Scaling Context Requires Rethinking Attention

Carles Gelada* **Jacob Buckman*** **Sean Zhang*** **Txus Bach**

Abstract

We argue that neither transformers nor sub-quadratic architectures are well suited to training at long sequence lengths: the cost of processing the context is too expensive in the former, too inexpensive in the latter. Approaches such as sliding window attention which reduce the cost-per-token of a transformer impair in-context learning, and so are also unsuitable. To address these limitations, we introduce *power attention*, an architectural layer for linear-cost sequence modeling whose state size can be adjusted independently of parameters, unlocking the advantages of linear attention on practical domains. We develop and open-source a set of GPU kernels for efficient power attention, identifying a novel pattern of operation fusion to avoid memory and bandwidth bottlenecks. Our experiments on the in-context learning of power attention shows that these models dominate both exponential attention and linear attention at long-context training.

1 Introduction

Many techniques to improve the performance of language models involve adding tokens to the context. One popular approach is to include reference material, such as by adding the content of a codebase to the context of a coding assistant [Jimenez et al., 2023]. Another approach is to introduce tokens sampled from the model itself, as is done by chain-of-thought LLMs [DeepSeek-AI et al., 2025, Wei et al., 2022]. A third approach is to use LLM agents, which iteratively interact with the world via tool use and adapt to feedback via context tokens [Yang et al., 2024, He et al., 2024, Schick et al., 2023]. If these context scaling techniques continue to pay off, one might expect a future where contexts regularly contain millions or even billions of tokens.

However, it remains unclear what architectures are best suited for training with long contexts. It is commonly argued that, despite their ubiquity, transformers [Vaswani et al., 2023] are poorly suited to long-context training due to their use of self-attention, whose compute cost grows quadratically with context length. The fact that modern transformer-based LLMs are trained primarily on context lengths between 4k and 32k tokens [Grattafiori et al., 2024, Meta, 2025, Google et al., 2025], with long-context training relegated to post-training (if at all), lends credence to this position. These concerns have motivated research on so-called *subquadratic sequence architectures* such as those proposed by Sun et al. [2023], Peng et al. [2023], Gu and Dao [2024]. These architectures primarily utilize variants of *linear attention*, an operation similar to the attention layer of transformers except that it allows for a recurrent linear-cost formulation.

In Section 3 we argue that any strong long-context architecture must possess three attributes:

1. A balanced weight-to-state ratio at long contexts.
2. Admits an efficient hardware-aware implementation on tensor cores.
3. Good in-context learning (ICL) ability.

We then show that neither attention-based architectures, nor existing subquadratic architectures, meet these criteria. Table 1 summarizes our perspective.

[1]Equal contribution. Correspondence to: cgel.saez@gmail.com, jacobbuckman@gmail.com, seanxwzhang@gmail.com

Preprint. Under review.

In Section 4 we introduce *power attention*, a powerful variant of linear attention. Power attention possesses a hyperparameter p which controls the state size independently of the parameter count, enabling us to balance the weight-state FLOPs ratio for architectures of any scale. It also admits a hardware-aware implementation for training on GPUs. Section 4.1 describes the implementation of our open-source kernels, which enable real wall-clock speedups over Flash Attention in practical settings (e.g. $p = 2$ is 8.6x faster at 64k context). Furthermore, our kernels still lag behind Flash Attention [Dao, 2023] in terms of hardware utilization, and so we expect future engineering efforts which close this gap to result in even larger speedups.

We evaluate power attention empirically in Section 5. Experiments in Section 5.1 show that power attention has better in-context learning than other balanced architectures. In Section 5.3, we show that when training on contexts of length 65536, power attention dominates both exponential and linear attention in terms of loss-per-FLOP.

ARCHITECTURE	BALANCE	EFFICIENCY	ICL
Transformer	✗	✓	✓
Classic RNNs	✗	✗	✓
Modern RNNs	✗	✓	✓
Windowed Attention	✓	✓	✗
Power Attention	✓	✓	✓

Table 1: Comparison of approaches. Section 3 justifies the importance of these criteria, and explains why each architecture passes or fails.

These results have two main limitations. Firstly, our experiments are limited to measuring negative log likelihood on a dataset of generic natural language text. We did not study other domains, modalities, or downstream tasks. Secondly, in our setting, the compute-optimal context grows relatively slowly, diminishing the value of long-context training. We leave to future work the replication of these results across a variety of settings and metrics, and exploration to identify domains with long compute-optimal contexts (perhaps tasks that require chain-of-thought reasoning, or modalities such as audio).

2 Background

Sequence modeling. Let $\mathcal{X}$ denote a finite set of tokens, referred to as the *vocabulary*. Let $\mathcal{X}^t$ denote the set of length-t sequences over $\mathcal{X}$, the *documents*. Given some distribution $\mathbb{D} \in \text{Dist}(\mathcal{X}^t)$ we are concerned with finding a model assigning the maximum probability to documents sampled from $\mathbb{D}$. A common approach is *causal sequence modeling*, based on a model f_θ mapping sequences $\mathcal{X}^i$ of arbitrary length $i \in \mathbb{N}$ to distributions over next-tokens, $\text{Dist}(\mathcal{X})$, where $\theta \in \Theta$ denotes the *parameters* of the model. Implicitly, such a model defines a distribution over the space of all documents $x \in \mathcal{X}^t$ via the autoregressive factorization:

$$f_\theta(x) = f_\theta(x_1, \ldots, x_t) = \prod_{i=1}^{t} f_\theta(x_i \mid x_{<i}) \tag{1}$$

The goal of causal sequence modeling is to learn parameters θ such that the induced distribution f_θ matches the data distribution, where error is typically measured by the *cross-entropy loss*:

$$\mathcal{L}_D(\theta) = \mathbb{E}_{x \sim \mathbb{D}} \left[-\log f_\theta(x) \right] \tag{2}$$

Recurrent neural networks. RNNs [Elman, 1990, Hochreiter and Schmidhuber, 1997, Cho et al., 2014] are models f_θ which can be expressed using a Markovian *state*, $S_i \in \mathbb{R}^n$, which summarizes the information of the entire input history $x_{\leq i} = x_1, \cdots, x_i$. The output of an RNN can be expressed as $y_i = g_\theta(x_i, S_i)$ and the state evolves according to a recurrent relation $S_{i+1} = h_\theta(x_i, S_i)$.

Attention. The causal self-attention layer, a critical piece of the *transformer* architecture [Vaswani et al., 2023], is defined as follows. Let $Q, K \in \mathbb{R}^{t \times d}$, $V \in \mathbb{R}^{t \times v}$ be the query, key and value matrices. We can also think of them as sequences of vectors $Q_i, K_i \in \mathbb{R}^d$ and $V_i \in \mathbb{R}^v$. The output of the attention layer is a matrix $\text{attn}_{\text{exp}}(Q, K, V) \in \mathbb{R}^{t \times v}$ defined as

$$\text{attn}_{\text{exp}}(Q, K, V)_i = \sum_{j=1}^{i} e^{Q_i^T K_j} V_j \tag{3}$$

This can be implemented efficiently in matrix form by using a mask $M \in \mathbb{R}^{t\times t}$ where $M_{ij} = \mathbf{1}_{i\leq j}$,

$$\text{attn}_{\text{exp}}(Q, K, V) = \left(\exp(QK^T) \odot M\right) V \tag{4}$$

where $\exp(A)$ denotes element-wise exponentiation of the matrix A.

Attention can be expressed in an RNN-like form. The outputs Y_i depend only on a state $S_i = (K_{\leq i}, V_{\leq i}) \in \mathbb{R}^{t\times d} \oplus \mathbb{R}^{t\times v}$, commonly called the *KV cache*. The main difference from conventional RNNs is that the state of attention does not have a fixed dimensionality; it grows with sequence length.

Normalization. To stabilize learning, attention usually requires normalization. The original (and most common) approach to normalization is to divide by the sum of the attention scores, turning them into a probability distribution [Vaswani et al., 2023]. We use this normalization throughout this work. One limitation of this approach is that it requires positive attention scores. Other approaches have been proposed [Gu et al., 2024, Ramapuram et al., 2024], but we do not consider them.

Sliding window attention. A variant of attention which chooses a window size w, and truncates the KV cache to this length using a first-in-first-out approach [Child et al., 2019]. The formula for the outputs is $\sum_{j=i-w}^{i} e^{Q_i^T K_j} V_j$.

Linear attention. Katharopoulos et al. [2020] removes the exponential from attention and projects the keys and queries using $\phi : \mathbb{R}^d \to \mathbb{R}^D$,

$$\text{attn}^{\phi}_{\text{lin}}(Q, K, V) = \left(\phi(Q)\phi(K)^T \odot M\right) V \tag{5}$$

where $\phi(A) \in \mathbb{R}^{t\times D}$ denotes application of ϕ to the rows of $A \in \mathbb{R}^{t\times d}$. The key property of linear attention is that it admits an alternative to the KV cache, a constant-size state $S_i \in \mathbb{R}^{v\times D}$ unrolled via the recurrence relation:

$$\text{attn}^{\phi}_{\text{lin}}(Q, K, V)_i = S_i\phi(Q_i) \qquad S_i = S_{i-1} + V_i\phi(K_i)^T \tag{6}$$

The array of t outputs can be computed with cost $O(tDv)$. For this reason, for long sequences, the *recurrent form* is preferred over the *attention form* on the KV cache, which has cost $O\left(t^2(d + v)\right)$. This recurrent form also highlights the motivation behind the inclusion of ϕ. Since $S \in \mathbb{R}^{Dv}$, the choice of ϕ can be used to adjust the state size, known as *state expansion* [Schlag et al., 2021].

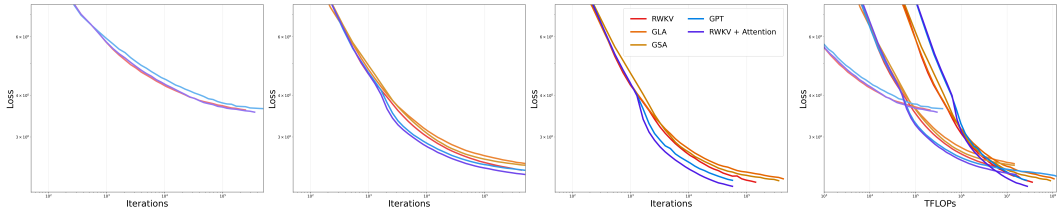
Chunked form. The recurrent form of linear transformers is rarely useful in practice. The states $S_i \in \mathbb{R}^{v\times D}$ are typically large, so having to compute and store in memory every state in the sequence becomes a major bottleneck. The *chunked form* [Buckman and Gelada, a, Sun et al., 2023] interpolates between the recurrent form and the attention form, capturing benefits of both. The key idea is to compute only a subset of all states: $S_0, S_c, S_{2c}, \cdots$, for some appropriately chosen chunk size $c \in \mathbb{N}$. The chunked form is given by the following equation:

$$Y_{nc+m} = S_{nc}\phi(Q_{nc+m}) + \sum_{j=nc+1}^{nc+m} (Q_{nc+m}^T K_j)V_j$$

For any i there exist $0 \leq n$ and $0 \leq m < c$ such that $i = nc + m$. So Y_{nc+m} can be computed with an interaction with the state S_{cn} of cost $O(vD)$ and an intra-chunk attention of cost $O(cd)$. Thus, the cost of the entire output sequence is $O(tDv + tcd)$.

Gating. On long-context tasks, it is common to give a mechanism for the network to directly avoid attending to old data. Originally, this was done at a fixed rate using techniques such as ALiBi [Press et al., 2021]. More recently, Lin et al. [2025] propose a learned gating value per timestep, which is the approach we adopt in this work. Gating has been demonstrated to be particularly important in linear attention [Zhang et al., 2024, Yang et al., 2023, Gu and Dao, 2024].

Architectures. Self-attention layers are merely one piece of a broader transformer architecture, which typically alternates between attention and MLP layers [Vaswani et al., 2023]. Modern architectures often also include components such as rotary embeddings [Su et al., 2024] and local convolutions [Yang et al., 2025]. In this work, we focus our study only on the attention layer, and in general do not modify other architectural components. We use the FLA codebase [Yang and Zhang, 2024] for all architectures.



