Fine-Tuning Pre-trained Transformers into Decaying Fast Weights

Huanru Henry Mao
Jenni
henry@jenni.ai

Abstract

Autoregressive Transformers are strong language models but incur $O(T)$ complexity during per-token generation due to the self-attention mechanism. Recent work proposes kernel-based methods to approximate causal self-attention by replacing it with recurrent formulations with various update rules and feature maps to achieve $O(1)$ time and memory complexity. We explore these approaches and find that they are unnecessarily complex, and propose a simple alternative - **decaying fast weights** - that runs fast on GPU, outperforms prior methods, and retains 99% of attention's performance for GPT-2. We also show competitive performance on WikiText-103 against more complex attention substitutes.

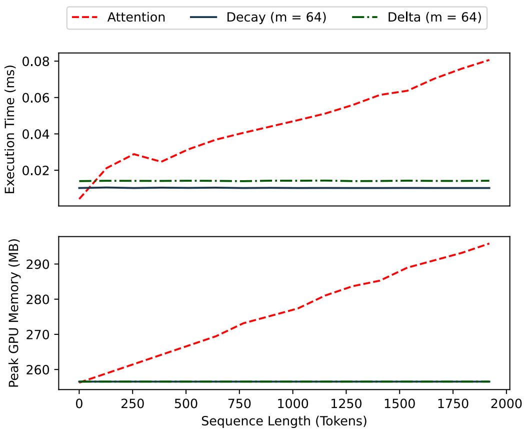

Figure 1: Plot of memory usage and execution time of **our decay rule**, **delta rule** and **attention** when generating the next token at various sequence lengths on Quadro RTX 4000. Decay and delta rule use approximately the same peak memory (overlapped in plot).

1 Introduction

Autoregressive Transformers (Vaswani et al., 2017) have demonstrated strong performance on text generation (Brown et al., 2020). The success of self-attention in Transformers over recurrent models (Hochreiter and Schmidhuber, 1997) can be attributed to its parallelizability (Hooker, 2021) and its effective gradient propagation over many time steps (Ke et al., 2018). However, self-attention has a high computation and memory cost. During inference sampling, it consumes $O(T)$ time and memory and grows linearly per token generated.

These drawbacks motivated recent work to *convert* or *fine-tune* attention into recurrent formulations with $O(1)$ memory and time complexity for auto-regressive generation. **Kernel-based methods** for self-attention (Tay et al., 2021) learn approximations of the exponential similarity function using m-dimensional feature maps to reformulate attention as a recurrent computation. They replace attention with "unlimited capacity" with fixed-capacity fast weights (Schmidhuber, 1992; Peng et al., 2022), where the memory-accuracy trade-off (Kerg et al., 2020) is controlled by m. Several works explored different feature maps and recurrent formulations (i.e., update rules). Katharopoulos et al. (2020) propose feature maps to maintain positive outputs, while Choromanski et al. (2021); Peng et al. (2021) carefully ensure their *random* feature maps are unbiased estimates of the softmax attention kernel. Schlag et al. (2021a); Peng et al. (2021) propose more sophisticated update rules to *forget* information in the recurrent state to improve performance. Recently, Kasai et al. (2021) showed that pre-trained Transformers can be fine-tuned into a recurrent formulation using learned ReLU feature maps with minor degradations. While promising, it is unclear which update rules or feature maps are critical for successful fine-tuning.

In this work, we investigate various update rule configurations to fine-tune pre-trained Transformers into RNNs for fast inference. We find that **prior proposals contain unnecessary operations**, leading us to propose a simple element-wise **decay update rule** with **no feature map**. We fine-tune GPT-2 (Radford et al., 2019) into our recurrent formulation to demonstrate that our rule outperforms prior methods and **recovers 99% of self-attention's performance**. We also show competitive performance on WikiText-103 (Merity et al., 2017) compared to more complex attention alternatives. Our results support the idea (Merity, 2019; Zhai et al., 2021) that it is unnecessary for attention alternatives to maintain a close analogy to self-attention, and it is more important to focus on designing an expressive update rule.

