 and Text8 [Hutter, 2012], we trained 6-layer models for 200 epochs, whereas the baseline model had 4 layers and was trained for only 29 epochs. Batch size was 128, and the BPTT window size was set to 256. Increasing the window size had no discernible effect in agreement with the findings of Khandelwal et al. [2018]. Due to the long time required to train these models, the best hyperparameters were selected from a random pool of 60 candidates. Since preliminary experiments indicated that the benefit

Table 4.1: Word-level perplexities of near state-of-the-art models. Names for models with our new results are in bold. On Wikitext-2, the baseline employed Mixture of Softmaxes [Yang et al., 2017b], but we found no benefit to that when used in conjunction with multiple dropout samples.

			No Dyneval		Dyneval	
			Val.	Test	Val.	Test
PTB	Transformer-XL [Dai et al., 2019]	24M	56.7	54.5		
	Mogrifier LSTM [Melis et al., 2020]	24M	52.1	51.0	45.1	45.0
	Mogrifier LSTM	24M	49.9	48.5	43.5	43.3
	Mogrifier RLSTM	24M	48.9	**47.9**	42.9	**42.9**
WT2	Mogrifier LSTM MoS2	35M	58.7	56.6	40.6	39.0
	Mogrifier LSTM	35M	57.4	55.8	40.0	38.6
	Mogrifier RLSTM	35M	56.7	**55.0**	39.3	**38.0**

of using multiple dropout samples was less than 0.01 bpc, we refrained from using more than one dropout sample to save time.

Model evaluation was performed with the standard, deterministic dropout; we refrained from using the more expensive Monte Carlo averaging [Gal and Ghahramani, 2016]. The optimal softmax temperature was selected at evaluation time to maximize the validation log-likelihood [Melis et al., 2018]. Finally, we report results with and without dynamic evaluation [Krause et al., 2017].

4.7 RESULTS

Due to the aforementioned loss of parallelization opportunities, we found that the RLSTM was about 10%–30% slower than the LSTM on NVIDIA p100 GPUs. Still, the RLSTM retained a significant advantage over the LSTM even when trained for the same wall clock time.

On word-level language modelling (see Table 4.1), only training for half the epochs, our Mogrifier LSTM outperformed the same model in Melis et al. [2020], the baseline. This was mostly due to the quicker convergence with multiple dropout samples and, to a small degree, to the initialization of the forget gate. Note that on 2-layer models, which we used on this task, residual connections are identical to skip connections, which are employed by the baseline. On top of these improvements, the RLSTM outperformed the LSTM by a small margin, and we established a new state of the art on both datasets with and without dynamic evaluation.

Table 4.2 shows our results on character-based language modelling. On Penn Treebank, again only training for half the epochs, we significantly boosted the

Table 4.2: Bits per character on character-based datasets of near state-of-the-art models. Names for models with our new results are in bold. The best test results with and without dynamic evaluation for a given dataset and model size are in bold unless there is a smaller model with a better result.

			No Dyneval		Dyneval	
			Val.	Test	Val.	Test
PTB	Mogrifier LSTM [Melis et al., 2020]	24M	1.149	1.131	1.098	1.088
	Mogrifier LSTM	24M	1.127	1.109	1.085	1.074
	Mogrifier RLSTM	24M	1.115	**1.096**	1.073	**1.061**
Enwik8	Transformer-XL (d24) [Dai et al., 2019]	277M		0.993		0.940
	Longformer [Beltagy et al., 2020]	277M		**0.97**		
	Transformer-XL (d18) [Dai et al., 2019]	88M		1.03		
	Longformer [Beltagy et al., 2020]	102M		**0.99**		
	Mogrifier LSTM [Melis et al., 2020]	96M	1.110	1.122	1.009	0.988
	Mogrifier LSTM	96M	1.057	1.072	0.963	0.946
	Mogrifier RLSTM	96M	1.028	1.042	0.952	**0.935**
	Transformer-XL (d12) [Dai et al., 2019]	41M		1.06		1.01
	Longformer [Beltagy et al., 2020]	41M	1.02	**1.00**		
	Mogrifier LSTM [Melis et al., 2020]	48M	1.135	1.146	1.035	1.012
	Mogrifier LSTM	48M	1.083	1.094	0.988	0.969
	Mogrifier RLSTM	48M	1.060	1.071	0.986	**0.968**
Text8	Transformer-XL (d24) [Dai et al., 2019]	277M		**1.08**		**1.038**
	Mogrifier LSTM	96M	1.033	1.106	0.977	1.047
	Mogrifier RLSTM	96M	1.022	1.096	0.975	1.044
	Longformer [Beltagy et al., 2020]	41M	1.04	**1.10**		
	Mogrifier LSTM	48M	1.063	1.140	1.005	1.075
	Mogrifier RLSTM	48M	1.044	1.119	0.998	1.068

results of the state-of-the-art baseline model by using multiple dropout samples and to a smaller degree by tuning the forget gate initialization. Our results were then further improved by switching to the RLSTM.

Our results on Enwik8 and Text8 were boosted greatly. However, some corners were cut due the high computational cost, thus the results can likely be improved further e.g. with proper tuning (instead of random sampling), by training even longer, and by using multiple dropout samples. We heuristically estimate a 0.015–0.03 bpc suboptimality due to these factors only.

Taking advantage of Two-Tailed Averaging's (2TA) online estimates and the fact that hyperparameters were selected randomly, we can estimate the contribution of training longer (200 vs 29 epochs). For example, in the 48M parameter setting on Enwik8, we observed that training longer lowered the validation bpc by about 0.035, leaving another 0.016 bpc for the contributions

of residual connections and the forget gate initialization. Finally, the RLSTM outperformed the LSTM by 0.022 bpc. Similar relative contributions were observed in all other settings on Enwik8 and Text8.

Some of the changes we made increased primarily the stability of training and the efficiency of hyperparameter tuning without discernible effect on the final results. Using Rectified Adam and restarting from a previous checkpoint on divergence greatly reduced the number of failed runs, and 2TA made tuning easier by removing a hyperparameter while opening the door for early stopping due to its online nature.

We found that in the early stages using D dropout samples allowed optimization to make more progress per step but less than the progress with a single sample and D times the number of steps. On datasets that are on the smaller side and where dropout rates are high, in the late stages of optimization, the multi-sample objective proved better in terms of validation perplexity. However, that was not the case on Enwik8 and Text8, possibly due to being bottlenecked by other issues such as the length of optimization.

4.8 CONCLUSIONS

§ Just because some recurrent models suffer from being hard to optimize and inefficient on today's hardware, they are not necessarily bad models of language. We demonstrated this by the extent to which these models can still be improved by a combination of a slightly better recurrent cell, architecture, objective, as well as optimization. In the process, we established a new state of the art for language modelling on small datasets and on Enwik8 with dynamic evaluation.

We strengthened purely recurrent models' language modelling results on a varied collection of datasets using various techniques and a slightly novel recurrent cell. These new baselines better represent the models' ability but do not necessarily allow for a fair comparison to previous results. In particular, while we cited and listed results of the best transformer-based models, we did not evaluate them ourselves using the same methodology. This is especially obvious where previous works did not report dynamic evaluation results. The comparisons we made to recurrent models were also rather targeted: we focussed solely on the state-of-the-art purely recurrent model, the Mogrifier LSTM, and ignored both the less performant models and those combined with attention [Merity, 2019, Lei, 2021]. Nevertheless, the combination of our strong results and little novelty highlight the extent to which minor details and computational efficiency on current hardware affect model comparisons.

With that in mind, the primary contribution of this chapter is to apply more resources to designing and evaluating recurrent models and to provide stronger baselines for model comparisons. Second, the set of improvements to models of the recurrent cell, overall architecture, training objective, and optimization that we employed or proposed might inform future practice and research. In addition, the fact that two quite different architectures (recurrent and attention-based) appear to perform similarly in terms of perplexity suggests that bigger changes are necessary to develop data-efficient language models.

5 LAX LATENTS

> All our words from loose using
> have lost their edge.
>
> *Ernest Hemingway*

We observed that recurrent and attention-based models perform similarly on small datasets, where differences in model bias should be more readily apparent than at large scale. As discussed in §1, there are good reasons to believe that much more data-efficient models exist. These factors prompted us to take bolder steps towards equipping our models with a stronger bias. While model biases can be altered in several ways (e.g. by changing regularization, the optimizer or the data), here we consider adding latent variables and explicitly structuring our models with conditional independence assumptions. Our hope is that these two tools provide a rich design space to explore with very direct effects on model biases. However, as we will see, training latent variable models at scale is challenging, and the common workhorse, the variational autoencoder, has a tendency to produce degenerate solutions. So before we can explore the design space, we must improve the tool.

Taking a broader view, our decision to focus on latent variable models rests upon their promise in learning about the underlying generative process, discovering structure in the data, principled representation learning, improved generalization and controllable generation; all made possible by judicious choice of model structure, such as the prior, the likelihood, and any conditional independence assumptions.

Variational autoencoders (VAEs, Kingma and Welling, 2013, Rezende et al., 2014) provide a general framework for statistical inference in latent variable models of the form $p_\theta(x, z) = p_\theta(x|z)p(z)$, where x is the observable data, z is a set of latent variables, and the objective is to learn the parameters θ so that the resulting marginal distribution $p_\theta(x)$ well approximates the empirical data distribution $p_D(x)$. The generality of VAEs comes at a price though, as their training objective introduces a new learnt component, the variational posterior $q_\phi(z|x)$, to approximate the true posterior $p_\theta(z|x)$. As we will see, the VAE objective pulls $q_\phi(z|x)$ towards the prior and also the two posteriors together, which biases the inference towards oversimplified posteriors relative to what

fitting the model $p_\theta(x, z)$ to the data would itself require. When having to match the data distribution does not strongly constrain the model posterior – as is the case when the decoder $p_\theta(x|z)$ is highly flexible (e.g. a neural network) – the biases of the inference method are fully on display and have been observed in the underuse of latent variables, whose extreme case, where the latents are ignored entirely, is known as posterior collapse [Zhao et al., 2019] and is the main focus of this chapter. The issue of posterior collapse is especially acute with auto-regressive decoders, which are capable of modelling the data without using the latents at all [Yang et al., 2017b]. Bowman et al. [2015] attributed this to a 'difficult learning problem', and dozens of attempts to remedy it followed [Alemi et al., 2017, Dieng et al., 2017, van den Oord et al., 2017, Kim et al., 2018] to help VAEs fulfil their promise in representation learning.

We aim to understand and remedy posterior collapse in VAEs with the long-term goal of facilitating research into latent variable models. While acknowledging that their ultimate evaluation is necessarily in terms of performance on down-stream tasks or as density models, we demonstrate that suboptimal inference can present a severe tradeoff between latent variable usage and data fit. This inefficiency of inference renders the posterior unfit for its purpose as a representation of the data. Therefore, instead of measuring the performance of the learned models on down-stream tasks, we evaluate this tradeoff in terms of their rate-distortion behaviour [Alemi et al., 2017], by measuring the rate as the mutual information between the observables x and the latents z, and distortion based on the negative log-likelihood assigned by the model to the data.

Several interacting factors (outlined in §5.1) play a role in posterior collapse, but the two most pertinent to this work are the looseness of the lower bound and underspecification. *The looseness of the variational lower bound* (such as the ELBO objective) biases solutions found by VAEs away from the theoretical optimum. Hence, designing tighter lower bounds has been a mainstay of research on variational inference [Rezende and Mohamed, 2015, Kingma et al., 2016, Tomczak and Welling, 2018], and one approach is to take multiple samples from a variational posterior distribution $q_\phi(z|x)$ to form an approximation of the marginal likelihood $p_\theta(x)$.[1] In these so-called Monte Carlo objectives [Mnih and Rezende, 2016], such as IWAE [Burda et al., 2015], $q_\phi(z|x)$ does not represent the true posterior $p_\theta(z|x)$ explicitly, but it can be interpreted as a factor in a more elaborate, implicit approximate posterior [Cremer et al., 2017]. In this context, it is thus more correct to refer to $q_\phi(z|x)$ as the proposal distribution.

[1]Throughout, we do not explicitly distinguish between densities and probability mass functions unless it is necessary; these are naturally dictated by the type (continuous/discrete) of the underlying variables.

The second factor we consider is *underspecification*. We use underspecification in the sense that models that are optimal in terms of marginal likelihood can differ greatly in their posteriors [Huszár, 2017a], thus the optimization aiming to fit the data by maximizing the likelihood is underspecified. This issue is most apparent in that, for VAEs with powerful function classes expressing the likelihoods, posterior collapse can be an optimal solution in terms of data fit because the usual evidence lower bound (ELBO) objective is neutral with respect to the mutual information between the latents and the data. However, as Huszár [2017a] argues and as we show in §5.3, the marginal likelihood objective leaves the posterior underspecified, so underspecification is a shortcoming of the ELBO in only as much as it does not correct for it. Many proposed methods aim to address underspecification by constraining the mutual information between the observable and the latent variables [Higgins et al., 2017, Phuong et al., 2018, Zhao et al., 2019] relying on the availability of a good posterior approximation. As discussed next, with Monte Carlo objectives we do not have access to a good posterior approximation, and this must be addressed if we hope constrain mutual information to reduce underspecification.

When the proposal $q_\phi(z|x)$ is close to $p_\theta(z|x)$, we can approximate the mutual information $I_p(X, Z) = \mathbb{E}_{p_\theta(x)} \mathrm{KL}(p_\theta(z|x) \parallel p(z))$ (where KL denotes the Kullback–Leibler divergence) with $\mathbb{E}_{p_\theta(x)} \mathrm{KL}(q_\phi(z|x) \parallel p(z))$, which features the proposal q_ϕ. Unfortunately, with Monte Carlo objectives we cannot expect the proposal to approximate the posterior well [Mnih and Rezende, 2016], and $\mathrm{KL}(q_\phi(z|x) \parallel p(z))$ (called the *representational KL*) is a highly biased estimate of $\mathrm{KL}(p_\theta(z|x) \parallel p(z))$ (the *true KL* from now on).

Our main contribution is a novel method to constrain the mutual information between the observable and the latent variables in the context of multi-sample Monte Carlo objectives, bringing research on loose bounds and underspecification together. More specifically, we introduce an optimization objective that features two terms, one coming from the variational lower bound and another from the mutual information, where both terms are based on multiple samples taken from the proposal distribution q_ϕ. Compared to the single-sample case, we get the benefit of the tighter lower bounds Monte Carlo objectives offer without having to give up control of the mutual information. At the same time, our multi-sample estimators for the mutual information are much more efficient than in the single-sample case and can better tolerate low-quality posterior approximations. Our mutual information term is computed from the recycled samples of the Monte Carlo estimator of the marginal likelihood, hence the method has negligible computational overhead. Combined with best-of-breed

gradient estimators, such as DReG [Tucker et al., 2018] and VIMCO [Mnih and Rezende, 2016] for a multi-sample objective, we train models with continuous and discrete latents at much improved rate-distortion.

The rest of the chapter is structured as follows.

- §5.1 provides an overview of the known causes of posterior collapse, which are all shortcomings of the inference method except for underspecification.
- In §5.3, we characterize underspecification as the lack of sufficient conditional independence assumptions, which may be partially offset by constraining mutual information.
- §5.4 proposes reusing samples from Monte Carlo objectives to better estimate the mutual information, which is the main contribution.
- §5.5 and §5.6 show that the representational KL, which underlies many mutual information estimates, corresponds to the single-sample case of our estimators, and the single-sample objectives built on them are equivalent to the β-VAE objective of Higgins et al. [2017].
- §5.7 experimentally verifies the effectiveness of the proposed methods on synthetic and language modelling tasks, emphasizing evaluation in terms of the data fit vs latent usage tradeoff.

5.1 VARIATIONAL AUTOENCODERS AND POSTERIOR COLLAPSE

This section introduces variational autoencoders and describes the known causes of posterior collapse. Contrary to what the name variational autoencoder may suggest, a VAE is not a model itself but an inference[2] mechanism for models of the form $p_\theta(x, z) = p_\theta(x|z)p(z)$ where $\{p_\theta(x|z)\}$ is a parametric family of conditional distributions [Kingma and Welling, 2013]. VAE training constructs an approximate maximum likelihood estimate of the model parameters θ with the aim of maximizing the probability over the empirical data distribution: $\arg\max_\theta \mathbb{E}_{x\sim p_D(x)}[\ln p_\theta(x)]$. Since $\ln p_\theta(x)$ has no analytic form in general, VAEs posit a variational family of distributions $\mathcal{Q} = \{q_\phi(z|x)\}$ parameterized by ϕ to approximate the true posterior $p_\theta(z|x)$ and construct a lower bound on the marginal likelihood $p_\theta(x)$, also called the evidence:

$$\ln p_\theta(x) \geqslant \mathrm{ELBO}(x, \theta, \phi) = \ln p_\theta(x) - \mathrm{KL}\big(q_\phi(z|x) \,\big\|\, p_\theta(z|x)\big) \tag{5.1}$$

$$= \mathop{\mathbb{E}}_{z\sim q_\phi(z|x)}[\ln p_\theta(x|z)] - \mathrm{KL}\big(q_\phi(z|x) \,\big\|\, p(z)\big). \tag{5.2}$$

[2] We use *inference* in the statistical sense, commonly referred to as *training* or *learning* in the machine learning literature.

