Parallelizing Linear Transformers with the Delta Rule over Sequence Length

Songlin Yang[◇] **Bailin Wang**[◇] **Yu Zhang**[†] **Yikang Shen**[‡] **Yoon Kim**[◇]
[◇]Massachusetts Institute of Technology [†]Soochow University [‡]MIT-IBM Watson AI Lab
yangsl66@mit.edu

Abstract

Transformers with linear attention (i.e., linear transformers) and state-space models have recently been suggested as a viable linear-time alternative to transformers with softmax attention. However, these models still underperform transformers especially on tasks that require in-context retrieval. While more expressive variants of linear transformers which replace the additive update in linear transformers with the delta rule [DeltaNet; 99] have been found to be more effective at associative recall, existing algorithms for training such models do not parallelize over sequence length and are thus inefficient to train on modern hardware. This work describes a hardware-efficient algorithm for training linear transformers with the delta rule, which exploits a memory-efficient representation for computing products of Householder matrices [11]. This algorithm allows us to scale up DeltaNet to standard language modeling settings. We train a 1.3B model for 100B tokens and find that it outperforms recent linear-time baselines such as Mamba [30] and GLA [116] in terms of perplexity and zero-shot performance on downstream tasks. We also experiment with two hybrid models which combine DeltaNet layers with (1) sliding-window attention layers every other layer or (2) two global attention layers, and find that these hybrids outperform strong transformer baselines.

1 Introduction

The attention mechanism [8, 112] has been shown to be an important primitive for accurate sequence modeling. Attention is moreover efficient during training as it is rich in matrix multiplications and can thus take advantage of highly parallel processing capabilities and specialized accelerators on modern GPUs. However, the complexity of attention is quadratic in sequence length, and hence it is a fundamentally expensive primitive. And while recent techniques have made it possible to scale attention to longer sequences through hardware-aware restructuring of the intermediate computations [19, 17, 57, 14], these methods still require storing the key/value vectors of previous elements, and this "KV cache" (whose size grows linearly) can be unwieldy to manage for long sequences.

Linear attention transformers [47] replace the exponential kernel in softmax attention with a dot-product over (possibly transformed) key and query vectors. This makes it possible to formulate linear attention as a linear RNN with matrix-valued hidden states, thus obviating the need for a KV cache and enabling constant-memory inference. While initial variants of linear attention generally underperformed softmax attention on language modeling, gated variants of linear attention which incorporate a data-dependent gating factor have recently been shown to be competitive against strong transformer baselines [116, 89, 9, 77]. These gated linear transformers, along with time-varying state space models such as Mamba [30, 18] (which can be reparameterized as a gated linear transformer [116]), have been suggested as a potential alternative to ordinary transformers. However, despite

The parallel DeltaNet layer is made available as part of the FLASHLINEARATTENTION library [116, 115]: https://github.com/sustcsonglin/flash-linear-attention

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

the competitive language modeling performance, these models have been shown to underperform transformers on recall-intensive tasks [6, 7], which is important for many practical downstream tasks of interest (e.g., in retrieval-augmented generation [52]).

To enhance associative recall over long contexts, Schlag et al. [99] propose DeltaNet, a variant of a linear transformer which uses a delta rule-like update [114] to retrieve and update a value vector that is associated with the current key. DeltaNet was found to be effective on synthetic tasks and small scale language modeling/machine translation. However, the original work used a sequential algorithm that did not parallelize across sequence length, thus resulting in hardware-inefficient training, and it has not been clear how to scale DeltaNet to larger models and datasets.

This work describes a hardware-efficient training algorithm for DeltaNets which parallelizes the forward/backward passes across sequence length. We reparameterize the DeltaNet as a matrix-valued RNN whose recurrence is given by a generalized Householder transformation. This reparameterization enables the use of the compact WY representation [11] for products of Householder matrices, eliminating the need to materialize the hidden states of matrix size at each time step during parallel training, which would otherwise result in high I/O costs. The memory-efficient representation makes it possible to straightforwardly extend the chunkwise parallel strategy for training linear attention models [33, 105, 116] to the DeltaNet case. We scale DeltaNets to moderate-scale language modeling benchmarks (1.3B models trained on 100B tokens), where DeltaNet is found to obtain better language modeling and zero-shot downstream task performance than strong linear recurrent models such as Mamba [30] and GLA [116]. For in-context retrieval and learning evaluation, we evaluate DeltaNet on synthetic and real benchmarks [4, 2, 82, 6], where it is again found to perform well against linear recurrent baselines. Finally, we experiment with a hybrid approach where we combine DeltaNet layers with sliding attention layers or global attention layers, and find that these hybrid models can improve upon ordinary transformers, as well as the pure DeltaNet transformer.