2 Background and Related Work

2.1 Kernel-based Self-Attention

Kernel-based approximations (Katharopoulos et al., 2020; Choromanski et al., 2021; Kasai et al., 2021) to self-attention reorders computation such that a typical $O(Td)$ (per token) memory and time complexity attention becomes $O(dm)$ for T time steps, dimension d and feature size m. Given input to the attention layer $x_t \in \mathbb{R}^{d\times 1}$ and learned weight matrices W_*, the causal self-attention (Vaswani et al., 2017) for query $q_t = W_q x_t \in \mathbb{R}^{d\times 1}$, key $k_t = W_k x_t \in \mathbb{R}^{d\times 1}$ and value $v_t = W_v x_t \in \mathbb{R}^{d\times 1}$ is defined as:

$$y_t = \sum_j^t \frac{\text{sim}(k_j, q_t)}{\sum_i^t \text{sim}(k_i, q_t)} v_j \qquad \text{sim}(x,y) = \exp(x^\intercal y/\sqrt{d}) \tag{1}$$

Kernel-based methods propose an approximation to the exponential similarity function $\widetilde{sim}(x,y) = \phi(x)^\intercal \phi(y)$ via a m-dimensional kernel feature map $\phi : \mathbb{R}^d \to \mathbb{R}^m$. This approximation enables us to rewrite Eq. 1 as

$$y_t = \frac{\sum_j^t v_j \phi(k_j)^\intercal \phi(q_t)}{\sum_i^t \phi(k_i)^\intercal \phi(q_t)} \tag{2}$$

due to the associative property of matrix multiplication. This lends itself to a recurrent formulation with state $S_t \in \mathbb{R}^{d\times m}$ and normalizer $z_t \in \mathbb{R}^{1\times m}$ that can be computed at every time step:

$$S_t = S_{t-1} + v_t \phi(k_t)^\intercal, z_t = z_{t-1} + \phi(k_t)^\intercal \tag{3}$$

The state recurrence resembles fast weight additive outer products (Schmidhuber, 1992). Finally, the output is computed by normalizing against $z_t\phi(q_t)$, which we refer to as **attention normalization**:

$$y_t = \frac{S_t \phi(q_t)}{z_t \phi(q_t)} \tag{4}$$

2.2 Update Rules

Because Eq. 3 is an **additive update rule**, it is unable to forget past memories. This can overwhelm the fixed state capacity and lead to poor performance. Peng et al. (2021) proposes a **gated rule** similar to gated RNNs (Chung et al., 2014) to decay old information and induce a recency bias:

$$\begin{aligned} S_t &= g_t S_{t-1} + (1-g_t) v_t \phi(k_t)^\intercal \\ z_t &= g_t z_{t-1} + (1-g_t)\phi(k_t)^\intercal \end{aligned} \tag{5}$$

where $g_t = \sigma(W_g x_t) \in \mathbb{R}$ is a learned scalar gate that determines how much new information overwrites existing information. They also analogously modify the attention normalizer to incorporate g_t. This rule is problematic as it overwrites all state elements equally without fine-grained control.

Schlag et al. (2021a) proposes improving Eq. 5 using a Fast Weight Programmer (Schmidhuber, 1992) **delta rule** to forget values associated with the current write key by removing the associated value before adding the new value:

$$S_t = S_{t-1} - g_t S_{t-1} \phi'(k_t)\phi'(k_t)^\intercal + g_t v_t \phi'(k_t)^\intercal \tag{6}$$

where g_t is a scalar that defines the extent to which the new value replaces the old value. To stabilize training, Schlag et al. (2021a) applies **sum normalization** to feature maps to enforce the outputs to have components that sum to 1 (i.e., $\phi'(k_t) = \phi(k_t)/\sum_j^d \phi(k_t)_j$). This normalization is applied to both key and query, and the output is computed as $y_t = S_t\phi'(q_t)/z_t\phi'(q_t)$. Schlag et al. (2021a) showed that dropping **attention normalization** (i.e., $y_t = S_t\phi'(q_t)$) works just as well and is redundant when combined with sum normalization.

2.3 Kernel Feature Map

One motivation for kernel-based methods is to closely approximate self-attention. Peng et al. (2021) proposes Random Feature Attention (RFA), which uses random feature maps to produce unbiased estimates of the exponential $\text{sim}(x,y)$ function. Choromanski et al. (2021) proposes FAVOR+, a random feature map with lower variance. Instead of rigorously approximating self-attention, several proposals aim simply to maintain positive outputs motivated by the positivity of attention weights. Katharopoulos et al. (2020) proposes the $\phi(x) = \text{ELU}(x) + 1$ (Clevert et al., 2016) feature map. Schlag et al. (2021a) proposes Deterministic Parameter-Free Projection (DPFP), a feature map that expands m without introducing additional learned parameters. Kasai et al. (2021) proposes using a simple learned ReLU feature map $\phi(x) = \text{ReLU}(W_\phi x + b_\phi)$ that introduces additional parameters and showed better performance than ELU and RFA. In this work, we use this ReLU feature map as a baseline for its strong language model fine-tuning results.