(a) Context length 32. (b) Context length 1024. (c) Context length 8192. (d) All curves, per-FLOP.

Figure 1: Exponential attention (blue) vs linear attention (red).

3 What does long-context attention require?

In this section, we provide a framework for understanding what attributes of attention techniques make them suitable for long-context training. We focus on classic attention and linear attention, and conclude that neither is suitable for this setting. All experiments are conducted on LongCrawl64 [Buckman, 2024], a dataset containing 6M documents each of length 64k tokens.

3.1 Long-context attention requires a large state.

In Figure 1 we compare the performance of classic exponential attention with that of linear attention at context lengths 32, 1024, and 8192. To check that our conclusions about attention are broadly applicable, we run these experiments across a range of architectures, each modified to use either exponential attention (blue) or linear attention (red). See Appendix D for the full experimental details.

At short context lengths, both forms of attention perform equivalently. But as context length grows, exponential attention gains an advantage. We hypothesize a simple explanation for these observations: state scaling improves performance. In this setting, at context length 32 (Figure 1a), the state size of linear and exponential attention is the same, explaining their equivalent performance per-update. Whereas at context length 8k (Figure 1c), the state size of exponential attention is 256x larger, explaining its better performance per-update.

This additional performance comes at a cost: the larger state requires additional FLOPs per update. Linear attention is often claimed to be superior to exponential attention because it reduces this cost [Sun et al., 2023]. However, Figure 1d reveals that this is misleading, as the best performance for any FLOP budget can be achieved by training with exponential attention.

3.2 Long-context attention requires state-weight balance.

We now explore the implications of the importance of state size on compute-optimal sequence architectures. The computations of a sequence model can be divided into *weight FLOPs*, which involve an activation and a parameter, and *state FLOPs*, which involve an activation and a state.[1] We have seen in the previous section that long-context performance scales with state size, and it is well-established that performance scales with parameter count [Kaplan et al., 2020].

We refer to as the relative proportion of these two types of FLOPs as the *weight-state FLOP ratio (WSFR)*, and we argue that for compute-optimal models, the WSFR should be somewhat close to 1:1. This is because, for any model with a skewed WSFR (for example 100:1), doubling the smaller dimension will be effectively free in terms of total FLOPs. Since both the state and weight scales have a large impact on model performance, it is unwise to not take advantage of free scaling, and doing so will cause the WSFR to approach 1:1.

In Figure 2, we explore this empirically. We train a 400M GPT-2 model on context length 4096, as well as two other models with approximately the same total FLOPs: a small model with a large state,

[1] In this work, we only consider architectures whose weight FLOPs are proportional to parameter count, and whose state FLOPs are proportional to state size. However, note that techniques such as mixture-of-experts [Shazeer et al., 2017] produce a distinction between parameter count and weight FLOPs, and would require more nuanced analysis.

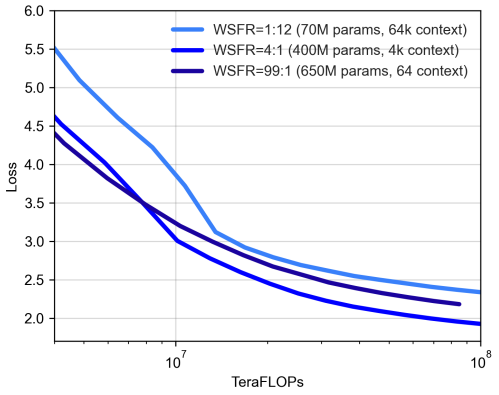


Figure 2: Compute-optimal transformers have a balanced WSFR. See Appendix D for details.

Attention	Context Length	WSFR
Exponential	1 024	8:1
Exponential	**8 192**	**1:1**
Exponential	65 536	1:8
Exponential	1 000 000	1:125
Linear	1 024	30:1
Linear	8 192	30:1
Linear	65 536	30:1
Linear	1 000 000	30:1
Window-8192	1 024	8:1
Window-8192	**8 192**	**1:1**
Window-8192	**65 536**	**1:1**
Window-8192	**1 000 000**	**1:1**

Table 2: WSFR comparison between attention techniques at various context lengths. Balanced architectures are in bold.

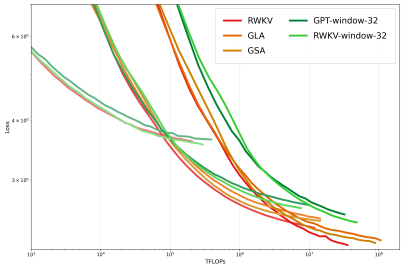
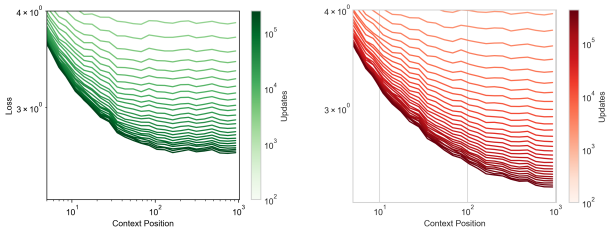


Figure 3: Comparing learning curves if linear vs window-32 attention for several architectures and context lengths.



(a) Window-32 attention. (b) Linear attention.

Figure 4: In-context learning across training for RWKV variants at context length 1024.

and a large model with a small state. [2] This set of models is therefore nearly identical except for WSFR, allowing us to isolate the impact of balance. We confirm that the most balanced architecture has the best performance.

Table 2 shows the WSFR of 124M-parameter GPT-2 models with various attention techniques and context lengths. Exponential attention is balanced for intermediate context lengths, but unbalanced for long context lengths, where it does far more state FLOPs than weight FLOPs. Linear attention, in contrast, is unbalanced at all context lengths in the opposite direction: far more weight FLOPs than state FLOPs. Thus, neither architecture is well-suited for long-context training.

How can we resolve this imbalance? One natural approach is to reduce the state size of exponential attention. In fact, many recent works in the transformer literature can be interpreted through this lens: hybrid architectures [Lieber et al., 2024] reduce the size of the state along the layer dimension, sparse attention [Child et al., 2019] reduces the size of the state along the time dimension, multi-query attention [Shazeer, 2019] reduces the size of the state along the head dimension, and latent attention [Liu et al., 2024] reduces the size of the state along the feature dimension. We use windowed attention to exemplify this family of *reduced-state exponential attention* approaches. Table 2 shows that windowed attention architectures have balanced WSFR for large context lengths, given appropriate selection of window size.

3.3 Long-context attention requires in-context learning.

In Section 3.1, we saw exponential attention outperform linear attention, and attributed this success to its larger state. In Figure 3, we perform a more fair comparison, by juxtaposing linear attention

[2] Since we adjust the state size via the context length, we also adjust the batch size inversely, to keep tokens-per-update identical between runs.

with windowed attention of equal state size. See Appendix D for details. We see a reversal of the previous trend: it is now linear attention that dominates at all context lengths and FLOP budgets. This indicates that linear attention makes better use of its state than windowed attention.

We can explain this gap using an *in-context learning (ICL) curve* of the training loss, which plots the negative log-likelihood at each context length throughout training. In Figure 4, we compare the in-context learning ability of RWKV with windowed attention to linear attention up to context length 1024. Figure 3a shows that no in-context learning occurs beyond 100 tokens for window-32 attention.[3] In contrast, linear attention can be seen in Figure 3b to demonstrate consistent in-context learning across the entire sequence.

Ultimately, a long-context model only has value if the extra context improves its predictions, so these results tell us that windowed attention, despite being balanced at all context lengths, is nonetheless a poor choice for a long-context architecture. We hypothesize that this limitation will be shared by the other reduced-state exponential attention approaches discussed in Section 3.2, although thorough investigation of this hypothesis is left to future work. Instead, in Section 4, we introduce a technique for the other natural approach to balanced long-context attention, *expanded-state linear attention*.

4 Power attention

If one substitutes the exponential in the classic attention formula by the p-th power, the result is *power attention*, variants of which have been studied by Arora et al. [2025], Kacham et al. [2024].

$$\text{attn}^p_{\text{pow}}(Q,K,V)_i = \sum_{j=1}^{i} \left(Q_i^T K_j\right)^p V_j \tag{7}$$

Power attention is a special case of linear attention because there exist functions ϕ s.t. $\phi(Q_i)^T\phi(K_j) = \left(Q_i^T K_j\right)^p$, granting the computational advantages of linear attention discussed in Section 2. Its simple inner-product attention form gives it an important computational advantage over other proposed state-expanded linear transformers, such as DPFP described in Schlag et al. [2021], which require the explicit expansion of $\phi(q), \phi(k)$ in the attention form. When large intermediate objects (such as expanded keys) are involved, the fused attention algorithms pioneered by Dao [2023] do not work, meaning such algorithms have poor hardware utilization in practice.

Lemma 4.1 *The function* $\text{TPOW}_p : \mathbb{R}^d \to \mathbb{R}^{d^p}$ *defined as*

$$\text{TPOW}_p(x) = \begin{bmatrix} x_1 \cdots x_1 \\ x_1 \cdots x_2 \\ \vdots \\ x_d \cdots x_d \end{bmatrix} = \begin{bmatrix} \vdots \\ \prod_k x_{i_k} \\ \vdots \end{bmatrix}_{(i_1,\cdots,i_p)\in\mathbb{N}_d^{\times p}} \tag{8}$$

Then, for $q, k \in \mathbb{R}^d$ *the following property holds* $\text{TPOW}_p(q)^T \text{TPOW}_p(k) = (q^T k)^p$

We therefore have that $\text{attn}^p_{\text{pow}}(Q,K,V) = \text{attn}^{\text{TPOW}_p}_{\text{lin}}(Q,K,V)$. The proofs for this section can be found in Appendix B.

However, a major disadvantage of $\text{TPOW}_p(x)$ is that it contains redundant entries. The theory of *symmetric powers* can be used to address this issue.

Lemma 4.2 *For any* $d, p \in \mathbb{N}$ *denote the set of non-decreasing multi-indices as* $NDMI^p_d = \{(i_1,\cdots,i_p) \in \mathbb{N}_d^{\times p} \mid i_1 \le \cdots \le i_p\}$. *Define* $\text{SPOW}_p : \mathbb{R}^d \to \mathbb{R}^D$ *to be the function*

$$\text{SPOW}_p(x) = \begin{bmatrix} \vdots \\ \sqrt{\frac{p!}{\prod_k hist_k(i)!}}\ \prod_k x_{i_k} \\ \vdots \end{bmatrix}_{i\in NDMI^p_d} \tag{9}$$

Where $hist_k(i_1,\cdots,i_2) = \sum_{j=1}^p 1(i_j = k)$ *is simply the function that counts how many times the index* k *occurs across the the multi index. Then, the following statements hold:*

[3] Note that this is a 12-layer model, so the effective context window is $12 * 32 = 384$ tokens.

1. *The dimensionality D is given by $\binom{d+p-1}{p}$ (the binomial n choose k)*

2. *The inner products $\text{SPOW}_p(q)^T \text{SPOW}_p(k) = (q^T k)^p$*

A few concrete examples might be helpful:

$$\text{SPOW}_2\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 x_1 \\ \sqrt{2}\, x_1 x_2 \\ x_2 x_2 \end{bmatrix} \qquad \text{SPOW}_3\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 x_1 x_1 \\ \sqrt{3}\, x_1 x_1 x_2 \\ \sqrt{3}\, x_1 x_2 x_2 \\ x_2 x_2 x_2 \end{bmatrix}$$

Table 3 compares the dimensions d^p and $\binom{d+p-1}{p}$ for TPOW and SPOW respectively. For large p, these reductions in D have a large impact on the runtime and memory utilization of chunked power attention.

Ultimately, SPOW_p is a state expansion that increases the state size by a factor of $\frac{\binom{d+p-1}{p}}{d}$ without introducing any parameters. For example, for a model with head size 64, $p = 2$ increases the state size by a factor of approximately 32, $p = 3$ by about 700, and $p = 4$ by about 12000.

