GATED DELTA NETWORKS: IMPROVING MAMBA2 WITH DELTA RULE

Anonymous authors
Paper under double-blind review

ABSTRACT

Linear Transformers have emerged as efficient alternatives to standard Transformers due to their inference efficiency, achieving competitive performance across various tasks, though they often struggle with recall-intensive tasks. Recently, two mechanisms—the gating mechanism and the delta update rule—have been used to enhance linear Transformers. We found these two mechanisms to be complementary: the gating mechanism enables fast, adaptive memory erasure, while the delta rule allows for more precise and targeted memory updates. In this work, we introduce the gated delta rule, which combines both mechanisms, and extend the delta rule's parallel algorithm to incorporate gating. Our experiments demonstrate that linear Transformers with the gated delta rule, dubbed Gated DeltaNet, consistently outperform Mamba2 (a gated linear transformer) and DeltaNet in language modeling, common sense reasoning, and real-world in-context recall-intensive tasks. Additionally, we explore hybrid models that combine Gated DeltaNet layers with sliding window attention or Mamba2 layers, further enhancing retrieval capabilities.

1 INTRODUCTION

The Transformer architecture has significantly advanced the capabilities of Large Language Models (LLMs), showcasing exceptional performance across a wide range of tasks due to its effective attention mechanism. This mechanism excels in precise sequence modeling and leverages the parallel processing capabilities of modern GPUs during training. However, the self-attention component scales quadratically with sequence length, leading to substantial computational demands that pose challenges for both training and inference.

To mitigate these issues, researchers have explored alternatives like Linear Transformers (Katharopoulos et al., 2020a), which replace traditional softmax-based attention with kernelized dot-product-based linear attention, substantially reducing memory requirements during inference by reframing as a linear RNN with matrix-valued states. While early versions of Linear Transformers underperformed in language modeling tasks compared to standard Transformers, recent enhancements—such as incorporating data-dependent gating mechanisms akin to those in LSTMs, exemplified by models like GLA (Yang et al., 2024a) and Mamba2 (Dao & Gu, 2024a)—have shown promising improvements. Despite these advancements, challenges remain in effectively managing stored information over long sequences, particularly in tasks requiring associative recall/learning where traditional Transformers still hold an advantage (Arora et al., 2023a; 2024a; Jelassi et al., 2024; Wen et al., 2024; Akyürek et al., 2024).

This phenomenon is not surprising: linear Transformers can be interpreted as implementing an outer-product-based key-value association memory, reminiscent of tensor product representation (Smolensky, 1990). However, the number of orthogonal key-value pairs they can store is *bounded* by the model's dimensionality. When the sequence length exceeds this dimension, memory collisions become inevitable, hindering exact retrieval (Schlag et al., 2021a).

Mamba2 addresses this limitation by introducing a simple gated update rule, $\mathbf{S}_t = \alpha_t \mathbf{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^{\intercal}$, which uniformly decays all key-value associations at each time step by a dynamic ratio, α_t. However, this approach does not account for the varying importance of different key-value associations, potentially leading to inefficient memory utilization. If the model needs to forget a specific key-value association, all key-value associations are equally forgotten, making the process less targeted and efficient.[1]

In contrast, the linear Transformer with the delta rule (Widrow et al., 1960), known as DeltaNet (Schlag et al., 2021a; Yang et al., 2024b), selectively updates memory by (softly) replacing an old key-value pair with the incoming one in a sequential manner. This method has demonstrated impressive performance in synthetic benchmarks for in-context associative retrieval and learning. However, since this process only modifies a single key-value pair at a time, the model lacks the ability to rapidly clear outdated or irrelevant information, especially during context switches where previous data needs to be erased. Consequently, DeltaNet has been found to perform moderately on real-world recall-intensive tasks and struggles to generalize to sequences longer than those seen during training (Yang et al., 2024b), likely due to the absence of a robust memory-clearing mechanism.

Recognizing the complementary advantages of the gated update rule and the delta rule in memory management, we propose the *gated delta rule*, a simple and intuitive mechanism that combines both approaches. The Linear Transformer with the gated delta rule, referred to as *Gated DeltaNet*, gains the flexibility to promptly clear memory by setting $\alpha_t \to 0$, while selectively updating memory when needed without affecting other content by setting $\alpha_t \to 1$ (i.e., switching to the pure delta rule).

The remaining challenge lies in implementing the gated delta rule in a hardware-efficient manner. Yang et al. (2024b) proposed an efficient algorithm that parallelizes the computation of the delta rule over the sequence length dimension using the WY representation (Bischof & Loan, 1985). We carefully extend this algorithm to incorporate the gating terms, resulting in an approach that still supports chunkwise parallelism (Hua et al., 2022; Sun et al., 2023a; Yang et al., 2024a), allowing for hardware-efficient training.

Our experiments demonstrate that linear Transformers with the gated delta rule, dubbed Gated DeltaNet, consistently outperform models like Mamba2 (a gated linear transformer) and DeltaNet in language modeling, commonsense reasoning, and real-world in-context recall-intensive tasks. Additionally, we explore hybrid models that combine Gated DeltaNet layers with sliding window attention or Mamba2 layers, further enhancing retrieval capabilities.

2 Preliminary

2.1 Linear Attention with Chunkwise Parallel Form

It is known that the linear transformer (Katharopoulos et al., 2020b) can be formulated as the following linear recurrence when excluding normalization and query/key activations:

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\intercal \in \mathbb{R}^{d_v \times d_k}, \qquad \boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t \in \mathbb{R}^{d_v}$$

where d_k and d_v represent the (head) dimensions for query/key and value, respectively. By expanding the recurrence, we can express it in both vector form (left) and matrix form (right) as follows:

$$\boldsymbol{o}_t = \sum_{i=1}^{t} (\boldsymbol{v}_i \boldsymbol{k}_i^\intercal) \boldsymbol{q}_t = \sum_{i=1}^{t} \boldsymbol{v}_i (\boldsymbol{k}_i^\intercal \boldsymbol{q}_t) \in \mathbb{R}^{d_v}, \qquad \mathbf{O} = (\mathbf{Q}\mathbf{K}^\intercal \odot \mathbf{M})\mathbf{V} \in \mathbb{R}^{L \times d_v}$$

where L is the sequence length, and $\mathbf{M} \in \mathbb{R}^{L \times L}$ is the causal mask defined by $\mathbf{M}_{ij} = 0$ when $i < j$, and 1 otherwise.

This formulation makes it clear that linear attention eliminates the softmax operation used in traditional attention mechanisms and instead leverages the linearity and associativity of matrix multiplications, leading to linear complexity. However, both the recurrent and parallel forms are not ideal for efficient training (Yang et al., 2024a), which motivates the use of the chunkwise parallel form (Hua et al., 2022; Sun et al., 2023a; Yang et al., 2024a) for hardware-efficient, linear-time training, as introduced below.

Chunkwise parallel form. To summarize, the chunkwise parallel form splits inputs and outputs into several chunks of size C, and computes outputs for each chunk based on the final state of the

[1]While a fine-grained gating mechanism (i.e., assigning each dimension its own decay ratio) could alleviate this issue, as seen in Mamba1, it limits the use of tensor cores, preventing efficient scaling of the state size.

previous chunk and the query/key/value blocks of the current chunk. Following the notation of Sun et al. (2023b); Yang et al. (2024a;b), let's take the query block, $\boldsymbol{q}$, as an example. We denote $\mathbf{Q}_{[t]} := \boldsymbol{q}_{tC+1:(t+1)C+1}$ as the query block for chunk t, and $\boldsymbol{q}_{[t]}^r := \boldsymbol{q}_{tC+r}$ as the r-th query within chunk t. The initial state of chunk t is defined as $\mathbf{S}_{[t]} := \mathbf{S}_{[t]}^0 = \mathbf{S}_{[t-1]}^C$. By partially expanding the recurrence, we have

$$\mathbf{S}_{[t]}^r = \mathbf{S}_{[t]} + \sum_{i=1}^{r} \boldsymbol{v}_{[t]}^i \boldsymbol{k}_{[t]}^{i\intercal} \in \mathbb{R}^{d_v \times d_k}, \qquad \boldsymbol{o}_{[t]}^r = \mathbf{S}_{[t]}^r \boldsymbol{q}_{[t]}^r = \mathbf{S}_{[t]} \boldsymbol{q}_{[t]}^r + \sum_{i=1}^{r} \boldsymbol{v}_{[t]}^i \left(\boldsymbol{k}_{[t]}^{i\intercal} \boldsymbol{q}_{[t]}^r\right) \in \mathbb{R}^{d_v}$$