3 Decay Update Rule

We propose the **decay update rule**, which replaces the self-attention mechanism in Transformers with decaying fast weights (Ba et al., 2016) that evolves with linear dynamics. We modify the additive update rule (Eq. 3) by adding a low-rank decay matrix $G_t \in (0,1)^{d\times m}$ to forget information.

$$\begin{aligned} S_t &= G_t \otimes S_{t-1} + v_t\phi(k_t)^\intercal \\ G_t &= \sigma(W_z x_t + b_z)\sigma(W_f x_t + b_f)^\intercal \end{aligned} \tag{7}$$

where $W_z \in \mathbb{R}^{d\times d}$, $W_f \in \mathbb{R}^{m\times d}$, $b_z \in \mathbb{R}^d$ and $b_f \in \mathbb{R}^m$ are newly added parameters. The sigmoid activation σ in G_t bounds the output values to ensure[1] stable linear recurrent dynamics (Miller and Hardt, 2019). We initialize b_z, b_f via Uniform Gate Initialization (Gu et al., 2020), then re-scale pre-trained weight W_v by $1-\sigma(b_z)$. This re-scaling prevents initial iterations from numerical overflows.

Our rule is based on the **gated rule** (Peng et al., 2021) and gated fast weights (Schlag and Schmidhuber, 2017). Unlike the **gated rule**, our gate G_t is a learned matrix and not a scalar, which enables more control when decaying S_t as feature size m increases. Compared to Schlag and Schmidhuber (2017); Peng et al. (2021), we do not gate the $v_t\phi(k_t)^\intercal$ term as it did not bring performance gains given our initialization scheme. Unlike the **delta rule** (Schlag et al., 2021a,b), our rule is a pure element-wise operation, where state dimensions m *do not mix* (Laurent and von Brecht, 2017; Chen, 2017). This can enable more efficient parallelization on GPU (Lei, 2021) (Fig. 1).

Finally, we choose a linear projection feature map and remove attention normalization (Schlag et al., 2021b) and its associated vector z_t from Eq. 4.

$$\phi(x) = W_\phi x, y_t = S_t\phi(q_t) \tag{8}$$

In practice, this feature map can be subsumed into W_k, W_q (Kasai et al., 2021) and is equivalent to the identity function (i.e., **no feature map**).

[1]Without sigmoid, training diverges.

3.1 Implementation Efficiency

During per-token generation, our proposal has the same $O(dm)$ time and memory complexity as prior kernel-based methods instead of the $O(Td)$ requirement of self-attention. Fig. 1 illustrates the practical benefits of delta and decay rules versus self-attention during generation. For training, we developed a memory-efficient CUDA kernel[2] that avoids materializing outer product matrices in memory, similar to Katharopoulos et al. (2020). However, we must store S for backpropagation due to the addition of G_t in Eq. 7, which makes the computation non-reversible (MacKay et al., 2018). Thus, training consumes $O(Tdm)$ memory and $O(T)$ parallel time. In practice, our implementation consumes less memory than self-attention when $m \ll T$. This implies that for memory-bound training, delta may be more favorable than our decay rule despite having a slower run-time. We note this trade off for practitioners to consider and leave memory optimizations for future work.

4 GPT-2 Fine-Tuning Experiments

We perform fine-tuning experiments to speed up existing pre-trained Transformers in a similar setting to Transformer-to-RNN (T2R) (Kasai et al., 2021). We choose GPT-2 small (Radford et al., 2019) as our candidate model to fine-tune, as it has a direct scale-up to large models such as GPT-3 (Brown et al., 2020). To facilitate reproducibility on publicly available datasets, we first fine-tune GPT-2 on The Pile (Gao et al., 2020) - a large general corpus of text. This fine-tuned GPT-2 serves as our baseline *target model* (14.5 PPL, Table 2). When replacing self-attention, we simply swap the layer with a new recurrence formulation. Following Kasai et al. (2021), the entire model is fine-tuned.