4.1 Hardware-aware implementation

An efficient implementation of chunked power attention requires careful consideration to the sizes of relevant objects. The main quantities that appear are the key dimension d, the value dimension v, the sequence dimension t, and the expanded key dimension D. At large problem sizes, d, v typically stay small, whereas t, D typically become large. For example, in Llama 3 [Grattafiori et al., 2024], the largest d is 128, whereas the largest t is 128000.

p	TPOW D	SPOW D	Savings
2	4096	2080	49%
3	262144	45760	82%
4	16777216	766480	95%
5	1073741824	10424128	99%
6	68719476736	119877472	99.8%

Table 3: State size comparison for tensor vs. symmetric power embeddings for $d = 64$.

The inputs and outputs of attention, Q, K, V, Y, are all in either $\mathbb{R}^{t\times d}$ or $\mathbb{R}^{t\times v}$. But some intermediate objects, most notably $\phi(Q), \phi(K)$, live in $\mathbb{R}^{t\times D}$. If materialized, these objects dominate memory consumption, and their IO bottlenecks computation and reduces arithmetic intensity. This is reminiscent of how in standard attention, memory and IO is dominated by the attention matrix, an intermediate object in $\mathbb{R}^{t\times t}$. This problem was addressed by Flash Attention [Dao et al., 2022] via operator fusion, whose central algorithmic innovation was the design of a kernel that avoids materializing the attention matrix in HBM.

We apply the same principles to design efficient kernels for chunked power attention. We factorize the algorithm as follows:

$$\text{update-state}(S, K, V) = S + V^T \phi(K) \qquad \text{query-state}(S, Q) = \phi(Q) S^T \tag{10}$$

Each of these functions centers around a matrix multiplication between a large expanded object and a smaller object. This computational structure can be exploited via a fused *expand-MMA* kernel, a matrix multiplication where the tiles of one operand are expanded on-the-fly.

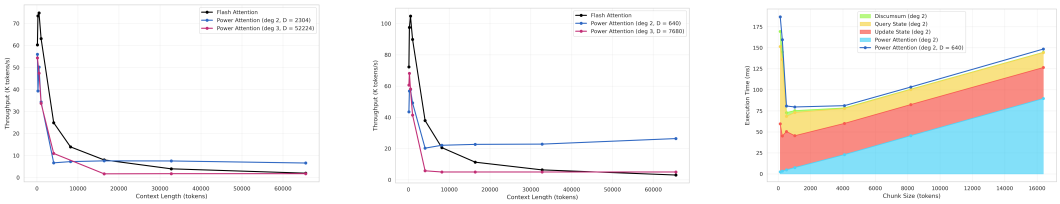
Our implementation of expand-MMA uses Triton [Tillet et al., 2019] with a custom templating system to handle multiple values of p. A complete implementation of chunked power attention also requires a kernel for intra-chunk attention and another for the cumulative gated sum of states (see Appendix F). We use Flash Attention [Dao et al., 2022] for the former and a simple CUDA kernel for the latter.

We have released our kernels open-source[4] to allow others to use power attention, and to enable research on other applications of the symmetric power in deep learning. In Appendix F, we provide more details on our implementation.

4.1.1 TSPOW

Based on Table 3, one would expect that chunked linear attention using SPOW would run faster than TPOW, because its smaller D translates into fewer FLOPs. However, modern GPUs are mainly opti-

[4] `https://github.com/m-a-n-i-f-e-s-t/power-attention`



(a) Throughput comparison at head size 64.

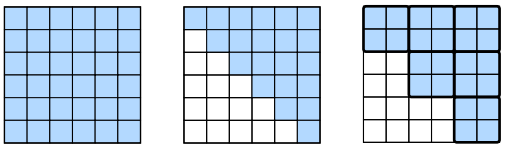
(b) Throughput comparison at head size 32.

(c) Effect of chunk size on total execution time.

Figure 6: Hardware efficiency of Power Attention kernels.

mized for matrix multiplications, and the TPOW expansion is more compatible with the computational structure of a matmul. TPOW calculations can be easily partitioned for parallel processing amongst CTAs/threads using a standard tiling approach. In contrast, the less-regular structure of a symmetric tensor is less compatible with standard GPU operations. The correction term $\sqrt{\frac{p!}{\prod_k \text{hist}_k(i)!}}$ must be computed on slower CUDA cores, causing thread divergence due to branching, and the jagged memory access patterns lead to share memory bank conflicts.

Our approach is to use the idea of tiling to interpolate between TPOW and SPOW, harnessing benefits of both. Our proposed *tiled symmetric power* expansion, TSPOW, operates on tiles of data (providing the GPU-friendly structure of TPOW) but only computes tiles of data with non-decreasing multi-indices (reducing data duplication like SPOW). Figure 5 paints the basic picture for $p = 2$. The dimension of every tile is $d\text{-}tile^p$, and the number of tiles with non-decreasing multi indices is $\binom{d/d\text{-}tile+p-1}{p}$. This means the dimension $D = \binom{d/d\text{-}tile+p-1}{p} d\text{-}tile^p$. Empirically, we find that $d\text{-}tile = 8$ is a good choice for $p = 2$. For $p = 3$ a smaller $d\text{-}tile = 4$ seems preferable.



(a) TPOW (b) SPOW (c) TSPOW

Figure 5: Illustration of TPOW, SPOW, and TSPOW.

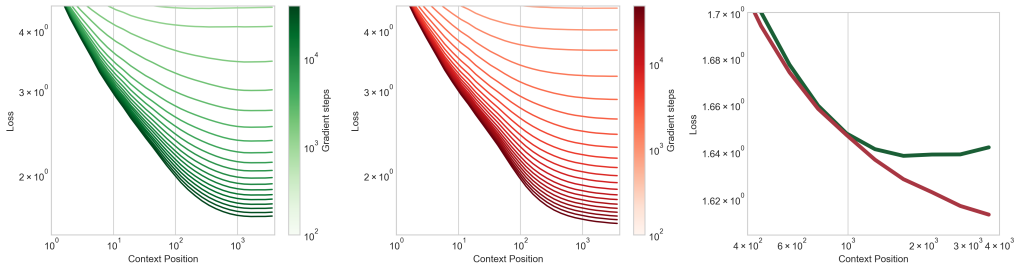
4.1.2 Benchmarks

To benchmark our progress, we compare the throughput (tokens per second) between Power Attention and Flash Attention kernels, with a batch size of 8 and 12 heads, on an A100 GPU. For short contexts, the attention form achieves higher throughput, but as the context size grows, Power Attention switches to the chunk form and retains a constant throughput from then on. On the other hand, the throughput of flash attention decays proportional to t. At context length 65536, degree-2 Power Attention achieves **3.3x** (for head size 64) and **8.6x** (for head size 32) higher throughput than Flash Attention.

Note that the performance of Power Attention is highly dependent on the chunk size c. Figure 6 shows the total execution time is broken down into its component operations for various c.

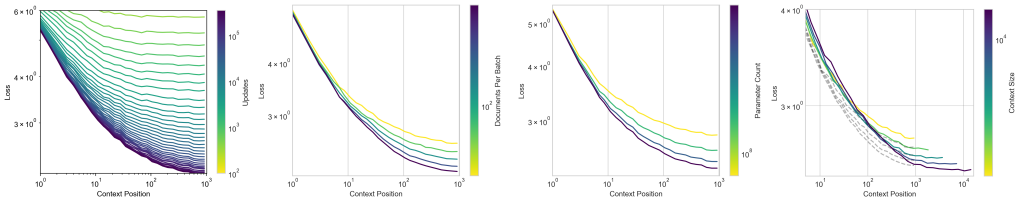
5 Empirical evaluation of power attention

In this section, we evaluate power attention on the basis of its in-context learning ability and long-context performance. To ensure that the dataset contains documents with true long-term structure [5], all of our experiments are conducted on LongCrawl64 [Buckman, 2024]. For these experiments, we use power attention with per-head gating, and normalize by the sum of the attention weights.



(a) Window-1k attention. (b) $p = 2$ power attention. (c) Close-up on ICL curves at 50k.

Figure 7: Power attention demonstrates more ICL per FLOP than equivalent windowed attention.



(a) Gradient updates. (b) Documents per batch. (c) Parameter count. (d) Context length.

Figure 8: The impact of conventional scaling axes on in-context learning of power attention.

5.1 In-context learning comparison

In Section 3, we saw that linear attention has better in-context learning than windowed attention. Now, we investigate whether this is also true for weight-state balanced linear attention, by using power attention. Figure 7 shows the progression through training of the ICL curves of two models with balanced weight-state ratios on context length 4096: window-1024 attention and power attention with $p = 2$. Both models are based on the RKWV architecture with the attention layer swapped for their respective attention mechanism (see Appendix D for full experimental details). In this setting, we see that the power attention architecture has the steepest ICL curve throughout training. Furthermore, as a result of its better in-context learning ability, power attention outperforms (per FLOP) an equivalent transformer in this setting (see Appendix E).

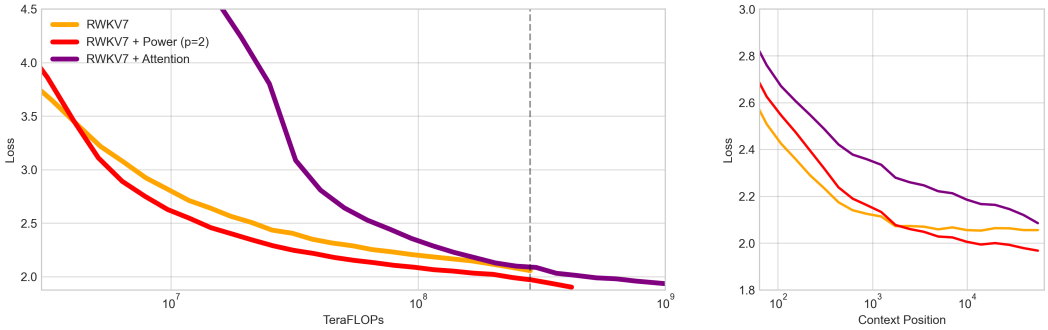
5.2 Factors impacting in-context learning

Figure 8 shows the context-wise loss curve of a $p = 2$ power transformer when varying four axes: number of gradient updates, documents per batch, parameter count, and context length. See Appendix D for experimental details. In all cases, the ICL curve becomes steeper as we scale the respective axis. This indicates that long-context predictions benefit more from scale than short-context predictions.

One phenomenon of note is that scaling context, as shown in Figure 8d, has two effects: additional opportunity for ICL and additional tokens-per-update (this second effect is similar to that of scaling the batch size). The dashed grey lines on this plot ablate these two factors by sampling long sequences but reshaping into a larger batch of shorter sequences, which removes the effect of ICL and so isolates the effect of the additional tokens. We see that the additional tokens are responsible for nearly all of the improvement, and so the same effect could have been achieved by scaling the batch size. [6] The takeaway is that increasing the context length is not always the best way to improve the in-context learning ability of a model, since *all* axes of scale improve in-context learning.

[5] This is not true of many common benchmark datasets. For example, most sequences in OpenWebText [Liu et al., 2019] have length less than 1k. Figure C in Appendix C shows the document length distribution.

[6] Although we do not explore it in depth in this work, we note that increasing the batch size typically increases the *diversity* of the tokens more quickly than increasing the context length does. This translates into better gradient estimates and improved learning, including improved in-context learning.



(a) Heldout best-context loss across training.

(b) ICL after 3e8 TeraFLOPs.

Figure 9: Comparison between different forms of attention on long context. The dashed line in 9a indicates the position of ICL measurement in 9b.

5.3 Long-context training

We now turn to the question of what architecture is best for training on long contexts. We compare three architectures, all based on RWKV, but with different attention layers: native RWKV linear attention, classic exponential attention, and $p = 2$ power attention. Updates are computed on batches of 32 documents, each of length 65536. See Appendix D for experimental details. We see in Figure 9a that power attention dominates other architectures in terms of loss-per-FLOP.