As evident in its *posterior-contrastive* form, the ELBO (5.1) is a lower bound on $\ln p_\theta(x)$ due to the non-negativity of the KL divergence [Kullback and Leibler, 1951]. In its *prior-contrastive* form (5.2), both the expectation and the KL divergence terms can be estimated by taking a single sample from q, which forms the basis of its optimization. Alternatively, the KL may be computable analytically. Gradient-based optimization with VAEs is performed jointly over parameters θ of the model and ϕ of the approximate posterior as

$$\arg\max_{\theta,\phi} \mathop{\mathbb{E}}_{x\sim p_D(x)} \mathrm{ELBO}(x,\theta,\phi).$$

In practice, the expectation in (5.2) is approximated with a single sample from $q_\phi(z|x)$ and the expectation over $p_D(x)$ with a minibatch, which goes by the name of doubly stochastic variational inference [Titsias and Lázaro-Gredilla, 2014]. For a broader context, we refer the reader to Zhang et al. [2018], who provide a comprehensive overview of developments in variational inference [Jordan et al., 1999]. From this point on, wherever possible we drop the subscripts in p_θ and q_ϕ to declutter the notation.

The possibility of posterior collapse (also referred to as over-pruning, Yeung et al. 2017, or information preference, Zhao et al. 2019) is most evident in the prior-contrastive ELBO (5.2). If the likelihood $p(x|z)$ is able to model the distribution of x without the latents z, then the reconstruction term $\mathbb{E}_{q(z|x)} \ln p(x|z)$ is just $\ln p(x)$ independently of $q(z|x)$. Since q does not affect the reconstruction term, we can just set it to the prior $p(z)$, which is often in $\mathcal{Q}$, so that the KL penalty is zero. In this case, $q(z|x) = p(z|x)$ is also satisfied, but unfortunately the two posteriors have now collapsed to match the prior $p(z)$, which renders the latent variables useless.

The issues in what we observe as posterior collapse (or its milder form, the underuse of latents) span a number of causes. Figure 5.1 summarizes the main known contributors to posterior collapse and their interactions. At a glance, the immediate causes are high-variance gradient estimates, a loose lower bound, and underspecification.

- **Gradient noise**: Optimization can be adversely affected by high-variance gradient estimators [Roeder et al., 2017, Tucker et al., 2018] and minibatch noise in stochastic gradient descent [Titsias and Lázaro-Gredilla, 2014]. Unlike minibatch noise, sampling noise – the variance induced by the latents – can be eliminated by ignoring the latents. Because the looseness of the lower bound is proportional to the sampling variance [Maddison et al., 2017], coupled with SGD's preference for flat minima [Hochreiter and Schmidhuber, 1995], this biases optimization towards posterior collapse.

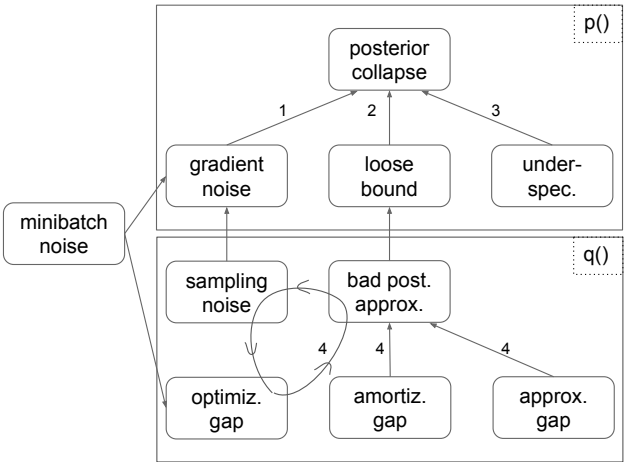


Figure 5.1: Causes of posterior collapse in VAEs. 1. High variance gradient estimates and SGD's preference for flat minima exerts pressure to reduce variance by ignoring the latents. 2. A loose lower bound underestimates the benefits of the latents. 3. Underspecification can allow posterior collapse to be the theoretical optimum. 4. Posterior collapse reduces or eliminates all of the gaps and the sampling noise.

- **Loose lower bound**: As its posterior-contrastive form (5.1) suggests, the ELBO is tight if $q(z|x)$ matches $p(z|x)$, and the worse the approximation, the looser the ELBO. However, perfect posterior approximation might be hard or impossible to achieve, depending on the following factors [Cremer et al., 2017].
  - The **approximation gap** is the distance of the true posterior from the variational family $\mathcal{Q}$ if $p(z|x) \notin \mathcal{Q}$ [Shu et al., 2018, Razavi et al., 2019].
  - The **amortization gap** is the gap caused by the encoder's inability to represent the optimal $q_{\phi^\star}(z|x)$ (which is $p_\theta(z|x)$) for all x with the same ϕ [Kim et al., 2018, Shu et al., 2018].
  - The **optimization gap** is caused the suboptimality of $q_\phi(z|x)$ found by the optimizer relative to $q_{\phi^\star}(z|x)$ [He et al., 2019].

  Notably, there is a negative feedback loop: optimization difficulties cause bad posterior approximation, which increases the variance induced by sampling the latents, which makes optimization harder.
- **Underspecification**: As argued by Alemi et al. [2017], the ELBO is neutral with respect to the mutual information, and even perfect optimization can land anywhere on the rate-distortion curve (the expected likelihood of the data as a function of the mutual information of the data and the latents). More generally, the optimization task for generative models of the form $p(x, z) = p(x|z)p(z)$ is often underspecified [Huszár, 2017a], which we explore in §5.3.

Unfortunately, posterior collapse reduces or eliminates all of the gaps and also the sampling noise, which makes the bound tight. For VAEs to use their latent variables reliably and optimally, all of the above issues must be addressed. The direction we take here delegates the problem of dealing with the looseness of the bound to a Monte Carlo estimator of the marginal likelihood, such as IWAE, and that of reducing gradient noise to a corresponding gradient estimator, such as DReG or VIMCO. Importantly, in addition to providing tighter bounds, Monte Carlo estimators employ multiple samples from $q(z|x)$, which benefits our efforts to tackle the issue of underspecification by designing better estimators of the mutual information.

5.2 RELATED WORKS

With several interacting issues to disentangle, the body of work on posterior collapse has grown large and disparate. Many works aim to make the prior or the posterior more flexible to tighten the lower bound [Rezende and Mohamed, 2015, Kingma et al., 2016, Tomczak and Welling, 2018]. At the same time, Shu et al. [2018] argue that while it is tempting to think that the variational family $\mathcal{Q}$ and $q_\phi(z|x)$ should be as large and as flexible possible, a more restrictive choice can prove useful in performing posterior regularization. Kim et al. [2018] propose reducing the amortization gap by only initializing the value of the latent variables according to the encoder, then optimizing their values separately for each example as in non-amortized variational inference [Jordan et al., 1999]. He et al. [2019] focus on improving the approximation quality of $q_\phi(z|x)$ by taking several optimization steps on ϕ for each update of θ in p_θ. Yang et al. [2017b] replace an auto-regressive decoder with dilated convolutions to restrict the available context and to thus enforce the use of the latents. Van den Oord et al. [2017] introduce categorically distributed latents with a fixed KL cost and replace the prior with the aggregate posterior, which eliminates any pressure to ignore the latents.

Instead of completely eliminating the pressure to ignore the latents, controlling the KL term in (5.2) by reducing its cost also received lots of attention. Downweighting or annealing it during training is common [Bowman et al., 2015, Higgins et al., 2017, Sønderby et al., 2016, Maalø e et al., 2019, Huang et al., 2018]. Kingma et al. [2016] flat spot the KL term so that small KL values are not penalized. Razavi et al. [2019] achieve a similar effect by choosing the variational family and the approximate posterior such that there is always an approximation gap.

Several works [Alemi et al., 2017, Zhao et al., 2019, McCarthy et al., 2019, Rezaabad and Vishwanath, 2020, Serdega and Kim, 2020] recognize the issue of underspecification and propose to constrain the mutual information. All work with single sample VAEs and the representational KL. In contrast, we propose combining multi-sample Monte Carlo objectives with estimates based on the true KL. As we will see, this results in our mutual information estimates being less tied to the quality of $q(z|x)$, while containing the single-sample, representational KL scenario as a degenerate case.

5.3 CIA AND POSTERIOR COLLAPSE

In this section, we explore *why* our models are underspecified to better understand whether constraining the mutual information is a good solution. More specifically, we ask under what conditional independence assumptions (CIA[3]) can posterior collapse be an optimal solution for *any* model to abstract away from finite capacities and optimization difficulties. The next proposition shows that the answer is in fact trivial: the independence assumptions can be satisfied with a model where the latent and the observable variables are independent and the model matches the data distribution perfectly if and only if the independence assumptions are compatible with the marginal distribution of the data.

In particular, we consider Bayesian networks, given as

$$p(x, z) = p(z) \prod_{i=1}^{n} p\big(x_i \,\big|\, \mathrm{pa}(i), \mathbb{1}_{[1,k]}(i) z\big),$$

where $\mathrm{pa}(i)$ denotes the set of parent nodes of x_i, i.e. those on which it directly depends, and the first k x_i directly depend also on z.

Proposition 8. *Let $\{x_i\}$, $i \in [1, n]$, be a partitioning of x, and $\mathrm{pa}(i) \subseteq \{x_i\}$. For any prior $p(z)$, there exists a distribution[4] $p(x, z)$ satisfying, for all x and z,*
(i) $p(x, z) = p(x)p(z)$ *(posterior collapse);*
(ii) $p(x) = p_D(x)$ *(perfectly modelling the data);*
(iii) $p(x, z) = p(z) \prod_{i=1}^{n} p\big(x_i \,\big|\, \mathrm{pa}(i), \mathbb{1}_{[1,k]}(i) z\big)$ *for some* $k \leqslant n$ *(CIA)*
if and only if $p_D(x) = \prod_{i=1}^{n} p_D(x_i|\mathrm{pa}(i))$ *(the data distribution is compatible with the CIA of (iii)).*

Proof. If $p_D(x) = \prod_{i=1}^{n} p_D(x_i|\mathrm{pa}(i))$, then it is easy to check that $p(x, z) =$

[3]Due to CIA being a collective noun in other contexts, we let its singular form stand also for its plural.

[4]Here we consider the set of all distributions and not only the parameterized family $\{p_\theta\}$.

$p_D(x)p(z)$ satisfies the conditions (i)–(iii) with $k = 0$. To prove the other direction, assume (i)–(iii) hold. Then (i) and (iii) imply that $p(x) = \prod_{i=1}^{n} p\big(x_i \,\big|\, \mathrm{pa}(i), \mathbb{1}_{[1,k]}(i)z\big)$ for all z. Also, from (i) it follows that $\big(\mathrm{pa}(i), x_i\big) \perp\!\!\!\perp z$, and hence for all z, $p(x_i \mid \mathrm{pa}(i), z) = p(x_i, \mathrm{pa}(i) \mid z)/p(\mathrm{pa}(i) \mid z) = p(x_i|\mathrm{pa}(i))$, giving $p(x) = \prod_{i=1}^{n} p(x_i|\mathrm{pa}(i))$. Finally, from (ii) we have that the two joints, $p_D(x)$ and $p(x)$ are the same, therefore their marginals and conditionals are the same too, which implies $p_D(x) = \prod_{i=1}^{n} p_D(x_i|\mathrm{pa}(i))$. □

The proposition says that just by specifying CIA for otherwise dependent parts of the data, any model that suffers posterior collapse (in the sense that x and z are independent) will be suboptimal in terms of the model evidence $p(x)$. This gives a degree of assurance that given such a structure, a well-optimized model with high enough capacity will not suffer posterior collapse. Latent variable image models, which model pixels or patches conditionally independently given the latents fall into this category, which explains while posterior collapse is not so prevalent in that case.

Conversely, in theory and in the absence of CIA, there is a trivial latent variable model $p(x, z) = p_D(x)p(z)$ that is optimal in terms of the marginal likelihood $p(x)$ but does not use the latents. The prototypical example for this case is auto-regressive likelihoods, such as RNN language models.

To summarize, *CIA must be made* to guarantee that latents are used given powerful enough models and inference methods. On the other hand, lacking the necessary CIA, we can still bias solutions by changing the objective. One such change to compensate for the lack of model structure is adding a constraint on mutual information.

5.4 MUTUAL INFORMATION AUGMENTED OBJECTIVES

Mutual information is often used as a measure of latent variable usage in trained models or as part of the training objective to control latent usage and reduce underspecification. First, as a measure of latent usage, it is a diagnostic of the inference method. In this role, it is but a proxy for the generalization ability of the model or for the performance on down-stream tasks. Second, as a constraint during training, it can be seen as compensating for the lack of structure in the model. However, its role is not essential in either of these: evaluating representations without the down-stream tasks is fraught with peril, and equipping the model with structure sounds a rather more appealing direction to pursue. Still, finding a good model structure is easier said than

done, and in practice mutual information is useful both as a diagnostic for inference and as a tool for model specification. Thus, we augment the marginal likelihood objective with a mutual information term and maximize

$$\mathop{\mathbb{E}}_{p_D(x)} \ln p(x) + \lambda I_p(X, Z), \tag{5.3}$$

where $p_D(x)$ is the data distribution and $\lambda \in \mathbb{R}, \lambda \geqslant 0$. Throughout, we assume that $I_p(X, Z)$ is bounded, which is satisfied by any model for which the optimization of $p(x)$ is well-posed and hence its $p(x|z)$ is bounded for all z. Also note that for any given $I_p(X, Z)$, the model achieves its global maximum when $p(x) = p_D(x)$, and we can reasonably expect models that suffer from posterior collapse to effectively balance data fit and mutual information.

Motivated by the identity $I_p(X, Z) = \mathbb{E}_{p(x)} \mathrm{KL}(p(z|x) \parallel p(z))$, the mutual information term can be estimated by the average KL divergence. While this true KL is hard to compute in general due to the intractable posterior, the availability of the variational posterior offers the compelling alternative of estimating $I_p(X, Z)$ with

$$I^{q,p}_{p_D}(X, Z) \coloneqq \mathop{\mathbb{E}}_{p_D(x)} \mathrm{KL}\big(q(z|x) \,\big\|\, p(z)\big).$$

Note the shortcut of sampling from $p_D(x)$ instead of $p(x)$. If we plan to use the representations obtained from the variational posterior $q(z|x)$ on some task, then $I^{q,p}_{p_D}$ is a natural quantity to track [Zhao et al., 2019, Rezaabad and Vishwanath, 2020]. However, in this work, our primary concern lies not with artifacts of variational inference but with the model $p(x, z)$ and its ability to capture information in the latents. Moreover, employing $I^{q,p}_{p_D}$ as a proxy objective is problematic because it overestimates I_p as q tends to underestimate the variance of the true posterior.[5] Even worse, with $I^{q,p}_{p_D}$ the quality of $q(z|x)$ would influence our conclusions about latent variable usage. Monte Carlo objectives, whose $q(z|x)$ in itself is no longer a direct approximation to $p(z|x)$ [Mnih and Rezende, 2016], exacerbate the problem with latent usage estimation. Experimentally, we found that, for models trained with Monte Carlo objectives, $I^{q,p}_{p_D}$ can wildly under- or overestimate I_p. Thus, to form a better estimate of I_p, we replace $q(z|x)$ with $p(z|x)$ in $I^{q,p}_{p_D}$:

$$I^{p}_{p_D}(X, Z) \coloneqq \mathop{\mathbb{E}}_{p_D(x)} \mathrm{KL}\big(p(z|x) \,\big\|\, p(z)\big). \tag{5.4}$$

This 'cross' mutual information $I^{p}_{p_D}$ is the average true KL over the data distribution p_D. Averaging over $p_D(x)$ instead of $p(x)$ is common practice as it

[5] For VAEs, this follows from the properties of the KL divergence, while for Monte Carlo objectives in general, it follows from how the looseness of the lower bound relates to the variance of the estimator [Maddison et al., 2017].

allows the mutual information to be estimated based on the current minibatch without sampling from the model in the typical doubly stochastic optimization setting [Titsias and Lázaro-Gredilla, 2014]. The price for efficiency is a possible generalization problem: the average KL may be different over $p_D(x)$ and $p(x)$. Generalization may eventually become a pressing issue, but as our experiments in §5.7 will demonstrate, we first have to deal with underfitting. In addition to being expedient, as we will show in §5.5 and §5.6, this choice makes the single-sample case of our estimators correspond to the representational KL and the β-VAE [Higgins et al., 2017].

Definition 3 (Mutual information augmented objective)**.** *The mutual information augmented objective, which is a combination of the usual marginal likelihood objective and $I^p_{p_D}$, is defined as*

$$\mathcal{O}(\lambda) = \mathop{\mathbb{E}}_{p_D(x)} \ln p(x) + \lambda I^p_{p_D}(X, Z). \tag{5.5}$$

Furthermore, the pointwise version of the objective is defined as

$$\mathcal{O}(\lambda, x) = \ln p(x) + \lambda \operatorname{KL}\big(p(z|x) \,\big\|\, p(z)\big), \tag{5.6}$$

which in turn satisfies $\mathcal{O}(\lambda) = \mathbb{E}_{p_D(x)} \mathcal{O}(\lambda, x)$.