2 Background

2.1 Linear Transformer: Transformers with Linear Attention

Given a sequence of d-dimensional input vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L$, transformers use the softmax attention mechanism to attend over the entire past,

$$\boldsymbol{q}_t,\ \boldsymbol{k}_t,\ \boldsymbol{v}_t = \boldsymbol{W}_Q\boldsymbol{x}_t, \boldsymbol{W}_K\boldsymbol{x}_t, \boldsymbol{W}_V\boldsymbol{x}_t, \qquad \boldsymbol{o}_t = \sum_{i=1}^{t} \frac{\exp(\boldsymbol{k}_i^\top \boldsymbol{q}_t)}{\sum_{j=1}^{t} \exp(\boldsymbol{k}_j^\top \boldsymbol{q}_t)} \boldsymbol{v}_i,$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d\times d}, \boldsymbol{q}_t, \boldsymbol{k}_t, \boldsymbol{v}_t, \boldsymbol{o}_t \in \mathbb{R}^d$. (Here we assume a single attention head for simplicity). Linear attention [47] replaces the exponential kernel $\exp(\boldsymbol{k}_i^\top \boldsymbol{q}_t)$ with the dot-product $\phi(\boldsymbol{k}_i)^\top \phi(\boldsymbol{q}_t)$ where $\phi : \mathbb{R}^d \to \mathbb{R}^n$ is a feature map. This makes it possible to rearrange computations to represent linear attention as a linear RNN with matrix-valued hidden states,

$$\boldsymbol{o}_t = \sum_{i=1}^{t} \frac{\phi(\boldsymbol{k}_i)^\top \phi(\boldsymbol{q}_t)}{\sum_{j=1}^{t} \phi(\boldsymbol{k}_j)^\top \phi(\boldsymbol{q}_t)} \boldsymbol{v}_i = \frac{\left(\sum_{i=1}^{t} \boldsymbol{v}_i \phi(\boldsymbol{k}_i)^\top\right) \phi(\boldsymbol{q}_t)}{\left(\sum_{j=1}^{t} \phi(\boldsymbol{k}_j)^\top\right) \phi(\boldsymbol{q}_t)} = \frac{\mathbf{S}_t \phi(\boldsymbol{q}_t)}{\mathbf{z}_t^\top \phi(\boldsymbol{q}_t)},$$

where $\mathbf{S}_t = \sum_{i=1}^{t} \boldsymbol{v}_i \phi(\boldsymbol{k}_i)^\top \in \mathbb{R}^{d\times n}$ and $\mathbf{z}_t = \sum_{i=1}^{t} \phi(\boldsymbol{k}_i) \in \mathbb{R}^n$. If we allow n to go to infinity, linear attention can use feature maps associated with polynomial kernels to compute a polynomial approximation to the exponential kernel as a dot product, and can thus approximate softmax attention arbitrarily well [6]. The denominator $\mathbf{z}_t^\top \phi(\boldsymbol{q}_t) \in \mathbb{R}$ can result in numerical instabilities [84] and is removed in recent works [98, 61]. It is also common to use the identity mapping for ϕ [61, 105], which results in the following simplified linear transformer: $\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top$, $\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$.

Efficient training. Let $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{L\times d}$ be the stacked query, key, value vectors, e.g., $\mathbf{Q}_i = \boldsymbol{q}_i$. We can then compute the output $\mathbf{O} \in \mathbb{R}^{L\times d}$ in parallel via $\mathbf{O} = \left(\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M}_L\right)\mathbf{V}$, where $\mathbf{M}_L \in \mathbb{R}^{L\times L}$ is the causal mask. This fully "parallel form" and the above "recurrent form" have different FLOPs and parallelization tradeoffs. The parallel form takes $O(L^2 d + L d^2)$ and thus requires more FLOPs than the recurrent form, which takes $O(L d^2)$. However, the parallel form is often much faster in practice for moderate-length sequences as it can be done in $O(1)$ steps. This sequence-level parallellism also enables high GPU occupancy. The recurrent form requires fewer

FLOPs but cannot be parallelized across sequence length[1] and the elementwise operations involved in recurrence moreover cannot make use of specialized matmul accelerators (e.g., tensor cores).