Equivalently, in matrix form:

$$\mathbf{S}_{[t+1]} = \mathbf{S}_{[t]} + \mathbf{V}_{[t]} \mathbf{K}_{[t]}^\intercal \in \mathbb{R}^{d_v \times d_k}, \qquad \mathbf{O}_{[t]} = \mathbf{Q}_{[t]} \mathbf{S}_{[t]}^\intercal + \left(\mathbf{Q}_{[t]} \mathbf{K}_{[t]}^\intercal \odot \mathbf{M}\right) \mathbf{V}_{[t]} \in \mathbb{R}^{C \times d_v}$$

where $\mathbf{M} \in \mathbb{R}^{C \times C}$ is the causal mask. The above equations are rich in matrix multiplications (matmuls), and by setting C to a multiple of 16, one can take advantage of tensor cores—specialized GPU units for efficient half-precision matmul operations—for hardware-efficient training. Typically, C is set to a small constant (e.g., 64 as implemented in FLA (Yang & Zhang, 2024)), ensuring that the overall computational complexity remains linear with respect to sequence length, enabling efficient modeling of extremely long sequences.

2.2 Mamba2: Linear attention with scalar-valued data-dependent decay

Mamba2 (Dao & Gu, 2024a) can be represented by the following linear recurrence (up to specific parameterization):

$$\mathbf{S}_t = \alpha_t \mathbf{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\intercal, \qquad \boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$$

where $\alpha_t \in (0, 1)$ is a **data-dependent** scalar-valued decay term. In the following, we will highlight the decay terms in red to facilitate a clearer comparison with vanilla linear attention. Define the cumulative decay product $\gamma_j = \prod_{i=1}^{j} \alpha_i$, and by expanding the recurrence, we can express the result in both a vector form (left) and a matrix parallel form (right):

$$\boldsymbol{o}_t = \sum_{i=1}^{t} \left(\frac{\gamma_t}{\gamma_i} \boldsymbol{v}_i \boldsymbol{k}_i^\intercal\right) \boldsymbol{q}_t = \sum_{i=1}^{t} \boldsymbol{v}_i \left(\frac{\gamma_t}{\gamma_i} \boldsymbol{k}_i^\intercal \boldsymbol{q}_t\right), \qquad \mathbf{O} = \left(\left(\mathbf{Q}\mathbf{K}^\intercal\right) \odot \Gamma\right) \mathbf{V}$$

Here, $\Gamma \in \mathbb{R}^{L \times L}$ is a decay-aware causal mask where $\Gamma_{ij} = \frac{\gamma_i}{\gamma_j}$ if $i \geq j$ and $\Gamma_{ij} = 0$ otherwise.

This parallel and recurrent formulation is referred to as state space duality (SSD) in Dao & Gu (2024a). Notably, this recurrence structure has also been employed in Gated RFA (Peng et al., 2021), xLSTM (Beck et al., 2024), and Gated RetNet (Sun et al., 2024b).

Chunkwise parallel form. Slightly abusing the notation, we define the local cumulative product of decays within the chunk as $\gamma_{[t]}^j = \prod_{i=tC+1}^{tC+j} \alpha_i$. Additionally, we define $(\Gamma_{[t]})_{ij} = \frac{\gamma_{[t]}^j}{\gamma_{[t]}^i}$ for $i \geq j$ and 0 otherwise. By partially expanding the recurrence, we obtain the following equations:

$$\mathbf{S}_{[t]}^r = \gamma_{[t]}^r \mathbf{S}_{[t]} + \sum_{i=1}^{r} \frac{\gamma_{[t]}^r}{\gamma_{[t]}^i} \boldsymbol{v}_{[t]}^i \boldsymbol{k}_{[t]}^{i\intercal}, \qquad \boldsymbol{o}_{[t]}^r = \gamma_{[t]}^r \mathbf{S}_{[t]}^r \boldsymbol{q}_{[t]}^r = \mathbf{S}_{[t]} \boldsymbol{q}_{[t]}^r + \sum_{i=1}^{r} \boldsymbol{v}_{[t]}^i \left(\frac{\gamma_{[t]}^r}{\gamma_{[t]}^i} \boldsymbol{k}_{[t]}^{i\intercal} \boldsymbol{q}_{[t]}^r\right)$$

This can be equivalently expressed in matrix form as:

$$\mathbf{S}_{[t+1]} = \gamma_{[t]}^C \mathbf{S}_{[t]} + \mathbf{V}_{[t]}^\intercal \mathrm{Diag}\left(\frac{\gamma_{[t]}^C}{\gamma_{[t]}}\right) \mathbf{K}_{[t]}$$

$$\mathbf{O}_{[t]} = \mathrm{Diag}\left(\gamma_{[t]}\right) \mathbf{Q}_{[t]} \mathbf{S}_{[t]}^\intercal + \left(\mathbf{Q}_{[t]} \mathbf{K}_{[t]}^\intercal \odot \Gamma_{[t]}\right) \mathbf{V}_{[t]}$$

We observe that the (cumulative) decay term can be seamlessly integrated into the matmuls with minimal computational overhead. This ensures that the chunkwise parallel form remains efficient and compatible with high-performance tensor core-based acceleration.

2.3 Delta Networks: Linear Attention with Delta Rule

The delta update rule (Widrow et al., 1960; Schlag et al., 2021b) *dynamically* erases the value ($\boldsymbol{v}_t^{\text{old}}$) associated with the current input key ($\boldsymbol{k}_t$) and writes a new value ($\boldsymbol{v}_t^{\text{new}}$), which is a linear combination of the current input value and the old value. This process updates a key-value association pair at each time step, where the scalar $\beta_t \in (0, 1)$ determines the extent to which the old association is replaced by the new one, as shown below.

$$\mathbf{S}_t = \mathbf{S}_{t-1} - \underbrace{(\mathbf{S}_{t-1}\boldsymbol{k}_t)}_{\boldsymbol{v}_t^{\text{old}}}\boldsymbol{k}_t^\intercal + \underbrace{(\beta_t\boldsymbol{v}_t + (1-\beta_t)\mathbf{S}_{t-1}\boldsymbol{k}_t))}_{\boldsymbol{v}_t^{\text{new}}}\boldsymbol{k}_t^\intercal = \mathbf{S}_{t-1}\left(\mathbf{I} - \beta_t\boldsymbol{k}_t\boldsymbol{k}_t^\intercal\right) + \beta_t\boldsymbol{v}_t\boldsymbol{k}_t^\intercal$$

Chunkwise parallel form. By partially expanding the recurrence, we have

$$\mathbf{S}_{[t]}^r = \mathbf{S}_{[t]}\underbrace{\left(\prod_{i=1}^{r}\mathbf{I} - \beta_{[t]}^i\boldsymbol{k}_{[t]}^i\boldsymbol{k}_{[t]}^{i\intercal}\right)}_{:=\mathbf{P}_{[t]}^r} + \underbrace{\sum_{i=1}^{r}\left(\beta_{[t]}^i\boldsymbol{v}_{[t]}^i\boldsymbol{k}_{[t]}^{i\intercal}\prod_{j=i+1}^{r}\left(\mathbf{I} - \beta_{[t]}^j\boldsymbol{k}_{[t]}^j\boldsymbol{k}_{[t]}^{j\intercal}\right)\right)}_{:=\mathbf{H}_t^r} \tag{1}$$

