CONFORMAL TRANSFORMATIONS FOR SYMMETRIC POWER TRANSFORMERS

Saurabh Kumar[*†] Stanford University
Jacob Buckman[*] Manifest AI
Carles Gelada Manifest AI
Sean Zhang Manifest AI

ABSTRACT

Transformers with linear attention offer significant computational advantages over softmax-based transformers but often suffer from degraded performance. The symmetric power (sympow) transformer, a particular type of linear transformer, addresses some of this performance gap by leveraging symmetric tensor embeddings, achieving comparable performance to softmax transformers. However, the finite capacity of the recurrent state in sympow transformers limits their ability to retain information, leading to performance degradation when scaling the training or evaluation context length. To address this issue, we propose the *conformal-sympow* transformer, which dynamically frees up capacity using data-dependent multiplicative gating and adaptively stores information using data-dependent rotary embeddings. Preliminary experiments on the LongCrawl64 dataset demonstrate that conformal-sympow overcomes the limitations of sympow transformers, achieving robust performance across scaled training and evaluation contexts.

1 INTRODUCTION

Transformers with softmax attention (Vaswani, 2017) have computational cost that is quadratic in context length. A popular solution is to remove the exponential in the softmax, resulting in a linear attention (Katharopoulos et al., 2020; Choromanski et al., 2020). Transformers with linear attention admit a corresponding recurrent formulation enabling linear time inference and a chunked formulation with sub-quadratic training cost. However, while linear transformers enjoy practical speedups, they suffer from degraded performance relative to softmax transformers (Kasai et al., 2021).

To bridge the performance gap with softmax transformers, recent work has proposed a variant of linear transformers called symmetric power (sympow) transformers (Buckman et al., 2024). Sympow transformers embed queries and keys using an embedding function based on the theory of symmetric tensors. Buckman et al. (2024) demonstrates that sympow achieves comparable performance to a softmax transformer baseline while maintaining a tractably small recurrent state size.

While the recurrent formulation enables efficient training and inference, a linear transformer's recurrent state is fundamentally constrained by its finite-dimensional representation. Additionally, in the attention formulation, a linear attention mechanism limits the class of attention score distributions, typically favoring more diffuse distributions relative to softmax attention. As a result, linear transformers may struggle in tasks that require synthesizing information from long contexts. Our experiments confirm that sympow transformers exhibit degraded performance both when increasing the training context and when evaluating on contexts longer than those seen during training.

In this paper, we introduce mechanisms that enable symmetric power transformers to manage their constrained capacity more effectively. We first apply data-dependent multiplicative gating developed in prior work (Dao & Gu, 2024) which erases information in the recurrent state, freeing up capacity for new information. We then introduce a novel approach for learning data-dependent rotary embeddings which iteratively applies dynamically chosen rotations to the recurrent state. Data-dependent rotations enable the model to adaptively determine where to store information in embedding space. The combination of data-dependent gating and rotary embeddings forms a learned *conformal linear transformation* to the recurrent state. We refer to the resulting conformal symmetric power

[*]Denotes equal contribution.
[†]Work completed during internship at Manifest AI. Correspondence to szk@stanford.edu.

transformer as *conformal-sympow*. Our experiments on the LongCrawl64 dataset (Buckman, 2024) demonstrate that conformal-sympow overcomes the limitations of sympow transformers, achieving robust performance across scaled training and evaluation contexts.

2 BACKGROUND: SYMMETRIC POWER TRANSFORMERS

We begin with a high level overview of symmetric power (sympow) transformers (Buckman et al., 2024). A more detailed review is in Appendix A.1.

In a linear transformer, the exponential in a softmax transformer is replaced with a kernel function along with an associated feature map. A symmetric power transformer is a particular type of linear transformer which uses kernel function $\mathbf{k}(v, w) = (v^T w)^p$ where p is referred to as the symmetric power. The corresponding feature map $\phi^p : \mathbb{R}^d \to \mathbb{R}^D$ satisfies the following: for input v, $\phi(v)$ contains the same information as $v \otimes \cdots \otimes v$, repeatedly taking the tensor product p times, while being more compact by removing symmetry.

In the *attention formulation* of a sympow transformer with power p, the inputs to an attention layer are sequences of $Q_i, K_i, V_i \in \mathbb{R}^d$ of queries, keys, and values, where i ranges from 1 to the sequence length t. The outputs are a sequence $Y_i \in \mathbb{R}^d$. The formula for the output vectors is:

$$Y_i = \sum_{j=1}^{i} A_{ij} V_j \qquad A_{ij} = \frac{B_{ij}}{\sum_{k=1}^{i} B_{ik}} \qquad B_{ij} = (Q_i^T K_j)^p \qquad \text{(sympow)}$$

We refer to A_{ij} as the attention scores and B_{ij} as the pre-attention scores.

A key feature of linear transformers is that the exact same outputs Y_i can be computed via a *recurrent formulation*. Using the sympow embedding function ϕ, we can write the recurrent equations:

$$Y_i = \frac{S_i \phi^p(Q_i)}{Z_i \phi^p(Q_i)} \qquad Z_i = Z_{i-1} + \phi^p(K_i)^T \qquad S_i = S_{i-1} + V_i \phi^p(K_i)^T$$

where $Z_0 \in \mathbb{R}^{1 \times D}$ and $S_0 \in \mathbb{R}^{d \times D}$ are 0 vectors in their respective spaces, and the tuple $(S_i, Z_i) \in \mathbb{R}^{(d+1) \times D}$ is the recurrent state that the sympow transformer stores, allowing for linear time inference. The equivalence between the attention and recurrent formulations arises from the fact that the embedding function ϕ^p satisfies the following property: for any two vectors $v, w \in \mathbb{R}^d$, $\phi^p(v)^T \phi^p(w) = (v^T w)^p$. The attention and recurrent forms give rise to a variety of algorithms for training linear transformers, which allow for subquadratic training cost and linear time inference.

Buckman et al. (2024) demonstrates that sympow achieves comparable performance to a softmax transformer baseline while maintaining a tractably small recurrent state size. However, our experiments in Section 4 show that sympow's performance degrades relative to a softmax transformer at longer context lengths. In this paper, we propose mechanisms to help sympow better manage its limited state capacity, which we hypothesize is a key factor behind this performance degradation.

3 CONFORMAL TRANSFORMATIONS

In this section, we propose learning conformal transformations to the sympow transformer's recurrent state in order to better manage its limited capacity. A *conformal linear transformation* is a type of linear transformation that preserves angles between vectors while allowing uniform scaling of lengths. Mathematically, a conformal linear transformation in n-dimensional Euclidean space can be expressed as

$$\mathbf{T}(\mathbf{x}) = s\mathbf{R}\mathbf{x},$$

where $s > 0$ is a scalar representing the scaling factor, $\mathbf{R}$ is an orthogonal matrix ($\mathbf{R}^T\mathbf{R} = \mathbf{I}$), and $\mathbf{x}$ is the input vector.