Chunkwise parallel form. The chunkwise parallel form [33, 105, 116] strikes a balance between the parallel and recurrent forms, allowing for fewer FLOPs than the parallel form and more sequence-level parallelism than the recurrent form. Concretely, suppose the query/key/value vectors are split into $\frac{L}{C}$ chunks where each chunk is of length C. Let $\mathbf{Q}_{[t]} \in \mathbb{R}^{C\times d}$ be all the query vectors for chunk t, and let $\boldsymbol{q}^i_{[t]} = \boldsymbol{q}_{tC+i}$ be the i-th query vector within the t'th chunk; the key/value chunks are defined similarly. Note that $t \in [0, L/C)$, $i \in [1, C]$. The state matrices are also re-indexed such that $\mathbf{S}^i_{[t]} = \mathbf{S}_{tC+i}$, and we additionally define $\mathbf{S}^0_{[t]} = \mathbf{S}^C_{[t-1]}$, i.e., the initial state of a chunk is the last state of the previous chunk. We can then obtain the following identity for the hidden state and output vector for the r-th element within the t-th chunk,

$$\mathbf{S}^r_{[t]} = \mathbf{S}^0_{[t]} + \sum_{i=1}^{r} \boldsymbol{v}^i_{[t]} \boldsymbol{k}^{i\intercal}_{[t]}, \qquad \boldsymbol{o}^r_{[t]} = \mathbf{S}^0_{[t]} \boldsymbol{q}^r_{[t]} + \sum_{i=1}^{r} \boldsymbol{v}^i_{[t]} \left(\boldsymbol{k}^{i\intercal}_{[t]} \boldsymbol{q}^r_{[t]}\right).$$

By further rewriting the intra-chunk computation based on the parallel form, we obtain following,

$$\mathbf{S}_{[t+1]} = \mathbf{S}_{[t]} + \mathbf{V}^\intercal_{[t]} \mathbf{K}_{[t]} \qquad \in \mathbb{R}^{d\times d}, \tag{1}$$

$$\mathbf{O}_{[t]} = \mathbf{Q}_{[t]} \mathbf{S}^\intercal_{[t]} + \left(\mathbf{Q}_{[t]} \mathbf{K}^\intercal_{[t]} \odot \mathbf{M}_C\right) \mathbf{V}_{[t]} \qquad \in \mathbb{R}^{C\times d} \tag{2}$$

where we let $\mathbf{S}_{[t]} = \mathbf{S}^0_{[t]}$ to reduce notational clutter. With this "chunkwise parallel form", information is propagated chunk-to-chunk through $\mathbf{S}_{[t]}$, and the intra-chunk states $\mathbf{S}^i_{[t]}$ for $i \in [1, C]$ need not be materialized, thus saving memory.

The complexity of the chunkwise parallel form is $O(LCd + Ld^2)$, and the number of steps (without chunk-level parallel scan) is $O(\frac{L}{C})$. Hence, $C = L$ recovers the fully parallel form and $C = 1$ recovers the recurrent form. The chunkwise parallel form allows us to interpolate between the two forms, in essence trading off the number of sequential computations against sequence-level parallelism. In practice C is set to a small constant (usually 64 or 128), allowing for subquadratic training. This chunkwise form enables practical speed-ups against parallel-form-only softmax attention even on moderate-length sequences, as demonstrated by FLASHLINEARATTENTION [116, 115]

2.2 DeltaNet: Linear Transformers with the Delta Update Rule

The above linear transformer employs a simple linear recurrence: $\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\intercal$. This can be seen as additively updating the memory $\mathbf{S}_{t-1}$ with new key-value associations at each time step. However, a purely additive update rule makes it difficult to deallocate past key-value associations, eventually leading to key "collisions" when $L > d$, as pointed out by Schlag et al. [98]. A model should ideally learn to remove less important key-value associations to make room for new ones, and this removal should depend on the interaction between the new key and the memory content.