We observe that $\mathbf{P}_{[t]}^j$ involves a cumulative matrix product of transition matrices, which Yang et al. (2024b) identify as being in the form of a (generalized) Householder matrix. This structure allows for a memory-efficient and compact computation using the classical WY representation (Bischof & Loan, 1985). Inspired by the WY representation, Yang et al. (2024b) introduce two new compact representations designed to optimize this process:

$$\mathbf{P}_{[t]}^r = \mathbf{I} - \sum_{i=1}^{r}\mathbf{w}_{[t]}^i\boldsymbol{k}_{[t]}^{i\intercal} \in \mathbb{R}^{d_k\times d_k} \qquad \mathbf{H}_{[t]}^r = \sum_{i=1}^{r}\mathbf{u}_{[t]}^i\boldsymbol{k}_{[t]}^{i\intercal} \in \mathbb{R}^{d_v\times d_k} \tag{2}$$

$$\mathbf{w}_{[t]}^r = \beta_{[t]}^r\left(\boldsymbol{k}_{[t]}^r - \sum_{i=1}^{r-1}\left(\mathbf{w}_{[t]}^i(\boldsymbol{k}_{[t]}^{i\intercal}\boldsymbol{k}_{[t]}^r)\right)\right) \quad \mathbf{u}_{[t]}^r = \beta_{[t]}^r\left(\boldsymbol{v}_{[t]}^r - \sum_{i=1}^{r-1}\left(\mathbf{u}_{[t]}^i(\boldsymbol{k}_{[t]}^{i\intercal}\boldsymbol{k}_{[t]}^r)\right)\right) \tag{3}$$

where $\mathbf{w}_{[t]}^r \in \mathbb{R}^{d_k}; \mathbf{u}_{[t]}^r \in \mathbb{R}^{d_v}$. Put them back to Eq.1, we have the following matrix form:

$$\mathbf{S}_{[t+1]} = \mathbf{S}_{[t]} + \left(\mathbf{U}_{[t]} - \mathbf{W}_{[t]}\mathbf{S}_{[t]}^\intercal\right)^\intercal\mathbf{K}_{[t]} \tag{4}$$

$$\mathbf{O}_{[t]} = \mathbf{Q}_{[t]}\mathbf{S}_{[t]}^\intercal + (\mathbf{Q}_{[t]}\mathbf{K}_{[t]}^\intercal \odot \mathbf{M})\left(\mathbf{U}_{[t]} - \mathbf{W}_{[t]}\mathbf{S}_{[t]}^\intercal\right) \tag{5}$$

where $\mathbf{M}$ is the standard causal mask.

3 Gated Delta Networks

3.1 Gated Delta Rule

The proposed gated delta rule offers a simple yet effective approach:

$$\mathbf{S}_t = \mathbf{S}_{t-1}\left(\alpha_t(\mathbf{I} - \beta_t\boldsymbol{k}_t\boldsymbol{k}_t^\intercal)\right) + \beta_t\boldsymbol{v}_t\boldsymbol{k}_t^\intercal \tag{6}$$

In comparison to the standard delta rule, it introduces a multiplicative, data-dependent **scalar-valued** decay term (or forget gate) $\alpha_t \in (0, 1)$, applied to the hidden state. This combination effectively merges the advantages of the gating mechanism with the flexibility of the delta update rule, enjoying the best of the two worlds.

However, despite its conceptual simplicity, the WY representation used for the delta rule no longer applies in this context, necessitating adaptations, which we will introduce below, with all changes highlighted in red.

Chunkwise parallel form. Likewise, by partially expanding the recurrence, we have

$$\mathbf{S}_{[t]}^r = \mathbf{S}_{[t]}\underbrace{\left(\prod_{i=1}^{r}\alpha_{[t]}^i\left(\mathbf{I} - \beta_{[t]}^i\boldsymbol{k}_{[t]}^i\boldsymbol{k}_{[t]}^{i\intercal}\right)\right)}_{:=\mathbf{P}_{[t]}^r} + \underbrace{\sum_{i=1}^{r}\left(\beta_{[t]}^i\boldsymbol{v}_{[t]}^i\boldsymbol{k}_{[t]}^{i\intercal}\prod_{j=i+1}^{r}\alpha_{[t]}^j\left(\mathbf{I} - \beta_{[t]}^j\boldsymbol{k}_{[t]}^j\boldsymbol{k}_{[t]}^{j\intercal}\right)\right)}_{:=\mathbf{H}_{[t]}^r}$$

We adapt the WY representation in Eq. 2-3 to incorporate the decay term as below,

$$\mathbf{P}_{[t]}^r = \gamma_{[t]}^r \left(\mathbf{I} - \sum_{i=1}^r \mathbf{w}_{[t]}^i \boldsymbol{k}_{[t]}^{i\intercal}\right) \qquad \mathbf{H}_{[t]}^r = \sum_{i=1}^r \frac{\gamma_t^r}{\gamma_t^i} \mathbf{u}_{[t]}^i \boldsymbol{k}_{[t]}^{i\intercal} \tag{7}$$

where

$$\mathbf{w}_{[t]}^r = \beta_{[t]}^r \left(\boldsymbol{k}_{[t]}^r - \sum_{i=1}^{r-1} \left(\mathbf{w}_{[t]}^i (\boldsymbol{k}_{[t]}^{i\intercal} \boldsymbol{k}_{[t]}^r)\right)\right) \quad \mathbf{u}_{[t]}^r = \beta_{[t]}^r \left(\boldsymbol{v}_{[t]}^r - \sum_{i=1}^{r-1} \left(\mathbf{u}_{[t]}^i (\frac{\gamma_{[t]}^r}{\gamma_{[t]}^i} \boldsymbol{k}_{[t]}^{i\intercal} \boldsymbol{k}_{[t]}^r)\right)\right) \tag{8}$$

and the proof of correctness can be found at Appendix. Then we have the following vector form:

$$\mathbf{S}_{[t]}^r = \gamma_{[t]}^r \mathbf{S}_{[t]}^0 + \sum_{i=1}^r \frac{\gamma_{[t]}^r}{\gamma_{[t]}^i} \left(\mathbf{u}_{[t]}^r - \left(\gamma_{[t]}^i \left(\mathbf{S}_{[t]}^0 \mathbf{w}_{[t]}^i\right)\right)\right) \boldsymbol{k}_{[t]}^{i\intercal}$$

$$\boldsymbol{o}_{[t]}^r = \mathbf{S}_{[t]}^r \boldsymbol{q}_{[t]}^r = \gamma_{[t]}^r \mathbf{S}_{[t]}^0 \boldsymbol{q}_{[t]}^r + \sum_{i=1}^r \left(\mathbf{u}_{[t]}^r - \left(\gamma_{[t]}^i \left(\mathbf{S}_{[t]}^0 \mathbf{w}_{[t]}^i\right)\right)\right) \left(\frac{\gamma_{[t]}^r}{\gamma_{[t]}^i} \boldsymbol{k}_{[t]}^{i\intercal} \boldsymbol{q}_{[t]}^r\right)$$

Equivalently, in matrix form:

$$\mathbf{S}_{[t+1]} = \gamma_{[t]}^C \mathbf{S}_{[t]} + \left(\mathbf{U}_{[t]} - \mathrm{Diag}\left(\gamma_{[t]}\right) \mathbf{W}_{[t]} \mathbf{S}_{[t]}^\intercal\right)^\intercal \mathbf{K}_{[t]} \tag{9}$$

$$\mathbf{O}_{[t]} = \mathrm{Diag}\left(\gamma_{[t]}\right) \mathbf{Q}_{[t]} \mathbf{S}_{[t]}^\intercal + (\mathbf{Q}_{[t]} \mathbf{K}_{[t]}^\intercal \odot \Gamma_{[t]}) \left(\mathbf{U}_{[t]} - \mathrm{Diag}\left(\gamma_{[t]}\right) \mathbf{W}_{[t]} \mathbf{S}_{[t]}^\intercal\right) \tag{10}$$

Hardware optimization using UT transform. Eq. 9 and 10 are are rich in matrix matmuls, making them well-suited for tensor core-based GPU acceleration. However, the construction of the extended WY representation is strictly sequential and, at first glance, cannot be represented as matmuls. Nonetheless, minimizing non-matmul FLOPs and maximizing matmul operations is critical to leveraging tensor cores effectively, as emphasized in works like Dao (2023); Fu et al. (2023); Yang et al. (2024a).