The gap in performance between power attention and exponential attention can be attributed to the difference in cost: exponential attention is much more expensive than power attention on long contexts, so the power attention architecture has the opportunity to perform many more steps of gradient descent. In contrast, the gap in performance between power attention and the original RWKV can by attributed to the difference between their in-context learning abilities, as seen in Figure 9b. RWKV obtains almost no benefit from additional context beyond 2000 tokens. Power attention allows RWKV to in-context learn nearly as well as exponential attention, while still retaining a large advantage in cost.

We note some limitations of this result. Firstly, a context length of 65536 is far larger than is compute-optimal in this setting, meaning that the dominance of power attention demonstrated here does not directly motivate its use on this dataset. Secondly, note that while power attention dominates in the FLOP regime of our experiments,[7] we expect that given sufficient training FLOPs, the attention model would overtake the power attention model, thanks to its larger state size.

6 Conclusions & future work

Our results indicate that linear attention with a hardware-efficient state expansion is the most effective architecture on long-context training, thanks to state-weight balance and strong in-context learning. We have proposed power attention, which is one such approach, and hope future architectural research will continue to study a variety of attention variants and state expansions. For example, one limitation of power attention as currently proposed is its use of the normalization from Vaswani et al. [2023], which requires positive inner products. This means only even powers are supported, and so the parameter-free adjustments to the state size enabled by adjusting p are coarse.

As discussed in Section 3.2 there are many techniques in the literature which reduce the state size of a transformer, including hybrid models, sparse attention, multi-query attention, and latent attention. We investigated one such approach, windowed attention, and found its in-context learning abilities to be worse than linear attention models of the same state size. In the future, a more comprehensive comparison to existing methods would be valuable. Furthermore, a complete characterization of the performance of these algorithms merits rigorous investigation under the framework of scaling laws [Kaplan et al., 2020]. Future work should quantitatively explore the impact of state size, context size,

[7] Our 10^9 TeraFLOPs corresponds to about 1000 H100-hours at 30% flop utilization.

and in-context learning on model performance, with the aim of fitting scaling laws dependent on these factors.

Our initial implementation uses Triton [Tillet et al., 2019], a high-level tool for writing GPU kernels that allows quick, Pythonic prototyping. However, without the flexibility provided by CUDA, our kernels cannot be optimized as thoroughly as e.g. Flash Attention [Dao, 2023]. As a result, our implementation, Power Attention, is not yet as dominant in wall-clock comparisons as FLOPs comparisons would indicate. Future implementations of Power Attention will move from Triton to CUDA in order to push wall-clock performance further.

Our experiments are limited to measuring negative log likelihood on a dataset of generic natural language text. We did not study other domains, modalities, or downstream tasks. In the future, we hope to validate our findings in these settings. Furthermore, we have observed that autoregressive prediction of natural language is largely dominated by short-context dependencies, even on long documents. This diminishes the value of long-context training in this setting. In future work, we hope to discover domains which are dominated by long-term dependencies. For example, we plan to explore tasks that require chain-of-thought reasoning, tool use, and modalities such as audio and video. In domains where performance is heavily dependent on long-term dependencies, the compute-optimal context will be large, and we expect that the dominance of power attention on long contexts will be of practical importance.

Acknowledgments and Disclosure of Funding

We would like to thank SF Compute for their generous support on computational resources for this research, Warfa Jibril for data and engineering contributions related to this project, and to Edward Hu, David Brandfonbrener, Eren Malach, and Zhixuan Lin for providing us with feedback on an early draft of this paper.

References

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff, 2025. URL https://arxiv.org/abs/2402.18668.

Jacob Buckman. Longcrawl64: A Long-Context Natural-Language Dataset. https://manifestai.com/articles/longcrawl64/, 2024. Accessed: 2025-05-15.

Jacob Buckman and Carles Gelada. Linear Transformers Are Faster, a.

Jacob Buckman and Carles Gelada. Compute-optimal Context Size, b.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014. URL https://arxiv.org/abs/1409.1259.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023. URL https://arxiv.org/abs/2307.08691.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang

Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL `https://arxiv.org/abs/2501.12948`.

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

Team Google, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul R. Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, Jack Krawczyk, Cosmo Du, Ed Chi, Heng-Tze Cheng, Eric Ni, Purvi Shah, Patrick Kane, Betty Chan, Manaal Faruqui, Aliaksei Severyn, Hanzhao Lin, YaGuang Li, Yong Cheng, Abe Ittycheriah, Mahdis Mahdieh, Mia Chen, Pei Sun, Dustin Tran, Sumit Bagri, Balaji Lakshminarayanan, Jeremiah Liu, Andras Orban, Fabian Güra, Hao Zhou, Xinying Song, Aurelien Boffy, Harish Ganapathy, Steven Zheng, HyunJeong Choe, Ágoston Weisz, Tao Zhu, Yifeng Lu, Siddharth Gopal, Jarrod Kahn, Maciej Kula, Jeff Pitman, Rushin Shah, Emanuel Taropa, Majd Al Merey, Martin Baeuml, Zhifeng Chen, Laurent El Shafey, Yujing Zhang, Olcan Sercinoglu, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, Alexandre Frechette, Charlotte Smith, Laura Culp, Lev Proleev, Yi Luan, Xi Chen, James Lottes, Nathan Schucher, Federico Lebron, Alban Rrustemi, Natalie Clay, Phil Crone, Tomas Kocisky, Jeffrey Zhao, Bartek Perz, Dian Yu, Heidi Howard, Adam Bloniarz, Jack W. Rae, Han Lu, Laurent Sifre, Marcello Maggioni, Fred Alcober, Dan Garrette, Megan Barnes, Shantanu Thakoor, Jacob Austin, Gabriel Barth-Maron, William Wong, Rishabh Joshi, Rahma Chaabouni, Deeni Fatiha, Arun Ahuja, Gaurav Singh Tomar, Evan Senter, Martin Chadwick, Ilya Kornakov, Nithya Attaluri, Iñaki Iturrate, Ruibo Liu, Yunxuan Li, Sarah Cogan, Jeremy Chen, Chao Jia, Chenjie Gu, Qiao Zhang, Jordan Grimstad, Ale Jakse Hartman, Xavier Garcia, Thanumalayan Sankaranarayana Pillai, Jacob Devlin, Michael Laskin, Diego de Las Casas, Dasha Valter, Connie Tao, Lorenzo Blanco, Adrià Puigdomènech Badia, David Reitter, Mianna Chen, Jenny Brennan, Clara Rivera, Sergey Brin, Shariq Iqbal, Gabriela Surita, Jane Labanowski, Abhi Rao, Stephanie Winkler, Emilio Parisotto, Yiming Gu, Kate Olszewska, Ravi Addanki, Antoine Miech, Annie Louis, Denis Teplyashin, Geoff Brown, Elliot Catt, Jan Balaguer, Jackie Xiang, Pidong Wang, Zoe Ashwood, Anton Briukhov, Albert Webson, Sanjay Ganapathy, Smit Sanghavi, Ajay Kannan, Ming-Wei Chang, Axel Stjerngren, Josip Djolonga, Yuting Sun, Ankur Bapna, Matthew Aitchison, Pedram Pejman, Henryk Michalewski, Tianhe Yu, Cindy Wang, Juliette Love, Junwhan Ahn, Dawn Bloxwich, Kehang Han, Peter Humphreys, Thibault Sellam, James Bradbury, Varun Godbole, Sina Samangooei, Bogdan Damoc, Alex Kaskasoli, Sébastien M. R. Arnold, Vijay Vasudevan, Shubham Agrawal, Jason Riesa, Dmitry Lepikhin, Richard Tanburn, Srivatsan Srinivasan, Hyeontaek Lim, Sarah Hodkinson, Pranav Shyam, Johan Ferret, Steven Hand,

Ankush Garg, Tom Le Paine, Jian Li, Yujia Li, Minh Giang, Alexander Neitz, Zaheer Abbas, Sarah York, Machel Reid, Elizabeth Cole, Aakanksha Chowdhery, Dipanjan Das, Dominika Rogozińska, Vitaliy Nikolaev, Pablo Sprechmann, Zachary Nado, Lukas Zilka, Flavien Prost, Luheng He, Marianne Monteiro, Gaurav Mishra, Chris Welty, Josh Newlan, Dawei Jia, Miltiadis Allamanis, Clara Huiyi Hu, Raoul de Liedekerke, Justin Gilmer, Carl Saroufim, Shruti Rijhwani, Shaobo Hou, Disha Shrivastava, Anirudh Baddepudi, Alex Goldin, Adnan Ozturel, Albin Cassirer, Yunhan Xu, Daniel Sohn, Devendra Sachan, Reinald Kim Amplayo, Craig Swanson, Dessie Petrova, Shashi Narayan, Arthur Guez, Siddhartha Brahma, Jessica Landon, Miteyan Patel, Ruizhe Zhao, Kevin Villela, Luyu Wang, Wenhao Jia, Matthew Rahtz, Mai Giménez, Legg Yeung, James Keeling, Petko Georgiev, Diana Mincu, Boxi Wu, Salem Haykal, Rachel Saputro, Kiran Vodrahalli, James Qin, Zeynep Cankara, Abhanshu Sharma, Nick Fernando, Will Hawkins, Behnam Neyshabur, Solomon Kim, Adrian Hutter, Priyanka Agrawal, Alex Castro-Ros, George van den Driessche, Tao Wang, Fan Yang, Shuo yiin Chang, Paul Komarek, Ross McIlroy, Mario Lučić, Guodong Zhang, Wael Farhan, Michael Sharman, Paul Natsev, Paul Michel, Yamini Bansal, Siyuan Qiao, Kris Cao, Siamak Shakeri, Christina Butterfield, Justin Chung, Paul Kishan Rubenstein, Shivani Agrawal, Arthur Mensch, Kedar Soparkar, Karel Lenc, Timothy Chung, Aedan Pope, Loren Maggiore, Jackie Kay, Priya Jhakra, Shibo Wang, Joshua Maynez, Mary Phuong, Taylor Tobin, Andrea Tacchetti, Maja Trebacz, Kevin Robinson, Yash Katariya, Sebastian Riedel, Paige Bailey, Kefan Xiao, Nimesh Ghelani, Lora Aroyo, Ambrose Slone, Neil Houlsby, Xuehan Xiong, Zhen Yang, Elena Gribovskaya, Jonas Adler, Mateo Wirth, Lisa Lee, Music Li, Thais Kagohara, Jay Pavagadhi, Sophie Bridgers, Anna Bortsova, Sanjay Ghemawat, Zafarali Ahmed, Tianqi Liu, Richard Powell, Vijay Bolina, Mariko Iinuma, Polina Zablotskaia, James Besley, Da-Woon Chung, Timothy Dozat, Ramona Comanescu, Xiance Si, Jeremy Greer, Guolong Su, Martin Polacek, Raphaël Lopez Kaufman, Simon Tokumine, Hexiang Hu, Elena Buchatskaya, Yingjie Miao, Mohamed Elhawaty, Aditya Siddhant, Nenad Tomasev, Jinwei Xing, Christina Greer, Helen Miller, Shereen Ashraf, Aurko Roy, Zizhao Zhang, Ada Ma, Angelos Filos, Milos Besta, Rory Blevins, Ted Klimenko, Chih-Kuan Yeh, Soravit Changpinyo, Jiaqi Mu, Oscar Chang, Mantas Pajarskas, Carrie Muir, Vered Cohen, Charline Le Lan, Krishna Haridasan, Amit Marathe, Steven Hansen, Sholto Douglas, Rajkumar Samuel, Mingqiu Wang, Sophia Austin, Chang Lan, Jiepu Jiang, Justin Chiu, Jaime Alonso Lorenzo, Lars Lowe Sjösund, Sébastien Cevey, Zach Gleicher, Thi Avrahami, Anudhyan Boral, Hansa Srinivasan, Vittorio Selo, Rhys May, Konstantinos Aisopos, Léonard Hussenot, Livio Baldini Soares, Kate Baumli, Michael B. Chang, Adrià Recasens, Ben Caine, Alexander Pritzel, Filip Pavetic, Fabio Pardo, Anita Gergely, Justin Frye, Vinay Ramasesh, Dan Horgan, Kartikeya Badola, Nora Kassner, Subhrajit Roy, Ethan Dyer, Víctor Campos Campos, Alex Tomala, Yunhao Tang, Dalia El Badawy, Elspeth White, Basil Mustafa, Oran Lang, Abhishek Jindal, Sharad Vikram, Zhitao Gong, Sergi Caelles, Ross Hemsley, Gregory Thornton, Fangxiaoyu Feng, Wojciech Stokowiec, Ce Zheng, Phoebe Thacker, Çağlar Ünlü, Zhishuai Zhang, Mohammad Saleh, James Svensson, Max Bileschi, Piyush Patil, Ankesh Anand, Roman Ring, Katerina Tsihlas, Arpi Vezer, Marco Selvi, Toby Shevlane, Mikel Rodriguez, Tom Kwiatkowski, Samira Daruki, Keran Rong, Allan Dafoe, Nicholas FitzGerald, Keren Gu-Lemberg, Mina Khan, Lisa Anne Hendricks, Marie Pellat, Vladimir Feinberg, James Cobon-Kerr, Tara Sainath, Maribeth Rauh, Sayed Hadi Hashemi, Richard Ives, Yana Hasson, Eric Noland, Yuan Cao, Nathan Byrd, Le Hou, Qingze Wang, Thibault Sottiaux, Michela Paganini, Jean-Baptiste Lespiau, Alexandre Moufarek, Samer Hassan, Kaushik Shivakumar, Joost van Amersfoort, Amol Mandhane, Pratik Joshi, Anirudh Goyal, Matthew Tung, Andrew Brock, Hannah Sheahan, Vedant Misra, Cheng Li, Nemanja Rakićević, Mostafa Dehghani, Fangyu Liu, Sid Mittal, Junhyuk Oh, Seb Noury, Eren Sezener, Fantine Huot, Matthew Lamm, Nicola De Cao, Charlie Chen, Sidharth Mudgal, Romina Stella, Kevin Brooks, Gautam Vasudevan, Chenxi Liu, Mainak Chain, Nivedita Melinkeri, Aaron Cohen, Venus Wang, Kristie Seymore, Sergey Zubkov, Rahul Goel, Summer Yue, Sai Krishnakumaran, Brian Albert, Nate Hurley, Motoki Sano, Anhad Mohananey, Jonah Joughin, Egor Filonov, Tomasz Kępa, Yomna Eldawy, Jiawern Lim, Rahul Rishi, Shirin Badiezadegan, Taylor Bos, Jerry Chang, Sanil Jain, Sri Gayatri Sundara Padmanabhan, Subha Puttagunta, Kalpesh Krishna, Leslie Baker, Norbert Kalb, Vamsi Bedapudi, Adam Kurzrok, Shuntong Lei, Anthony Yu, Oren Litvin, Xiang Zhou, Zhichun Wu, Sam Sobell, Andrea Siciliano, Alan Papir, Robby Neale, Jonas Bragagnolo, Tej Toor, Tina Chen, Valentin Anklin, Feiran Wang, Richie Feng, Milad Gholami, Kevin Ling, Lijuan Liu, Jules Walter, Hamid Moghaddam, Arun Kishore, Jakub Adamek, Tyler Mercado, Jonathan Mallinson, Siddhinita Wandekar, Stephen Cagle, Eran Ofek, Guillermo Garrido, Clemens Lombriser, Maksim Mukha, Botu Sun, Hafeezul Rahman Mohammad, Josip Matak, Yadi Qian, Vikas Peswani, Pawel Janus, Quan Yuan, Leif Schelin, Oana David, Ankur Garg, Yifan He,