We explore different update rules by fine-tuning the target **GPT-2** (Eq. 1) model into **add** (Eq. 3), **gated** (Eq. 5), **delta** (Eq. 6)[3] and our proposed **decay** rule (Eq. 7). In all settings, we hold state capacity $m = 4$ constant for fair comparison. As a **baseline**, we train all rules with ReLU feature maps (Kasai et al., 2021) and attention normalization. For attention normalization, Eq. 5 is used for the gated rule and Eq. 3 is used otherwise. We subsequently ablate attention normalization and ReLU

[2]Our optimized CUDA kernels are available at `https://github.com/jenni-ai/T2FW`.

[3]Following Schlag et al. (2021a), we apply sum normalization to the delta rule. We also initialize $g_t \approx 0.007$ to a small value. Without these changes, the model diverges.

Rule	Baseline	Norm ✗	ϕ ✗
Add	22.6	*	20.0
Delta	21.6	20.1	*
Gated	*	19.7	**17.1**
Decay	N/A	18.8	**17.1**

Table 1: Perplexity (PPL) of different update rule configurations on The Pile validation set. **Baseline** uses both attention normalization and ReLU feature maps. **Norm ✗** means we removed normalization from the baseline. ϕ ✗ means we removed both ReLU feature maps and normalization from the baseline. * indicates divergence.

Rule	m	PPL
GPT-2	-	14.5
Local Attention	-	19.4
Gated	16	16.8
Decay	16	16.1
Gated	32	16.4
Decay	32	**14.6**

Table 2: Update rules on The Pile validation set as m increases. No normalization or feature maps are used.

feature maps (to no feature map) to analyze their effects.

4.1 Results

4.1.1 Normalization Ablation

We first ablate each update rules' proposed attention normalization method to test its effects (**Baseline** vs **Norm ✗** in Table 1). Peng et al. (2021)'s modification of attention normalization (Eq. 5) causes the gated rule to diverge, likely due to division by small z_t values. Without normalization, the gated rule trained with better stability. Our results indicate that only the additive rule with ReLU feature maps requires normalization to converge. Otherwise, normalization is redundant, corroborating Schlag et al. (2021a)'s findings.

4.1.2 Feature Map Ablation

We remove attention normalization on all rules, and further investigate if positive feature maps are necessary (ϕ ✗ in Table 1). For each rule, we replace the ReLU feature map (Kasai et al., 2021) with a linear projection (i.e., no feature map). This simplification improves performance for all rules[4] except the delta rule, which diverges. Our results suggest that positive feature maps are only necessary when using attention or sum normalization to avoid small divisors during normalization. To verify this, we also trained the additive rule with normalization but no feature map and it diverged.

[4]For the additive rule, we re-scaled the pre-trained value weights W_v by $1/512$ to prevent initial iteration overflow.

Model	Memory	Test (0)	Test (480)
Base	-	20.5	19.0
†Linformer	2×64	27.2	30.7
†ABC$_{\text{MLP}}$	2×32	21.9	20.5
Decay	32	22.1	20.7
Decay	64	21.9	20.5

Table 3: Comparison under Baevski and Auli (2019) setting on WikiText-103 test set with context sizes 0 and 480. † results are from Peng et al. (2022), which stores both key and value vectors, doubling the memory size. We do not compare with memory size ABC$_{\text{MLP}}$ 2×64 as it effectively stores 128 vectors.

4.1.3 Update Rule Comparison

When comparing update rules under their best configuration (Table 1), the delta rule performs the worse. Surprisingly, under its best configuration, the simple additive rule outperforms the delta rule. This could be due to a lack of stable recurrent dynamics (Chen, 2017) during the fine-tuning setting. Both the gated and decay rules outperform other rules in all configurations under $m = 4$.

4.1.4 Scaling to Larger m

Finally, we explore if we can increase state capacity m to close the performance gap with **GPT-2** baseline (Table 2). We also compare against **local attention** (window size 32), which is regarded as a strong baseline (Xiong et al., 2022). We train larger state capacity variants of the decay and gated rule where $m = \{16, 32\}$. Our results show that the decay rule scales to better performance relative to the gated rule as m increases. At $m = 32$, our decay rule recovers 99% of GPT-2's performance.