Fortunately, by applying the UT transform (Joffrain et al., 2006), we observe that much of the computation can be rewritten as matmuls. This technique, which has been used to optimize the WY transform on modern hardware (Dominguez & Orti, 2018), allows us to reframe most of the operations in a more hardware-friendly manner.

$$\mathbf{W}_{[t]} = \mathbf{A}_{[t]}^W \,\mathrm{Diag}(\beta_{[t]}) \mathbf{K}_{[t]}, \qquad \mathbf{A}_{[t]}^W = \left(\mathbf{I} - \mathrm{lower}(\mathrm{Diag}(\beta_{[t]}) \mathbf{K}_{[t]} \mathbf{K}_{[t]}^\intercal)\right)^{-1}$$

$$\mathbf{U}_{[t]} = \mathbf{A}_{[t]}^U \,\mathrm{Diag}\left(\beta_{[t]}\right) \mathbf{V}_{[t]}, \qquad \mathbf{A}_{[t]}^U = \left(\mathbf{I} - \Gamma_{[t]} \odot \mathrm{lower}\left(\mathrm{Diag}(\beta_{[t]}) \mathbf{K}_{[t]} \mathbf{K}_{[t]}^\intercal\right)\right)^{-1}$$

where $\mathrm{lower}(\cdot) := \mathrm{tril}(\cdot, -1)$; and the inverse of a lower triangle matrix can be calculated efficiently by back substitution.

Remarks on Speed. As we can see in . UT transform can be used to speedup the computation for both delta rule and gated delta rule. We observe that the running speed of the gated delta rule is nearly identical to that of the delta rule, as the introduced overhead is minimal—all matmul operations remain intact, with only additional efficient elementwise operations required to handle the gating terms. This is analogous to the comparison between Mamba2 and vanilla linear attention. As a result, Gated DeltaNet maintains similar training throughput to DeltaNet.

3.2 Neural Architecture

Token Mixer Block Design. The basic Gated DeltaNet follows the macro architecture of the Llama Transformer, stacking token mixer layers with SwiGLU MLP layers, but replaces the standard self-attention mechanism with a gated delta rule token mixing layer. Fig. 1 (right) illustrates a single gated delta rule token mixing layer. First, the hidden states are projected to create the query, key, and value vectors. Additionally, two more projections are made to generate the forget gate α and the output gate $\mathbf{g}$. The forget gate is parameterized similarly to Mamba2, with some details omitted for brevity. The transformed query, key, and value vectors are then projected into a new space

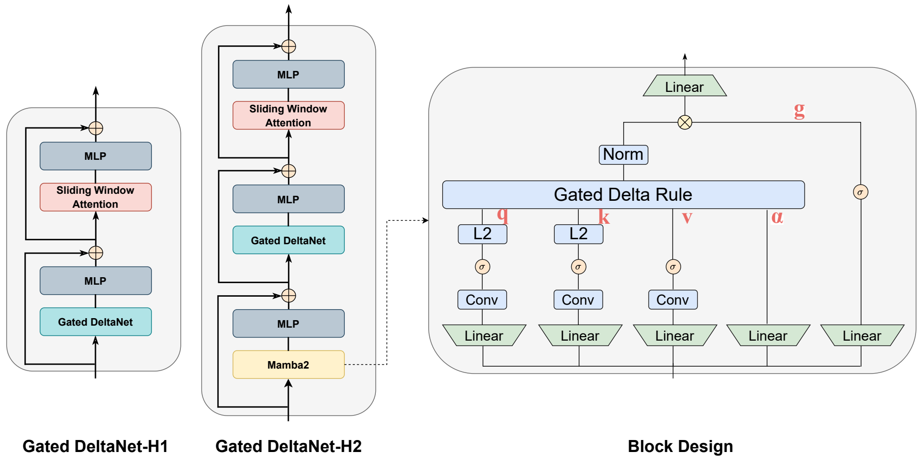

Figure 1: Visualization of the architecture and block design of Gated DeltaNet models. Gated DeltaNet-H1 and Gated DeltaNet-H2 consist of Gated DeltaNet + SWA and Mamba2 + Gated DeltaNet + SWA patterns, respecrively. We use L2 normalization and SiLU feature map in the block design.

using a short convolution, consisting of a 1D convolution followed by a `SiLU` activation function, as employed in both Mamba2 and DeltaNet. To ensure that the eigenvalues of the transition matrices remain less than one, as recommended by Yang et al. (2024b), L2 normalization is applied to the query and key vectors, resulting in the final query $\boldsymbol{q}$, key $\boldsymbol{k}$, and value $\boldsymbol{v}$. Subsequently, $\boldsymbol{q}$, $\boldsymbol{k}$, $\boldsymbol{v}$, and α are used to produce the output $\boldsymbol{o}$ based on the recurrence in Eq. 6. To stabilize training, RMS normalization is applied to the output $\boldsymbol{o}$, a technique shown to be effective by Qin et al. (2022) and Sun et al. (2023a). This is followed by a Swish-activated output gating mechanism, which has also proven effective in prior work (Sun et al., 2023a; Peng et al., 2023), as shown below.

$$\boldsymbol{o}'_i = \mathrm{RMSNorm}(\boldsymbol{o}_i) \odot \mathrm{Swish}(\boldsymbol{g}_i)$$

This representation $\boldsymbol{o}'$ is then passed through the output projection layer.

Hybrid Architectures. Linear transformers face challenges in handling local shifts and comparisons as effectively as attention-based mechanisms (Arora et al., 2024a). To address this, we follow the recent trend of hybridizing linear recurrent layers with sliding window attention (SWA), as seen in models like Griffin (De et al., 2024) and Samba (Ren et al., 2024). We propose two hybrid models, Gated DeltaNet-H1 and Gated DeltaNet-H2, as illustrated on the left-hand side of Fig.1. For an ablation study on various design integration patterns in the Gated DeltaNet-H2 model, please refer to the Appendix.

4 Experiments

4.1 Setup

Training We trained models from scratch with 400M and 1.3B parameters for 15B and 100B tokens, respectively on the same subset of the FineWeb-Edu dataset (Penedo et al., 2024). Our experiments include a wide variety of recent SOTA models from purely Transformer and RNN-based to hybrid approaches. Specifically, we compare against the following baseline: RetNet (Sun et al., 2023a), Mamba (Gu & Dao, 2023), Mamba2 (Dao & Gu, 2024b), Samba (Ren et al., 2024) and DeltaNet (Yang et al., 2024b).

Evaluation Tasks To evaluate the effectiveness of model, we evaluate the zeroshot performance on various commonsense reasoning benchmarks. These tasks include PIQA (Bisk et al., 2020), HellaSwag (Hella.; Zellers et al., 2019), WinoGrande (Wino.; Sakaguchi et al., 2021), ARC-easy (ARC-e) and ARC-challenge (Arc-c) (Clark et al., 2018), SIQA (Sap et al., 2019), BoolQ (Clark et al., 2019) Wikitext (Wiki.; Merity et al., 2016) and LAMBADA (LMB.; Paperno et al., 2016). All evaluations are performed by using `lm-evaluation-harness` (Gao et al., 2021).

Furthermore, we evaluate the performance of models for associative-recall tasks on SWDE (Lockard et al., 2019), SQuAD (Rajpurkar et al., 2018), FDA (Arora et al., 2023b), TriviaQA (Joshi et al., 2017), Drop (Dua et al., 2019) and NQ (Kwiatkowski et al., 2019). Specifically, SWDE is designed to extract structured relations in HTML files while FDA is focused on key-value information retrieval of PDF files. In addition, SQuAD, TriviaQA, Drop and NQ are question-answering tasks that are designed for in-context information grounding in documents. Since our pretrained models are not instruction-tuned, we use the script provided by Arora et al. (2024b) with Cloze Completion Formatting prompts for evaluation, which aligns more closely with the next-word-prediction training objective of these language models.