In the following, we propose estimators of $\mathcal{O}(\lambda, x)$ and the true KL within it to estimate $\mathcal{O}(\lambda)$ and the mutual information in a manner suitable for the doubly stochastic optimization setting.

5.4.1 *The KL Objective*

To find the maximum of $\mathcal{O}(\lambda)$ in θ, both of its terms must be estimated well. We delegate the task of estimating the marginal log-likelihood $\ln p(x)$ to a 'base' Monte Carlo estimator of the form $\hat{S}^K(x, z_{1:K}) = \ln\big(\frac{1}{K}\sum_{i=1}^K f(x, z_i)\big)$, where $z_{1:K} = (z_1, \ldots, z_K)$ are independent samples from the proposal distribution $q(z|x)$, and f is some function of the observable and latent variables [Mnih and Rezende, 2016]. Ideally, $\hat{S}^K$ is chosen to have low bias and low variance, allowing optimization to strike a better balance with mutual information. For our first contribution, the KL objective, we rewrite the true KL in a form more amenable to importance sampling:

$$\begin{aligned}\operatorname{KL}\big(p(z|x) \,\big\|\, p(z)\big) &= \mathop{\mathbb{E}}_{p(z|x)} \ln \frac{p(z|x)}{p(z)} \\ &= \mathop{\mathbb{E}}_{p(z|x)} \big[\ln p(x|z)\big] - \ln p(x)\end{aligned}$$

$$= \mathop{\mathbb{E}}_{q(z\,|\,x)}\left[\frac{p(z\,|\,x)}{q(z\,|\,x)}\ln p(x\,|\,z)\right] - \ln p(x)$$
$$= \frac{1}{p(x)}\mathop{\mathbb{E}}_{q(z\,|\,x)}\left[\frac{p(x,z)}{q(z\,|\,x)}\ln p(x\,|\,z)\right] - \ln p(x).$$

Plugging this into the definition of $\mathcal{O}(\lambda, x)$ in (5.6) and grouping the $\ln p(x)$ terms, we get

$$\mathcal{O}(\lambda, x) = (1-\lambda)\ln p(x) + \lambda\frac{1}{p(x)}\mathop{\mathbb{E}}_{q(z\,|\,x)}\frac{p(x,z)}{q(z\,|\,x)}\ln p(x\,|\,z).$$

In this form of the mutual information augmented objective, the first term, $(1-\lambda)\ln p(x)$, can be estimated with the base Monte Carlo estimator $\hat{S}^K$, so we only have the task of dealing the second term

$$\frac{1}{p(x)}\mathop{\mathbb{E}}_{q(z\,|\,x)}\frac{p(x,z)}{q(z\,|\,x)}\ln p(x\,|\,z). \tag{5.7}$$

Here, $p(x)$ could be estimated with the simple K-sample importance sampling estimator

$$\hat{p}^K(x, z_{1:K}) \coloneqq \frac{1}{K}\sum_{i=1}^{K}\frac{p(x,z_i)}{q(z_i\,|\,x)}. \tag{5.8}$$

However, combining $\hat{p}^K$ with an importance sampling estimate of the expectation in (5.7) using different samples begot very high variance in preliminary experiments. Instead, we approximate (5.7) with the self-normalized importance sampling estimator

$$\hat{U}^K(x, z_{1:K}) \coloneqq \hat{p}^K(x, z_{1:K})^{-1}\frac{1}{K}\sum_{i=1}^{K}\frac{p(x,z_i)}{q(z_i\,|\,x)}\ln p(x\,|\,z_i), \tag{5.9}$$

which uses the same samples for estimating $1/p(x)$ and the expectation. This leads to our first estimator for $\mathcal{O}(\lambda, x)$.

Definition 4 (KL objective)**.** *Let $\hat{S}^K$ be any K-sample Monte Carlo estimator of* $\ln p(x)$*. Then the augmented objective $\mathcal{O}(\lambda, x)$ can be estimated by the KL objective*

$$\mathcal{O}_{\mathrm{KL}}(\hat{S}, K, \lambda, x) = \mathop{\mathbb{E}}_{z_{1:K}\sim q(z\,|\,x)}\hat{O}_{\mathrm{KL}}(\hat{S}, K, \lambda, x, z_{1:K}), \tag{5.10}$$

where

$$\hat{\mathcal{O}}_{\mathrm{KL}}(\hat{S}, K, \lambda, x, z_{1:K}) = (1-\lambda)\hat{S}^K(x, z_{1:K}) + \lambda\hat{U}^K(x, z_{1:K}). \tag{5.11}$$

Note that $\hat{\mathcal{O}}_{\mathrm{KL}}(\hat{S}^K, K, \lambda, x, z_{1:K})$ uses the same samples $z_{1:K} \sim q(z\,|\,x)$ to estimate both terms of the augmented objective (5.6). Grouping the terms differently, we can separate out the estimate of the KL:

$$\hat{\mathcal{O}}_{\mathrm{KL}}(\hat{S}, K, \lambda, x, z_{1:K}) = \underbrace{\hat{S}^K(x, z_{1:K})}_{\approx \ln p(x)} + \lambda \underbrace{\left(\hat{U}^K(x, z_{1:K}) - \hat{S}^K(x, z_{1:K})\right)}_{\approx \mathrm{KL}(p(z\,|\,x) \parallel p(z))}. \tag{5.12}$$

We assume throughout that the estimators $\hat{S}^K(x, z_{1:K})$ and $\hat{U}^K(x, z_{1:K})$ have finite variance. The estimators $\hat{\mathcal{O}}_{\mathrm{KL}}(\hat{S}^K, K, \lambda, x)$ of $\mathcal{O}(\lambda, x)$ and $\hat{U}^K(x, z_{1:K}) - \hat{S}^K(x, z_{1:K})$ of $\mathrm{KL}(p(z|x) \parallel p(z))$ have the following properties:

Proposition 9 (properties of the KL objective)**.**

(i) If $\hat{S}^K$ converges in probability or almost surely to $\ln p(x)$ as $K \to \infty$, then so do $\hat{\mathcal{O}}_{\mathrm{KL}}$ and $\hat{U}^K - \hat{S}^K$ to $\mathcal{O}(\lambda, x)$ and $\mathrm{KL}(p(z|x) \parallel p(z))$, respectively.

(ii) If $\mathbb{E}_{z_{1:K} \sim q(z\,|\,x)} \hat{S}^K(x, z_{1:K}) \leqslant \ln p(x)$, then

$$\mathop{\mathbb{E}}_{z_{1:K} \sim q(z\,|\,x)} [\hat{U}^K(x, z_{1:K}) - \hat{S}^K(x, z_{1:K})] \geqslant \mathrm{KL}(p(z|x) \parallel p(z)).$$

That is, the estimator is biased upward.

(iii) The bias of the self-normalized importance sampling estimator $\hat{U}^K$ is bounded if $p(x, z)$ is bounded as shown in Proposition 7 of Metelli et al. [2020].

(iv) The variance of $\hat{U}^K$ decays with K, but unlike in non-normalized importance sampling, for any given K, there is no proposal distribution with which the variance is zero unless $\hat{U}^K$ is constant for all $z_{1:K}$ with probability 1.

Proof. These follow from the properties of self-normalized importance sampling [Owen, 2013] except where noted. □

Note that although the objective is biased upwards, its bias is decreased with more samples and is bounded if $p(x, z)$ is bounded. It can be assumed that $p(x, z)$ is bounded, else the optimization of $\ln p(x)$ is ill-posed even without the mutual information term. Consequently we may be able to rely on λ to counteract the bias thus bounded. Importantly, the computation of $\hat{U}^K$ imposes minimal overhead as it needs to evaluate only $p(x|z_i)$, $p(z_i)$ and $q(z_i|x)$: the same quantities and same z_i as needed for computing $\hat{S}^K$. Referring back to (5.3), we argue that this KL objective allows for effective interpolation between fitting the data and capturing information in the latent variables.

5.4.2 *The Rényi Objective*

In this section, we introduce a second estimator of the augmented objective $\mathcal{O}(\lambda, x)$ (5.6), based on the Rényi divergence, to address a potential issue with the KL objective's estimate (5.11). This issue lies in the fact that the KL objective

linearly combines two estimators ($\hat{S}^K$ and $\hat{U}^K$) of different quantities. How their biases and variances relate deserves some consideration. As we have seen, with $\hat{S}^K$ that underestimate $\ln p(x)$ (e.g. IWAE), $\hat{U}^K - \hat{S}^K$ overestimates the true KL. Luckily, both biases can be reduced with more samples.

However, taking more samples may not help if the two estimators have very different variances, in which case optimizing the objective may be difficult. This issue could be addressed by designing a $\hat{U}^K$ for every $\hat{S}^K$, but this would limit the applicability of our method in practice. Instead, we apply $\hat{S}$ not only to estimate $\ln p(x)$ but *a second time* too to estimate the Rényi divergence, itself a biased estimate of the true KL. As we will see later, this works surprisingly well in practice despite the presence of the bias.

The Rényi divergence between two distributions $f(x)$ and $g(x)$ is defined as $D_\alpha(f \parallel g) = \frac{1}{\alpha-1} \ln \mathbb{E}_{g(x)} f(x)^\alpha g(x)^{-\alpha}$, where α is a positive real number. For $\alpha < 1$, $D_\alpha(f \parallel g) \leqslant \mathrm{KL}(f \parallel g)$, while for $\alpha > 1$, $D_\alpha(f \parallel g) \geqslant \mathrm{KL}(f \parallel g)$. Since $\lim_{\alpha \to 1} D_\alpha(f \parallel g) = \mathrm{KL}(f \parallel g)$), $D_\alpha(f \parallel g)$ can approximate $\mathrm{KL}(f \parallel g)$ arbitrarily closely when α is sufficiently close to 1. This latter property motivates the use of $D_\alpha(p(z|x) \parallel p(z))$ as an approximation to the true KL. To construct an estimator, we first rewrite the Rényi divergence as:

$$\begin{aligned}
(\alpha - 1) D_\alpha\big(p(z|x) \,\big\|\, p(z)\big) &= \ln \mathop{\mathbb{E}}_{p(z)} \frac{p(z|x)^\alpha}{p(z)^\alpha} \\
&= \ln \mathop{\mathbb{E}}_{p(z)} \left[\frac{p(z|x)^\alpha p(x)^\alpha}{p(z)^\alpha} \right] - \alpha \ln p(x) \\
&= \ln \mathop{\mathbb{E}}_{p(z)} \left[p(x|z)^\alpha \right] - \alpha \ln p(x) \\
&= \ln p^\alpha(x) - \alpha \ln p(x), \qquad (5.13)
\end{aligned}$$

where $p^\alpha(x) \coloneqq \mathbb{E}_{p(z)} p(x|z)^\alpha$.

We note in passing that $p^\alpha(x)$ has an intuitive interpretation, particularly when $\alpha \in \mathbb{N}$. Consider the task of modelling the distribution of discrete data over some discrete set $\mathcal{X}$ duplicated α times, that is $p_D^\alpha(x, \ldots, x) \coloneqq p_D(x)$. That is, p_D^α is a probability distribution over $\mathcal{X}^\alpha$, whereas p_D is over $\mathcal{X}$, and $p_D^\alpha(x_1, \ldots, x_\alpha) = 0$ unless all x_i are identical. On this task, $\alpha \ln p(x) = \ln p(x)^\alpha$ acts as the uninformed baseline, in which a separate set of latents is used for each branch $p(x_i|z)$, thus the cost of information in the latents must be paid α times. This means that, based on its alternative form $\ln p^\alpha(x) - \alpha \ln p(x)$ in (5.13), we can interpret the Rényi divergence as a measure of how much better the α-duplicated model $p^\alpha(x)$ does at modelling the duplicated data compared to the worst-case solution $p(x)^\alpha$, which does not use the latents to amortize the cost of encoding the data multiple times.

We now derive a biased approximation to $\mathcal{O}(\lambda, x)$:

$$\begin{aligned}\mathcal{O}(\lambda, x) &= \ln p(x) + \lambda \,\mathrm{KL}\big(p(z|x) \,\big\|\, p(z)\big) \\ &\approx \ln p(x) + \lambda D_\alpha\big(p(z|x) \,\big\|\, p(z)\big) \\ &= \ln p(x) + \frac{\lambda}{\alpha - 1}\big(\ln p^\alpha(x) - \alpha \ln p(x)\big) \\ &= \frac{\lambda}{\alpha - 1} \ln p^\alpha(x) - \left(\frac{\lambda\alpha}{\alpha - 1} - 1\right) \ln p(x).\end{aligned}$$

Definition 5 (Rényi objective)**.** *Let $\lambda, \alpha > 0$, and let $\hat{S}^K(x, z_{1:K})$ and $\hat{S}^K_\alpha(x, z_{1:K})$ be K-sample Monte Carlo estimators for $\ln p(x)$ and $\ln p^\alpha(x)$, respectively. Then the augmented objective $\mathcal{O}(\lambda, x)$ can be estimated by the Rényi objective*

$$\mathcal{O}_R(\hat{S}, K, \lambda, \alpha, x) = \mathop{\mathbb{E}}_{z_{1:K} \sim q(z|x)} \hat{\mathcal{O}}_R(\hat{S}, K, \lambda, \alpha, x, z_{1:K}), \tag{5.14}$$

where

$$\hat{\mathcal{O}}_R(\hat{S}, K, \lambda, \alpha, x, z_{1:K}) = \frac{\lambda}{\alpha - 1}\hat{S}^K_\alpha(x, z_{1:K}) - \left(\frac{\lambda\alpha}{\alpha - 1} - 1\right)\hat{S}^K(x, z_{1:K}). \tag{5.15}$$

Separating out the estimate of the Rényi divergence yields the alternative form

$$\hat{\mathcal{O}}_R(\hat{S}, K, \lambda, \alpha, x, z_{1:K}) = \underbrace{\hat{S}^K(x, z_{1:K})}_{\approx \ln p(x)} + \lambda \underbrace{\left(\frac{1}{\alpha - 1}\left(\hat{S}^K_\alpha(x, z_{1:K}) - \alpha\hat{S}^K(x, z_{1:K})\right)\right)}_{\approx D_\alpha(p(z|x) \,\|\, p(z))}. \tag{5.16}$$

Our goal was to address the mismatched biases and variances of the KL objective's $\hat{S}^K$ and $\hat{U}^K$. Having eliminated $\hat{U}^K$, we are left with only $\hat{S}^K$ and $\hat{S}^K_\alpha$, estimating two closely related quantities, $\ln p(x)$ and $\ln p^\alpha(x)$. Note that $\hat{S}^K_\alpha$ can be obtained with a slight modification of $\hat{S}^K$, as explained in the next section. In §5.7, we validate experimentally that the benefits this scheme affords outweigh the obvious drawback of additional bias in the Rényi objective.

Estimating $p^\alpha(x)$ with IWAE

From $\ln p(x) = \ln\left(\mathbb{E}_{p(z)}\, p(x|z)\right)$ and $\ln p^\alpha(x) = \ln\left(\mathbb{E}_{p(z)}\, p(x|z)^\alpha\right)$, we have that if $\hat{S}^K(x, z_{1:K})$ is computed explicitly in terms of $p(x|z)$, then $\hat{S}^K_\alpha(x, z_{1:K})$ can be derived simply by replacing $p(x|z)$ with $p(x|z)^\alpha$ in $\hat{S}^K(x, z_{1:K})$. All base estimators considered here have this property and we elucidate the derivation through the example of the IWAE.

The ELBO of variational autoencoders can be rather loose, and the variance of the resulting approximate posterior is usually smaller than that of the true posterior. To tackle these issues, Burda et al. [2015] proposed the importance weighted autoencoder (IWAE). Whereas the ELBO is single-sample, the IWAE bound is based on $K \in \mathbb{N}$ samples:

$$\ln p(x) = \ln \mathop{\mathbb{E}}_{p(z)} p(x|z) = \ln \mathop{\mathbb{E}}_{\substack{z_1,\ldots,z_K \\ \sim q(z|x)}} \frac{1}{K} \sum_{i=1}^{K} \frac{p(x|z_i)p(z_i)}{q(z_i|x)}$$
$$\geqslant \mathop{\mathbb{E}}_{\substack{z_1,\ldots,z_K \\ \sim q(z|x)}} \ln \frac{1}{K} \sum_{i=1}^{K} \frac{p(x|z_i)p(z_i)}{q(z_i|x)}.$$

In importance sampling terms, $p(x|z)$ is the integrand, the function whose expectation we want to compute with respect to $p(z)$. While $p(x|z)$ here is a probability mass function, importance sampling does not require this. In fact, we can estimate the expectation of $p(x|z)^\alpha$ analogously:

$$\ln p^\alpha(x) \geqslant \mathop{\mathbb{E}}_{\substack{z_1,\ldots,z_K \\ \sim q(z|x)}} \ln \frac{1}{K} \sum_{i=1}^{K} \frac{p(x|z_i)^\alpha p(z_i)}{q(z_i|x)}. \tag{5.17}$$

To estimate the expectation in the above lower bound, we can recombine $p(x|z_i)$, $p(z_i)$ and $q(z_i|x)$, quantities already computed for the base estimator, with negligible computational overhead.