Oleksii Duzhyi, Anton Älgmyr, Timothée Lottaz, Qi Li, Vikas Yadav, Luyao Xu, Alex Chinien, Rakesh Shivanna, Aleksandr Chuklin, Josie Li, Carrie Spadine, Travis Wolfe, Kareem Mohamed, Subhabrata Das, Zihang Dai, Kyle He, Daniel von Dincklage, Shyam Upadhyay, Akanksha Maurya, Luyan Chi, Sebastian Krause, Khalid Salama, Pam G Rabinovitch, Pavan Kumar Reddy M, Aarush Selvan, Mikhail Dektiarev, Golnaz Ghiasi, Erdem Guven, Himanshu Gupta, Boyi Liu, Deepak Sharma, Idan Heimlich Shtacher, Shachi Paul, Oscar Akerlund, François-Xavier Aubet, Terry Huang, Chen Zhu, Eric Zhu, Elico Teixeira, Matthew Fritze, Francesco Bertolini, Liana-Eleonora Marinescu, Martin Bölle, Dominik Paulus, Khyatti Gupta, Tejasi Latkar, Max Chang, Jason Sanders, Roopa Wilson, Xuewei Wu, Yi-Xuan Tan, Lam Nguyen Thiet, Tulsee Doshi, Sid Lall, Swaroop Mishra, Wanming Chen, Thang Luong, Seth Benjamin, Jasmine Lee, Ewa Andrejczuk, Dominik Rabiej, Vipul Ranjan, Krzysztof Styrc, Pengcheng Yin, Jon Simon, Malcolm Rose Harriott, Mudit Bansal, Alexei Robsky, Geoff Bacon, David Greene, Daniil Mirylenka, Chen Zhou, Obaid Sarvana, Abhimanyu Goyal, Samuel Andermatt, Patrick Siegler, Ben Horn, Assaf Israel, Francesco Pongetti, Chih-Wei "Louis" Chen, Marco Selvatici, Pedro Silva, Kathie Wang, Jackson Tolins, Kelvin Guu, Roey Yogev, Xiaochen Cai, Alessandro Agostini, Maulik Shah, Hung Nguyen, Noah Ó Donnaile, Sébastien Pereira, Linda Friso, Adam Stambler, Adam Kurzrok, Chenkai Kuang, Yan Romanikhin, Mark Geller, ZJ Yan, Kane Jang, Cheng-Chun Lee, Wojciech Fica, Eric Malmi, Qijun Tan, Dan Banica, Daniel Balle, Ryan Pham, Yanping Huang, Diana Avram, Hongzhi Shi, Jasjot Singh, Chris Hidey, Niharika Ahuja, Pranab Saxena, Dan Dooley, Srividya Pranavi Potharaju, Eileen O'Neill, Anand Gokulchandran, Ryan Foley, Kai Zhao, Mike Dusenberry, Yuan Liu, Pulkit Mehta, Ragha Kotikalapudi, Chalence Safranek-Shrader, Andrew Goodman, Joshua Kessinger, Eran Globen, Prateek Kolhar, Chris Gorgolewski, Ali Ibrahim, Yang Song, Ali Eichenbaum, Thomas Brovelli, Sahitya Potluri, Preethi Lahoti, Cip Baetu, Ali Ghorbani, Charles Chen, Andy Crawford, Shalini Pal, Mukund Sridhar, Petru Gurita, Asier Mujika, Igor Petrovski, Pierre-Louis Cedoz, Chenmei Li, Shiyuan Chen, Niccolò Dal Santo, Siddharth Goyal, Jitesh Punjabi, Karthik Kappaganthu, Chester Kwak, Pallavi LV, Sarmishta Velury, Himadri Choudhury, Jamie Hall, Premal Shah, Ricardo Figueira, Matt Thomas, Minjie Lu, Ting Zhou, Chintu Kumar, Thomas Jurdi, Sharat Chikkerur, Yenai Ma, Adams Yu, Soo Kwak, Victor Ähdel, Sujeevan Rajayogam, Travis Choma, Fei Liu, Aditya Barua, Colin Ji, Ji Ho Park, Vincent Hellendoorn, Alex Bailey, Taylan Bilal, Huanjie Zhou, Mehrdad Khatir, Charles Sutton, Wojciech Rzadkowski, Fiona Macintosh, Roopali Vij, Konstantin Shagin, Paul Medina, Chen Liang, Jinjing Zhou, Pararth Shah, Yingying Bi, Attila Dankovics, Shipra Banga, Sabine Lehmann, Marissa Bredesen, Zifan Lin, John Eric Hoffmann, Jonathan Lai, Raynald Chung, Kai Yang, Nihal Balani, Arthur Bražinskas, Andrei Sozanschi, Matthew Hayes, Héctor Fernández Alcalde, Peter Makarov, Will Chen, Antonio Stella, Liselotte Snijders, Michael Mandl, Ante Kärrman, Paweł Nowak, Xinyi Wu, Alex Dyck, Krishnan Vaidyanathan, Raghavender R, Jessica Mallet, Mitch Rudominer, Eric Johnston, Sushil Mittal, Akhil Udathu, Janara Christensen, Vishal Verma, Zach Irving, Andreas Santucci, Gamaleldin Elsayed, Elnaz Davoodi, Marin Georgiev, Ian Tenney, Nan Hua, Geoffrey Cideron, Edouard Leurent, Mahmoud Alnahlawi, Ionut Georgescu, Nan Wei, Ivy Zheng, Dylan Scandinaro, Heinrich Jiang, Jasper Snoek, Mukund Sundararajan, Xuezhi Wang, Zack Ontiveros, Itay Karo, Jeremy Cole, Vinu Rajashekhar, Lara Tumeh, Eyal Ben-David, Rishub Jain, Jonathan Uesato, Romina Datta, Oskar Bunyan, Shimu Wu, John Zhang, Piotr Stanczyk, Ye Zhang, David Steiner, Subhajit Naskar, Michael Azzam, Matthew Johnson, Adam Paszke, Chung-Cheng Chiu, Jaume Sanchez Elias, Afroz Mohiuddin, Faizan Muhammad, Jin Miao, Andrew Lee, Nino Vieillard, Jane Park, Jiageng Zhang, Jeff Stanway, Drew Garmon, Abhijit Karmarkar, Zhe Dong, Jong Lee, Aviral Kumar, Luowei Zhou, Jonathan Evens, William Isaac, Geoffrey Irving, Edward Loper, Michael Fink, Isha Arkatkar, Nanxin Chen, Izhak Shafran, Ivan Petrychenko, Zhe Chen, Johnson Jia, Anselm Levskaya, Zhenkai Zhu, Peter Grabowski, Yu Mao, Alberto Magni, Kaisheng Yao, Javier Snaider, Norman Casagrande, Evan Palmer, Paul Suganthan, Alfonso Castaño, Irene Giannoumis, Wooyeol Kim, Mikołaj Rybiński, Ashwin Sreevatsa, Jennifer Prendki, David Soergel, Adrian Goedeckemeyer, Willi Gierke, Mohsen Jafari, Meenu Gaba, Jeremy Wiesner, Diana Gage Wright, Yawen Wei, Harsha Vashisht, Yana Kulizhskaya, Jay Hoover, Maigo Le, Lu Li, Chimezie Iwuanyanwu, Lu Liu, Kevin Ramirez, Andrey Khorlin, Albert Cui, Tian LIN, Marcus Wu, Ricardo Aguilar, Keith Pallo, Abhishek Chakladar, Ginger Perng, Elena Allica Abellan, Mingyang Zhang, Ishita Dasgupta, Nate Kushman, Ivo Penchev, Alena Repina, Xihui Wu, Tom van der Weide, Priya Ponnapalli, Caroline Kaplan, Jiri Simsa, Shuangfeng Li, Olivier Dousse, Fan Yang, Jeff Piper, Nathan Ie, Rama Pasumarthi, Nathan Lintz, Anitha Vijayakumar, Daniel Andor, Pedro Valenzuela, Minnie Lui, Cosmin Paduraru, Daiyi Peng, Katherine Lee, Shuyuan Zhang, Somer Greene, Duc Dung Nguyen, Paula Kurylowicz, Cassidy Hardin, Lucas Dixon, Lili