To update the recurrent state of a sympow transformer, we right multiply the state with a conformal transformation before adding new information:

$$S_i = S_{i-1}(s\mathbf{R}) + V_i\phi^p(K_i)^T \tag{1}$$

In this paper, we consider conformal transformations for which the scalar satisfies $s < 1$ and the matrix $\mathbf{R}$ is a rotation matrix, leveraging the benefits of gating and rotary embeddings. Gating (multiplying the state by a scalar between 0 and 1), serves to erase information that is no longer needed and applying rotations serves to store information in the state more efficiently. In Section 3.1, we apply data-dependent gating developed in prior work to sympow transformers. In Section 3.2, we introduce a novel approach to learn data-dependent rotations. We unify these concepts to form the conformal-sympow transformer in Section 3.3. Proofs of the propositions in this section are in Appendix A.3.

3.1 Data Dependent Gating

The basic idea of gating is that at each time step, the state matrix $S \in \mathbb{R}^{d\times D}$ will be discounted by a scalar $\gamma \in [0, 1]$. Discounting the state "erases" past information stored in the state. This technique has been used extensively throughout the linear transformer literature (Peng et al., 2021; Mao, 2022; Katsch, 2023). In this paper, we apply the technique proposed in Dao & Gu (2024) and used in Peng et al. (2021), Sun et al. (2024), and Beck et al. (2024) to symmetric power transformers. Scalar discount values for gating are computed in a data-dependent manner using parameters W_γ in each attention head. The discount value at timestep i is $\gamma_i = \sigma(W_\gamma X_i)$ where σ refers to the sigmoid function, $W_\gamma \in \mathbb{R}^{d\times 1}$ and $X_1, X_2, ..., X_i, ...$ is the input sequence from which the keys, queries, and values are computed (e.g. $K_i = W_K X_i$). When using symmetric power attention with power p, the recurrent state update is simply

$$Z_i = \gamma_i Z_{i-1} + \phi^p(K_i')^T \qquad S_i = \gamma_i S_{i-1} + V_i\phi^p(K_i')^T \qquad \mu_i = \mu_{i-1} + \theta$$

Here, $K_i' = R^i K_i = R(\mu_i)K_i$ uses the notation for computing and applying rotary positional embeddings, discussed in Appendix A.1.2. $R(\theta)$ (shorthand R) is the rotation matrix used to compute rotary positional embeddings at each time step using a vector of rotation rates θ. The rotation applied to keys at position i is R^i which is equivalent to $R(\theta)^i = R(i\theta) = R(\mu_i)$.

To write the attention formulation we define $b_{ij} = \Pi_{k=j+1}^{i}\gamma_m$. Then, in the attention formulation, the preattention scores become

$$B_{ij} = b_{ij}\ ({Q_i'}^T K_j')^p \qquad \text{(sympow+gating)}$$

Proposition 1. *When applying scalar gating to sympow transformers, the attention formulation of the output Y_i at time step i is equivalent to its recurrent formulation. Specifically,*

$$Y_i = \sum_{j=1}^{i} \frac{b_{ij}\left(Q_i'^T K_j'\right)^p V_j}{\sum_{k=1}^{i} b_{ik}(Q_i'^T K_k')^p} \qquad \textit{is equivalent to} \qquad Y_i = \frac{S_i\phi^p(Q_i')}{Z_i\phi^p(Q_i')}$$

3.2 Data Dependent Rotations

We now introduce an approach to learn rotation rates in rotary positional embeddings. For a review of rotary embeddings and their compatibility with sympow transformers, see Appendix A.1.2.

There are many possible ways to learn rotation rates. For example, the model could independently decide how much to rotate each pair of dimensions in the query or key vector. In this paper, we start with the fixed rotation rates $\theta_1, \theta_2, ..., \theta_{\frac{d}{2}}$ where $\theta_i = \frac{2\pi}{N^{\frac{2(i-1)}{d}}}$, and we equip the model with the ability to uniformly scale the rotation rates. Similar to the data-dependent gating approach, we add parameters W_β to each attention layer. At each time step i, the attention layer outputs a scalar $\beta_i = 1 + \tanh(W_\beta X_i)$ which it uses to scale the fixed vector θ that rotary embeddings typically apply. Recall that $X_1, X_2, ..., X_i, ...$ is the input sequence, so the scalar β_i is data dependent. In the recurrent formulation, this produces the equation

$$\mu_i = \mu_{i-1} + \beta_i\theta \quad \text{where} \quad \beta_i = 1 + \tanh(W_\beta X_i)$$

Intuitively, since the range of $1 + \tanh(\cdot)$ is $(0, 2)$, the model can decide to either speed up the rotation rates by up to 2 times or reduce the rotation rates until effectively zero rotation is applied.

While learned gating affects how information in the state is erased, learned rotations affect where information gets stored in embedding space.

To write the attention formulation we define $c_{ij} = \sum_{k=j+1}^{i} \beta_i$. Then, in the attention formulation, the preattention scores become

$$B_{ij} = (Q_i R(c_{ij}\theta) K_j)^p \qquad \text{(sympow learned rotary)}$$

3.3 Conformal Sympow

We combine data dependent gating and data dependent rotary embeddings into the following manner of computing preattention scores:

$$B_{ij} = b_{ij}\,(Q_i R(c_{ij}\theta) K_j)^p \qquad \text{(conformal-sympow)}$$

The corresponding recurrent state update is

$$Z_i = \gamma_i Z_{i-1} + \phi^p(R(\mu_i)K_i)^T \qquad S_i = \gamma_i S_{i-1} + V_i \phi^p(R(\mu_i)K_i)^T \qquad \mu_i = \mu_{i-1} + \beta_i \theta \quad (2)$$

As written, the recurrent state update above does not match the form of a conformal transformation applied to the state as in Equation 1. While in Equation 1, the state is right multiplied by a rotation matrix, in the above update in Equation 2, the rotation appears when transforming the key vectors $K_i' = R(\mu_i)K_i$. There is an equivalent form to the above state update which does right multiply the state by a conformal transformation. The construction of this form uses the following result.