Note that there is an optimal proposal distribution $q_{\text{opt}}(z|x) = p(x|z)^\alpha p(z)/p^\alpha(x)$ that leads to exactly computing $\ln p^\alpha(x)$ (i.e. estimating it with zero bias and variance). On the other hand, the other term in the Rényi objective is best estimated using, in general, a different proposal distribution. One solution would be to apply separate proposal distributions, another is to choose λ and α such that in $\ln p(x) + \lambda D_\alpha$ the $\ln p(x)$ term is cancelled out, which we explore briefly in §5.4.3 below.

Other Estimators

In the interest of space, we omit re-derivations of further estimators of $\ln p^\alpha(x)$ and their gradient estimators. As in the IWAE case, all we need to show is that the estimators do not depend on properties of $p(x|z)$ that $p^\alpha(x|z)$ does not have, and for performance, that $p(x|z)$ is explicitly computed so that computing $p(x|z)^\alpha$ is cheap. These are true for the estimators used in our experiments: REINFORCE [Williams, 1987, Mnih and Gregor, 2014], VIMCO [Mnih and Rezende, 2016], STL [Roeder et al., 2017] and DReG [Tucker et al., 2018].

5.4.3 *The Power Objective*

Notice that, in the Rényi estimator (5.15), if λ is set to $(\alpha - 1)/\alpha$, then the coefficient of the $\ln p(x)$ term becomes zero, yielding a simpler form.

Definition 6 (Power objective)**.** *We call the Rényi objective with $\lambda = (\alpha - 1)/\alpha$ the* power objective*:*

$$\hat{\mathcal{O}}_P(\hat{S}, K, \alpha, x, z_{1:K}) = \alpha^{-1}\hat{S}^K_\alpha(x, z_{1:K}). \tag{5.18}$$

Note that for optimization the constant α^{-1} can be dropped from the objective and that if $\alpha = 1$, the power objective is equal to the log-likelihood $\ln p(x)$. Next we show that if $\alpha > 1$, maximizing $p^\alpha(x)$ optimizes a lower bound on $p(x)$, and this lower bound is tight when the latents fully determine the observables.

Proposition 10. *Assume that X is concentrated on the countable set $\mathcal{X}$, $\alpha > 1$, and let $x \in \mathcal{X}$ be arbitrary. Then $p^\alpha(x) \leqslant p(x)$ with equality for all $x \in \mathcal{X}$ if and only if $H_p(X|Z) = 0$. Second, $p(x)^\alpha \leqslant p^\alpha(x)$ and $\ln p^\alpha(x) - \alpha \ln p(x) = (\alpha - 1)D_\alpha(p(z|x) \parallel p(z))$.*

Proof. Since X is countable, $p(x|z)$ is discrete, hence $p(x|z) \leqslant 1$ for all x and z. Therefore, since $\alpha > 1$, $p(x|z)^\alpha \leqslant p(x|z)$ with equality if and only if $p(x|z) \in \{0, 1\}$. Thus, $p^\alpha(x) = \mathbb{E}_{p(z)}\, p(x|z)^\alpha \leqslant \mathbb{E}_{p(z)}\, p(x|z) = p(x)$, with equality if and only if $p(x|z) \in \{0, 1\}$ for all x, for almost all z. The latter holds if and only if X is a deterministic function of Z with probability 1, which is equivalent to $H_p(X|Z) = 0$. The second statement of is a direct consequence of (5.13) and both $\alpha - 1$ and the Rényi divergence being non-negative. □

So the power objective with $\alpha > 1$ is an upper bound on the KL (and $I^p_{p_D}$ when averaged over x), but this upper bound nevertheless becomes tight when the latents determine the observable variables (i.e. when $H_p(X|Z) = 0$).

In summary, the power objective has three important differences from the Rényi objective, of which it is a special case. First, with $\alpha > 1$, it is guaranteed to be a lower bound on $\ln p(x)$. Second, since there is only a single quantity, $\ln p^\alpha(x)$, being estimated, the question of using separate $q(z|x)$ proposal distributions to estimate the different terms does not arise. Third, λ and α are tied, so it may be harder to find a good balance between ease of optimization

and low bias. Whether the power objective's benefits outweigh its downsides is investigated in §5.B.3.

5.5 CONNECTION TO THE REPRESENTATIONAL KL

Proposition 11 (Single-sample KL estimate)**.** *In the single-sample case with* $\hat{S}^1(x, z_1) = \ln \frac{p(x,z_1)}{q(z_1|x)}$*, the KL objective's estimate of the true KL in* (5.12) *is the representational KL in expectation:*

$$\mathop{\mathbb{E}}_{z_1 \sim q(z|x)} \left[\hat{U}^1(x, z_1) - \hat{S}^1(x, z_1)\right] = \mathrm{KL}\big(q(z|x) \,\big\|\, p(z)\big).$$

Proof. With $\hat{p}^1(x, z_1) = p(x, z_1)/q(z_1|x)$ from (5.8), we have that

$$\begin{aligned}
&\hat{U}^1(x,z_1) - \hat{S}^1(x, z_1) \\
&\quad = \hat{p}^1(x, z_1)^{-1} \frac{p(x, z_1)}{q(z_1|x)} \ln p(x|z_1) - \ln \frac{p(x, z_1)}{q(z_1|x)} \\
&\quad = \ln p(x|z_1) - \ln \frac{p(x, z_1)}{q(z_1|x)} = \ln \frac{q(z_1|x)}{p(z_1)}.
\end{aligned}$$

□

So not only are the expectations the same, but our single-sample estimate is the same as the trivial single-sample estimate of the representational KL. We now prove a similar result for the Rényi objective.

Proposition 12 (Single-sample Rényi estimate)**.** *In the single-sample case with* $\hat{S}^1(x, z_1) = \ln \frac{p(x,z_1)}{q(z_1|x)}$*, the Rényi objective's estimate of* $D_\alpha p(z|x)p(z)$ *in* (5.16) *is the representational KL in expectation:*

$$\mathop{\mathbb{E}}_{z_1 \sim q(z|x)} \left[\frac{1}{\alpha - 1}\left(\hat{S}^1_\alpha(x, z_1) - \alpha \hat{S}^1(x, z_1)\right)\right] = \mathrm{KL}\big(q(z|x) \,\big\|\, p(z)\big).$$

Proof. Let us rewrite the expression in the expectation.

$$\begin{aligned}
&\frac{1}{\alpha - 1}\left(\hat{S}^1_\alpha(x, z_1) - \alpha \hat{S}^1(x, z_1)\right) \\
&\quad = \frac{1}{\alpha - 1}\left(\ln \frac{p(x|z_1)^\alpha p(z_1)}{q(z_1|x)} - \alpha \ln \frac{p(x, z_1)}{q(z_1|x)}\right) \\
&\quad = \frac{1}{\alpha - 1}\left(\ln \frac{p(z_1)}{q(z_1|x)} - \alpha \ln \frac{p(z_1)}{q(z_1|x)}\right) \\
&\quad = \ln \frac{q(z_1|x)}{p(z_1)}
\end{aligned}$$

□

In the single-sample case, where $q(z|x)$ approximates the true posterior much better compared to the multi-sample Monte Carlo estimators, the representational KL approximates the true KL with a relatively small bias. This bias is, however, asymptotically eliminated by our KL estimator as the number of samples K grows (and reduced for our Rényi estimator), which significantly improves the usage of the latent variables. This is demonstrated in our experiments (e.g. Figures 5.4 and 5.7) where we compare methods using KL estimates based on different number of samples from $q(z|x)$.

5.6 CONNECTION TO THE β-VAE

In parallel to the single-sample case of our estimators, we now show that the β-VAE [Higgins et al., 2017] objective defined as

$$\mathcal{O}_{\beta\text{-VAE}}(\beta, x) = \mathop{\mathbb{E}}_{z_1 \sim q(z|x)} \big[\ln p(x|z_1)\big] - \beta \,\mathrm{KL}\big(q(z|x) \,\big\|\, p(z)\big) \tag{5.19}$$

is equal to the KL, Rényi and power objectives with suitably chosen parameters.

Proposition 13 (β-VAE equivalence)**.** *In the single-sample case with* $\hat{S}^1(x, z_1) = \ln \frac{p(x,z_1)}{q(z_1|x)}$*, the* β*-VAE and our single-sample objectives are equal with a suitable choice of* λ *or* α*:*

$$\begin{aligned}
\mathcal{O}_{\beta\text{-VAE}}(\beta, x) &= \mathcal{O}_{\mathrm{KL}}(\hat{S}, 1, 1-\beta, x) && (\lambda = 1-\beta)\\
&= \mathcal{O}_{R}(\hat{S}, 1, 1-\beta, \alpha, x) && (\lambda = 1-\beta)\\
&= \mathcal{O}_{P}(\hat{S}, 1, \beta^{-1}, x) && (\alpha = \beta^{-1}).
\end{aligned}$$

Proof. We prove the KL objective case from (5.12) and Proposition 11:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{z_1 \sim q(z|x)} \hat{\mathcal{O}}_{\mathrm{KL}}(\hat{S}, 1, \lambda, x, z_1)\\
&\quad= \mathop{\mathbb{E}}_{z_1 \sim q(z|x)} \big[\hat{S}^1(x, z_1)\big] + \lambda \mathop{\mathbb{E}}_{z_1 \sim q(z|x)} \big[\hat{U}^1(x, z_1) - \hat{S}^1(x, z_1)\big]\\
&\quad= \mathop{\mathbb{E}}_{z_1 \sim q(z|x)} \left[\ln \frac{p(x, z_1)}{q(z_1|x)}\right] + \lambda \,\mathrm{KL}\big(q(z|x) \,\big\|\, p(z)\big)\\
&\quad= \mathop{\mathbb{E}}_{z_1 \sim q(z|x)} \big[\ln p(x|z_1)\big] + (\lambda - 1)\,\mathrm{KL}\big(q(z|x) \,\big\|\, p(z)\big).
\end{aligned}$$

The proof for the Rényi case follows a similar route, starting from (5.16) and using Proposition 12. Finally, for the power objective, $\lambda = (\alpha - 1)/\alpha$, so $\alpha = \beta^{-1}$ recovers $\lambda = 1 - \beta$. □

The goal of our experiments is to demonstrate the difficulty of inference with VAEs and Monte Carlo objectives and to evaluate the proposed methods. Our results indicate the presence of a severe tradeoff between data fit and latent variable usage. We emphasize that, for progress to be made, the choice of evaluation method must acknowledge the existence of this tradeoff. In this work, evaluation is performed in terms of Pareto frontiers of data likelihood vs latent usage curves; reporting a single, best data likelihood would always pick the point with zero latent usage. Results with our proposed estimators, either with continuous latents and DReG or discrete latents and the VIMCO base estimator, markedly improve on their baselines, which do not have multiple samples or cannot use them as efficiently. The improvement is especially significant with discrete latents.

Instead of trying to improve the predictive performance directly, first we demonstrate the difficulty of inference on a simple, synthetic data set and a model to which posterior collapse comes easy. These experiments focus exclusively on data fit to highlight the tradeoff against latent usage. After the experiments on synthetic data, we move on to language modelling with recurrent networks, a very hostile application for VAEs [Bowman et al., 2015].

5.7.1 *Experiments with Synthetic Data*

Every data point is a single symbol drawn from a discrete uniform distribution over a vocabulary of 10000 symbols. The optimal solution is to assign probability 1/10000 to each symbol, which can be trivially satisfied by a simple model that ignores the latents. Note that $\lambda > 0$ makes such solutions suboptimal. In fact, for all priors such that $H(Z) \geqslant H(X)$, the optimal solution must capture all information in the latents (i.e. $I(X, Z) = H(X)$). Our goal with these experiments is to demonstrate the difficulty of fitting the data while using the latents.

Model Architecture

The encoder implementing q assigns an embedding [Mikolov et al., 2013] to each word in the vocabulary, feeds the embedding to a two-layer, densely connected neural network with tanh non-linearities. For continuous latents, $q(z|x)$ follows an isotropic normal distribution $\mathcal{N}(f(o), \text{diag}(\exp(g(o))))$, where f and g are affine transformations parameterized as densely connected layers, and o is the

output of the last encoder layer for a given x. For categorical latents, $q(z|x)$ is proportional to $\exp(o)$ (i.e. it is a categorical distribution parameterized with $\text{softmax}(o)$). The decoder is similar to the encoder but with a final softmax layer. In the decoder, values of continuous latents are fed directly as input to the first layer, but values of categorical latents are first embedded. To compensate for this discrepancy, decoders with categorical latents have only one hidden layer. All embeddings and hidden layers are of size 128.

Evaluation Methodology

Since there is no single number to summarize the tradeoff between latent usage and the quality of the model, we plot the model's latent usage and the expected negative log-likelihood (NLL). The NLL of optimal solutions is the entropy of the data distribution, $\ln 10000 \approx 9.210$. The models' NLLs are estimated using IWAE with 100 samples. We quantify latent variable usage as the mutual information $I_p(X, Z)$, estimated as the average $\hat{U}^K(x, z_{1:K}) - \ln \hat{p}^K(x, z_{1:K})$ (5.9) with $K = 100$ samples. We validated empirically that the variance of these estimates is small (< 0.01) over all feasible latent usage values. Zero latent usage corresponds to posterior collapse.

These plots (like Figure 3b in Alemi et al. 2017) carry the same information as rate-distortion curves, which can be recovered by subtracting the rate I_p from the NLL. Only the measurements on the Pareto frontier are shown, and we tune hyperparemeters, such as the learning rate and λ, of the augmented objective (5.5) to push the Pareto frontier towards more efficient latent usage. See §5.c for details.

Overview of Results

Results for continuous and discrete latents are presented in separate sections, following a similar progression:

- We first show that the choice of base estimator matters: the variance issues of IWAE and REINFORCE limit the potential of the proposed method.
- Next, with DReG and VIMCO we present clear improvements over the single-sample baselines.
- Finally, as an ablation study, we verify that using multiple samples in both the marginal likelihood and the mutual information terms of the objective is necessary for best performance.

We also note that without augmenting the objective with a mutual information

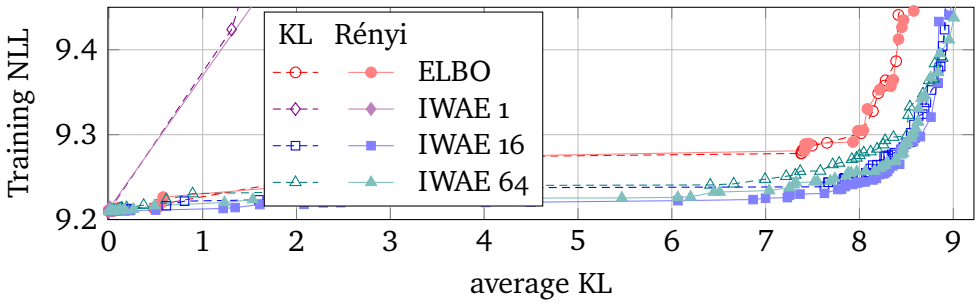


Figure 5.2: KL and Rényi objectives (empty and full markers) on continuous synthetic data with base estimators ELBO and IWAE N, where N is the number of samples used for estimating both $p(x)$ and $\mathrm{KL}(p(z|x) \parallel p(z))$. Here and in the following plots, without augmenting the objective with the mutual information term, models tend to fit the data perfectly with negligible latent usage. These degenerate curves consisting of a single point are omitted from the plots. Also, with base estimators ELBO and IWAE 1, our estimators are equivalent to a β-VAE with analytical and estimated KL (see §5.6), respectively.

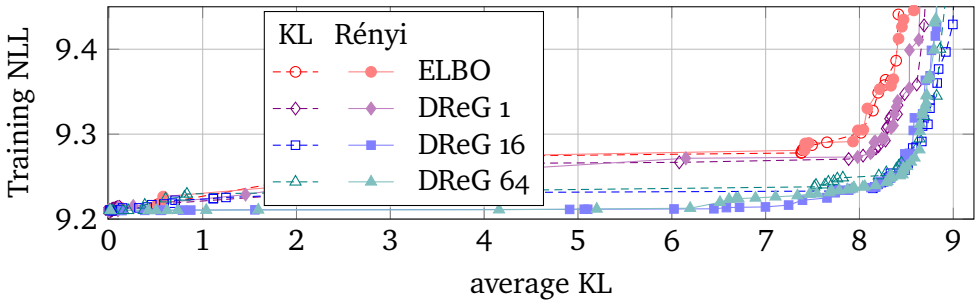


Figure 5.3: KL and Rényi objectives on continuous synthetic data with base estimators ELBO, DReG.

term, our model tends to fit the data perfectly with negligible latent usage. These degenerate curves consisting of a single point are omitted from the plots.