Janzer, Kiam Choo, Ziqiang Feng, Biao Zhang, Achintya Singhal, Dayou Du, Dan McKinnon, Natasha Antropova, Tolga Bolukbasi, Orgad Keller, David Reid, Daniel Finchelstein, Maria Abi Raad, Remi Crocker, Peter Hawkins, Robert Dadashi, Colin Gaffney, Ken Franko, Anna Bulanova, Rémi Leblond, Shirley Chung, Harry Askham, Luis C. Cobo, Kelvin Xu, Felix Fischer, Jun Xu, Christina Sorokin, Chris Alberti, Chu-Cheng Lin, Colin Evans, Alek Dimitriev, Hannah Forbes, Dylan Banarse, Zora Tung, Mark Omernick, Colton Bishop, Rachel Sterneck, Rohan Jain, Jiawei Xia, Ehsan Amid, Francesco Piccinno, Xingyu Wang, Praseem Banzal, Daniel J. Mankowitz, Alex Polozov, Victoria Krakovna, Sasha Brown, MohammadHossein Bateni, Dennis Duan, Vlad Firoiu, Meghana Thotakuri, Tom Natan, Matthieu Geist, Ser tan Girgin, Hui Li, Jiayu Ye, Ofir Roval, Reiko Tojo, Michael Kwong, James Lee-Thorp, Christopher Yew, Danila Sinopalnikov, Sabela Ramos, John Mellor, Abhishek Sharma, Kathy Wu, David Miller, Nicolas Sonnerat, Denis Vnukov, Rory Greig, Jennifer Beattie, Emily Caveness, Libin Bai, Julian Eisenschlos, Alex Korchemniy, Tomy Tsai, Mimi Jasarevic, Weize Kong, Phuong Dao, Zeyu Zheng, Frederick Liu, Fan Yang, Rui Zhu, Tian Huey Teh, Jason Sanmiya, Evgeny Gladchenko, Nejc Trdin, Daniel Toyama, Evan Rosen, Sasan Tavakkol, Linting Xue, Chen Elkind, Oliver Woodman, John Carpenter, George Papamakarios, Rupert Kemp, Sushant Kafle, Tanya Grunina, Rishika Sinha, Alice Talbert, Diane Wu, Denese Owusu-Afriyie, Cosmo Du, Chloe Thornton, Jordi Pont-Tuset, Pradyumna Narayana, Jing Li, Saaber Fatehi, John Wieting, Omar Ajmeri, Benigno Uria, Yeongil Ko, Laura Knight, Amélie Héliou, Ning Niu, Shane Gu, Chenxi Pang, Yeqing Li, Nir Levine, Ariel Stolovich, Rebeca Santamaria-Fernandez, Sonam Goenka, Wenny Yustalim, Robin Strudel, Ali Elqursh, Charlie Deck, Hyo Lee, Zonglin Li, Kyle Levin, Raphael Hoffmann, Dan Holtmann-Rice, Olivier Bachem, Sho Arora, Christy Koh, Soheil Hassas Yeganeh, Siim Põder, Mukarram Tariq, Yanhua Sun, Lucian Ionita, Mojtaba Seyedhosseini, Pouya Tafti, Zhiyu Liu, Anmol Gulati, Jasmine Liu, Xinyu Ye, Bart Chrzaszcz, Lily Wang, Nikhil Sethi, Tianrun Li, Ben Brown, Shreya Singh, Wei Fan, Aaron Parisi, Joe Stanton, Vinod Koverkathu, Christopher A. Choquette-Choo, Yunjie Li, TJ Lu, Abe Ittycheriah, Prakash Shroff, Mani Varadarajan, Sanaz Bahargam, Rob Willoughby, David Gaddy, Guillaume Desjardins, Marco Cornero, Brona Robenek, Bhavishya Mittal, Ben Albrecht, Ashish Shenoy, Fedor Moiseev, Henrik Jacobsson, Alireza Ghaffarkhah, Morgane Rivière, Alanna Walton, Clément Crepy, Alicia Parrish, Zongwei Zhou, Clement Farabet, Carey Radebaugh, Praveen Srinivasan, Claudia van der Salm, Andreas Fidjeland, Salvatore Scellato, Eri Latorre-Chimoto, Hanna Klimczak-Plucińska, David Bridson, Dario de Cesare, Tom Hudson, Piermaria Mendolicchio, Lexi Walker, Alex Morris, Matthew Mauger, Alexey Guseynov, Alison Reid, Seth Odoom, Lucia Loher, Victor Cotruta, Madhavi Yenugula, Dominik Grewe, Anastasia Petrushkina, Tom Duerig, Antonio Sanchez, Steve Yadlowsky, Amy Shen, Amir Globerson, Lynette Webb, Sahil Dua, Dong Li, Surya Bhupatiraju, Dan Hurt, Haroon Qureshi, Ananth Agarwal, Tomer Shani, Matan Eyal, Anuj Khare, Shreyas Rammohan Belle, Lei Wang, Chetan Tekur, Mihir Sanjay Kale, Jinliang Wei, Ruoxin Sang, Brennan Saeta, Tyler Liechty, Yi Sun, Yao Zhao, Stephan Lee, Pandu Nayak, Doug Fritz, Manish Reddy Vuyyuru, John Aslanides, Nidhi Vyas, Martin Wicke, Xiao Ma, Evgenii Eltyshev, Nina Martin, Hardie Cate, James Manyika, Keyvan Amiri, Yelin Kim, Xi Xiong, Kai Kang, Florian Luisier, Nilesh Tripuraneni, David Madras, Mandy Guo, Austin Waters, Oliver Wang, Joshua Ainslie, Jason Baldridge, Han Zhang, Garima Pruthi, Jakob Bauer, Feng Yang, Riham Mansour, Jason Gelman, Yang Xu, George Polovets, Ji Liu, Honglong Cai, Warren Chen, XiangHai Sheng, Emily Xue, Sherjil Ozair, Christof Angermueller, Xiaowei Li, Anoop Sinha, Weiren Wang, Julia Wiesinger, Emmanouil Koukoumidis, Yuan Tian, Anand Iyer, Madhu Gurumurthy, Mark Goldenson, Parashar Shah, MK Blake, Hongkun Yu, Anthony Urbanowicz, Jennimaria Palomaki, Chrisantha Fernando, Ken Durden, Harsh Mehta, Nikola Momchev, Elahe Rahimtoroghi, Maria Georgaki, Amit Raul, Sebastian Ruder, Morgan Redshaw, Jinhyuk Lee, Denny Zhou, Komal Jalan, Dinghua Li, Blake Hechtman, Parker Schuh, Milad Nasr, Kieran Milan, Vladimir Mikulik, Juliana Franco, Tim Green, Nam Nguyen, Joe Kelley, Aroma Mahendru, Andrea Hu, Joshua Howland, Ben Vargas, Jeffrey Hui, Kshitij Bansal, Vikram Rao, Rakesh Ghiya, Emma Wang, Ke Ye, Jean Michel Sarr, Melanie Moranski Preston, Madeleine Elish, Steve Li, Aakash Kaku, Jigar Gupta, Ice Pasupat, Da-Cheng Juan, Milan Someswar, Tejvi M., Xinyun Chen, Aida Amini, Alex Fabrikant, Eric Chu, Xuanyi Dong, Amruta Muthal, Senaka Buthpitiya, Sarthak Jauhari, Nan Hua, Urvashi Khandelwal, Ayal Hitron, Jie Ren, Larissa Rinaldi, Shahar Drath, Avigail Dabush, Nan-Jiang Jiang, Harshal Godhia, Uli Sachs, Anthony Chen, Yicheng Fan, Hagai Taitelbaum, Hila Noga, Zhuyun Dai, James Wang, Chen Liang, Jenny Hamer, Chun-Sung Ferng, Chenel Elkind, Aviel Atias, Paulina Lee, Vít Listík, Mathias Carlen, Jan van de Kerkhof, Marcin Pikus, Krunoslav Zaher, Paul Müller, Sasha Zykova, Richard Stefanec, Vitaly Gatsko, Christoph Hirnschall, Ashwin Sethi, Xingyu Federico Xu, Chetan Ahuja, Beth Tsai,

Anca Stefanoiu, Bo Feng, Keshav Dhandhania, Manish Katyal, Akshay Gupta, Atharva Parulekar, Divya Pitta, Jing Zhao, Vivaan Bhatia, Yashodha Bhavnani, Omar Alhadlaq, Xiaolin Li, Peter Danenberg, Dennis Tu, Alex Pine, Vera Filippova, Abhipso Ghosh, Ben Limonchik, Bhargava Urala, Chaitanya Krishna Lanka, Derik Clive, Yi Sun, Edward Li, Hao Wu, Kevin Hongtongsak, Ianna Li, Kalind Thakkar, Kuanysh Omarov, Kushal Majmundar, Michael Alverson, Michael Kucharski, Mohak Patel, Mudit Jain, Maksim Zabelin, Paolo Pelagatti, Rohan Kohli, Saurabh Kumar, Joseph Kim, Swetha Sankar, Vineet Shah, Lakshmi Ramachandruni, Xiangkai Zeng, Ben Bariach, Laura Weidinger, Tu Vu, Alek Andreev, Antoine He, Kevin Hui, Sheleem Kashem, Amar Subramanya, Sissie Hsiao, Demis Hassabis, Koray Kavukcuoglu, Adam Sadovsky, Quoc Le, Trevor Strohman, Yonghui Wu, Slav Petrov, Jeffrey Dean, and Oriol Vinyals. Gemini: A family of highly capable multimodal models, 2025. URL `https://arxiv.org/abs/2312.11805`.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido,

Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL `https://arxiv.org/abs/2312.00752`.

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. *arXiv preprint arXiv:2410.10781*, 2024.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*, 2024.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `https://doi.org/10.1162/neco.1997.9.8.1735`.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Praneeth Kacham, Vahab Mirrokni, and Peilin Zhong. Polysketchformer: Fast transformers via sketching polynomial kernels, 2024. URL https://arxiv.org/abs/2310.01655.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020. URL https://arxiv.org/abs/2006.16236.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

Zhixuan Lin, Evgenii Nikishin, Xu Owen He, and Aaron Courville. Forgetting transformer: Softmax attention with a forget gate. *arXiv preprint arXiv:2503.02130*, 2025.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Meta. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/, 2025. Accessed: 2025-04-05.

Maxim Milakov and Natalia Gimelshein. Online normalizer calculation for softmax, 2018. URL https://arxiv.org/abs/1805.02867.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilimbi, Benoit Prabhakaran, Michael Rabbat, Zachary Taylor, and Vladislav Petrov. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Jiaju Lin, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Bolun Wang, Johan S. Wind, Stanislaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. Rwkv: Reinventing rnns for the transformer era, 2023. URL https://arxiv.org/abs/2305.13048.

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.

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

Jason Ramapuram, Federico Danieli, Eeshan Dhekane, Floris Weers, Dan Busbridge, Pierre Ablin, Tatiana Likhomanenko, Jagrit Digani, Zijin Gu, Amitis Shidani, et al. Theory, analysis, and best practices for sigmoid self-attention. *arXiv preprint arXiv:2409.04431*, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers, 2021. URL `https://arxiv.org/abs/2102.11174`.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

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.

Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL `https://arxiv.org/abs/1706.03762`.

Shanshan Wang, Gaurav Naithani, Archontis Politis, and Tuomas Virtanen. Deep neural network based low-latency speech separation with asymmetric analysis-synthesis window pair. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pages 301–305. IEEE, 2021.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Songlin Yang and Yu Zhang. FLA: A Triton-Based Library for Hardware-Efficient Implementations of Linear Attention Mechanism, January 2024. URL `https://github.com/fla-org/flash-linear-attention`.

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.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule, 2025. URL `https://arxiv.org/abs/2412.06464`.

Yu Zhang, Songlin Yang, Rui-Jie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, et al. Gated slot attention for efficient linear-time sequence modeling. *Advances in Neural Information Processing Systems*, 37:116870–116898, 2024.