DeltaNet uses the delta update rule [114] to operationalize this mechanism. Specifically, it first retrieves the old value using the current key, $\boldsymbol{v}^{\text{old}}_t = \mathbf{S}_{t-1}\boldsymbol{k}_t$. It then obtains a new value $\boldsymbol{v}^{\text{new}}_t$ by interpolating between the old value and the current value $\boldsymbol{v}_t$, which replaces $\boldsymbol{v}^{\text{old}}_t$ in the memory:

$$\boldsymbol{v}^{\text{new}}_t = \beta_t \boldsymbol{v}_t + (1-\beta_t)\, \boldsymbol{v}^{\text{old}}_t, \qquad \mathbf{S}_t = \mathbf{S}_{t-1} \underbrace{-\boldsymbol{v}^{\text{old}}_t \boldsymbol{k}_t^\intercal}_{\text{remove}} \underbrace{+\boldsymbol{v}^{\text{new}}_t \boldsymbol{k}_t^\intercal}_{\text{write}}$$

Here $\beta_t = \sigma(\mathbf{W}_\beta \boldsymbol{x}_t) \in (0, 1)$ is a soft "writing strength": when $\beta_t = 1$, the old value is completely removed and $\boldsymbol{v}^{\text{new}}_t = \boldsymbol{v}_t$; when $\beta_t = 0$, the memory remains unmodified and we have $\mathbf{S}_t = \mathbf{S}_{t-1}$. The output computation is the same as vanilla linear attention, i.e., $\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$. The complexity of this recurrent form is the same as that of vanilla linear attention, i.e., $O(Ld^2)$. This DeltaNet is a special case of *fast weight programmers* [100], and Schlag et al. [98] and Irie et al. [36] show that this type of linear transformer outperforms ordinary linear transformers on small-scale language modeling and synthetic in-context retrieval tasks.

[1] It is possible in theory to use parallel scan [13] to parallelize the recurrent form, which would enable the computations to be performed in $O(\log L)$ steps and $O(Ld^2)$ FLOPs. However, this approach requires materializing the 2D hidden state for each time step, which would incur significant memory I/O cost unless the state size is small enough such that materialization can happen in faster memory (i.e., as in Mamba [30]).

Since the old value vector depends on the previous hidden state $\mathbf{S}_{t-1}$, it is not possible to straightforwardly apply the above chunkwise parallel strategy for training DeltaNet transformers. While the official implementation from Schlag et al. [98] avoids materializing the $\mathbf{S}_t$'s (thus minimizing I/O cost) by using the linear-time-constant-memory algorithm from Katharopoulos et al. [47, §3.3.1], it still uses the pure recurrent form and thus does not parallelize across the sequence dimension, which makes it difficult to scale DeltaNet to modern language modeling settings.

3 Parallelizing DeltaNet Across the Sequence Dimension

In the same spirit as the chunkwise form of linear attention, we derive a chunkwise form for DeltaNet that enables hardware-efficient training through parallelizing across the sequence dimension.

3.1 A Memory-efficient Reparameterization

We first observe that $\mathbf{S}_t$ admits a purely additive representation of the form $\mathbf{S}_t = \sum_{i=1}^{t} \boldsymbol{u}_i \boldsymbol{k}_i^\top$ for $\boldsymbol{u}_i, \boldsymbol{k}_i \in \mathbb{R}^d$, since we can simply set $\boldsymbol{u}_i = \boldsymbol{v}_i^{\text{new}} - \boldsymbol{v}_i^{\text{old}} = \beta_i(\boldsymbol{v}_i - \boldsymbol{v}_i^{\text{old}})$. Recall from §2.1 that simple linear attention has the form $\mathbf{S}_t = \sum_{i=1}^{t} \boldsymbol{v}_i \boldsymbol{k}_i^\top$. Thus, DeltaNet simply replaces the value vector $\boldsymbol{v}_i$ in linear attention with the "pseudo" value vector $\boldsymbol{u}_i$. Once the $\boldsymbol{u}_i$'s have been constructed, the rest of computation can proceed as in ordinary linear attention, i.e., $\mathbf{O} = \left(\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M}\right)\mathbf{U}$ where $\mathbf{U} \in \mathbb{R}^{L \times d}$ is the row-wise concatenation of the $\boldsymbol{u}_i$ vectors.