Proposition 2. *For any vector $k \in \mathbb{R}^d$, if $P \in \mathbb{R}^{d\times d}$ is a rotation matrix, then there exists another rotation matrix $\bar{P} \in \mathbb{R}^{D\times D}$ s.t.*

$$\phi^p(Pk) = \bar{P}\phi^p(k)$$

Using the above result, the recurrent state update can be written as follows:

$$Z_i = Z_{i-1}(\gamma_i \bar{R}(\theta, \beta_i)) + \phi^p(K_i)^T \qquad S_i = S_{i-1}(\gamma_i \bar{R}(\theta, \beta_i)) + V_i \phi^p(K_i)^T \quad (3)$$

where $\bar{R}(\theta, \beta_i)$ is a rotation matrix that depends on the fixed rotation rates θ and the scalar β_i.

In practice, the recurrent state update (2) is more straightforward to implement, but it is equivalent to the conformal recurrent state update (3). We now have the conformal-sympow transformer in both attention and recurrent formulations.

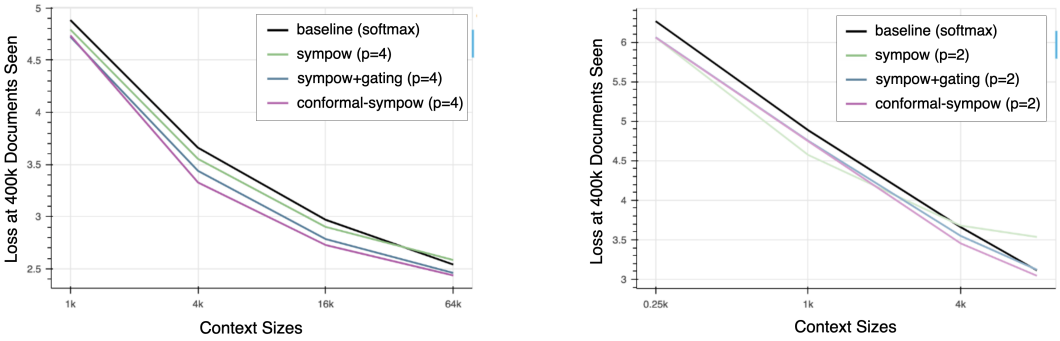
4 Experiments

There are two main goals of our experiments: (1) evaluate sympow transformers on longer training and evaluation contexts than in prior work, and (2) determine the effectiveness of conformal transformations in closing any performance gap that arises at longer contexts.

For all experiments, we used the LongCrawl64 dataset (Buckman, 2024), with a batch size of 524288 tokens. We used a transformer architecture that is similar to the 124M-parameter GPT-2 architecture but with rotary positional encoding and an additional layernorm after input embeddings. We performed optimization in bf16 mixed-precision using Adam with learning rate .0006 and no scheduling. Each model was trained on a node of 8 H100s. Additional results are in Appendix A.4.

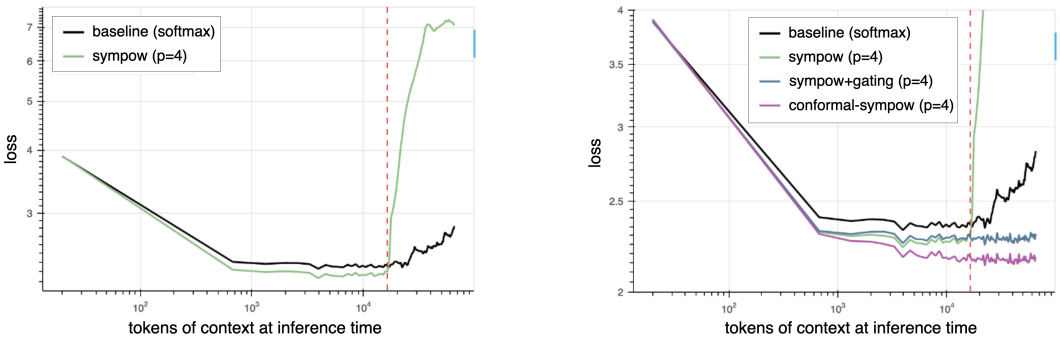
4.1 Are Sympow Transformers Robust to Context Scaling?

Prior work introducing sympow transformers demonstrated that sympow with a tractably small recurrent state (power $p = 4$) closes the performance gap with a softmax transformer baseline at context size 4096 on the LongCrawl64 dataset (Buckman et al., 2024). We ran experiments with the same setup but scaled the training context size from 1024 up to $65,536$ when using sympow with $p = 4$. We additionally ran experiments from training context size 256 up to $8,192$ using sympow with $p = 2$. Using $p = 2$ results in a smaller recurrent state size (39 MB) than when using $p = 4$ (14 GB). Buckman et al. (2024) consider state sizes under 80 GB (the memory capacity of A100 and H100 GPUs) as tractable.



(a) $p = 4$ (b) $p = 2$

Figure 1: The training performance of sympow degrades relative to a softmax transformer baseline as the context size grows. Sympow with data-dependent gating (sympow+gating) closes this performance gap. Training performance further improves when adding data-dependent rotations with conformal-sympow. In contrast to sympow, conformal-sympow does not suffer from the degraded scaling of training context, either when (a) $p = 4$ or (b) $p = 2$.



(a) (b)

Figure 2: Average loss at different evaluation context lengths ranging from 1 to $65,536$ tokens. The training context size is $16,384$, indicated by the dashed red line. (a) Sympow is unable to generalize beyond the training context size of $16,384$. (b) Gated sympow generalizes well and conformal-sympow improves performance further.

Our results in Figure 1 demonstrate that symmetric power transformers suffer degraded training performance compared to the softmax baseline as the training context size grows. While the effect is much more pronounced for $p = 2$ model than $p = 4$, solving poor scaling of the training context size will be important even for models with larger state sizes when scaling to millions of tokens in the training context. We further evaluate the ability of a trained model to make predictions at different evaluation context lengths, including ones longer than the ones used during training. In Figure 2(a), we see that sympow does not generalize well beyond the training context size.

4.2 Does Conformal-Sympow Close the Performance Gap?

In this subsection, we determine whether conformal-sympow overcomes the limitations of sympow transformers. We implement conformal-sympow in its attention formulation which consists of parameters W_γ and W_β in each attention layer to learn data-dependent discounts and rotation scaling, as described in Sections 3.1 and 3.2. We repeat the same experiments as in the previous subsection, scaling the training context sizes from 1024 to $65,536$ when using sympow with $p = 4$ and from 256 to $8,192$ when using sympow with $p = 2$.

Our results in Figure 1 demonstrate that conformal-sympow does not suffer from degraded performance compared to the softmax baseline as the training context size grows. When evaluating

conformal-sympow on held-out data, we find that it generalizes beyond the training context size, as shown in Figure 2(b).

4.3 Related Work

The challenge of scaling transformers to long training and evaluation contexts while maintaining computational efficiency has inspired a rich body of research. This section highlights relevant work on gating mechanisms and positional embeddings, both of which play key roles towards computationally efficient context scaling.