Continuous Latents

With 40 continuous latents and an isotropic standard normal prior, the base estimator is either the standard ELBO, IWAE or IWAE-DReG. In the ELBO, the KL term (5.2) is computed analytically, while in IWAE 1, the single sample from the latents is used to estimate it. An improvement to IWAE 1 is the STL estimator from Roeder et al. [2017], which removes a zero-expectation term from the objective and whose gradients have zero variance when the variational approximation is exact. IWAE–DReG, whose objective is also identical to that of

IWAE, is a generalization of STL to multiple samples, fixing the signal-to-noise problem of gradient estimates of IWAE [Rainforth et al., 2018], wherein the magnitude of the gradient decreases faster with more samples than the variance of the gradient estimates. Although DReG can also be applied to Reweighted Wake-Sleep [Bornschein and Bengio, 2014] and Jackknife Variational Inference [Nowozin, 2018], we only use IWAE–DReG in our experiments and let DReG stand for IWAE–DReG without ambiguity.

Our results in Figures 5.2 and 5.3 are in agreement with these previous works. In the context of this work, our findings can be summarized as follows.

- Compared to the theoretical optimum, a horizontal line at $\ln 10000 \approx 9.210$, all Pareto curves slope increasingly upwards with more latent usage.
- Higher-variance estimators have steeper curves than lower variance estimators, which is most apparent in the contrast between two of our baselines, IWAE 1 and DReG 1.
- The single sample estimators ELBO, IWAE 1, DReG 1 – which are equivalent to a β-VAE (§5.6) – are less efficient in their latent usage than our proposed multi-sample estimators.
- The Rényi objective performs slightly better than the KL objective with the same base estimator.

In more detail, Figure 5.2 shows that both of our estimators perform much worse with IWAE 1 as the base estimator than with the ELBO. As the number of samples grows, this is reversed, although with a high number of samples we do see the predicted degradation [Rainforth et al., 2018].

Next, Figure 5.3 confirms DReG's advantage over the IWAE, performing more efficiently than the ELBO even with a single sample. Our multi-sample objectives both improve on the stronger baseline DReG 1 represents. The Rényi objective outperforms the KL objective by a small but consistent margin before all estimators start degenerating quickly nearing the maximal average KL possible.

In Figure 5.4, we take a closer look at the best performing DReG 16 base estimator to determine whether the improvements are due to a better estimate of the marginal likelihood $\ln p(x)$, or the mutual information term $I^p_{p_D}(X, Z)$ in (5.5). DReG 16/1 and DReG 1/16 (which use multiple samples to estimate only the marginal likelihood or the mutual information, respectively) are less efficient than DReG 16, sometimes being outperformed even by DReG 1. Out of the two, DReG 1/16 performs better, indicating that the variance of the base estimator DReG 1 is low and the problem lies with the mutual information estimate. These results support not only our observations in §5.4 on the unsuitability of $I^{q,p}_{p_D}$ as an estimate of I_p in the multi-sample case (DReG 16/1), when $q(z|x)$ is not a

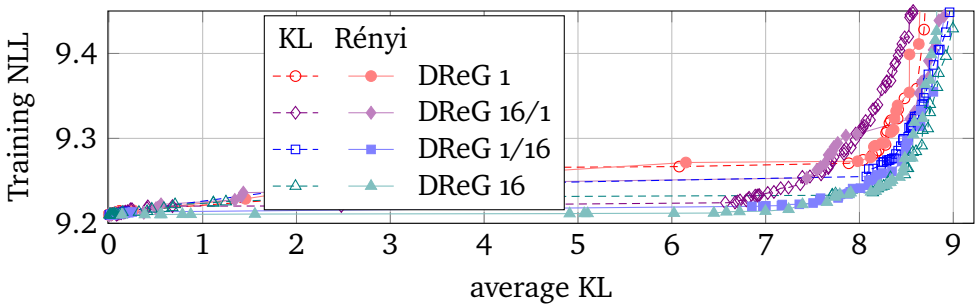


Figure 5.4: KL and Rényi objectives on continuous synthetic data with base estimator DReG. DReG 1 uses a single sample to estimate both $p(x)$ and $\mathrm{KL}(p(z|x) \parallel p(z))$. DReG 16/1 uses 16 samples to estimate $p(x)$ and 1 sample for the KL. DReG 1/16 uses 1 sample to estimate $p(x)$ and 16 samples for the KL. Finally, DReG 16 uses the same 16 samples for both terms.

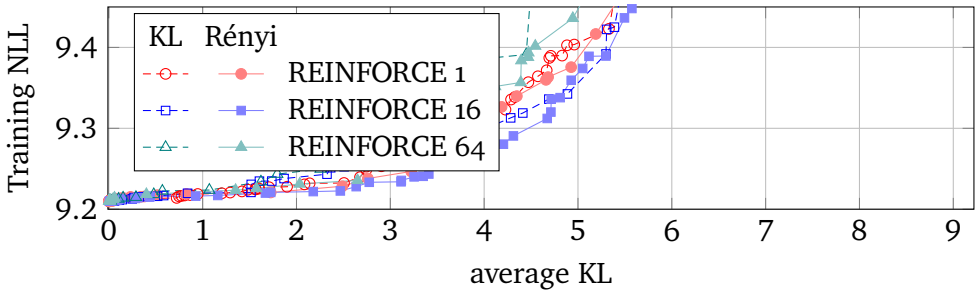


Figure 5.5: KL and Rényi objectives on discrete synthetic data with base estimator REINFORCE.

good approximation of $p(z|x)$, but also demonstrate that $I_{p_D}^{q,p}$ can be improved by taking multiple samples even when $\ln p(x)$ is estimated with a single sample and $q(z|x)$ better approximates the true posterior $p(z|x)$.

Discrete Latents

Next we performed experiments with 8 categorical latent variables, each with a uniform prior over 10 categories. Using the high-variance REINFORCE base estimator (Figure 5.5), we could only get a small improvement over the single-sample case with 16 samples and a similar degradation with 64.

However, with the much lower variance VIMCO estimator, we achieved almost perfect results (see Figure 5.6) with 16 samples. For our purposes, it suffices to think of VIMCO as IWAE applied to discrete latents with lower variance gradients. It may be surprising that these results are even better than with continuous latents, especially at near-maximal latent usage. To intuit why,

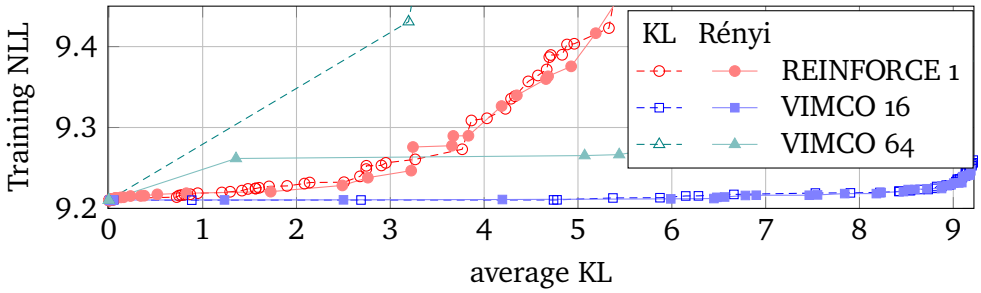


Figure 5.6: KL and Rényi objectives on discrete synthetic data with base estimator VIMCO. The single-sample VIMCO is equivalent to REINFORCE.

consider the minimum variance posterior achievable for a given level of average KL. For discrete latents, a hard posterior (i.e. of zero variance) is possible depending on the number of latents and categories. For continuous latents, the posterior can never be hard: the mutual information determines a lower bound on the average variance of the posterior.

Increasing the number of samples further to 64 made results much worse, indicating a potential issue with VIMCO, which may be similar to the signal-to-noise issue that DReG addresses, but this is beyond the scope of the present work. Note that there is no single-sample VIMCO since its baseline for the contribution of a sample is computed as the average over the rest of the samples, which is undefined. Assuming a zero baseline, we recover REINFORCE 1, which we use for comparison wherever VIMCO 1 would be needed.

Similarly to Figure 5.4 in the continuous case, in Figure 5.7, we try to tease apart the contributions of using multiple samples for estimating the marginal likelihood and mutual information terms of (5.5). The results are much more pronounced here. Both VIMCO 16/1 and VIMCO 1/16 improve significantly on REINFORCE 1 but only with the Rényi objective. Still, both fall way short of VIMCO 16, which employs multiple samples for both terms. Again, these results support not only our observations in §5.4 on the unsuitability of $I_{p_D}^{q,p}$ as an estimate of I_p in the multi-sample case (DReG 16/1), when $q(z|x)$ is not a good approximation of $p(z|x)$, but also demonstrate that $I_{p_D}^{q,p}$ can be improved by taking multiple samples even when $\ln p(x)$ is estimated with a single sample and $q(z|x)$ better approximates the true posterior $p(z|x)$.

In §5.A, we also present results for the VQ-VAE [van den Oord et al., 2017] *without* augmenting its objective with a mutual information term. Since the KL cost in VQ-VAEs is determined by the latent space and is fixed during training, we tune the number of latent variables and the number of categories to control

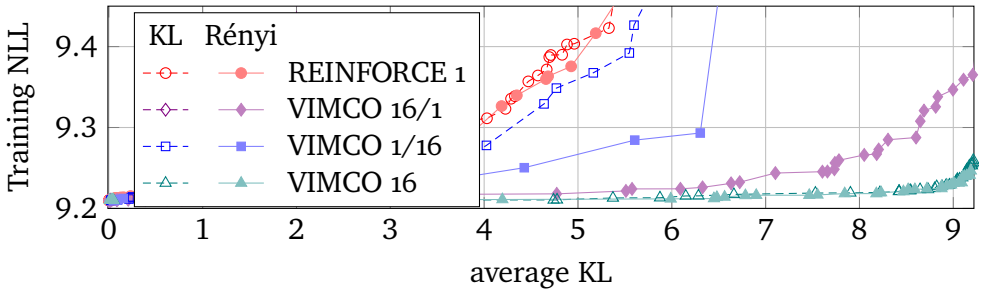


Figure 5.7: KL and Rényi objectives on discrete synthetic data with base estimator VIMCO. REINFORCE 1 (standing in for VIMCO 1) uses a single sample to estimate both $p(x)$ and $\mathrm{KL}(p(z|x) \parallel p(z))$. VIMCO 16/1 uses 16 samples to estimate $p(x)$ and 1 samples for the KL. VIMCO 1/16 uses 1 sample to estimate $p(x)$ and 16 samples for the KL. Finally, VIMCO 16 uses the same 16 samples for both terms.

the mutual information. Results of the VQ-VAE are better than REINFORCE but much worse than VIMCO 16 with either the KL or the Rényi objective.

5.7.2 *Language Modelling Experiments*

One of the most challenging applications of variational autoencoders is language modelling with per-sentence latents, as found by Bowman et al. [2015]. They recognize the generality of the underspecification issue and attribute the increased difficulty to 'the sensitivity of the LSTM to subtle variations in its hidden state as introduced by the sampling process'. In this section, we first show that the data fit vs latent usage tradeoff is even more pronounced in the language modelling case than on the synthetic task, then confirm that the proposed estimators improve validation set results in terms of the Pareto frontiers. Once again, the improvement is strongest with discrete latents.

Data Set

We do sentence-level language modelling on the Penn Treebank (PTB) corpus by Marcus et al. [1993] with preprocessing from Mikolov et al. [2010]. Our goal here is to compare inference methods, not to establish a new state of the art, so to reduce the computational burden brought about by hyperparemeter tuning, we truncated sentences to 35 tokens in both the training and validation sets, with a reduction of 3% in the total number of tokens. This truncation is non-standard.

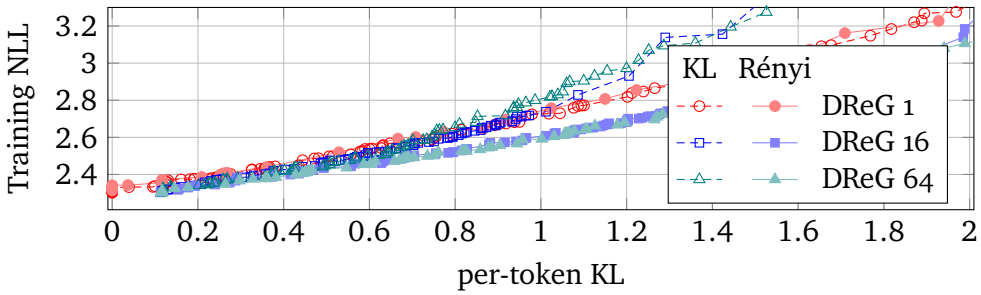


Figure 5.8: KL and Rényi objectives (empty and full markers, respectively) on Penn Treebank with base estimators DReG N, where N is the number of samples used for estimating both $p(x)$ and $\mathrm{KL}(p(z|x) \parallel p(z))$.

Model Architecture

The model architecture is like in the synthetic case except the encoder embeds the input tokens and feeds them to a one-layer, bidirectional LSTM [Hochreiter and Schmidhuber, 1997b] and the output o (from which the parameters of the variational posterior $q(z|x)$ are computed) is the average of the last states of LSTMs corresponding to the two directions. There is a fixed number of latents for all sentences, regardless of their length. The decoder is a unidirectional LSTM whose input is the embedding of the previous token plus the values of the latent variables. Unless stated otherwise, the embedding and hidden sizes are all 128. Similarly to the synthetic case, we use either 40 real-valued latents with an isotropic standard normal prior or 8 categorical latent variables, each with a uniform prior over 10 categories.

Evaluation Methodology

Following previous works, both the reported NLL and the average KL values are averages over tokens in the data set. Since there is only a single set of latents per sentence, this means that the average sentence-level KL is about 21 (the average number of tokens per sentence) times larger. For expediency, only the base estimators that performed best on the synthetic data set are considered, leaving us with DReG for continuous, and VIMCO for discrete latents. Training and validation NLLs are estimated using IWAE with 100 samples. Test NLLs in Table 5.1 are estimated using IWAE with 500 samples.

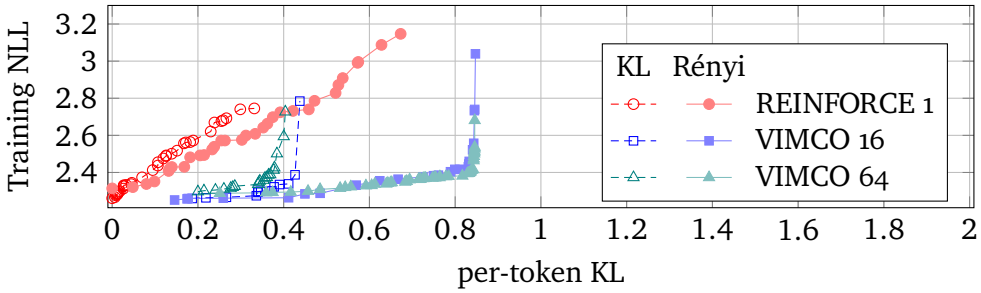


Figure 5.9: KL and Rényi objectives on PTB with base estimator VIMCO.

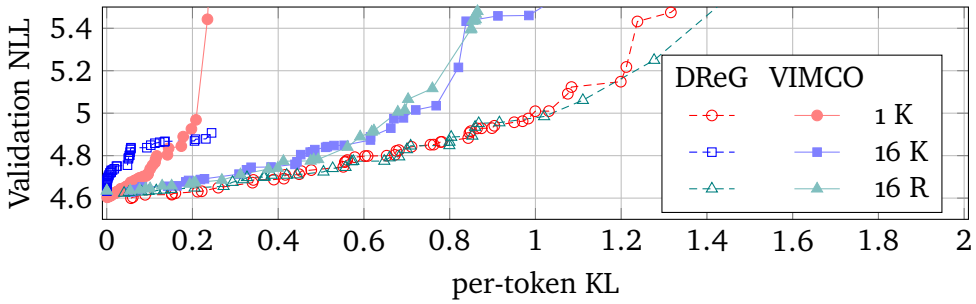


Figure 5.10: Validation NLL with naive dropout using DReG and VIMCO on PTB. The row with *16 K* refers to 16-sample DReG or VIMCO with the KL objective, while *R* stands for the Rényi objective.

Training Fit

In terms of training fit, the language modelling results are similar to those of the synthetic case. In Figure 5.8, there is improvement in the continuous case with multiple samples, initially with both the KL and the Rényi objectives but only with the Rényi at high latent usage.

Figure 5.9 shows the discrete latents case. Once again, with either the KL or the Rényi objective, the results improve markedly, outperforming the continuous models. Note that the KL range is limited by the entropy of the latent space at around $\ln(10^8)/21 \approx 0.87$. Furthermore, as §5.B.2 shows, improvements in training fit require multiple samples in both terms, even more so than in the experiments on the synthetic data earlier.