Hyperparameters For all models, we use the AdamW optimizer with a peak learning rate of 4e-4, weight decay of 0.1 and gradient clipping of 1. Cosine annealing is used with warm up over 150M and 1B tokens for models with 340M and 1.3B parameters, respectively. We also use global batch sizes of 512 and 1024 for 340M and 1.3B model, respectively. In addition, all models have a vocabulary size of 32000, use the Llama2 tokenizer and trained with sequence length of 4096. Models denoted with SWA use a local sliding window attention of size 2048. We use 128 and 32 NVIDIA A100 GPUs for training all 1.3B and 340M models, respectively.

Model	**Wiki.** ppl ↓	**LMB.** ppl ↓	**LMB.** acc ↑	**PIQA** acc ↑	**Hella.** acc_n ↑	**Wino.** acc ↑	**ARC-e** acc ↑	**ARC-c** acc_n ↑	**SIQA** acc ↑	**BoolQ** acc ↑	**Avg.**
400M params / 15B tokens											
Transformer++	30.63	37.37	29.64	64.27	37.72	51.53	54.95	27.36	38.07	61.59	45.64
RetNet	29.92	46.83	29.16	65.23	36.97	51.85	56.01	27.55	37.30	59.66	45.47
HGRN2	32.33	47.14	26.12	64.52	35.45	52.24	55.97	25.51	37.35	59.02	44.52
Mamba	29.22	39.88	29.82	65.72	37.93	50.11	58.37	26.70	37.76	61.13	45.94
Mamba2	26.34	33.19	32.03	65.77	39.73	**52.48**	59.00	27.64	37.92	60.72	46.91
DeltaNet	27.69	44.04	29.96	64.52	37.03	50.82	56.77	27.13	38.22	60.09	45.57
Gated DeltaNet	25.47	29.24	34.40	65.94	40.46	51.46	59.80	28.58	37.43	60.03	47.26
Gated DeltaNet-H2	24.19	**28.09**	**36.77**	**66.43**	**40.79**	52.17	59.55	**29.09**	**39.04**	58.56	47.69
Gated DeltaNet-H1	**24.06**	28.72	36.00	65.50	40.73	51.30	**60.69**	28.49	37.71	**61.77**	**47.88**
1.3B params / 100B tokens											
Transformer++	18.53	18.32	42.60	70.02	50.23	53.51	68.83	35.10	40.66	57.09	52.25
RetNet	19.08	17.27	40.52	70.07	49.16	54.14	67.34	33.78	40.78	60.39	52.02
HGRN2	19.10	17.69	39.54	70.45	49.53	52.80	69.40	35.32	40.63	56.66	51.79
Mamba	17.92	15.06	43.98	71.32	52.91	52.95	69.52	35.40	37.76	61.13	53.12
Samba	16.13	13.29	44.94	70.94	53.42	55.56	68.81	36.17	39.96	62.11	54.00
Mamba2	16.56	12.56	45.66	71.87	55.67	55.24	**72.47**	37.88	40.20	60.13	54.89
DeltaNet	17.71	16.88	42.46	70.72	50.93	53.35	68.47	35.66	40.22	55.29	52.14
Gated DeltaNet	16.42	12.17	46.65	72.25	55.76	57.45	71.21	38.39	40.63	60.24	55.32
Gated DeltaNet-H2	**15.91**	12.55	**48.76**	72.19	**56.88**	57.77	71.33	39.07	**41.91**	61.55	56.18
Gated DeltaNet-H1	16.07	**12.12**	47.73	**72.57**	56.53	**58.40**	71.75	**40.10**	41.40	**63.21**	**56.40**

Table 1: Zero-shot performance comparison of 400M and 1.3B parameter models that are trained for 15B and 100B tokens respectively. Gated DeltaNet-H1 and Gated DeltaNet-H2 denote hybrid variants comprising of Gated DeltaNet + SWA and Mamba2 + Gated DeltaNet + SWA, respectively. All models are trained from scratch on FineWeb-Edu dataset (Penedo et al., 2024).

4.2 Empirical results

Commonsense Reasoning. In Table 1, we present the language modeling perplexity and zero-shot accuracy on commonsense reasoning benchmarks for models with 400M and 1.3B parameters. Gated DeltaNet consistently outperforms other linear models, including RetNet, HGRN2, Mamba, Mamba2, and DeltaNet, at both scales. As expected, the hybrid variant further enhances performance.

In-context recall-intensive tasks. Table 2 presents the results of recall-intensive tasks. As expected, linear models exhibit a notable performance gap compared to Transformers, with Mamba2 standing out as a strong baseline recurrent model, outperforming all other pure recurrent baseline models.

State size is *strongly* correlated with final performance. With a $128\times Ld$ state size and 400M parameters, Gated DeltaNet clearly outperforms DeltaNet, underscoring the importance of the gating mechanism. When using a $256\times Ld$ state size, Gated DeltaNet outperforms Mamba2 across both model scales, demonstrating the effectiveness of the delta update rule. However, for the 0.4B models, Gated DeltaNet with a $128\times Ld$ state size underperforms compared to Mamba2 with a $256\times Ld$ state

Models	State size	SWDE ↑	SQuAD ↑	FDA ↑	TriviaQA ↑	NQ ↑	Drop ↑	Avg
400M params / 15B tokens								
Transformer++	N/A	22.1	28.3	30.2	43.1	15.6	17.5	26.1
Samba	$2062\times Ld$	**23.1**	29.9	31.0	45.1	16.3	16.7	27.0
RetNet	$512\times Ld$	6.0	19.6	1.5	39.4	8.7	14.9	15.0
HGRN2	$128\times Ld$	6.1	15.3	1.0	36.9	7.6	12.1	13.1
Mamba	$32\times Ld$	6.8	15.7	1.1	37.8	8.0	12.2	13.6
Mamba2	$256\times Ld$	12.0	24.9	10.8	43.3	11.8	17.3	20.1
DeltaNet	$128\times Ld$	7.4	22.4	6.5	41.8	12.3	16.7	17.8
Gated DeltaNet	$128\times Ld$	11.3	26.0	4.5	42.2	10.2	18.0	18.7
Gated DeltaNet	$256\times Ld$	13.6	26.5	9.8	48.3	13.7	16.0	21.3
Gated DeltaNet-H2	$1418\times Ld$	20.1	31.8	**41.0**	48.9	17.5	**19.1**	**29.7**
Gated DeltaNet-H1	$2112\times Ld$	20.7	**33.2**	33.1	**49.8**	**19.5**	18.9	29.2
1.3B params / 100B tokens								
Transformer++	N/A	29.5	38.0	**52.2**	58.3	22.5	21.6	37.0
Samba	$2062\times Ld$	33.0	39.2	50.5	57.7	23.5	20.2	37.3
RetNet	$512\times Ld$	14.0	28.5	7.0	54.4	16.2	17.3	22.9
HGRN2	$128\times Ld$	8.3	25.3	4.8	51.2	14.2	16.9	20.1
Mamba	$32\times Ld$	9.8	25.8	3.7	54.3	14.9	17.4	21.0
Mamba2	$256\times Ld$	19.1	33.6	25.3	61.0	20.8	19.2	29.8
DeltaNet	$128\times Ld$	17.9	30.9	18.4	53.9	17.3	18.6	26.2
Gated DeltaNet	$256\times Ld$	25.4	34.8	23.7	60.0	20.0	19.8	30.6
Gated DeltaNet-H2	$1461\times Ld$	**38.2**	**40.4**	50.7	**63.3**	**24.8**	**23.3**	**40.1**
Gated DeltaNet-H1	$2176\times Ld$	35.6	39.7	52.0	60.1	24.6	22.2	39.0

Table 2: Performance comparison on associative-recall tasks for models with 400M and 1.3B parameter models which are trained on 15B and 100B tokens, respectively. Gated DeltaNet-H1 and Gated DeltaNet-H2 denote hybrid variants comprising of Gated DeltaNet + SWA and Mamba2 + Gated DeltaNet + SWA, respectively. In state size column, L denotes the number of layer while d denotes model dimension.

size. This highlights the importance of maintaining consistent state sizes for fair model comparisons, and we recommend future work ensure state size consistency when evaluating models.