Gated Linear Attention. In architectures with linear attention, gating serves to erase information from the finite recurrent state, freeing up memory. Gating has been shown to improve performance of linear transformers, both at training time and at inference time when extrapolating to long evaluation contexts up to 10 times the train context size (Yang et al., 2023). The design of gating mechanisms for linear attention should ensure compatibility with both attention-based and recurrent formulations, preserving the computational efficiency inherent to linear transformers. To this end, several types of gating have been proposed, including using a global, non-data-dependent decay factor (Qin et al., 2023), per head non-data-dependent decay factors (Sun et al., 2023), scalar and per-head data-dependent decay factors (Dao & Gu, 2024; Peng et al., 2021; Sun et al., 2024; Beck et al., 2024), and a combination of a non-data-dependent matrix with a data-dependent vector (Gu & Dao, 2023). In this work, we adopt the scalar data-dependent gating mechanism introduced in Dao & Gu (2024), as it strikes a balance between expressiveness and efficiency, making it particularly well-suited for integration with sympow transformers. Our results demonstrate that scalar data-dependent gating provides significant benefits to sympow transformers, just as it has for other linear transformer architectures.

Positional Embeddings. In this paper, we adopt rotary embeddings as positional encodings for sympow transformers, leveraging their intrinsic connection to conformal transformations. Rotary embeddings enhance extrapolation performance in softmax transformers compared to sinusoidal position embeddings, which are fixed vectors added to token embeddings before the first transformer layer (Vaswani, 2017; Su et al., 2024; Peng et al., 2021). An alternative approach, the T5 bias model, replaces rotary embeddings with a learned, shared bias added to each query-key score, conditioned on the distance between the query and key (Raffel et al., 2020). While Peng et al. (2021) demonstrate that this method improves extrapolation performance, it incurs a higher computational cost. In contrast, Attention with Linear Biases (ALiBi) offers further enhancements in extrapolation performance while reducing computational overhead (Peng et al., 2021). ALiBi biases query-key attention scores with a penalty that is proportional to the distance between the query and key. In Appendix A.2, we demonstrate that ALiBi can be interpreted as a form of scalar gating, further underscoring the versatility of gating mechanisms in transformer architectures. Recent work studying the effect of positional embeddings on length generalization suggests that positional embeddings are not essential for inference time extrapolation when using softmax transformers (Kazemnejad et al., 2024).

5 Conclusion

We present the conformal-sympow transformer, which addresses the shortcomings of sympow transformers by achieving robust performance across scaled training and evaluation contexts. We further verify the compatibility of learned gating and rotary embeddings with sympow transformers in both the attention and recurrent formulations. While we focus on a specific instantiation of learned conformal transformations, we do not extensively explore alternative gating strategies—such as fixed, learned but data-independent, or data-dependent vector gating—nor the broader landscape of learning rotary embeddings. A more comprehensive investigation of these design choices is left for future work.

Acknowledgements

We would like to thank Txus Bach and Jono Ridgway for insightful discussions and Anmol Kagrecha and Wanqiao Xu for valuable feedback on an early version of the paper.

REFERENCES

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *arXiv preprint arXiv:2405.04517*, 2024.

Jacob Buckman. Longcrawl64: A Long-Context Natural-Language Dataset, 2024.

Jacob Buckman, Carles Gelada, and Sean Zhang. Symmetric Power Transformers, 2024.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A Smith. Finetuning pretrained transformers into rnns. *arXiv preprint arXiv:2103.13076*, 2021.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Tobias Katsch. Gateloop: Fully data-controlled linear recurrence for sequence modeling. *arXiv preprint arXiv:2311.01927*, 2023.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, 36, 2024.

Huanru Henry Mao. Fine-tuning pre-trained transformers into decaying fast weights. *arXiv preprint arXiv:2210.04243*, 2022.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. *arXiv preprint arXiv:2103.02143*, 2021.

Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

Zhen Qin, Dong Li, Weigao Sun, Weixuan Sun, Xuyang Shen, Xiaodong Han, Yunshen Wei, Baohong Lv, Fei Yuan, Xiao Luo, et al. Scaling transnormer to 175 billion parameters. *arXiv preprint arXiv:2307.14995*, 2023.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. You only cache once: Decoder-decoder architectures for language models. *arXiv preprint arXiv:2405.05254*, 2024.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

A APPENDIX

A.1 REVIEW OF SYMMETRIC POWER TRANSFORMERS

In this section, we provide a detailed review of symmetric power transformers. We also review rotary embeddings (Su et al., 2024) which are an essential component of the conformal transformations discussed in this paper. We demonstrate that rotary embeddings are compatible with sympow transformers.

A.1.1 SYMMETRIC POWER TRANSFORMERS

In a linear transformer, the exponential in a softmax transformer is replaced with a kernel function along with an associated feature map. A symmetric power transformer is a particular type of linear transformer which uses kernel function $\mathbf{k}(v, w) = (v^T w)^p$ where p is referred to as the symmetric power. The corresponding feature map $\phi^p : \mathbb{R}^d \to \mathbb{R}^D$ satisfies the following: for input v, $\phi(v)$ contains the same information as $v \otimes \cdots \otimes v$, repeatedly taking the tensor product p times, while being more compact by removing symmetry.

In a sympow transformer, the inputs to an attention layer are sequences of $Q_i, K_i, V_i \in \mathbb{R}^d$ of queries, keys, and values, where i ranges from 1 to the sequence length t. The outputs are a sequence $Y_i \in \mathbb{R}^d$. In the *attention formulation* of a sympow transformer with power p, the formula for the output vectors is:

$$Y_i = \sum_{j=1}^{i} A_{ij} V_j \qquad A_{ij} = \frac{B_{ij}}{\sum_{k=1}^{i} B_{ik}} \qquad B_{ij} = (Q_i^T K_j)^p \qquad \text{(sympow)}$$

We refer to A_{ij} as the attention scores and B_{ij} as the pre-attention scores.

In practice, it is important that the symmetric power p is even because that guarantees that each B_{ij} is non-negative, which makes the set of attention scores $A_{i1}, \cdots, A_{ii}$ a valid probability distribution. In turn, this makes the outputs Y_i a convex combination of the value vectors $V_1, \cdots, V_i$.