Validation and Test Results

Our initial results on the validation set were not very positive. We tuned hyperparameters on the validation set, where following Melis et al. [2017],

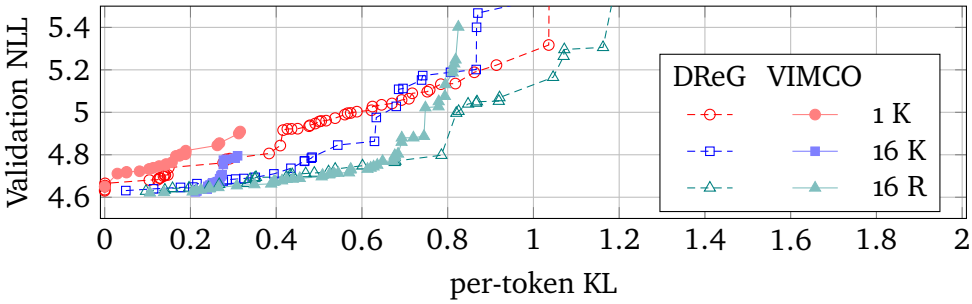


Figure 5.11: Validation NLL with L2 regularization using DReG and VIMCO on PTB.

we introduced three additional hyperparameters in the decoder: the rate of dropout applied to the input embedding, the recurrent state, and the output embedding. As Figure 5.10 shows, with this strong form of regularization, the multi-sample results brought no improvement with continuous latents. With discrete latents, the results improved but not nearly to the extent observed on the training set, and discrete latents performed considerably worse than their continuous counterparts.

We suspected that the stochasticity from dropout may accentuate the problems of variational inference. To verify, we repeated the validation experiment with L2 regularization instead of dropout (see Figure 5.11). In this setting, the validation results are more consistent with training fit. As before, multiple samples significantly improve efficiency. Unlike the training fit though, validation NLL with discrete latents degrades faster than with continuous ones. These results suggest that the additional stochasticity of dropout indeed poses a challenge. On the other hand, with only L2 regularization, the best NLL is higher than with dropout. This translates to a few perplexity points, which may seem small on the graphs presented here, but in language modelling, kingdoms have been won or lost on such differences [Merity et al., 2017].

To have the best of both worlds, the best NLL of dropout and the Pareto curves of L2 regularization, we went back to dropout, but this time we tried using the same dropout mask for all latent samples belonging to the same sentence in a minibatch. As Figure 5.12 shows, this change was successful, and we observed that our proposed estimators improve latent usage for both continuous and discrete latents, and discrete and continuous latents are on par up to an average KL of about 0.5. This constitutes a significant advance in modelling with discrete latents.

Contrary to results of Pelsmaeker and Aziz [2019], we did not find that introducing latent variables significantly improves outright perplexity. As Table 5.1

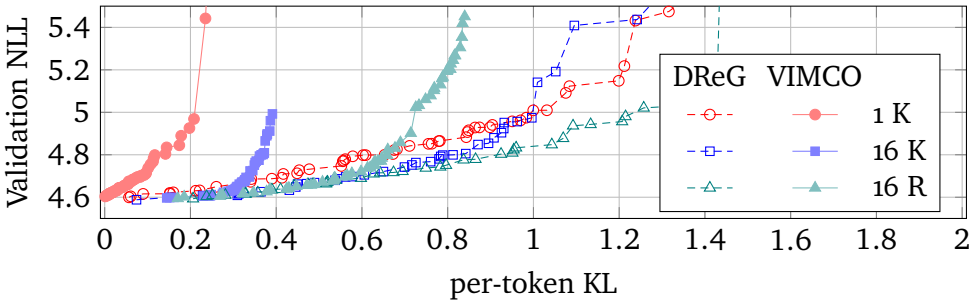


Figure 5.12: Validation NLL with DReG and VIMCO on PTB, using the same dropout mask for all latent samples belonging to the same sentence in a minibatch.

Table 5.1: Best test results for DReG and VIMCO with 1 or 16 samples and the KL or the Rényi objective in the continuous and discrete cases, estimated with IWAE and 500 samples. These optima are at low (or, in the single-sample case, mostly negligible) latent usage.

		DReG			VIMCO		
		1	16		1	16	
		K / R	K	R	K / R	K	R
hs=128	Perplexity	92.5	91.7	91.6	94.6	92.1	92.5
	NLL	4.527	4.518	4.517	4.549	4.523	4.527
	Per-token KL	0.047	0.131	0.126	0.000	0.164	0.120
hs=256	Perplexity	83.9	82.8	82.2	84.7	83.4	83.1
	NLL	4.429	4.416	4.409	4.439	4.424	4.420
	Per-token KL	0.000	0.140	0.138	0.001	0.167	0.094

shows, the best test perplexity improves only slightly with more samples for this combination of a small model and strong regularization.

We also performed experiments with the power objective (§5.B.3) to better understand the tradeoff it represents. We found that its trivial implementation cost comes at the price of decreased efficiency relative to the general Rényi objective, of which it is a special case.

In closing, we would like to emphasize the importance of the results in Figure 5.12. It is not only that small and large improvements have been made, but that the evaluation throughout focussed on the apparent tradeoff between data fit and latent usage. Every point on the Pareto curves is the result of tuning several hyperparemeters: the learning rate, λ, α for the Rényi objective, and three different dropout rates. These curves capture and communicate what most published experiments do not and what single numbers (e.g. in Table 5.1)

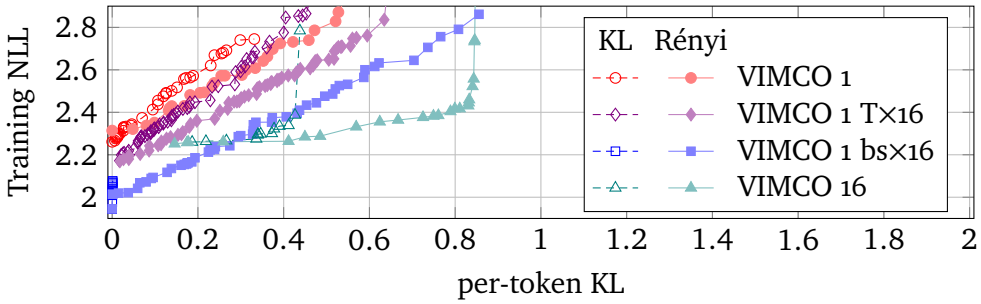


Figure 5.13: Training NLL on PTB with KL and Rényi objectives and base estimator VIMCO. VIMCO 1 T×16 is trained 16 times longer. VIMCO 1 bs×16 has a 16 times larger batch size. While their best NLL at very low latent usage is lower than that of VIMCO 16, they lose this advantage at higher latent usage.

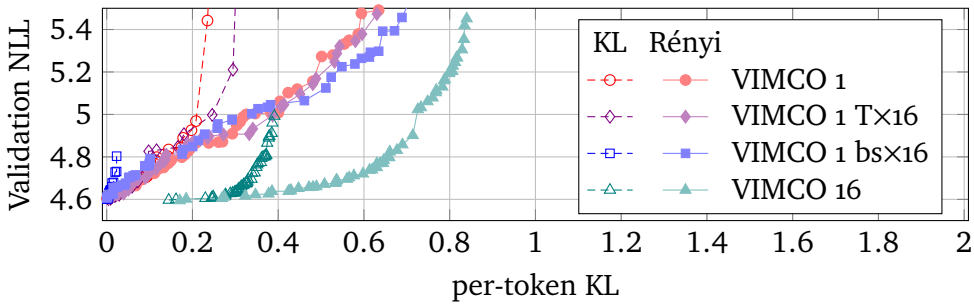


Figure 5.14: Validation NLL on PTB with KL and Rényi objectives and base estimator VIMCO. VIMCO 1 T×16 is trained 16 times longer. VIMCO 1 bs×16 has a 16 times larger batch size. VIMCO 16 performs better than either.

cannot: a reliable comparison of a latent variable model to a strongly regularized baseline over a wide range of latent usage.

Single- vs multi-sample at the same computational budget

The argument for using more samples in Monte Carlo objectives is that their lower bound is tighter, resulting in a better data fit vs latent usage tradeoff. Having equipped multi-sample Monte Carlo objectives with mutual information constraints, we have indeed demonstrated improvements in this tradeoff. Here, we present additional experiments to answer whether performing more computation with a single-sample estimator can compensate the advantages of multiple samples. To this end, we compare 16-sample to single-sample estimators, where the latter are trained for 16 times more optimization steps (*T×16*) or with a batch size that is 16 times larger (*bs×16*) to make the computational cost more equal. In fact, these perform an equal number of gradient calculations

but more forward computation as they evaluate $q(z|x)$ 16 times more than the 16-sample Monte Carlo estimator.

In Figure 5.13, we observe that with more computation, the best training fit of VIMCO 1 improves beyond that of VIMCO 16, but the tradeoff remains severe, as evidenced by that the shapes of the Pareto curves hardly change from the single-sample baseline.

With regards to validation results, we expected the strong regularization provided by tuning three different dropout rates to compensate for possible regularization effects of fewer optimization steps and smaller batch sizes. Consequently, all single-sample estimators shall exhibit similar validation results at zero latent usage. Figure 5.14 confirms this, and shows that the steeper training curves translate to steeper validation curves, indicating that increasing the computation does not address the inefficiency of the inference method, and the apparent improvement in training comes at the cost of more overfitting. In §5.B.1, Figure 5.24 and Figure 5.25 tell a similar story for DReG, but the results are less conclusive there since the curves are much closer.

5.8 CONCLUSIONS

§ Posterior collapse is a common failure mode of density models trained as variational autoencoders, wherein they model the data without relying on their latent variables, rendering these variables useless. We focussed on two factors contributing to posterior collapse, which have been studied separately in the literature. First, the underspecification of the model, which in an extreme but common case allows posterior collapse to be the theoretical optimum. Second, the looseness of the variational lower bound and the related underestimation of the utility of the latents. We weaved these two strands of research together, specifically the tighter bounds of multi-sample Monte Carlo objectives and constraints on the mutual information between the observable and the latent variables. The main obstacle was that the usual method of estimating the mutual information as the average Kullback–Leibler divergence between the easily available variational posterior $q(z|x)$ and the prior does not work with Monte Carlo objectives because their $q(z|x)$ is not a direct approximation to the model's true posterior $p(z|x)$. Hence, we constructed estimators of the Kullback–Leibler divergence of the true posterior from the prior by recycling samples used in the objective, with which we trained models of continuous and discrete latents at much improved rate-distortion and no posterior collapse. While alleviated, the tradeoff between modelling the data and using the latents

still remains, and we urge for evaluating inference methods across a range of mutual information values.

We showed that the representational KL, often used in mutual information constraints, corresponds to the single-sample version of our estimators. Taking more samples both tightens the lower bound and reduces the variance of the estimate of the true KL. Our experimental results support these theoretical predictions and underline the need to use multiple samples for both terms of the objective.

Recognizing that the problem of underfitting becomes more acute with increased latent usage, we emphasized evaluating estimators on the training set, where regularization does not cloud the picture, instead of going outright for improvements in held-out performance. In addition, evaluation was performed in terms of the Pareto frontier of negative log-likelihood vs latent usage curves since reporting a single number cannot capture the tradeoff between the two quantities.

The results demonstrated increased efficiency in latent usage on both the synthetic and language modelling tasks. For discrete latent spaces in particular, the improvements have been dramatic: from a very weak baseline, data fit improved beyond that of models with continuous latents on both data sets. In terms of validation results on Penn Treebank, the best continuous and discrete models and estimators are closely matched up until a significant, per-token KL of 0.5 (about 10.5 as a per-sentence KL).

This work is towards opening the door to making the latents truly useful. We believe that substantial gains in generalization and utility in down-stream tasks can be achieved by shaping the latent space. It also remains to be explored how to best encode the true posterior as the representation for use in down-stream models since the approximate posterior is no longer a suitable choice in the context of Monte Carlo objectives.

In summary, our Mutual Information constrained Monte Carlo Objectives (MICMCOs) help achieve a better tradeoff between modelling the data and using the latent variables to drive the generative process: a prerequisite for fulfilling the promise of generative modelling. This tradeoff is still quite severe though, and there is a lot of room for improvement.

APPENDICES

5.A ADDITIONAL EXPERIMENTS ON SYNTHETIC DATA

Since the KL cost in VQ-VAEs is determined by the latent space and is fixed during training, we tune the number of latent variables and the number of categories to control the mutual information. As Figure 5.15 shows, the VQ-VAE is generally better than REINFORCE 1 but quite far from the optimum and much worse than VIMCO 16 with either the KL or the Rényi objective.

5.B ADDITIONAL LANGUAGE MODELLING EXPERIMENTS

5.B.1 *Robustness*

We performed additional experiments to verify that our results about the relative merits of the estimators are robust to different choices of batch size, number of parameters and optimization length. With continuous latents, we focussed on DReG, the best performing base estimator and varied the batch size (Figure 5.16), the length of optimization (Figure 5.17), and the length of optimization again at two times the model size (Figure 5.18). For VIMCO, the best performing base estimator for the discrete latents, Figures 5.19 to 5.21 tell a similar story. In most cases, we observed that varying these nuisance factors only shifted the Pareto curves downwards or upwards, leaving their relative positions the same. The exception is Figure 5.18, DREG with 256 hidden units, where the differences between the three estimators are very small.

Finally, similar to those in §5.7.2, we present experiments with single-sample DReG at an increased computational budget. As training (Figure 5.24) and validation (Figure 5.25) results show, the overall picture may be the same as for VIMCO, (that is, steeper training curves translate to steeper validation curves), but it is harder to assess this since the curves are closer.

5.B.2 *Asymmetric Samples*

We now investigate whether the improvements are due to a better estimate of the marginal likelihood $\ln p(x)$, or the mutual information term $I^p_{p_D}(X, Z)$ in (5.5). Figures 5.22 and 5.23 show that improvements in training fit require

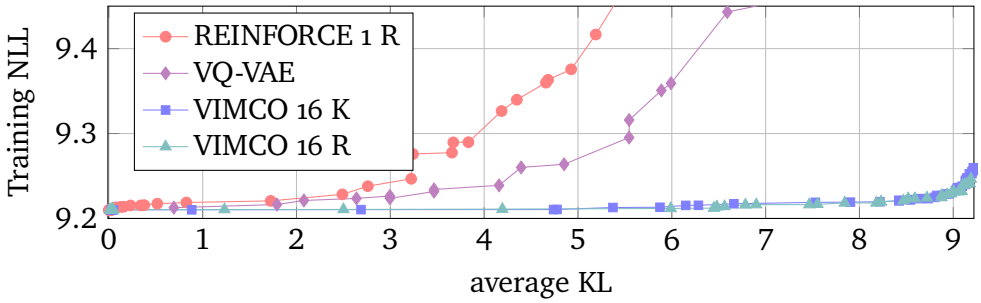


Figure 5.15: VQ-VAE on synthetic data with tuned latent sizes compared to REINFORCE 1 and VIMCO 16 with the KL and Rényi objectives.

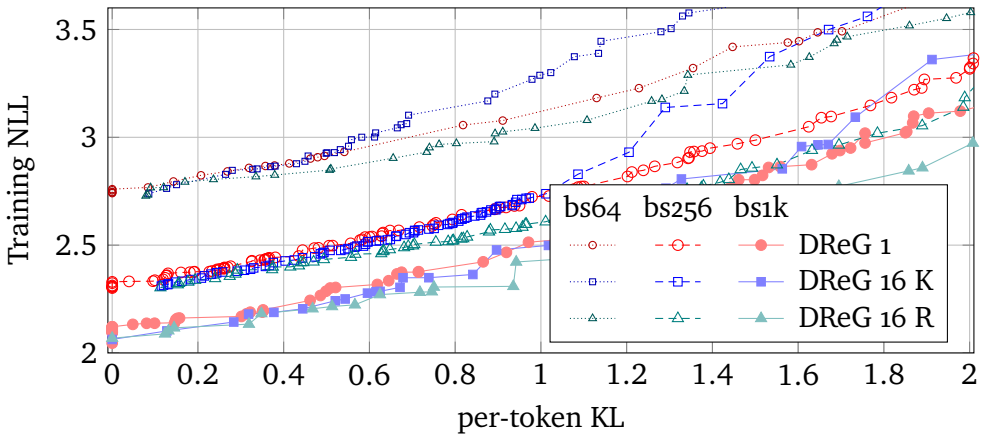


Figure 5.16: The effect of batch size with DReG and 128 hidden units on PTB.

multiple samples in both terms, even more so than in the experiments on the synthetic data earlier (Figures 5.4 and 5.7).

5.B.3 *Experiments with the Power Objective*

Recall that the power objective (5.18) is a special case of the Rényi objective (5.15) with the choice of $\lambda = (\alpha - 1)/\alpha$. With this λ, the objective simplifies to $\mathbb{E}_{p_D(x)} \ln p^\alpha(x)$. Since $\ln p^\alpha(x) = \ln(p(x|z)^\alpha p(z)) = \alpha \ln p(x|z) + \ln p(z)$, implementing the power objective is as easy as upweighting the log-likelihood term $\ln p(x|z)$. Here we investigate whether at the same time, by tying λ and α, we can maintain parity with the Rényi objective in terms efficiency of latent usage. As Figures 5.26 to 5.29 show, the power objective is often better than the KL objective but lags the Rényi objective as α determines both λ, the weight of the mutual information term and its bias with respect to the KL.