However, computing $\boldsymbol{u}_t$ naïvely requires explicitly materializing $\mathbf{S}_{t-1}$ to compute $\boldsymbol{v}_t^{\text{old}}$, which would require $O(d^2)$ memory. We now show that we can obtain the $\boldsymbol{u}_t$'s *without* explicitly materializing $\mathbf{S}_{t-1}$ in $O(d)$ memory. Our simple proof (by induction) relies on an application of the WY representation for products of Householder matrices [11]. The base case is clear since we have $\mathbf{S}_1 = \beta_1 \boldsymbol{v}_1 \boldsymbol{k}_1^\top$, so $\boldsymbol{u}_1 = \beta_1 \boldsymbol{v}_1$. For the inductive step, we first observe that the DeltaNet update is given by,

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

which can be seen as applying a generalized Householder transformation (i.e., matmul with an identity plus rank-one matrix) to the previous state. The inductive step is then given by,

$$\mathbf{S}_t = \mathbf{S}_{t-1}(\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top) + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\top = \sum_{i=1}^{t-1} \boldsymbol{u}_i \boldsymbol{k}_i^\top + \underbrace{\beta_t \left(\boldsymbol{v}_t - \sum_{i=1}^{t-1} \boldsymbol{u}_i \left(\boldsymbol{k}_i^\top \boldsymbol{k}_t \right) \right)}_{\boldsymbol{u}_t} \boldsymbol{k}_t^\top = \sum_{i=1}^{t} \boldsymbol{u}_i \boldsymbol{k}_i^\top \quad (3)$$

Note that $\boldsymbol{u}_t$ does not require materializing any of the hidden states and requires $O(d)$ memory to compute, thus completing the proof. While we have avoided materializing $\mathbf{S}_t$'s, computing $\boldsymbol{u}_t$'s for all L (that is, $\mathbf{U}$) takes $O(L^2 d)$ and moreover cannot be fully parallelized, unlike in linear attention where we can calculate all the value vectors $\mathbf{V}$ in parallel in $O(1)$ steps. We now show that the above trick still enables an efficient chunkwise parallel form for DeltaNet.

3.2 Chunkwise Parallel Form for DeltaNet

To derive the chunkwise parallel form, we first unroll the recurrence,

$$\mathbf{S}_t = \mathbf{S}_{t-1}(\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top) + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\top = \sum_{i=1}^{t} \beta_i (\boldsymbol{v}_i \boldsymbol{k}_i^\top) \left(\prod_{j=i+1}^{t} (\mathbf{I} - \beta_j \boldsymbol{k}_j \boldsymbol{k}_j^\top) \right). \quad (4)$$

We then define the following variables: $\mathbf{P}_i^j = \prod_{t=i}^{j} (\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top) \in \mathbb{R}^{d \times d}$, $\mathbf{H}_i^j = \sum_{t=i}^{j} \beta_t (\boldsymbol{v}_t \boldsymbol{k}_t^\top) \mathbf{P}_{t+1}^j \in \mathbb{R}^{d \times d}$, where we let $\mathbf{P}_i^j = \mathbf{I}$ whenever $i > j$. Intuitively, $\mathbf{P}_i^j$ is the "decay factor" to be applied to $\mathbf{S}_i$ for obtaining $\mathbf{S}_j$, and $\mathbf{H}_i^j$ represents the contributions to $\mathbf{S}_j$ starting from token i. (Hence $\mathbf{S}_t = \mathbf{H}_1^t$). The chunkwise recurrence can then be written as,

$$\mathbf{S}_{[t]}^r = \mathbf{S}_{[t]}^0 \mathbf{P}_{[t]}^r + \mathbf{H}_{[t]}^r \quad (5)$$

where we define the chunkwise variables $\mathbf{S}_{[t]}^i = \mathbf{S}_{tC+i}$, $\mathbf{P}_{[t]}^r = \mathbf{P}_{tC+1}^{tC+r}$, $\mathbf{H}_{[t]}^r = \mathbf{H}_{tC+1}^{tC+r}$. Here we have $\frac{L}{C}$ chunks of size C. The trick is to now efficiently represent the $\mathbf{P}_{[t]}^r, \mathbf{H}_{[t]}^r \in \mathbb{R}^{d \times d}$ matrices

using a similar approach described in §3.1, so that these matrices can be stored in $O(d)$ memory,