A key feature of linear transformers is that the exact same outputs Y_i can be computed via a *recurrent formulation*. In the case of sympow transformers, doing so involves the feature map $\phi^p : \mathbb{R}^d \to \mathbb{R}^D$, which is the symmetric power embedding function. Given an input vector $v \in \mathbb{R}^d$, the vector $\phi^p(v) \in \mathbb{R}^D$ contains the same information as $v \otimes \cdots \otimes v \in \mathbb{R}^{d^p}$, repeatedly taking tensor product p times. It does so much more efficiently because it removes a lot of symmetry in the tensor product (hence the name symmetric power). Thus $D << d^p$. Using this embedding function, we can write the recurrent equations:

$$Y_i = \frac{S_i \phi^p(Q_i)}{Z_i \phi^p(Q_i)} \qquad Z_i = Z_{i-1} + \phi^p(K_i)^T \qquad S_i = S_{i-1} + V_i \phi^p(K_i)^T$$

where $Z_0 \in \mathbb{R}^{1 \times D}$ and $S_0 \in \mathbb{R}^{d \times D}$ are 0 vectors in their respective spaces, and the tuple $(S_i, Z_i) \in \mathbb{R}^{(d+1) \times D}$ is the recurrent state that the sympow transformer stores, allowing for linear time inference.

The equivalence between the attention and recurrent formulations arises from the fact that the embedding function ϕ^p satisfies the following property: for any two vectors $v, w \in \mathbb{R}^d$, $\phi^p(v)^T \phi^p(w) = (v^T w)^p$. The attention and recurrent forms give rise to a variety of algorithms for training linear transformers, which allow for subquadratic training cost and linear time inference.

A.1.2 COMPATIBILITY OF ROTARY EMBEDDINGS WITH SYMPOW

Rotary embeddings (Su et al., 2024) are a type of positional encoding which encode time information by rotating the keys and queries by an amount proportional to their corresponding timestep. In particular, the rotation matrix $R \in \mathbb{R}^{d \times d}$ tells us how much we want to rotate every timestep, so that:

$$Q'_i = R^i Q_i \qquad K'_j = R^j K_j$$

Softmax transformers with rotary embeddings achieve strong performance.

We now show that rotary embeddings are compatible with sympow transformers. Specifically, after applying rotary embeddings, the attention scores remain a distribution. Further, the attention formulation with rotary embeddings has an equivalent recurrent formulation.

In sympow transformers, the pre-attention is changed to:

$$B_{ij} = \left({Q'_i}^T K'_j\right)^p = \left({Q_i}^T (R^{i-j})^T K_j\right)^p \qquad \text{(sympow rotary)}$$

Importantly, rotary embeddings are compatible with the attention formulation when p is even: each B_{ij} is non-negative which makes $A_{i1}, ..., A_{ii}$ a valid probability distribution.

It is evident that the effect of rotation of the embeddings is *relative* because it modulates interaction between Q_i and K_j depending only on the time difference $i - j$.

The rotation matrix R is constructed in a particular way. We start with rotation rates $\theta_1, \theta_2, \cdots, \theta_{\frac{d}{2}}$ distributed in the range $(0, 2\pi)$. Specifically, $\theta_i = \frac{2\pi}{N^{\frac{2(i-1)}{d}}}$, where N is the maximum document size. The vector $\theta = (\theta_1, \theta_2, ..., \theta_{\frac{d}{2}})$ contains these rotation rates. Then, the rotation matrix is the following block diagonal matrix:

$$R(\theta) = \begin{pmatrix} \cos(\theta_1) & -\sin(\theta_1) & \cdots & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \cos(\theta_{d/2}) & -\sin(\theta_{d/2}) \\ 0 & 0 & \cdots & \sin(\theta_{d/2}) & \cos(\theta_{d/2}) \end{pmatrix},$$

When multiplying a query or key vector by this rotation matrix, each pair of dimensions indexed by $2j - 1$ and $2j$ for $j \in \{1, 2, ..., \frac{d}{2}\}$ is rotated by a different amount θ_j. Rotating each pair by a different angle helps break symmetry and increases the expressiveness of the positional encodings.

A computational advantage of using rotation matrices of this form is that there is an efficient way to compute $R(\theta)^k = R(k\theta)$, which massively simplifies the cost of computing all the Q'_i and K'_j. We include a derivation of this fact in Appendix A.3.

Now we want to find the recurrent formulation of rotary embeddings with sympow transformers. A simple way we can do that is by including one extra vector in the recurrent state which is now a tuple (S, Z, μ), where $\mu \in \mathbb{R}^{\frac{d}{2}}$. The recurrent equations are given by

$$Z_i = Z_{i-1} + \phi^p(R(\mu_i)K_i)^T \qquad S_i = S_{i-1} + V_i\phi^p(R(\mu_i)K_i)^T \qquad \mu_i = \mu_{i-1} + \theta$$

Note we rotate the keys by $R(\mu_i)$ before using them.

Given S_i and Z_i, the outputs are the same as before, except that we rotate the queries by $R(\mu)$ before using them:

$$Y_i = \frac{S_i\phi^p(R(\mu_i)Q_i)}{Z_i\phi^p(R(\mu_i)Q_i)}$$

Proposition 3. *When using rotary embeddings with sympow transformers, the attention formulation of the output Y_i at time step i is equivalent to its recurrent formulation. Specifically,*

$$Y_i = \sum_{j=1}^{i} \frac{({Q'_i}^T K'_j)^p V_j}{\sum_{k=1}^{i} ({Q'_i}^T K'_k)^p} \quad \textit{is equivalent to} \quad Y_i = \frac{S_i\phi^p(R(\mu_i)Q_i)}{Z_i\phi^p(R(\mu_i)Q_i)}.$$

The proof is in Appendix A.3.

A.2 Equivalence between Gating and ALiBi

Press et al. (2021) proposes Attention with Linear Biases (ALiBi), a type of positional encoding that significantly improves the ability of softmax transformers to extrapolate to evaluation contexts longer than the training context size. ALiBi biases query-key attention scores with a penalty that is

proportional to the distance between the query and key. We now show that ALiBi is equivalent to applying scalar gating.

In a softmax transformer, the attention scores are computed as

$$A_{ij} = \frac{B_{ij}}{\sum_{k=1}^{i} B_{ik}} \qquad B_{ij} = \exp(Q_i^T K_j) \qquad \text{(softmax)}$$

Recall that we refer to A_{ij} as the attention scores and B_{ij} as the pre-attention scores.

The pre-attention scores after applying ALiBi are

$$B_{ij} = \exp(Q_i^T K_j + m(j-i)) \qquad \text{(softmax + ALiBi)}$$

where $0 < m < 1$ is a head-specific value that is fixed before training.

Note that

$$\exp(Q_i^T K_j + m(j-i)) = \gamma^{(i-j)} \exp(Q_i^T K_j)$$

where $\gamma = \exp(-m)$. Since $-m < 0$, $0 < \gamma < 1$. Thus, the application of ALiBi is equivalent to applying scalar gating.