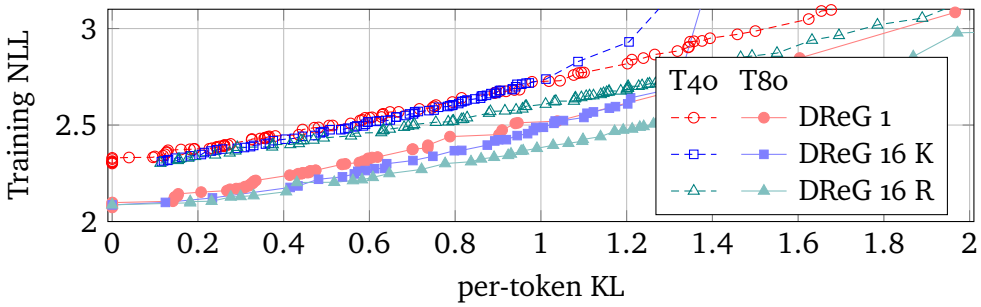


Figure 5.17: The effect of optimization length (40 or 80 thousand optimization steps) with DReG and 128 hidden units on PTB.

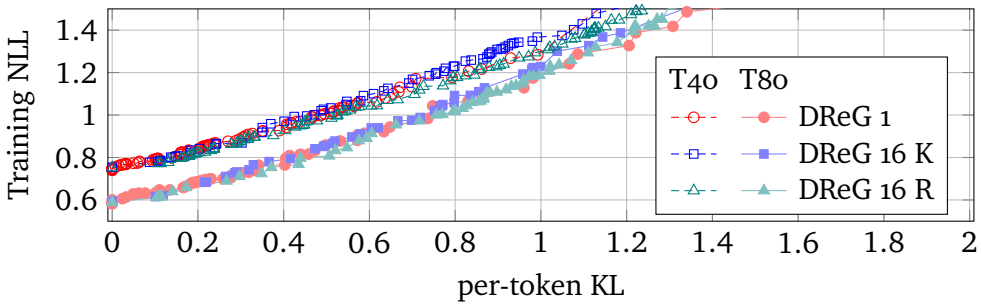


Figure 5.18: The effect of optimization length (40 or 80 thousand optimization steps) with DReG with 256 hidden units on PTB.

5.C OPTIMIZATION SETTINGS

In all experiments, we use the Adam optimizer [Kingma and Ba, 2014] with $\beta_1 = 0$, $\beta_2 = 0.999$, and $\epsilon = 1e-8$. We tune hyperparameters using a black-box hyperparameter tuner based on batched Gaussian Process Bandits [Golovin et al., 2017]. Hyperparameters and their ranges are listed in Table 5.2. The learning rate is the only hyperparameter tuned in all experiments. The rest only apply in specific circumstances. *Input and output dropout* are the dropout rates applied to the inputs and outputs of the LSTM, respectively, while *state dropout* is the dropout rate for the LSTM's recurrent state from the previous time step [Gal and Ghahramani, 2016]. For a description of the VQ-VAE parameters see van den Oord et al. [2017].

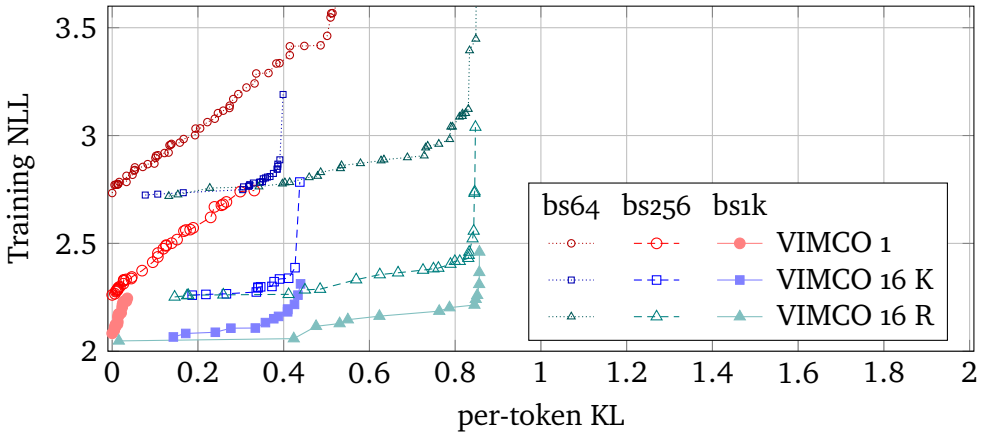


Figure 5.19: The effect of batch size with VIMCO and 128 hidden units on PTB.

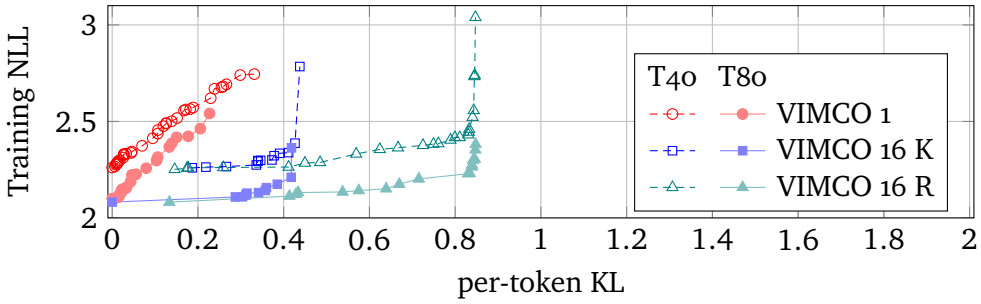


Figure 5.20: The effect of optimization length (40 or 80 thousand optimization steps) with VIMCO and 128 hidden units on PTB.

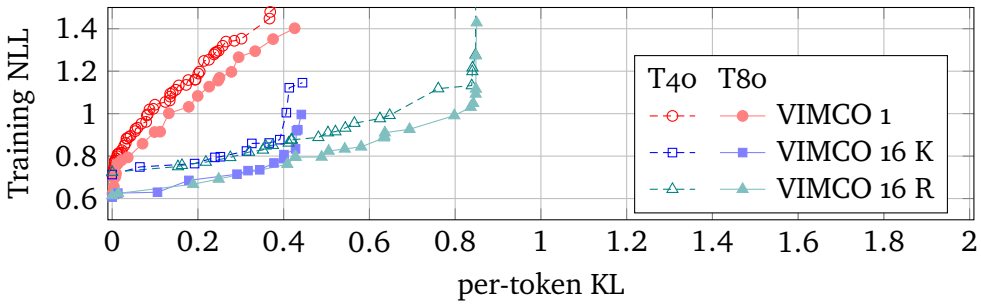


Figure 5.21: The effect of optimization length (40 or 80 thousand optimization steps) with VIMCO and 256 hidden units on PTB.

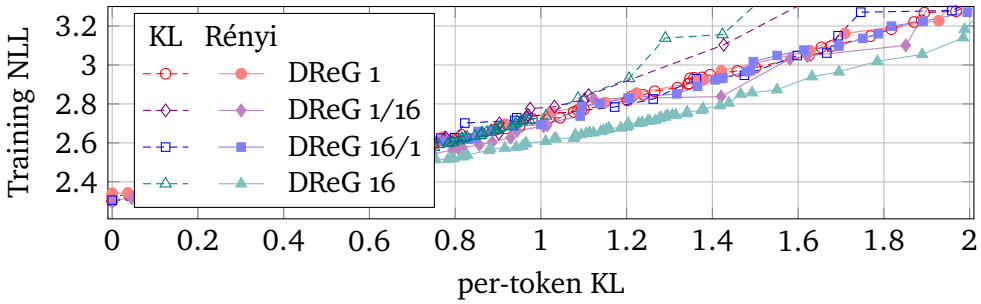


Figure 5.22: KL and Rényi objectives on PTB with base estimator DReG. DReG 1 uses a single sample to estimate both $p(x)$ and $\mathrm{KL}(p(z|x) \parallel p(z))$. DReG 16/1 uses 16 samples to estimate $p(x)$ and 1 samples for the KL. DReG 1/16 uses 1 sample to estimate $p(x)$ and 16 samples for the KL. Finally, DReG 16 uses the same 16 samples for both terms.

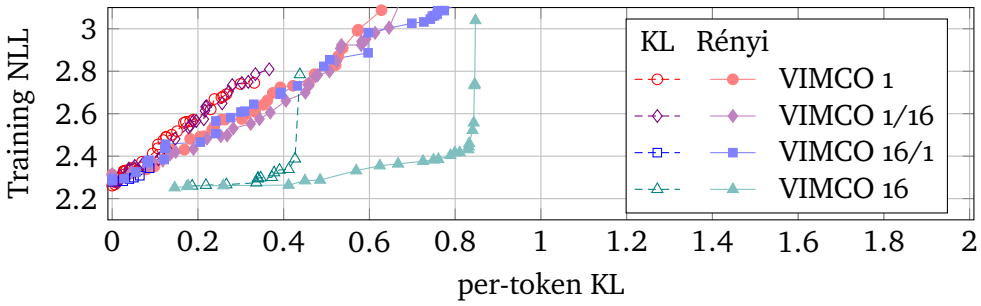


Figure 5.23: KL and Rényi objectives on PTB with base estimator VIMCO. VIMCO 1 uses a single sample to estimate both $p(x)$ and $\mathrm{KL}(p(z|x) \parallel p(z))$. VIMCO 16/1 uses 16 samples to estimate $p(x)$ and 1 samples for the KL. VIMCO 1/16 uses 1 sample to estimate $p(x)$ and 16 samples for the KL. Finally, VIMCO 16 uses the same 16 samples for both terms.

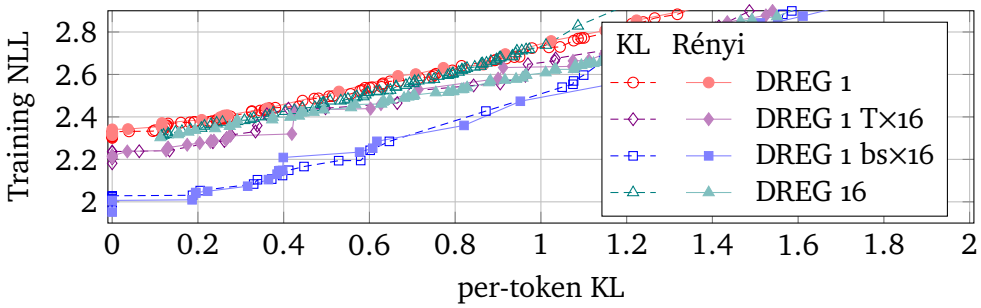


Figure 5.24: Training NLL on PTB with KL and Rényi objectives and base estimator DREG. DREG 1 T×16 is trained 16 times longer. DREG 1 bs×16 has a 16 times larger batch size. While their best NLL at very low latent usage is lower than that of DREG 16, they lose this advantage at higher latent usage.

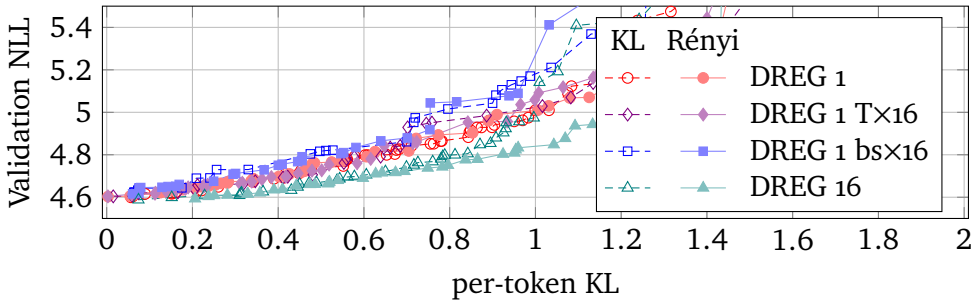


Figure 5.25: Validation NLL on PTB with KL and Rényi objectives and base estimator DREG. DREG 1 T×16 is trained 16 times longer. DREG 1 bs×16 has a 16 times larger batch size. DREG 16 generalizes better than either.

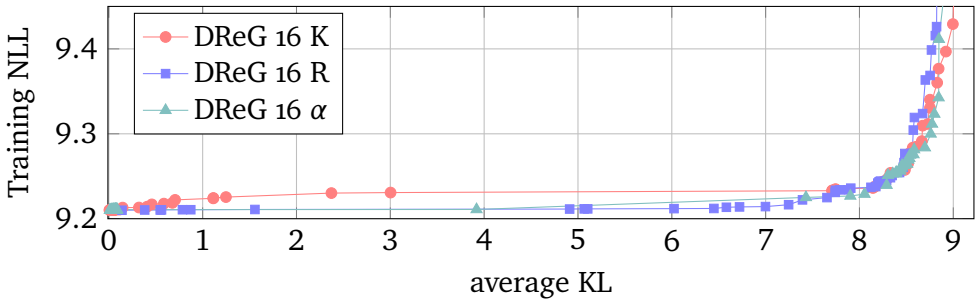


Figure 5.26: The power objective with DReG on the synthetic data set compared to the KL and Rényi objectives.

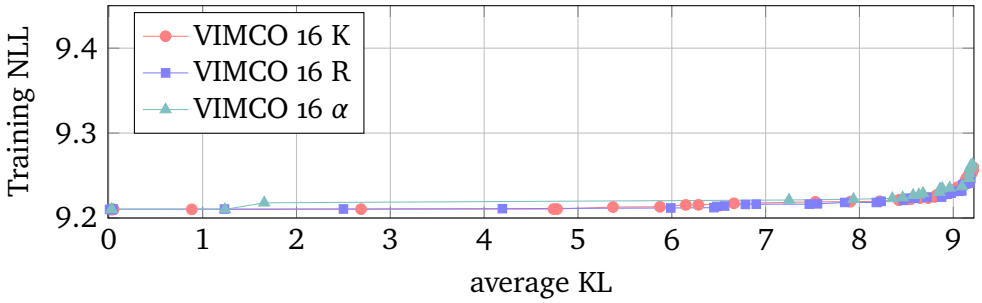


Figure 5.27: The power objective with VIMCO on the synthetic data set compared to the KL and Rényi objectives.

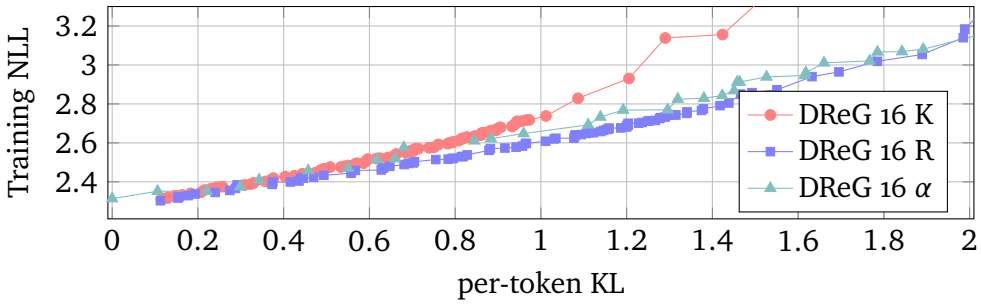


Figure 5.28: The power objective with DReG on PTB compared to the KL and Rényi objectives.

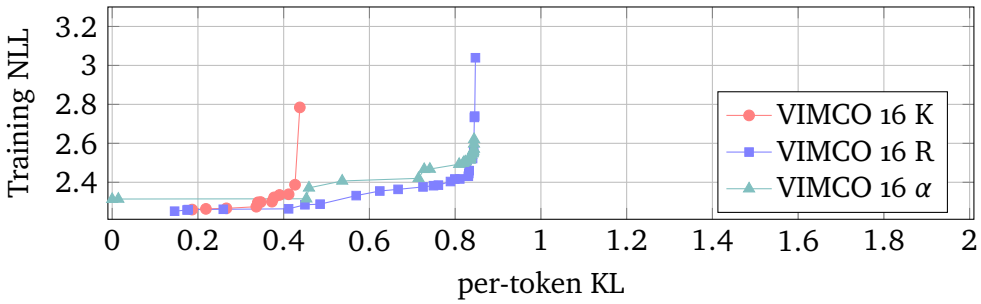


Figure 5.29: The power objective with VIMCO on PTB compared to the KL and Rényi objectives.

Table 5.2: Hyperparameter tuning ranges.

	min	max	scale	
learning rate	0.0001	0.01	log	
λ	-0.05	0.9999		KL and Rényi only
α	0.86	16.0	log	power objective only
input dropout	0.0	0.9		validation only
state dropout	0.0	0.8		validation only
output dropout	0.0	0.95		validation only
L2 penalty coefficient	5e-6	1e-3	log	validation only
number of latents	1	8		VQ-VAE only
number of categories	2	20		VQ-VAE only
VQ β	0.01	20.0	log	VQ-VAE only
VQ decay	0.9	0.99999	log	VQ-VAE only

6 CONCLUSION

> It is at this point that normal language gives up, and goes and has a drink.
>
> *Terry Pratchett*

Our journey towards better generative models of language ends here. On this long and winding road, it was easy to lose sight of the end goal. Throughout, our aim was simply to enable building better language models, whether it be by designing architectures, optimization or inference methods. As it turns out, our contributions in these areas stand on their own beyond the field of language modelling and even natural language processing. The motivation for the Mogrifier was to contextualize input embeddings in language modelling, but its benefits were shown to extend beyond that setting, and it has already seen applications in unrelated fields [Zhang et al., 2022, Cui et al., 2022]. Two-Tailed Averaging is a technique for making optimization for generalization more efficient. It is an easy to use extension on top of stochastic optimization, which we hope will prove to be widely applicable in practice. Finally, MICMCO explores the causes of posterior collapse in the context of variational autoencoders and how to address them. It brings great improvements in the discrete latent case, opening the door for designing latent variable models in which the latents are less of a hindrance and can be employed to shape the model's bias. We hope to see each of these contributions flourish individually.