For models utilizing sliding window attention (SWA), the KV cache size is used as the state size. Recurrent models enhanced by SWA exhibit larger state sizes and significantly higher recall performance: Samba outperforms Transformer++ in both configurations, while Gated DeltaNet-H1 surpasses Samba. Interestingly, Gated DeltaNet-H2 exceeds Gated DeltaNet-H1 despite a smaller state size, indicating the potential benefits of hybridizing multiple models. Further exploration of this hybridization is left as a direction for future research.

Length extrapolation. As observed in Yang et al. (2024b) and illustrated in Fig. 2, DeltaNet struggles to extrapolate to sequences longer than its training length (in this case, 4K tokens). We speculate that this limitation arises from its slow forgetting mechanism, which hinders the model's ability to efficiently clear outdated memory content. As a result, when the evaluated sequence length exceeds the training length, the model's memory becomes saturated, leaving no room to accommodate new information.

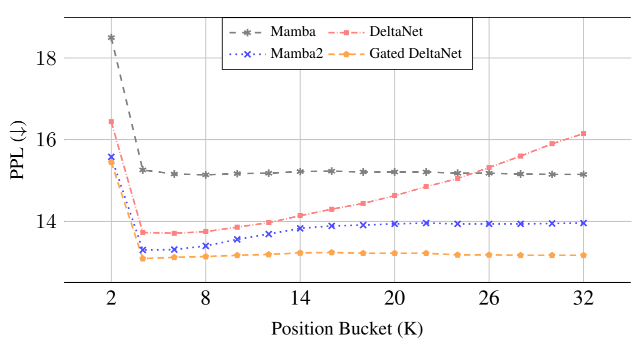

Figure 2: Length extrapolation results in PG19 test set.

Similarly, Mamba2 faces a related issue, with perplexity increasing as sequence length grows, though to a lesser extent than DeltaNet, due to its forgetting mechanism. This suggests that while the simple gated update rule improves memory management, it does not fully solve the challenge of handling extended contexts. In contrast, Mamba1 does not exhibit a significant increase in perplexity with longer sequences, thanks to its more fine-grained gating mechanism, which allows for different decay rates for each hidden dimension. However, this fine-grained control prevents efficient use of tensor cores and limits the state size, ultimately resulting in higher perplexity due to these computational constraints.

Gated DeltaNet demonstrates clear advantages over these approaches, due to the superiority of the gated delta rule in memory management. This enables the model to effectively process much longer sequences with a finite state size, making it more adaptable to extended contexts.

Ablation Study. Table 3 shows the ablation study of the Gated DeltaNet block. We found that both the short convolution and output gate are crucial to performance, while output normalization provides a slight improvement. Similar to Yang et al. (2024b), we observed that L2 normalization is essential for optimal performance, whereas the specific choice of feature map is less critical. That said, SiLU consistently performed the best, in line with findings by Qin et al. (2023). Regarding head dimension, we found that setting it to 128 strikes a good balance between performance and efficiency.

5 Related Work

Gated Linear RNN. Large linear recurrent language models have garnered significant attention due to their training and inference efficiency. The field of linear RNNs has rapidly evolved from using data-independent decay mechanisms, as seen in models like S4 (Gu et al., 2022), S5 (Smith et al., 2023), RWKV4/5 (Peng et al., 2023), and RetNet, to adopting data-dependent decay mechanisms in more recent models like HGRN1/2 (Qin et al., 2024a;b), Mamba1/2, RWKV6 (Peng et al., 2024), and GSA (Zhang et al., 2024). This shift is largely due to the unique advantages of gating/forgetting mechanisms (referred to as selective mechanisms in Mamba), a classical concept that originated in the gated RNN literature (Gers et al., 2000) and whose significance has been repeatedly validated (Greff et al., 2015; Jing et al., 2017; van der Westhuizen & Lasenby, 2018; Qin et al., 2024b).

Table 3: Ablation study on the Gated DeltaNet block. Avg-PPL and Avg-Acc denote average perplexity and zero-shot commonsense reasoning accuracy (as in Table 1), respectively. All models have 400M parameters and are trained for 15B tokens on the same subset of FineWeb-Edu dataset (Penedo et al., 2024).

Gated DeltaNet Ablations (400M)	Avg-PPL (↓)	Avg-Acc (↑)
Gated DeltaNet *w* Head Dim 128,	27.35	47.26
Macro Design		
w. naive Delta Rule	30.87	45.12
w/o. Short Conv	28.95	46.16
w/o. Output Gate	29.12	45.46
w/o. Output Norm	27.55	47.07
Normalization & Feature Map		
w. L_1-norm & ReLU	30.79	45.92
w. L_1-norm & 1+ELU	30.34	46.05
w. L_1-norm & SiLU	30.18	46.09
w. L_2-norm & ReLU	27.67	46.94
w. L_2-norm & 1+ELU	27.58	47.17
Model Dimensions		
w. Head Dim 64	28.31	46.35
w. Head Dim 256	27.13	47.38

Modern forget gates differ from traditional designs like those in LSTM by removing the dependency on the previous hidden state, relying solely on input data. This enables efficient parallelism across sequence lengths (Martin & Cundy, 2018; Qin et al., 2024b; De et al., 2024). The absence of a forget gate has been a key limitation in DeltaNet, and our gated extension of DeltaNet addresses this gap in a way that is both natural and effective.

Delta Rule. The delta learning rule has been shown to offer superior memory capacity compared to the Hebbian learning rule (Gardner, 1988; Prados & Kak, 1989). While linear transformers rely on a Hebbian-like learning rule, DeltaNet utilizes the delta rule, and this advantage in memory capacity is empirically evident in synthetic in-context learning tasks. Moreover, this superiority extends across various applications, including language modeling (Irie et al., 2021; Yang et al., 2024b), reinforcement learning (Irie et al., 2022), and image generation (Irie & Schmidhuber, 2023). Yang et al. (2024b) further parallelized delta rule computations across sequence lengths and highlighted the increased expressiveness of DeltaNet's data-dependent identity-plus-low-rank structured transition matrix ($\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\intercal$) compared to Mamba2's data-dependent diagonal matrices ($\alpha_t \mathbf{I}$). This shift from diagonal to structured dense matrices significantly enhances the model's ability to tackle complex reasoning tasks, such as regular language processing (Fan et al., 2024) and state-tracking tasks beyond the TC^0 complexity class (Merrill et al., 2024), which are critical for applications like coding.

The delta rule also exhibits an interesting connection to online (meta) learning via gradient descent (Munkhdalai et al., 2019). Recent studies, such as Longhorn (Liu et al., 2024) and TTT (Sun et al., 2024a), revisit this link by framing state space learning as a gradient-based online learning problem.

Notably, Longhorn's closed-form solution with L2 loss closely mirrors the delta update rule, while the TTT-linear variant recovers the delta rule when layer normalization is excluded.

Despite these strengths, the delta rule still faces theoretical limitations, as highlighted by Irie et al. (2023), and has shown moderate performance on real-world data (Yang et al., 2024b). Extensions of DeltaNet, such as the *Recurrent DeltaNet* (Irie et al., 2021) and the *Modern Self-referential Weight Matrix* (Irie & Schmidhuber, 2023), introduce strict recurrence to improve expressiveness, albeit at the cost of parallelizability during training. In contrast, our proposed Gated DeltaNet incorporates a gating mechanism that enhances DeltaNet's expressiveness while preserving efficient training on modern hardware.

6 Conclusion

In this work, we introduced Gated DeltaNet, which combines the gated update mechanism from Mamba2 with the delta update rule from DeltaNet to create more expressive recurrent models. We extended the delta rule parallel algorithm (Yang et al., 2024b) to incorporate gating terms, enabling chunkwise parallelism and hardware-efficient training. Experiments on commonsense reasoning and recall-intensive tasks demonstrate the advantages of Gated DeltaNet over both Mamba2 and DeltaNet, validating its effectiveness in enhancing model performance.