A.3 Derivations

Proposition. *Given a vector of angles $\theta = (\theta_1, \theta_2, ..., \theta_{\frac{d}{2}}$, the block-diagonal rotation matrix*

$$R(\theta) = \begin{pmatrix} \cos(\theta_1) & -\sin(\theta_1) & \cdots & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \cos(\theta_{d/2}) & -\sin(\theta_{d/2}) \\ 0 & 0 & \cdots & \sin(\theta_{d/2}) & \cos(\theta_{d/2}) \end{pmatrix},$$

satisfies $R(\theta)^k = R(k\theta)$ for any positive integer k.

Proof. We prove this statement by induction on k for a single 2×2 rotation matrix $R(\theta_i)$, and then extend it to the full block diagonal matrix.

For the base case $k = 1$, we have:

$$R(\theta_i)^1 = R(\theta_i),$$

which is equivalent to $R(1 \cdot \theta_i) = R(\theta_i)$. Thus, the base case holds.

Assume that for some positive integer k, the property holds:

$$R(\theta_i)^k = R(k\theta_i).$$

We need to show that $R(\theta_i)^{k+1} = R((k+1)\theta_i)$. Using the definition of matrix exponentiation:

$$R(\theta_i)^{k+1} = R(\theta_i) R(\theta_i)^k.$$

By the inductive hypothesis, $R(\theta_i)^k = R(k\theta_i)$. Substituting this:

$$R(\theta_i)^{k+1} = R(\theta_i) R(k\theta_i).$$

The product of two rotation matrices corresponds to a rotation by the sum of their angles. Therefore:

$$R(\theta_i) R(k\theta_i) = R(\theta_i + (k\theta_i)) = R((k+1)\theta_i).$$

Thus, $R(\theta_i)^{k+1} = R((k+1)\theta_i)$, completing the inductive step. By induction, the property holds for all $k \geq 1$.

Extension to Block Diagonal Matrices

Consider the block diagonal matrix $R(\theta)$, where:

$$R(\theta) = \begin{pmatrix} R(\theta_1) & 0 & \cdots & 0 \\ 0 & R(\theta_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R(\theta_{d/2}) \end{pmatrix}.$$

Since each block $R(\theta_i)$ is independent of the others, the k-th power of $R(\theta)$ is the block diagonal matrix with each block raised to the k-th power:

$$R(\theta)^k = \begin{pmatrix} R(\theta_1)^k & 0 & \cdots & 0 \\ 0 & R(\theta_2)^k & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R(\theta_{d/2})^k \end{pmatrix}.$$

Using the result for a single rotation matrix, $R(\theta_i)^k = R(k\theta_i)$, we get:

$$R(\theta)^k = \begin{pmatrix} R(k\theta_1) & 0 & \cdots & 0 \\ 0 & R(k\theta_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R(k\theta_{d/2}) \end{pmatrix}.$$

This is equivalent to the block diagonal matrix $R(k\theta)$, where $k\theta = (k\theta_1, k\theta_2, \ldots, k\theta_{d/2})$.

Thus, by induction and the block diagonal structure, $R(\theta)^k = R(k\theta)$ for any positive integer k. $\square$

Proposition 3. *When using rotary embeddings with sympow transformers, the attention formulation of the output Y_i at time step i is equivalent to its recurrent formulation. Specifically,*

$$Y_i = \sum_{j=1}^{i} \frac{(Q_i'^T K_j')^p V_j}{\sum_{k=1}^{i} (Q_i'^T K_k')^p} \quad \textit{is equivalent to} \quad Y_i = \frac{S_i \phi^p(R(\mu_i)Q_i)}{Z_i \phi^p(R(\mu_i)Q_i)}.$$

Proof. We begin by writing the output Y_i at time step i in the attention formulation. For notational simplicity, let $C_i = \sum_{k=1}^{i} (Q_i'^T K_k')^p = \sum_{k=1}^{i} \phi^p(Q_i')^T \phi^p(K_k')$.

$$\begin{aligned}
Y_i &= \sum_{j=1}^{i} \frac{\left(Q_i^T (R^{i-j})^T K_j\right)^p V_j}{C_i} \\
&= \sum_{j=1}^{i} \frac{\left(Q_i^T (R^{i})^T R^j K_j\right)^p V_j}{C_i} \\
&= \sum_{j=1}^{i} \frac{\left(Q_i^T R(\mu_i)^T R(\mu_j) K_j\right)^p V_j}{C_i} \\
&= \sum_{j=1}^{i} \frac{\left((R(\mu_i)Q_i)^T R(\mu_j) K_j\right)^p V_j}{C_i} \\
&= \sum_{j=1}^{i} \frac{\left(\phi^p(R(\mu_i)Q_i)^T \phi^p(R(\mu_j)K_j)\right) V_j}{C_i} \\
&= \sum_{j=1}^{i} \frac{V_j \phi^p(R(\mu_j)K_j)^T \phi^p(R(\mu_i)Q_i)}{C_i} \\
&= \frac{\left(\sum_{j=1}^{i} V_j \phi^p(R(\mu_j)K_j)^T\right) \phi^p(R(\mu_i)Q_i)}{C_i} \\
&= \frac{S_i \phi^p(R(\mu_i)Q_i)}{Z_i \phi^p(R(\mu_i)Q_i)}
\end{aligned}$$

which is the recurrent formulation of the output Y_i. The last line above uses the fact that

$$\sum_{j=1}^{i} V_j \phi^p(R(\mu_j)K_j)^T = S_i$$

and

$$\begin{aligned}
C_i &= \sum_{k=1}^{i} \phi^p(Q_i')^T \phi^p(K_k') \\
&= \sum_{k=1}^{i} \phi^p(R(\mu_i)Q_i)^T \phi^p(R(\mu_k)K_k) \\
&= \sum_{k=1}^{i} \phi^p(R(\mu_k)K_k)^T \phi^p(R(\mu_i)Q_i) \\
&= Z_i \phi^p(R(\mu_i)Q_i)
\end{aligned}$$

□

Proposition 1. *When applying scalar gating to sympow transformers, the attention formulation of the output Y_i at time step i is equivalent to its recurrent formulation. Specifically,*

$$Y_i = \sum_{j=1}^{i} \frac{b_{ij}\left(Q_i'^T K_j'\right)^p V_j}{\sum_{k=1}^{i} b_{ik}(Q_i'^T K_k')^p} \qquad \textit{is equivalent to} \qquad Y_i = \frac{S_i \phi^p(Q_i')}{Z_i \phi^p(Q_i')}$$

Proof. We begin by writing the output Y_i at time step i in the attention formulation. For notational simplicity, let $C_i = \sum_{k=1}^{i} b_{ik}(Q_i'^T K_k')^p = \sum_{k=1}^{i} b_{ik}\phi^p(Q_i')^T\phi^p(K_k')$.