A Derivation of chunked algorithm

Here we we prove that the *chunked form* of linear attention is equivalent to the *attention form*. Recall that the chunk-form says

$$Y_{(i)_c} = S_{ci} Q_{(i)_c} + V_{(i)_c} \left(Q_{(i)_c} K_{(i)_c}^T \odot M \right) \qquad S_{c(i+1)} = S_{ci} + V_{(i)_c} K_{(i)_c}^T \tag{11}$$

Because this is in matrix form, if we look at output at each position i, it becomes

$$Y_i = S_{ci}Q_i + \sum_{j=\lfloor \frac{i}{c} \rfloor c+1}^{i} (Q_i K_j^T)V_j \tag{12}$$

$$= (S_{c(i-1)} + V_{(i)_c} K_{(i)_c}^T)Q_i + \sum_{j=\lfloor \frac{i}{c} \rfloor c+1}^{i} (Q_i K_j^T)V_j \tag{13}$$

$$= S_{c(i-1)}Q_i + \sum_{j=\lfloor \frac{i-1}{c} \rfloor c+1}^{\lfloor \frac{i}{c} \rfloor c} V_j K_j^T Q_i + \sum_{j=\lfloor \frac{i}{c} \rfloor c+1}^{i} (Q_i K_j^T)V_j \tag{14}$$

$$= S_{c(i-1)}Q_i + \sum_{j=\lfloor \frac{i-1}{c} \rfloor c+1}^{\lfloor \frac{i}{c} \rfloor c} (K_j^T Q_i)V_j + \sum_{j=\lfloor \frac{i}{c} \rfloor c+1}^{i} (Q_i K_j^T)V_j \tag{15}$$

$$= S_{c(i-1)}Q_i + \sum_{j=\lfloor \frac{i-1}{c} \rfloor c+1}^{\lfloor \frac{i}{c} \rfloor c} (Q_i K_j^T)V_j + \sum_{j=\lfloor \frac{i}{c} \rfloor c+1}^{i} (Q_i K_j^T)V_j \tag{16}$$

$$= S_{c(i-1)}Q_i + \sum_{j=\lfloor \frac{i-1}{c} \rfloor c+1}^{i} (Q_i K_j^T)V_j \tag{17}$$

$$\vdots \tag{18}$$

$$= S_0 Q_i + \sum_{j=\lfloor \frac{0}{c} \rfloor c+1}^{i} (Q_i K_j^T)V_j \tag{19}$$

$$= \sum_{j=1}^{i} (Q_i K_j^T)V_j \quad \text{assuming initial state is 0} \tag{20}$$

This concludes the proof.

B Derivation of SPOW

B.1 Tensor product and tensor power

A convenient way to define the tensor product is, given vectors $x, y \in \mathbb{R}^d$, their tensor product $x \otimes y = xy^T \in \mathbb{R}^{d \times d}$. The generic tensor product of p vectors in $\mathbb{R}^d$ can be written as $T = \bigotimes_{k=1}^{p} x_k \in \mathbb{R}^{d \times \cdots \times d}$ where, evaluated at a multi-index $(i_1 \cdots i_p) \in \mathbb{N}_d^{\times p}$, the tensor T has value $T_i = \prod_k x_{k,i_k}$. (For example, if $T = a \otimes b \otimes c$ then $T_{1,2,3} = a_1 b_2 c_3$.)

In this work, a central focus is on the pth *tensor power*, defined as taking the tensor product of a vector with itself p times, which we denote using $x^{\otimes p}$. We can define the helpful $\text{TPOW}(x, p) = \text{flat}\,(x^{\otimes p}) \in \mathbb{R}^{d^p}$, which gives us the flattened tensor power as a vector.

$$\text{TPOW}(x, p) = \begin{bmatrix} x_1 \cdots x_1 \\ x_1 \cdots x_2 \\ \vdots \\ x_d \cdots x_d \end{bmatrix} = \begin{bmatrix} \vdots \\ \prod_k x_{i_k} \\ \vdots \end{bmatrix}_{(i_1, \cdots, i_p) \in \mathbb{N}_d^{\times p}} \tag{21}$$

The central property that makes TPOW useful to us is:

$$\text{TPOW}(x,p)^T \text{TPOW}(y,p) = \sum_{(i_1,\cdots)\in\mathbb{N}_d^{\times p}} x_{i_1}\cdots x_{i_p} y_{i_1}\cdots y_{i_p} \tag{22}$$

$$= \sum_{i_1\in\mathbb{N}_d} x_{i_1} y_{i_1} \sum_{i_2\in\mathbb{N}_d} x_{i_2} y_{i_2} \cdots \tag{23}$$

$$= (x^T y)^p \tag{24}$$

Thus, letting $\phi = \text{TPOW}$ we see that power attention can be expressed as a special case of linear attention, $Y_i^{\text{attn}_{\text{pow}}^p} = Y_i^{\text{attn}_{\text{lin}}^{\text{TPOW}(\cdot,p)}}$. Power attention therefore inherits all of the desirable properties of linear attention described in Section 2, including a constant-size state and parallelizable chunked form. TPOW is a state expansion, mapping keys and queries into $\mathbb{R}^{d^p}$, and so power attention possesses a state of size $d^p v$.

B.2 Symmetric power

Here we prove that symmetric power SPOW is a mathematically equivalent state expansion function to TPOW.

Recall from Lemma 4.2 that

$$\text{SPOW}_p(x) = \begin{bmatrix} \vdots \\ \sqrt{\frac{p!}{\text{hist}_k(i)!}} \prod_k x_{i_k} \\ \vdots \end{bmatrix}_{i\in NDMI_d^p} \tag{25}$$

Where each $i = (i_1, ..., i_p)$ is the set of non-decreasing-multi-indices that determines a given entry in the embedded vector. One can use a different set of multi-indices $\alpha = (\alpha_1, ..., \alpha_d)$ to represent the same embedding, where

$$\alpha_j = \begin{cases} 1 & \text{if } \exists k \in \{1,2,...,p\}, \text{s.t.} i_k = j \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

in other words, we can include all the dimensions of x in each entry of the expanded vector, and mask out unnecessary dimension by raising them to a power of 0.

With this setup, let $x, y \in \mathbb{R}^d$, $\text{SPOW}_p(x) \in \mathbb{R}^{\binom{d+p-1}{p}}$ be the symmetric power embedding indexed by the multi-indixes $\alpha = (\alpha_1, ...\alpha_d)$, satisfying $\sum_{i=1}^d \alpha_i = p$.

$$\text{SPOW}_p(x)_\alpha = \sqrt{\frac{p!}{\alpha_1!\cdots\alpha_d!}}\, x_1^{\alpha_1}\cdots x_d^{\alpha_d}. \tag{27}$$

Then, by the multinomial theorem

$$\left\langle \text{SPOW}_p(x),\, \text{SPOW}_p(y) \right\rangle = \sum_\alpha \frac{p!}{\alpha_1!\cdots\alpha_d!}\, x_1^{\alpha_1}\cdots x_d^{\alpha_d}\, y_1^{\alpha_1}\cdots y_d^{\alpha_d} \tag{28}$$

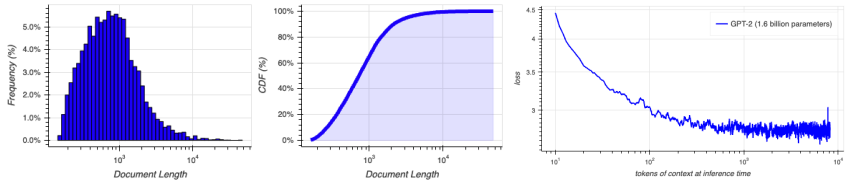
$$= \sum_\alpha \frac{p!}{\alpha_1!\cdots\alpha_d!} (x_1y_1)^{\alpha_1}\cdots(x_dy_d)^{\alpha_d} \tag{29}$$

$$= \left(x_1y_1 + \cdots + x_dy_d\right)^p \tag{30}$$

$$= (x^T y)^p \tag{31}$$

Therefore

$$\langle \text{SPOW}_p(x), \text{SPOW}_p(y)\rangle = (x^T y)^p = \langle \text{TPOW}_p(x), \text{TPOW}_p(y)\rangle \tag{32}$$



(a) Openwebtext document length distribution. (b) GPT-2 trained with 8192 context.

Figure 10

C LongCrawl64

LongCrawl64 [Buckman, 2024] consists of 6,661,465 pre-tokenized documents, each of which is 65,536 tokens long, for a total token count of 435 billion. The data is sourced from the Common Crawl, a typical source for language modeling datasets, but pretokenized and filtered down to only include long sequences. This is a prerequisite for in-context learning. In Figure C, it is clear that there is little potential for in-context learning beyond the lengths of the majority of documents in the dataset. The figure above and the discussion on the need for long documents were originally presented by Buckman and Gelada [b].

D Experimental details

Our experiments were implemented in PyTorch [Paszke et al., 2019], based around the FLA codebase [Yang and Zhang, 2024] whose implementations of all architectures we use. Since this work is focused specifically on the attention layer, we typically separate out the architecture (which we consider everything *except* the attention) from the attention itself. Also, since comparing loss between models of different context lengths can be nuanced, we report the best-context loss as described by Buckman and Gelada [b] whenever plotting a scalar-valued loss for a training curve. When selecting the best context, and also when plotting in-context learning curves, we smooth across the sequence dimension by binning, using exponentially-growing bins so that the bin widths are equal on a log plot.

Unless otherwise noted, we used the following hyperparameters for training: LongCrawl64 (train set for training and report losses on the heldout set), batch size 32, context length 1024, learning rate 3e-4 with a 2000 step warmup from 0 and a cosine decay over the full range of training to 1e-5, one epoch of training (or until convergence), AdamW with weight decay of .1, beta1 of .9, beta2 of .999, gradient clipping of 1, bf16 training, and activation checkpointing for memory reduction. We use a common set of model sizes, taken from Radford et al. [2019]. Small models have width 768, 12 hidden layers, 12 heads, and a MLP ratio of 4. Medium models have width 1024, 24 hidden layers, 16 heads, and a MLP ratio of 4. Large models have width 1280, 36 hidden layers, 20 heads, and a MLP ratio of 4. For experiments involving the RWKV architecture, we use RWKV7.

Figure 1 and Figure 3 contain a variety of architectures, labeled in the legend; all are small models with default hyperparameters. Figure 4 contains two small RWKV models with default hyperparameters and differing attention layers.

In Figure 2, all three curves are RWKV + attention models. The model with 1:12 WSFR has 6 heads, 8 layers, head size of 64, hidden dimension of 512, a context of 65536, and a batch size of 32. The model with 4:1 WSFR is a medium model with batch size 512 and context size 4096. The model with 99:1 WSFR has 26 layers, 20 heads, head size of 64, hidden dimension of 1280, batch size 32768, and context size 64.

Figure 7 and Table 4 contain the resuls of experiments run using the RWKV architeture at large (750M) size. The transformer was run using a batch size of 7680 and context of 1024. The other runs used batch size 1920 and context size 4096.

Figure 8a consists of one large $p = 2$ transformer. Figure 8b consists of small $p = 2$ transformer models evaluated at iteration 64k. Figure 8c consists of large $p = 2$ transformers evaluated at iteration 64k. The curve corresponding to the smallest model scale has 512 width, 6 layers, and 8 heads; the

other three are small, medium, and large models. Figure 8d involves a sweep over context lengths on a $p = 2$ transformer with 512 width, 6 layers, and 8 heads, evaluated at 170k iterations. The grey curves are the same, but after sampling documents (of any length), the data is reshaped to have a block size of 1024 before training. This means the actual tokens are kept constant, but the model has no opportunity to use long context to learn.

Figure 9a shows three small RWKV models with different attention layers, trained on context length of 65536 tokens at a batch size of 32.

All experiments in this work were run on Nvidia H100 GPUs, typically on nodes of 8 GPUs. In the course of conducting the experimental portion of this work, we had access to between 32 and 300 H100s for two months. The majority of the compute was spent on research, with only about 20% of compute spent on experiments in this paper.

E Power attention is compute-optimal under inference latency constraints

To a first-order approximation, inference latency is proportional to the sum of the parameter count and the state size. This is relevant when choosing an optimal training strategy for a model whose ultimate usage will have inference constraints, for example a speech model model [Wang et al., 2021]. A reasonable approach is to choose the parameter count and state size to be at some tolerable scale, and spend the compute budget scaling other axes of training, such as batch size and context length.

In this setting, we can compare the best achievable performance of a transformer to that of other models, keeping all variants equivalent in terms of state size, parameter count, tokens per update, and total FLOPs. In Table 4, we construct three such models, and compare the final best-context loss. The transformer is trained on batches of 7680 document of length 1024 (a context length which keeps its state size equivalent to the other approaches), while the windowed transformer and power attention transformer use batches of 1920 documents of length 4096. Power attention is the only architecture to outperform the transformer. See Appendix D for experimental details.