As to the results, we take them to indicate that recurrent and attention-based architectures are not too different, and further innovation will be needed. Latent variable models and conditional independence assumptions offer a principled approach to injecting structure into the model. Building upon MICMCO, we plan to pursue this avenue of research further. That said, there are other ways of shaping the biases: regularization, tweaking the optimizer, data augmentation and selection, to name just the most obvious ones.

Stepping back, the field of language modelling has changed unrecognizably during the last few years. Scaling models to billions of parameters and humongous amounts of data is now routine. In that context, recurrent architectures can be seen as irrelevant. While it is true that typical recurrent models are considerably slower on contemporary parallel hardware than attention-based

ones, decoupling transient hardware characteristics from model evaluation is a worthy goal that serves to further our understanding and inform future research. As to the evaluation itself, we have shown that recurrent models are underestimated by simply training longer and better with some help from the Rewired Mogrifier LSTM, outperforming or matching transformers on the small datasets considered. But do model biases matter, or is it enough to have lots of data? The importance of data is hard to overstate, but we still manage. Suppose that our goal is to solve every possible task encodable in language. If data and scaling are enough to achieve this goal, then model biases are unimportant. In other words, if we can solve an extremely rich class of problems just by overfitting a huge dataset, then model biases do not matter. As unlikely as that sounds, we are currently making great advances with just scaling. When that stops, research into memory, planning, reasoning, and situated agents will see a resurgence. At the same time, model biases will become a pressing concern, and we will once again need to turn back to the fundamental question of model design to push the frontiers.

BIBLIOGRAPHY

Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. Analyzing the behavior of visual question answering models. *arXiv preprint arXiv:1606.07356*, 2016.

Alexander A Alemi, Ben Poole, Ian Fischer, Joshua V Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken elbo. *arXiv preprint arXiv:1711.00464*, 2017.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128. PMLR, 2016.

Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. *Advances in neural information processing systems*, 29, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Trellis networks for sequence modeling. *arXiv preprint arXiv:1810.06682*, 2018.

Bram Bakker. Reinforcement learning with long short-term memory. In *Advances in neural information processing systems*, pages 1475–1482, 2002.

Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*, 2017.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Jörg Bornschein and Yoshua Bengio. Reweighted wake-sleep. In *International Conference on Learning Representations*, 2014.

Kaj Bostrom and Greg Durrett. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.414. URL `https://aclanthology.org/2020.findings-emnlp.414`.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576*, 2016.

Mark Braverman, Xinyi Chen, Sham Kakade, Karthik Narasimhan, Cyril Zhang, and Yi Zhang. Calibration, entropy rates, and memory in language models. In *International Conference on Machine Learning*, pages 1089–1099. PMLR, 2020.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Kris Cao and Laura Rimell. You should evaluate your language model on marginal likelihood over tokenisations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2104–2114, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.161. URL `https://aclanthology.org/2021.emnlp-main.161`.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pages 2067–2075, 2015.

Chris Cremer, Quaid Morris, and David Duvenaud. Reinterpreting importance-weighted autoencoders. *arXiv preprint arXiv:1704.02916*, 2017.

Alejandrina Cristia, Emmanuel Dupoux, Michael Gurven, and Jonathan Stieglitz. Child-directed speech is infrequent in a forager-farmer population: A time allocation study. *Child development*, 2017.

ShaoDong Cui, YiLa Su, Ren Qing dao er ji, and YaTu Ji. An end-to-end network for irregular printed mongolian recognition. *International Journal on Document Analysis and Recognition (IJDAR)*, pages 1–10, 2022.

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Adji Bousso Dieng, Dustin Tran, Rajesh Ranganath, John Paisley, and David Blei. Variational inference via χ upper bound minimization. In *Advances in Neural Information Processing Systems*, pages 2732–2741, 2017.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality, 2023.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*, 2018.

Jakob N Foerster, Justin Gilmer, Jascha Sohl-Dickstein, Jan Chorowski, and David Sussillo. Input switched affine networks: An rnn architecture designed for interpretability. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1136–1145. JMLR. org, 2017.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1019–1027, 2016.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. Google Vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1487–1495. ACM, 2017.

Chengyue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. Frage: frequency-agnostic word representation. In *Advances in Neural Information Processing Systems*, pages 1334–1345, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. Exposing the implicit energy networks behind masked language models via metropolis–hastings. *arXiv preprint arXiv:2106.02736*, 2021.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Hao Guo, Jiyong Jin, and Bin Liu. Stochastic weight averaging revisited. *arXiv preprint arXiv:2201.00519*, 2022.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=rkpACe1lx`.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. In *International Conference on Learning Representations*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, et al. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017.

John Hewitt, Christopher D Manning, and Percy Liang. Truncation sampling as language model desmoothing. *arXiv preprint arXiv:2210.15191*, 2022.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. Beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Machine Learning*, 2017.

Geoffrey E Hinton and David C Plaut. Using fast weights to deblur old memories. In *Proceedings of the 9th annual conference of the cognitive science society*, pages 177–186, 1987.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Sepp Hochreiter and Jürgen Schmidhuber. Simplifying neural nets by discovering flat minima. In *Advances in neural information processing systems*, pages 529–536, 1995.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997a.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997b.

Sepp Hochreiter and Jürgen Schmidhuber. LSTM can solve hard long time lag problems. In *Advances in neural information processing systems*, pages 473–479, 1997c.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Sara Hooker. The hardware lottery. *CoRR*, abs/2009.06489, 2020. URL `https://arxiv.org/abs/2009.06489`.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

Chin-Wei Huang, Shawn Tan, Alexandre Lacoste, and Aaron C Courville. Improving explorability in variational inference with annealed variational objectives. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Ferenc Huszár. Is maximum likelihood useful for representation learning? URL http://web.archive.org/web/20190704042553/https://www.inference.vc/maximum-likelihood-for-representation-learning-2/, 2017a. Accessed: 2020-04-15.

Ferenc Huszár. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017b.

Marcus Hutter. The human knowledge compression contest, 2012. URL `http://prize.hutter1.net`.

Hakan Inan, Khashayar Khosravi, and Richard Socher. Tying word vectors and word classifiers: A loss framework for language modeling. *CoRR*, abs/1611.01462, 2016. URL `http://arxiv.org/abs/1611.01462`.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. *arXiv preprint arXiv:1804.06059*, 2018.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Prateek Jain, Sham Kakade, Rahul Kidambi, Praneeth Netrapalli, and Aaron Sidford. Parallelizing stochastic gradient descent for least squares regression: mini-batching, averaging, and model misspecification. *Journal of Machine Learning Research*, 18, 2018.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2): 183–233, 1999.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. Learning to create and reuse words in open-vocabulary neural language modeling. *arXiv preprint arXiv:1704.06986*, 2017.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. Sharp nearby, fuzzy far away: How neural language models use context. *arXiv preprint arXiv:1805.04623*, 2018.

Yoon Kim, Sam Wiseman, Andrew C Miller, David Sontag, and Alexander M Rush. Semi-amortized variational autoencoders. *arXiv preprint arXiv:1802.02550*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

Ben Krause, Liang Lu, Iain Murray, and Steve Renals. Multiplicative LSTM for sequence modelling. *CoRR*, abs/1609.07959, 2016. URL `http://arxiv.org/abs/1609.07959`.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. Dynamic evaluation of neural sequence models. *arXiv preprint arXiv:1709.07432*, 2017.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. Dynamic evaluation of transformer language models. *arXiv preprint arXiv:1904.08378*, 2019.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, 2018.

Simon Lacoste-Julien, Mark Schmidt, and Francis Bach. A simpler approach to obtaining an o (1/t) convergence rate for the projected stochastic subgradient method. *arXiv preprint arXiv:1212.2002*, 2012.

Andrew K. Lampinen, Ishita Dasgupta, Stephanie C. Y. Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L. McClelland, Jane X. Wang, and Felix Hill. Can language models learn from explanations in context?, 2022. URL `https://arxiv.org/abs/2204.02329`.

Tao Lei. When attention meets fast recurrence: Training language models with reduced compute. *CoRR*, abs/2102.12459, 2021. URL `https://arxiv.org/abs/2102.12459`.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535, 2016.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

Lars Maaløe, Marco Fraccaro, Valentin Liévin, and Ole Winther. Biva: A very deep hierarchy of latent variables for generative modeling. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Chris J Maddison, John Lawson, George Tucker, Nicolas Heess, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Teh. Filtering variational objectives. In *Advances in Neural Information Processing Systems*, pages 6573–6583, 2017.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2): 313–330, 1993.

James Martens. New insights and perspectives on the natural gradient method. *The Journal of Machine Learning Research*, 21(1):5776–5851, 2020.

Hermann Mayer, Faustino Gomez, Daan Wierstra, Istvan Nagy, Alois Knoll, and Jürgen Schmidhuber. A system for robotic heart surgery that learns to tie knots using recurrent neural networks. *Advanced Robotics*, 22(13-14):1521–1537, 2008.

Arya D McCarthy, Xian Li, Jiatao Gu, and Ning Dong. Improved variational neural machine translation by promoting mutual information. *arXiv preprint arXiv:1909.09237*, 2019.

Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. Typical decoding for natural language generation. *arXiv preprint arXiv:2202.00666*, 2022.

Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*, 2017.

Gábor Melis, Charles Blundell, Tomáš Kočiskỳ, Karl Moritz Hermann, Chris Dyer, and Phil Blunsom. Pushing the bounds of dropout. *arXiv preprint arXiv:1805.09208*, 2018.

Gábor Melis. Two-tailed averaging: Anytime adaptive once-in-a-while optimal iterate averaging for stochastic optimization, 2022. URL `https://arxiv.org/abs/2209.12581`.

Gábor Melis, Tomáš Kočiský, and Phil Blunsom. Mogrifier LSTM. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SJe5P6EYvS`.

Stephen Merity. Single headed attention rnn: Stop thinking with your head. *arXiv preprint arXiv:1911.11423*, 2019.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *CoRR*, abs/1609.07843, 2016. URL `http://arxiv.org/abs/1609.07843`.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*, 2017.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. An analysis of neural language modeling at multiple scales. *arXiv preprint arXiv:1803.08240*, 2018.

Alberto Maria Metelli, Matteo Papini, Nico Montali, and Marcello Restelli. Importance sampling techniques for policy optimization. *J. Mach. Learn. Res.*, 21:141–1, 2020.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*, 2014.

Andriy Mnih and Danilo J Rezende. Variational inference for monte carlo objectives. *arXiv preprint arXiv:1602.06725*, 2016.

Nafise Sadat Moosavi and Michael Strube. Lexical features in coreference resolution: To be used with caution. *arXiv preprint arXiv:1704.06779*, 2017.

Hermann Ney, Ute Essen, and Reinhard Kneser. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language*, 8(1):1–38, 1994.

Hyeonwoo Noh, Tackgeun You, Jonghwan Mun, and Bohyung Han. Regularizing deep neural networks by noise: Its interpretation and optimization. *Advances in Neural Information Processing Systems*, 30, 2017.

Sebastian Nowozin. Debiasing evidence approximations: On importance-weighted autoencoders and jackknife variational inference. 2018.

Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

Tom Pelsmaeker and Wilker Aziz. Effective estimation of deep generative language models. *arXiv preprint arXiv:1904.08194*, 2019.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

George Philipp, Dawn Song, and Jaime G Carbonell. The exploding gradient problem demystified-definition, prevalence, impact, origin, tradeoffs, and solutions. *arXiv preprint arXiv:1712.05577*, 2017.

Mary Phuong, Max Welling, Nate Kushman, Ryota Tomioka, and Sebastian Nowozin. The mutual autoencoder: Controlling information in latent code representations, 2018. URL `https://openreview.net/forum?id=HkbmWqxCZ`.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

Ofir Press and Lior Wolf. Using the output embedding to improve language models. *CoRR*, abs/1608.05859, 2016. URL `http://arxiv.org/abs/1608.05859`.

Geoffrey K Pullum and Barbara C Scholz. Empirical assessment of stimulus poverty arguments. *The linguistic review*, 18(1-2):9–50, 2002.

Tom Rainforth, Adam R Kosiorek, Tuan Anh Le, Chris J Maddison, Maximilian Igl, Frank Wood, and Yee Whye Teh. Tighter variational bounds are not necessarily better. *arXiv preprint arXiv:1802.04537*, 2018.

Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*, 2011.

Ali Razavi, Aaron van den Oord, Ben Poole, and Oriol Vinyals. Preventing posterior collapse with delta-VAEs. In *International Conference on Learning Representations*, 2019.

Ali Lotfi Rezaabad and Sriram Vishwanath. Learning representations by maximizing mutual information in variational autoencoders. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 2729–2734. IEEE, 2020.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Herbert Robbins and Sutton Monro. A stochastic approximation method. In *Herbert Robbins Selected Papers*, pages 102–109. Springer, 1985.

Geoffrey Roeder, Yuhuai Wu, and David K Duvenaud. Sticking the landing: Simple, lower-variance gradient estimators for variational inference. In *Advances in Neural Information Processing Systems 30*, 2017.

Nicolas Le Roux. Anytime tail averaging, 2019.

David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

David Ruppert. Efficient estimations from a slowly convergent Robbins-Monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

Hasim Sak, Andrew W. Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR*, abs/1402.1128, 2014. URL `http://arxiv.org/abs/1402.1128`.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Andriy Serdega and Dae-Shik Kim. Vmi-vae: Variational mutual information maximization framework for vae with discrete and continuous priors. *arXiv preprint arXiv:2005.13953*, 2020.

Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International conference on machine learning*, pages 71–79. PMLR, 2013.

Laura A Shneidman and Susan Goldin-Meadow. Language input and acquisition in a mayan village: How important is directed speech? *Developmental science*, 15(5): 659–673, 2012.

Rui Shu, Hung H. Bui, Shengjia Zhao, Mykel J. Kochenderfer, and Stefano Ermon. Amortized inference regularization. In *NeurIPS*, pages 4398–4407, 2018.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. *Advances in neural information processing systems*, 29:3738–3746, 2016.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Corentin Tallec and Yann Ollivier. Can recurrent neural networks warp time? *arXiv preprint arXiv:1804.11188*, 2018.

Yee Whye Teh. A Bayesian interpretation of interpolated Kneser-Ney. Technical Report TRA2/06, School of Computing, National University of Singapore, 2006.

Michalis Titsias and Miguel Lázaro-Gredilla. Doubly stochastic variational bayes for non-conjugate inference. In *International Conference on Machine Learning*, pages 1971–1979, 2014.

Jakub Tomczak and Max Welling. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223. PMLR, 2018.

George Tucker, Dieterich Lawson, Shixiang Gu, and Chris J Maddison. Doubly reparameterized gradient estimators for monte carlo objectives. *arXiv preprint arXiv:1810.04152*, 2018.

Aaron van den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Chong Wang, Yining Wang, Po-Sen Huang, Abdelrahman Mohamed, Dengyong Zhou, and Li Deng. Sequence modeling via segmentations. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3674–3683. JMLR. org, 2017.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022. URL `https://arxiv.org/abs/2206.07682`.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

R Williams. A class of gradient-estimation algorithms for reinforcement learning in neural networks. In *Proceedings of the International Conference on Neural Networks*, pages II–601, 1987.

Terry Winograd. Understanding natural language, 1973.

Yuhuai Wu, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Ruslan R Salakhutdinov. On multiplicative integration with recurrent neural networks. In *Advances in neural information processing systems*, pages 2856–2864, 2016.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. Breaking the softmax bottleneck: a high-rank RNN language model. *arXiv preprint arXiv:1711.03953*, 2017a.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In *International conference on machine learning*, pages 3881–3890. PMLR, 2017b.

Serena Yeung, Anitha Kannan, Yann Dauphin, and Li Fei-Fei. Tackling over-pruning in variational autoencoders. *arXiv preprint arXiv:1706.03643*, 2017.

Lu Yu, Krishnakumar Balasubramanian, Stanislav Volgushev, and Murat A Erdogdu. An analysis of constant step size SGD in the non-convex regime: Asymptotic normality and bias. *arXiv preprint arXiv:2006.07904v2*, 2020.

Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

Yiyuan Zhang, Sanyuan Zhao, Yuhao Kang, and Jianbing Shen. Modality synergy complement learning with cascaded aggregation for visible-infrared person re-identification. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XIV*, pages 462–479. Springer, 2022.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Balancing learning and inference in variational autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5885–5892, 2019.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016. URL `http://arxiv.org/abs/1611.01578`.