References

Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. In-Context Language Learning: Architectures and Algorithms, January 2024. URL http://arxiv.org/abs/2401.12973. arXiv:2401.12973 [cs].

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. *CoRR*, abs/2312.04927, 2023a.

Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes, April 2023b. arXiv:2304.09433 [cs].

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. *CoRR*, abs/2402.18668, 2024a. arXiv: 2402.18668.

Simran Arora, Aman Timalsina, Aaryan Singhal, Benjamin Spector, Sabri Eyuboglu, Xinyi Zhao, Ashish Rao, Atri Rudra, and Christopher Ré. Just read twice: closing the recall gap for recurrent language models, 2024b. URL https://arxiv.org/abs/2407.05483.

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

Christian H. Bischof and Charles Van Loan. The WY representation for products of householder matrices. In *SIAM Conference on Parallel Processing for Scientific Computing*, 1985. URL https://api.semanticscholar.org/CorpusID:36094006.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *CoRR*, abs/2307.08691, 2023. doi: 10.48550/ARXIV.2307.08691.

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

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10041–10071. PMLR, 21–27 Jul 2024b. URL https://proceedings.mlr.press/v235/dao24a.html.

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando De Freitas, and Caglar Gulcehre. Griffin: Mixing Gated Linear Recurrences with Local Attention for Efficient Language Models, February 2024. URL http://arxiv.org/abs/2402.19427. arXiv:2402.19427 [cs].

Andrés E. Tomás Dominguez and Enrique S. Quintana Orti. Fast blocking of householder reflectors on graphics processors. *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pp. 385–393, 2018. URL https://api.semanticscholar.org/CorpusID:46960439.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs, 2019.

Ting-Han Fan, Ta-Chung Chi, and Alexander Rudnicky. Advancing regular language reasoning in linear recurrent neural networks. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 45–53, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-short.4. URL https://aclanthology.org/2024.naacl-short.4.

Daniel Y. Fu, Hermann Kumbong, Eric Nguyen, and Christopher Ré. Flashfftconv: Efficient convolutions for long sequences with tensor cores. *CoRR*, abs/2311.05908, 2023.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, September 2021.

E. Gardner. The space of interactions in neural network models. *Journal of Physics A*, 21:257–270, 1988. URL https://api.semanticscholar.org/CorpusID:15378089.

Felix A. Gers, Jürgen Schmidhuber, and Fred A. Cummins. Learning to forget: Continual prediction with LSTM. *Neural Comput.*, 12(10):2451–2471, 2000.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28: 2222–2232, 2015. URL https://api.semanticscholar.org/CorpusID:3356463.

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. 2023.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc V. Le. Transformer Quality in Linear Time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9099–9117. PMLR, 2022.

Kazuki Irie and Jürgen Schmidhuber. Images as weight matrices: Sequential image generation through synaptic learning rules. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=ddad0PNUvV.

Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. Going beyond linear transformers with recurrent fast weight programmers. *Advances in Neural Information Processing Systems*, 34:7703–7717, 2021.

Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. A modern self-referential weight matrix that learns to modify itself. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9660–9677. PMLR, 2022. URL https://proceedings.mlr.press/v162/irie22b.html.

Kazuki Irie, Róbert Csordás, and Jürgen Schmidhuber. Practical computational power of linear transformers and their recurrent and self-referential extensions. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9455–9465, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.588. URL https://aclanthology.org/2023.emnlp-main.588.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat After Me: Transformers are Better than State Space Models at Copying. *CoRR*, abs/2402.01032, 2024. doi: 10.48550/ARXIV.2402.01032. URL https://doi.org/10.48550/arXiv.2402.01032. arXiv: 2402.01032.

Li Jing, Çaglar Gülçehre, John Peurifoy, Yichen Shen, Max Tegmark, Marin Soljac$i, and Yoshua Bengio. Gated orthogonal recurrent units: On learning to forget.$ *Neural Computation*, $31: 765--783, 2017. URL.$

Thierry Joffrain, Tze Meng Low, Enrique S. Quintana-Ortí, Robert A. van de Geijn, and Field G. Van Zee. Accumulating householder transformations, revisited. *ACM Trans. Math. Softw.*, 32:169–179, 2006. URL https://api.semanticscholar.org/CorpusID:15723171.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, July 2017. Association for Computational Linguistics.

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

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

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. @articleDBLP:journals/corr/abs-2407-14207, author = Bo Liu and Rui Wang and Lemeng Wu and Yihao Feng and Peter Stone and Qiang Liu, title = Longhorn: State Space Models are Amortized Online Learners, journal = CoRR, volume = abs/2407.14207, year = 2024, url = https://doi.org/10.48550/arXiv.2407.14207, doi = 10.48550/ARXIV.2407.14207, eprinttype = arXiv, eprint = 2407.14207, timestamp = Fri, 23 Aug 2024 08:12:16 +0200, biburl =

https://dblp.org/rec/journals/corr/abs-2407-14207.bib, bibsource = dblp computer science bibliography, https://dblp.org : State space models are amortized online learners. *CoRR*, abs/2407.14207, 2024. 10.48550/ARXIV.2407.14207. URL https://doi.org/10.48550/arXiv.2407.14207.

Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. OpenCeres: When Open Information Extraction Meets the Semi-Structured Web. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3047–3056, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. 10.18653/v1/N19-1309. URL https://aclanthology.org/N19-1309.

Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

William Merrill, Jackson Petty, and Ashish Sabharwal. The Illusion of State in State-Space Models, April 2024. URL http://arxiv.org/abs/2404.08819. arXiv:2404.08819 [cs].

Tsendsuren Munkhdalai, Alessandro Sordoni, Tong Wang, and Adam Trischler. Metalearned Neural Memory. *ArXiv*, July 2019. URL https://www.semanticscholar.org/paper/a513bb6e1967f5a31ad4f38954e66d4169b613e5.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context, June 2016. URL http://arxiv.org/abs/1606.06031. arXiv:1606.06031 [cs].

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*, 2024.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran G. V, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Xiangru Tang, Bolun Wang, Johan S. Wind, Stanislaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the Transformer Era. *CoRR*, abs/2305.13048, 2023. 10.48550/ARXIV.2305.13048. arXiv: 2305.13048.

Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Xingjian Du, Teddy Ferdinan, Haowen Hou, Przemysław Kazienko, Kranthi Kiran GV, Jan Kocoń, Bartłomiej Koptyra, Satyapriya Krishna, Ronald McClelland Jr., Niklas Muennighoff, Fares Obeid, Atsushi Saito, Guangyu Song, Haoqin Tu, Stanisław Woźniak, Ruichong Zhang, Bingchen Zhao, Qihang Zhao, Peng Zhou, Jian Zhu, and Rui-Jie Zhu. Eagle and Finch: RWKV with Matrix-Valued States and Dynamic Recurrence, April 2024. URL http://arxiv.org/abs/2404.05892. arXiv:2404.05892 [cs].

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

DL Prados and SC Kak. Neural network capacity using delta rule. *Electronics Letters*, 3(25): 197–199, 1989.

Zhen Qin, Xiaodong Han, Weixuan Sun, Dongxu Li, Lingpeng Kong, Nick Barnes, and Yiran Zhong. The devil in linear transformer. *arXiv preprint arXiv:2210.10340*, 2022.

Zhen Qin, Dong Li, Weigao Sun, Weixuan Sun, Xuyang Shen, Xiaodong Han, Yunshen Wei, Baohong Lv, Fei Yuan, Xiao Luo, Y. Qiao, and Yiran Zhong. Transnormerllm: A faster and better large language model with improved transnormer. 2023. URL https://api.semanticscholar.org/CorpusID:260203124.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. *arXiv preprint arXiv:2404.07904*, 2024a.

Zhen Qin, Songlin Yang, and Yiran Zhong. Hierarchically gated recurrent neural network for sequence modeling. *Advances in Neural Information Processing Systems*, 36, 2024b.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Melbourne, Australia, 2018. Association for Computational Linguistics.

Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. *arXiv preprint arXiv:2406.07522*, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9355–9366. PMLR, 2021a.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear Transformers Are Secretly Fast Weight Programmers. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9355–9366. PMLR, 2021b.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artif. Intell.*, 46(1-2):159–216, 1990. 10.1016/0004-3702(90)90007-M. URL https://doi.org/10.1016/0004-3702(90)90007-M.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): Rnns with expressive hidden states. *CoRR*, abs/2407.04620, 2024a. 10.48550/ARXIV.2407.04620. URL https://doi.org/10.48550/arXiv.2407.04620.

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*, 2023a.

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*, 2023b.

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

Jos van der Westhuizen and Joan Lasenby. The unreasonable effectiveness of the forget gate. *CoRR*, abs/1804.04849, 2018.

Kaiyue Wen, Xingyu Dang, and Kaifeng Lyu. RNNs are not Transformers (Yet): The Key Bottleneck on In-context Retrieval. *CoRR*, abs/2402.18510, 2024. 10.48550/ARXIV.2402.18510. URL https://doi.org/10.48550/arXiv.2402.18510. arXiv: 2402.18510.

Bernard Widrow, Marcian E Hoff, et al. Adaptive switching circuits. In *IRE WESCON convention record*, volume 4, pp. 96–104. New York, 1960.

Songlin Yang and Yu Zhang. Fla: A triton-based library for hardware-efficient implementations of linear attention mechanism, January 2024.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 56501–56523. PMLR, 21–27 Jul 2024a. URL https://proceedings.mlr.press/v235/yang24ab.html.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *NeurIPS*, 2024b.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Yu Zhang, Songlin Yang, Ruijie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, Peng Zhou, and Guohong Fu. Gated slot attention for efficient linear-time sequence modeling. 2024. URL https://api.semanticscholar.org/CorpusID:272593079.

A APPENDIX

A.1 EXTENDED WY REPRESENTATION FOR GATED DELTA RULE

To reduce notation clutter, we only consider the first chunk here.

For $\mathbf{S}_t$, the extended WY representation is

$$\mathbf{S}_t = \sum_{i=1}^{t} \frac{\gamma_t}{\gamma_i} \mathbf{u}_i \boldsymbol{k}_i^{\intercal}, \qquad \mathbf{u}_t = \beta_t \left(\boldsymbol{v}_t - \sum_{i=1}^{t-1} \frac{\gamma_t}{\gamma_i} \mathbf{u}_i \boldsymbol{k}_i^{T} \boldsymbol{k}_t \right)$$

We proof this by mathmetical induction.

Proof.

$$\begin{aligned}
\mathbf{S}_{t+1} &= \mathbf{S}_t \left(\alpha_{t+1} (\mathbf{I} - \beta_{t+1} \boldsymbol{k}_{t+1} \boldsymbol{k}_{t+1}^{\intercal}) \right) + \beta_{t+1} \boldsymbol{v}_{t+1} \boldsymbol{k}_{t+1}^{\intercal} \\
&= \alpha_{t+1} (\sum_{i=1}^{t} \frac{\gamma_t}{\gamma_i} \mathbf{u}_i \boldsymbol{k}_i^{\intercal}) - \alpha_{t+1} \beta_{t+1} (\sum_{i=1}^{t} \frac{\gamma_t}{\gamma_i} \mathbf{u}_i \boldsymbol{k}_i^{\intercal} \boldsymbol{k}_i \boldsymbol{k}_{t+1}^{\intercal}) + \beta_{t+1} \boldsymbol{v}_{t+1} \boldsymbol{k}_{t+1}^{\intercal} \\
&= \sum_{i=1}^{t} \frac{\gamma_{t+1}}{\gamma_i} \mathbf{u}_i \boldsymbol{k}_i^{\intercal} + \underbrace{\beta_{t+1} \left(\boldsymbol{v}_{t+1} - \sum_{i=1}^{t} \frac{\gamma_{t+1}}{\gamma_i} \mathbf{u}_i \boldsymbol{k}_i^{T} \boldsymbol{k}_{t+1} \right)}_{\mathbf{u}_{t+1}} \boldsymbol{k}_{t+1}^{\intercal} \\
&= \sum_{i=1}^{t} \frac{\gamma_{t+1}}{\gamma_i} \mathbf{u}_i \boldsymbol{k}_i^{\intercal} + \underbrace{\frac{\gamma_{t+1}}{\gamma_{t+1}}}_{1} \mathbf{u}_{t+1} \boldsymbol{k}_{t+1}^{\intercal} \\
&= \sum_{i=1}^{t+1} \frac{\gamma_{t+1}}{\gamma_i} \mathbf{u}_i \boldsymbol{k}_i^{\intercal}
\end{aligned}$$

□

For $\mathbf{P}_t$,

$$\begin{aligned}
\mathbf{P}_t &= \prod_{i=1}^{t} \alpha_t \left(\mathbf{I} - \beta_i \boldsymbol{k}_i \boldsymbol{k}_i^{\intercal} \right) \\
&= \underbrace{\left(\prod_{i=1}^{t} \alpha_t \right)}_{\gamma_t} \underbrace{\left(\prod_{i=1}^{t} (\mathbf{I} - \beta_i \boldsymbol{k}_i \boldsymbol{k}_i^{\intercal}) \right)}_{\mathbf{I} - \sum_{i=1}^{t} \mathbf{w}_i \boldsymbol{k}_i^{\intercal}}
\end{aligned}$$

and

$$\prod_{i=1}^{t} (\mathbf{I} - \beta_i \boldsymbol{k}_i \boldsymbol{k}_i^{\intercal}) = \mathbf{I} - \sum_{i=1}^{t} \mathbf{w}_i \boldsymbol{k}_i^{\intercal}, \quad \mathbf{w}_n = \beta_n \boldsymbol{k}_n - \beta_n \sum_{t=1}^{n-1} (\mathbf{w}_t (\boldsymbol{k}_t^{\intercal} \boldsymbol{k}_n)$$

has already been proved in Yang et al. (2024b).

A.2 ABLATION STUDY

In this section, we present an ablation study for different hybrid integration patterns that were considered for designing the Gated DeltaNet-H2 model. This model comprises of Gated DeltaNet, Mamba2 and SWA blocks. However, it is not readily clear how these different blocks should be integrated. As shown in Table S.1, we study four different patterns based on different ordering of the aforementioned blocks. In addition, with a 12 layer network architecture, we keep the number of layer comprising of each block type the same to ensure fairness. Hence, the total number of parameters increase to 500M and the performance is generally better than 400M parameter models that were introduced in Table 1.

As seen in Table S.1, the model with Mamba2 + Gated DeltaNet + SWA hybrid design pattern outperforms other models in term of average accuracy. In addition, it shows better perplexity values. Hence, we chose this pattern as part of the Gated DeltaNet-H2 model.

Model	**Wiki.** ppl ↓	**LMB.** ppl ↓	**LMB.** acc ↑	**PIQA** acc ↑	**Hella.** acc_n ↑	**Wino.** acc ↑	**ARC-e** acc ↑	**ARC-c** acc_n ↑	**SIQA** acc ↑	**BoolQ** acc ↑	**Avg.**
Hybrid Ablations (500M/15B)											
Gated DeltaNet + SWA + Mamba2	24.02	28.20	34.77	67.08	40.84	50.74	60.35	28.83	38.94	61.49	47.88
Gated Gated DeltaNet + Mamba2 + SWA	23.69	26.83	36.17	67.51	41.51	51.85	61.19	29.77	38.58	53.73	47.54
Mamba2 + SWA + Gated DeltaNet	24.14	25.21	36.79	64.96	41.18	52.01	60.90	30.03	38.07	59.44	47.92
Mamba2 + Gated DeltaNet + SWA	**23.54**	**24.11**	36.92	66.48	41.70	52.72	61.06	30.54	39.91	60.51	**48.73**

Table S.1: Ablation studies of Gated DeltaNet models. All evaluations are performed by using `lm-evaluation-harness` (Gao et al., 2021). All models use the Mistral tokenizer and are trained on the same subset of the FineWeb-Edu dataset (Penedo et al., 2024).