5 WikiText-103 Fine-Tuning Experiments

We perform a similar experiment to fine-tune a pre-trained Transformer into our decay rule with $m = \{32, 64\}$ on the WikiText-103 (Merity et al., 2017) language modeling dataset. We fine-tune from the publicly available checkpoint from Baevski and Auli (2019) (**Base**) by swapping attention with our decay rule and tuning the entire model, with similar hyperparameters as Peng et al. (2022). We compare against recently proposed self-attention approximations including a causal variant of **Lin-**

former (Wang et al., 2020) and Attention with Bounded Memory Control (**ABC$_{\mathbf{MLP}}$**) (Peng et al., 2022) evaluated under the same setting of 0 and 480 token context sizes on the WikiText-103 test set. Our method is simpler than these approaches and performs competitively[5] under similar state memory sizes (Table 3).

6 Limitations

To compare update rules, our experiments use the simple feature map from Kasai et al. (2021) for all rules. However, we acknowledge that some of these rules are jointly proposed with their feature maps, which may be required for good performance. For example, when training Peng et al. (2021)'s gated rule (**baseline**), we attempted various initialization schemes and tried adding small constants to the normalization divisor to help the model converge. However, none of our efforts worked. We note that Peng et al. (2021)'s original proposal of the gated rule does not use the more recently proposed learned ReLU feature maps (Kasai et al., 2021), which may be incompatible with their original proposal. Similarly, Schlag et al. (2021a)'s delta rule was also proposed along with their feature map. Our results suggest that their proposed update rules may not be robust to alternative feature maps or the fine-tuning setting.

The majority of our experiments are performed on The Pile with GPT-2 and assumes a large dataset pre-training setup. While this setup is typical of many NLG applications, it may not generalize to settings without a large dataset, such as low-resource NLP settings. In these cases, more complex approaches may provide better inductive biases than our simple approach.

Our work focuses on the specific case of fine-tuning pre-trained auto-regressive Transformer language models into fast recurrent variants for inference applications. However, we do not explore training these models from scratch or on different architectures. It may be the case that our approach only works in the fine-tuning setting and that self-attention pre-training is required (Kasai et al., 2021).

Our work also exclusively focuses on language modeling and does not consider evaluation on downstream tasks. While language modeling performance typically correlates to downstream task performance (Brown et al., 2020), future work should further validate our proposal on these practical areas of interest.

Finally, as mentioned in Sec. 3.1, our proposed update rule requires more memory (i.e., $O(Tdm)$) than the add and delta rule (i.e., $O(Td)$) during training. This may lead to memory issues when training state sizes with larger m, in which the delta rule may be preferred over the decay rule. Future work should explore extensions of our approach with better space complexity.

[5]Linformer and ABC are trained from scratch, while our work focuses on the fine-tuning setting.

7 Ethical Considerations

Our work uses datasets crawled from the public web, including WikiText-103 (Merity et al., 2017) and The Pile (Gao et al., 2020) and may contain sensitive information or contain undesirable biases. We refer readers to the dataset descriptions from their respective papers for details.

Our work focuses on improving the inference speed of generating from language models by fine-tuning pre-trained models, which may benefit downstream applications that require the deployment of language models. Our work does not explore the issues of bias and disinformation in language models nor specifically aims to mitigate these issues. Our models will likely exhibit the same biases and issues that large language models exhibit (Brown et al., 2020). Practitioners who deploy models with our proposal should not assume our method mitigates these issues. Text sampled from these models may contain offensive content and biases and we advise practical uses of these models to involve some form of human supervision.

References

Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. 2016. Using fast weights to attend to the recent past. *Advances in Neural Information Processing Systems*, 29.

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Minmin Chen. 2017. Minimalrnn: Toward more interpretable and trainable recurrent neural networks. *arXiv preprint arXiv:1711.06788*.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. 2021. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (elus). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Albert Gu, Caglar Gulcehre, Thomas Paine, Matt Hoffman, and Razvan Pascanu. 2020. Improving the gating mechanism of recurrent neural networks. In *International Conference on Machine Learning*, pages 3800–3809. PMLR.

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

Sara Hooker. 2021. The hardware lottery. *Communications of the ACM*, 64(12):58–65.

Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A. Smith. 2021. Finetuning pretrained transformers into rnns. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 10630–10643. Association for Computational Linguistics.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR.

Nan Rosemary Ke, Anirudh Goyal ALIAS PARTH GOYAL, Olexa Bilaniuk, Jonathan Binas, Michael C Mozer, Chris Pal, and Yoshua Bengio. 2018. Sparse attentive backtracking: Temporal credit assignment through reminding. *Advances in neural information processing systems*, 31.