$$\begin{aligned}
Y_i &= \sum_{j=1}^{i} \frac{b_{ij}\left(Q_i'^T K_j'\right)^p V_j}{C_i} \\
&= \sum_{j=1}^{i} \frac{b_{ij}\left(\phi^p(Q_i')^T\phi^p(K_j')\right) V_j}{C_i} \\
&= \sum_{j=1}^{i} \frac{b_{ij} V_j \phi^p(K_j')^T\phi^p(Q_i')}{C_i} \\
&= \frac{\left(\sum_{j=1}^{i} b_{ij} V_j \phi^p(K_j')^T\right)\phi^p(Q_i')}{C_i} \\
&= \frac{S_i \phi^p(Q_i')}{Z_i \phi^p(Q_i')}
\end{aligned}$$

which is the recurrent formulation of the output Y_i. The last line above uses the fact that $\sum_{j=1}^{i} b_{ij} V_j \phi^p(K_j')^T = S_i$ and

$$\begin{aligned}
C_i &= \sum_{k=1}^{i} b_{ik}\phi^p(Q_i')^T\phi^p(K_k') \\
&= \sum_{k=1}^{i} b_{ik}\phi^p(K_k')^T\phi^p(Q_i') \\
&= Z_i \phi^p(Q_i')
\end{aligned}$$

We prove that $S_i = \sum_{j=1}^{i} b_{ij} V_j \phi^p(K_j')^T$ by induction. As the base case, note that $S_1 = V_1\phi^p(K_1')^T$. For the inductive step, suppose that for $k > 1$, $S_k = \sum_{j=1}^{k} b_{kj} V_j \phi^p(K_j')^T$. Then

$$\begin{aligned}
S_{k+1} &= \gamma_{k+1} S_k + V_{k+1}\phi^p(K_{k+1}')^T \\
&= \gamma_{k+1}\left(\sum_{j=1}^{k} b_{kj} V_j \phi^p(K_j')^T\right) + V_{k+1}\phi^p(K_{k+1}')^T \\
&= \left(\sum_{j=1}^{k} b_{(k+1)j} V_j \phi^p(K_j')^T\right) + V_{k+1}\phi^p(K_{k+1}')^T \\
&= \sum_{j=1}^{k+1} b_{(k+1)j} V_j \phi^p(K_j')^T
\end{aligned}$$

This completes the inductive step. □

Proposition 2. *For any vector $k \in \mathbb{R}^d$, if $P \in \mathbb{R}^{d\times d}$ is a rotation matrix, then there exists another rotation matrix $\bar{P} \in \mathbb{R}^{D\times D}$ s.t.*

$$\phi^p(Pk) = \bar{P}\phi^p(k)$$

Proof. Note that the symmetric power embedding function is equivalent to applying the tensor product and removing redundant information resulting from symmetry. For mathematical simplicity, we prove the corresponding result for which the embedding function is the repeated tensor product $\otimes^p$. The corresponding proposition is stated below.

Let V be a vector space with dimension d and basis vectors $\{v_1, v_2, \ldots, v_d\}$, and let $P \in \mathbb{R}^{d\times d}$ be a rotation matrix. Define the linear map $\bar{P} : V^{\otimes p} \to V^{\otimes p}$ (the tensor product of p copies of V) by its action on the basis elements as

$$\bar{P}(v_{i_1} \otimes v_{i_2} \otimes \cdots \otimes v_{i_p}) = (Pv_{i_1}) \otimes (Pv_{i_2}) \otimes \cdots \otimes (Pv_{i_p}) \quad \text{for all } i_1, i_2, \ldots, i_p.$$

Then $\bar{P} \in \mathbb{R}^{d^p\times d^p}$ is a rotation matrix.

We need to show that $\bar{P}$ satisfies the properties of a rotation matrix, namely: 1. $\bar{P}$ is an orthogonal matrix, i.e., $\bar{P}^T\bar{P} = I$. 2. $\det(\bar{P}) = 1$, so that $\bar{P}$ represents a proper rotation.

Step 1: Orthogonality of $\bar{P}$.

Since $\bar{P}$ is defined by its action on the basis elements of $V^{\otimes k}$ as

$$\bar{P}(v_{i_1} \otimes v_{i_2} \otimes \cdots \otimes v_{i_p}) = (Pv_{i_1}) \otimes (Pv_{i_2}) \otimes \cdots \otimes (Pv_{i_p}),$$

and P is an orthogonal matrix, i.e., $P^TP = I_d$, where I_d is the identity matrix in $\mathbb{R}^d$, we need to verify that $\bar{P}$ preserves the inner product in the tensor product space. The inner product of two basis elements $v_{i_1} \otimes v_{i_2} \otimes \cdots \otimes v_{i_p}$ and $v_{j_1} \otimes v_{j_2} \otimes \cdots \otimes v_{j_p}$ in $V^{\otimes p}$ is given by:

$$\langle v_{i_1} \otimes v_{i_2} \otimes \cdots \otimes v_{i_p}, v_{j_1} \otimes v_{j_2} \otimes \cdots \otimes v_{j_k}\rangle = \prod_{p=1}^{p} \langle v_{i_p}, v_{j_p}\rangle.$$

Applying $\bar{P}$ to this inner product, we get:

$$\langle \bar{P}(v_{i_1} \otimes \cdots \otimes v_{i_p}), \bar{P}(v_{j_1} \otimes \cdots \otimes v_{j_p})\rangle = \prod_{p=1}^{p} \langle Pv_{i_p}, Pv_{j_p}\rangle.$$

Since P is orthogonal, we have $\langle Pv_{i_p}, Pv_{j_p}\rangle = \langle v_{i_p}, v_{j_p}\rangle$ for each p. Therefore, $\bar{P}$ preserves the inner product, meaning that $\bar{P}$ is an orthogonal matrix, i.e., $\bar{P}^T\bar{P} = I_{d^p}$.

Step 2: Determinant of $\bar{P}$.

Next, we show that $\det(\bar{P}) = 1$. Since $\bar{P} = P \otimes P \otimes \cdots \otimes P$ (a p-fold tensor product of P with itself), we can use the property of the determinant for tensor products of matrices. Specifically, if A and B are square matrices, then:

$$\det(A \otimes B) = \det(A)^{\dim(B)} \det(B)^{\dim(A)}.$$

In our case, since $\bar{P} = P \otimes P \otimes \cdots \otimes P$, we have:

$$\det(\bar{P}) = \det(P)^{p\cdot d}.$$

Since P is a rotation matrix in $\mathbb{R}^d$, we know that $\det(P) = 1$. Therefore:

$$\det(\bar{P}) = 1^{p\cdot d} = 1.$$

Thus, $\bar{P}$ is a proper rotation matrix. □

A.4 EXPERIMENTS

A.4.1 TRAINING DETAILS

Since the goal of our experiments is to understand the performance of sympow and conformal-sympow, we implemented the attention formulation of sympow (with quadratic cost) instead of the more efficient chunked formulation, which would require writing custom CUDA kernels.