	Best context length	Loss
Window (1k)	1878	1.638
Attention	1024	1.631
Power ($p = 2$)	4096	**1.613**

Table 4: Power attention is compute-optimal given sufficient train FLOPs, when inference latency is equal.

F Algorithms in power attention

In this section we present the four algorithms used in power attention.

F.1 Attention

As shown below, the *attention* kernel in Power Attention is very similar to Flash Attention, apart from an extra step of log-space power and an extra output for the normalization term l. We chose to raise the attention score matrix to a power p in the log space because it is more numeric stable than performing the power operation directly.

F.2 Update state

The *update-state* operation concerns with creating a new state $S_{i+1} \in R^{D\times v}$ based on the past state $S_i \in R^{D\times v}$ and all the keys $K_i \in R^{c\times d}$ and values $V_i \in R^{c\times v}$ in the current chunk. There are many variants to this formulation in modern RNNs. Specifically, past state S_i are usually gated with a decay factor γ_i, which often depend on input as well.

$$\text{update-state}(S_i, K_i, V_i) = S_i + \phi(K_i)^T V_i \tag{33}$$

$$\text{gated-update-state}(S_i, K_i, V_i) = S_i \odot \gamma_i + \phi(K_i)^T V_i \tag{34}$$

Regardless the exact state evolution formula, the fusion of state expansion and a subsequent matrix multiplication is the fundamental building block, which we termed **fused spow-mma (expand M)** kernel. We use **expand M** here as **M, N, K** are commonly used to denote the 3 dimensions of a

Step	Flash Attention	Power Attention (attention form)
1. Query-Key Inner Product	$S = QK^T$	$S = QK^T$
2. Softmax Scaling	$S = S \odot \text{scale}$	$S = S \odot \text{scale}$
3. Log-space Power		$S = p \log(\lvert S \rvert + \epsilon)$
4. Row max scaling	$S = S - \text{rowmax}(S)$	$S = S - \text{rowmax}(S)$
5. Masked Exponential	$P = \exp(S \odot M)$	$P = \exp(S \odot M)$
6. Normalization	$P = P \odot D^{-1}(\text{rowsum}(P))$	$\zeta = \text{rowsum}(P)$
7. Matmul with Value	$O = PV$	$O = PV$
8. Output	O	O, ζ

Table 5: Procedural comparison between Flash Attention and Power Attention (attention form). $Q, K \in R^{t \times d}$, $V \in R^{t \times v}$; p stands for the degree of power; ϵ is a small constant to avoid taking the log of zero; rowmax(P) refers to the operation of taking the max of the $t \times t$ attention score matrix, an often-used techniques for stabilizing softmax [Milakov and Gimelshein, 2018]; rowsum(P) refers to the operation of summing up each row of the softmax matrix; D^{-1} refers to the operation of converting a vector into a diagonal matrix and take its inverse; ζ is the normalization term (sum of attention scores) used for combining attention output and *query-state* output

matrix multiplication problem, and this kernel is expanding the state along $\mathbf{M}$ axis ($K_i^T \in R^{d \times c} \to \phi(K_i)^T \in R^{D \times c}$). We might also use the term *update-state* interchangeably with **fused spow-mma (expand M)** kernel, as the gated summation is done in the *discumsum* kernel. Note that in practice, the *update-state* kernel would also produce a normalization term (a.k.a. sum of expanded keys) γ, which is used to combine the outputs of chunked attention and *query-state* such that the output is normalized. We denote this kernel with *fused-update-state*.

$$\text{fused-update-state}(S_i, K_i, V_i) = (\phi(K_i)^T V_i, \phi(K_i)^T \mathbf{1}) \tag{35}$$

Algorithm 1 Fused Update State

Require: Matrices A of size $\mathbf{d} \times \mathbf{K}$, B of size $\mathbf{K} \times \mathbf{N}$
Ensure: Output matrix C of size $\mathbf{D} \times \mathbf{N}$, normalization factor γ of size $\mathbf{D}$

Define degree of power $\mathbf{p}$, tile size for expansion $\mathbf{d_{tile}}$, expanded tile size $\mathbf{D_{tile}} = \mathbf{d_{tile}}^{\mathbf{p}}$
Denote the ordered list of non-decreasing-multi-indices $\mathbf{NDMI}^{\mathbf{p}}_{\mathbf{d}/\mathbf{d_{tile}}}$ with λ, of size $\mathbf{L} \times \mathbf{p}$
Divide A into $N_A = \lceil \frac{\mathbf{K}}{\mathbf{TK}} \rceil$ tiles, $A_1, ..., A_{N_A}$, each of size $\mathbf{d} \times \mathbf{TK}$; divide each A_k further into $\mathbf{N_d} = \frac{\mathbf{d}}{\mathbf{d_{tile}}}$ subtiles, $A_k^1, ..., A_k^{\mathbf{d}/\mathbf{d_{tile}}}$, each of size $\mathbf{d_{tile}} \times \mathbf{TK}$
Divide B into $N_B = \lceil \frac{\mathbf{N}}{\mathbf{TN}} \rceil$ tiles, $B_1, ..., B_{N_B}$, each of size $\mathbf{K} \times \mathbf{TN}$; divide each B_j further into $\lceil \frac{\mathbf{K}}{\mathbf{TK}} \rceil$ subtitles, $B_j^1, ..., B_j^{\lceil \mathbf{K}/\mathbf{TK} \rceil}$, each of size $\mathbf{TK} \times \mathbf{TN}$
for $1 \le l \le \mathbf{L}$, in parallel **do**
    for $1 \le j \le N_B$, in parallel **do**
        Initialize accumulation registers: $C_{l,j} \leftarrow 0$ of shape $\mathbf{D_{tile}} \times \mathbf{TN}$, D
        Initialize register for matrix multiplication $\hat{A}_k \leftarrow 0$ of shape $\mathbf{D_{tile}} \times \mathbf{TK}$
        Initialize register for normalization factor: γ_l of shape $\mathbf{D_{tile}}$
        for $1 \le k \le \lceil \frac{\mathbf{K}}{\mathbf{TK}} \rceil$ **do**
            Load A_k from global memory to on-chip SRAM
            Load B_j^k from global memory to on-chip SRAM
            for $1 \le z \le \mathbf{p}$ **do**
                Load $A_k^{\lambda(l,z)}$ from on-chip SRAM into registers
            end for
            $\hat{A}_k \leftarrow A_k^{\lambda(l,1)} \otimes \cdots \otimes A_k^{\lambda(l,\mathbf{p})}$
            $\gamma_l \leftarrow \text{rowsum}(\hat{A}^k) + \gamma_l$
            $C_{l,j} \leftarrow \hat{A}_k B_j^k + C_{l,j}$
        end for
        Write $C_{l,j}, \gamma_l$ to global memory
    end for
end for

F.3 Discumsum

The *discumsum* operation involves discounting and accumulative-summing states $S \in R^{n \times D \times d}$ for each chunk in a sequence (hence the name), where $n = \lceil \frac{t}{c} \rceil$. It takes the output produced by *update-state* kernel, and a gating factor $\lambda \in R^n$ and produced the discounted accumulative sum. Discounting is necessary when gating is involved. The discumsum kernel used in in paper was implemented by a custom CUDA kernel.

$$\text{discumsum}(S, \lambda) = \begin{bmatrix} S_1 \\ S_1 \odot \lambda_1 + S_2 \\ \cdots \\ S_1 \odot \lambda_1 + \cdots + S_i \odot \prod_{j=1}^{i} \lambda_j + S_n \end{bmatrix} \tag{36}$$

F.4 Query state

The *query-state* kernel involves querying the past state S_i using the queries $Q_i \in R^{c \times d}$ in the current chunk.

$$\text{query-state}(S_i, Q_i) = \phi(Q) S_i \tag{37}$$

Notice that as opposed to the *update-state* kernel, the *query-state* kernel expands the queries along the dimension of reduction in matrix multiplication. Therefore in the inner loop of the kernel, we go through all the nondecreasing-multi-indices.

In practice, we also fuse the summation of the intra-chunk output from attention $Y \in R^{c \times v}$ and $\phi(Q)S_i$ into the *query-state* kernel itself. We also chose to fuse the normalization into it. The algorithm for *fused-query-state* is shown below.

$$\text{fused-query-state}(S_i, Q_i, Y_i, \zeta_i, \gamma_i) = \frac{Y_i + \phi(Q_i) S_i}{\zeta_i + \phi(Q_i)\gamma_i} \tag{38}$$

Algorithm 2 Fused Query State

Require: Matrices A of size $\mathbf{M} \times \mathbf{d}$, B of size $\mathbf{D} \times \mathbf{N}$, Y of size $\mathbf{M} \times \mathbf{N}$, γ of size $\mathbf{D}$, ζ of size $\mathbf{M}$
Ensure: Output matrix C of size $\mathbf{M} \times \mathbf{N}$

Define degree of power $\mathbf{p}$, tile size for expansion $\mathbf{d_{tile}}$, expanded tile size $\mathbf{D_{tile}} = \mathbf{d_{tile}}^{\mathbf{p}}$
Denote the ordered list of non-decreasing-multi-indices $\mathbf{NDMI}^{\mathbf{p}}_{\mathbf{d}/\mathbf{d_{tile}}}$ with λ, of size $\mathbf{L} \times \mathbf{p}$
Divide A into $N_A = \lceil \frac{M}{TM} \rceil$ tiles, $A_1, ..., A_{N_A}$, each of size $\mathbf{TM} \times \mathbf{d}$; divide each A_i further into $N_d = \frac{\mathbf{d}}{\mathbf{d_{tile}}}$ subtitles, $A_i^1, ..., A_i^{N_d}$, each of size $\mathbf{TM} \times \mathbf{d_{tile}}$
Divide B into $N_B = \lceil \frac{N}{TN} \rceil$ tiles, $B_1, ..., B_{N_B}$, each of size $\mathbf{D} \times \mathbf{TN}$; divide each B_i further into $\mathbf{L}$ subtitles, $B_i^1, ..., B_i^{\mathbf{L}}$, each of size $\mathbf{D_{tile}} \times \mathbf{TN}$
Divide Y into N_A tiles, $Y_1, ...Y_{N_A}$, each of size $\mathbf{TM} \times \mathbf{N}$; divide each Y_i further into N_B subtiles, $Y_i^1, ..., Y_i^{N_B}$, each of size $\mathbf{TM} \times \mathbf{TN}$
Divide γ into $\mathbf{L}$ tiles, $\gamma_1, ...\gamma_L$, each of size $\mathbf{D_{tile}}$; divide ζ into N_A tiles, $\zeta_1, ..., \zeta_{N_A}$, each of size $\mathbf{TM}$
for $1 \leq i \leq N_A$, in parallel **do**
    for $1 \leq j \leq N_B$, in parallel **do**
        Initialize accumulation registers: $C_{i,j} \leftarrow 0$ of shape $\mathbf{TM} \times \mathbf{TN}$
        Initialize register for matrix multiplication $\hat{A}_i \leftarrow 0$, of shape $\mathbf{TM} \times \mathbf{D_{tile}}$
        Initialize register for normalization $s \leftarrow 0$, of shape $\mathbf{TM}$
        Load A_i from global memory to on-chip SRAM
        for $1 \leq l \leq \mathbf{L}$ **do**
            Load B_j^l, γ_l from global memory to on-chip SRAM
            for $1 \leq z \leq \mathbf{p}$ **do**
                Load $A_i^{\lambda(l,z)}$ from on-chip SRAM into registers
            end for
            $\hat{A}_i \leftarrow A_i^{\lambda(l,1)} \otimes \cdots \otimes A_i^{\lambda(l,\mathbf{p})}$
            $s \leftarrow \hat{A}_i \lambda_l + s$
            $C_{i,j} \leftarrow \hat{A}_i B_j^l + C_{i,j}$
        end for
        Load Y_i^j, ζ_i from global memory to on-chip SRAM
        $C_{i,j} \leftarrow \frac{Y_i^j + C_{i,j}}{\zeta_i + s}$
        Write $C_{i,j}$ to global memory
    end for
end for