Giancarlo Kerg, Bhargav Kanuparthi, Anirudh Goyal, Kyle Goyette, Yoshua Bengio, and Guillaume Lajoie. 2020. Untangling tradeoffs between recurrence and self-attention in artificial neural networks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Thomas Laurent and James von Brecht. 2017. A recurrent neural network without chaos. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Tao Lei. 2021. When attention meets fast recurrence: Training language models with reduced compute. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7633–7648. Association for Computational Linguistics.

Matthew MacKay, Paul Vicol, Jimmy Ba, and Roger B Grosse. 2018. Reversible recurrent neural networks. *Advances in Neural Information Processing Systems*, 31.

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. 2017. Mixed precision training. *arXiv preprint arXiv:1710.03740*.

John Miller and Moritz Hardt. 2019. Stable recurrent models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Hao Peng, Jungo Kasai, Nikolaos Pappas, Dani Yogatama, Zhaofeng Wu, Lingpeng Kong, Roy Schwartz, and Noah A. Smith. 2022. ABC: attention with bounded-memory control. pages 7469–7483.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. 2021. Random feature attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. 2021a. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pages 9355–9366. PMLR.

Imanol Schlag, Tsendsuren Munkhdalai, and Jürgen Schmidhuber. 2021b. Learning associative inference using fast weight memory. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Imanol Schlag and Jürgen Schmidhuber. 2017. Gated fast weights for on-the-fly neural program generation. In *NIPS Metalearning Workshop*.

Jürgen Schmidhuber. 1992. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2021. Long range arena : A benchmark for efficient transformers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

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

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Wenhan Xiong, Barlas Oguz, Anchit Gupta, Xilun Chen, Diana Liskovich, Omer Levy, Scott Yih, and Yashar Mehdad. 2022. Simple local attentions remain competitive for long-context tasks. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 1975–1986. Association for Computational Linguistics.

Shuangfei Zhai, Walter Talbott, Nitish Srivastava, Chen Huang, Hanlin Goh, Ruixiang Zhang, and Josh Susskind. 2021. An attention free transformer. *arXiv preprint arXiv:2105.14103*.

A GPT-2 Experimental Details

When fine-tuning GPT-2 small on The Pile[6], we started fine-tuning from the publicly available checkpoint provided by Hugging Face[7]. The Pile consists of 825 GB of diverse text (English-dominant) including stories, websites, code and mathematical questions. It is intended for large-scale language model pre-training. Due to the scale of The Pile, our fine-tuning only sees a small subset of the training set. We evaluate on the provided validation set from The Pile.

GPT-2 small (Radford et al., 2019) has approximately 124M parameters. All models are fine-tuned with a batch size of 32 for 100,000 iterations with a learning rate of 6×10^{-4} and gradient clipping of 1.0 on a single NVIDIA A100 GPU for 13 hours. Training was performed on Pytorch 1.11. We use mixed precision training (Micikevicius et al., 2017) except for computing normalizer z_t (Eq. 3), which is prone to numerical overflows. We evaluate on the validation set every 4000 training iterations and report the best validation perplexity achieved for a single run.

B WikiText-103 Experimental Details

WikiText-103 (Merity et al., 2017) is an English dataset of articles scraped from the Good and Featured articles on Wikipedia with 103K training, 217K validation and 245K test word tokens. The dataset is available under the Creative Commons Attribution-ShareAlike License. Compared to The Pile, WikiText-103 is a significantly smaller dataset with approximately 528 MB of text. We fine-tune starting from the checkpoint provided by Baevski and Auli (2019)[8], which has 242M parameters.

Training was performed using the Fairseq library[9] on Pytorch 1.11. A few manual hyperparameter searches were performed based on hyperparameters used in Peng et al. (2022), which included adjustments to the learning rate and the choice of optimizer. For both models, we trained with a batch size of 26 for 100,000 iterations. We used Adam optimizer with 4000 warm-up steps and decayed the learning rate using a cosine schedule to 2×10^{-6} for the rest of the training iterations. We used a maximum learning rate of 10^{-4} and gradient clipping of 0.1. Training was performed on a single NVIDIA A40 GPU for 35 hours. We report the test perplexity achieved at the end of the training for a single run.

[6]https://pile.eleuther.ai/

[7]https://huggingface.co/gpt2

[8]https://github.com/pytorch/fairseq/tree/main/examples/language_model

[9]https://github.com/pytorch/fairseq