A.5 COMPUTE OVERHEAD OF DATA-DEPENDENT GATING AND DATA-DEPENDENT ROTARY EMBEDDINGS

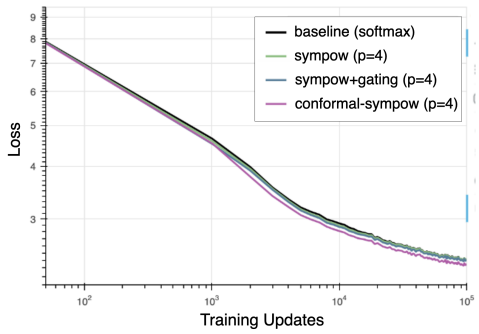
Adding data-dependent gating and rotary embeddings does not add significant compute overhead. Adding data-dependent gating increases the total parameter count by 0.09%. Together, adding data-dependent gating and data-dependent rotary embeddings increases the parameter count by 0.18%. The increase in wall clock time of the training conformal sympow relative to sympow with training context size $16,384$ for 10^5 training updates is $1.06\times$. When using data-dependent gating only, the increase is $1.03\times$. Note that in this paper, all training was done using the attention formulation.

A.5.1 HOW IMPORTANT IS LEARNING ROTARY EMBEDDINGS?

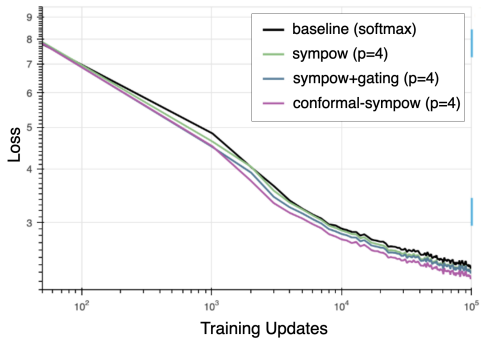
Our conformal-sympow architecture has two components: data-dependent gating and data-dependent rotary embeddings. Data-dependent gating has been shown in prior work to improve the performance of linear transformers (Yang et al., 2023). We isolate the addition of data-dependent rotary embeddings to determine whether scaling rotations is helpful in improving performance. Our results in Figure 1 and Figure 2(b) demonstrate that conformal-sympow further improves both training and evaluation performance over sympow with only learned gating (sympow+gating).

A.5.2 TRAINING CURVES

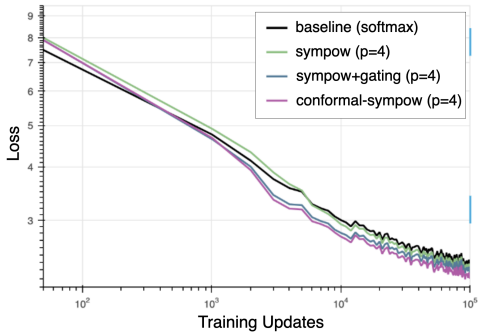
We present full training curves of sympow, sympow with gating, and conformal-sympow with $p = 4$ (Figure 3) and $p = 2$ (Figure 4). We can see that both sympow+gating and conformal-sympow improve optimization over sympow throughout training.



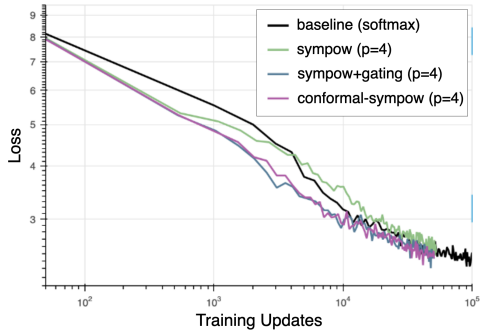
(a) context size = 1024



(b) context size = 4096

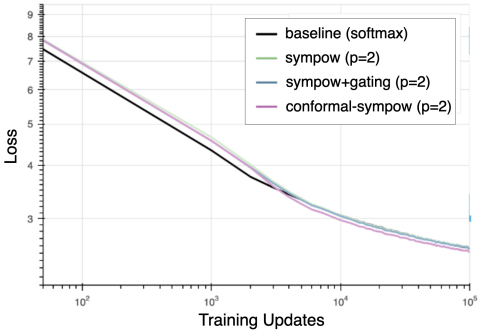


(c) context size = 16384

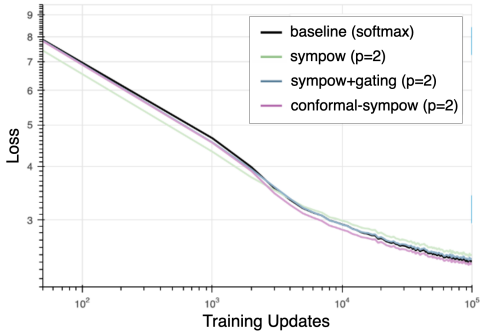


(d) context size = 65536

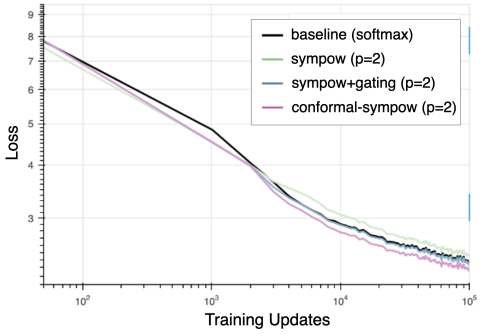
Figure 3: Training curves for sympow, sympow+gating, and conformal-sympow with $p = 4$ at different training context lengths. We can see that both sympow+gating and conformal-sympow improve optimization over sympow throughout training.



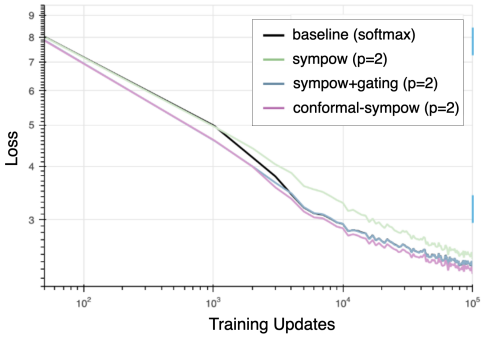
(a) context size = 256



(b) context size = 1024



(c) context size = 4096



(d) context size = 8192

Figure 4: Training curves for sympow, sympow+gating, and conformal-sympow with $p = 2$ at different training context lengths. We can see that both sympow+gating and conformal-sympow improve optimization over sympow throughout training.