$$\mathbf{P}_{[t]}^r = \mathbf{I} - \sum_{i=1}^{r} \boldsymbol{w}_{[t]}^i \boldsymbol{k}_{[t]}^{i\intercal}, \qquad \mathbf{H}_{[t]}^r = \sum_{t=1}^{r} \boldsymbol{u}_{[t]}^i \boldsymbol{k}_{[t]}^{i\intercal} \quad \in \mathbb{R}^{d\times d} \tag{6}$$

$$\boldsymbol{w}_{[t]}^r = \beta_{[t]}^r \left(\boldsymbol{k}_{[t]}^r - \sum_{i=1}^{r-1} \boldsymbol{w}_{[t]}^i (\boldsymbol{k}_{[t]}^{i\intercal} \boldsymbol{k}_{[t]}^r) \right), \quad \boldsymbol{u}_{[t]}^r = \beta_{[t]}^r \left(\boldsymbol{v}_{[t]}^r - \sum_{i=1}^{r-1} \boldsymbol{u}_{[t]}^i (\boldsymbol{k}_{[t]}^{i\intercal} \boldsymbol{k}_{[t]}^r) \right) \quad \in \mathbb{R}^d \tag{7}$$

The derivations for the above can be found in the appendix. Subsequently, based on Eq. 5, we can obtain the chunk-level recurrence for hidden states and outputs as,

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

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

Letting $\mathbf{S}_{[t]} = \mathbf{S}_{[t]}^0$, the above can be simplified to matrix notations similarly to Eq.1-2,

$$\mathbf{S}_{[t+1]} = \mathbf{S}_{[t]} + \left(\mathbf{U}_{[t]} - \mathbf{W}_{[t]} \mathbf{S}_{[t]}^\intercal \right)^\intercal \mathbf{K}_{[t]}, \tag{8}$$

$$\mathbf{O}_{[t]} = \mathbf{Q}_{[t]} \mathbf{S}_{[t]}^\intercal + (\mathbf{Q}_{[t]} \mathbf{K}_{[t]}^\intercal \odot \mathbf{M}) \left(\mathbf{U}_{[t]} - \mathbf{W}_{[t]} \mathbf{S}_{[t]}^\intercal \right) \tag{9}$$

where $\square_{[t]} = \square_{[t]}^{1:C} \in \mathbb{R}^{C\times d}$ for $\square \in \{\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{O}, \mathbf{U}, \mathbf{W}\}$ defines the chunkwise matrices that are formed from stacking the $\boldsymbol{q}_t, \boldsymbol{k}_t, \boldsymbol{v}_t, \boldsymbol{o}_t, \boldsymbol{u}_t, \boldsymbol{w}_t$ vectors.

Practical considerations. In the above, Eq. 7 is fully recurrent and thus cannot use tensor cores written as is. To solve this, we further leverage the *UT transform* [43, 22]:

$$\mathbf{T}_{[t]} = \left(\mathbf{I} - \operatorname{tril}(\operatorname{Diag}(\beta_{[t]}) \mathbf{K}_{[t]} \mathbf{K}_{[t]}^\intercal, -1) \right)^{-1} \operatorname{Diag}\left(\beta_{[t]}\right) \tag{10}$$

$$\mathbf{W}_{[t]} = \mathbf{T}_{[t]} \mathbf{K}_{[t]}, \qquad \mathbf{U}_{[t]} = \mathbf{T}_{[t]} \mathbf{V}_{[t]} \tag{11}$$

to rewrite most operations in matmuls. The inverse of lower triangular matrices could be solved efficiently using forward substitution. Once computed, the hidden state updates (Eq. 8) and the output computations (Eq. 9) are largely the same as in vanilla linear attention. We adapt FLASHLINEARATTENTION [116] to implement Eq. 8 and 9 with hidden states recomputed during the backward pass for saving GPU memory. The PyTorch pseudocode for the forward pass is shown in Listing 1.

Speed comparison. We implement both the pure recurrent form[2] and the chunkwise parallel form in Triton [109] and show the speed-ups for various sequence lengths (L) and head dimensions (d_{head}) in the right figure, where the model dimension d is 2048 and we vary batch size and sequence length so that they multily to 16384.[3] Our chunkwise algorithm achieves greater speed-ups as sequence length L and head dimension d_{head} increase, where the use of sequence-level parallelism (for high GPU occupancy) and tensor core (for fast matmuls) become more important [116, §3].

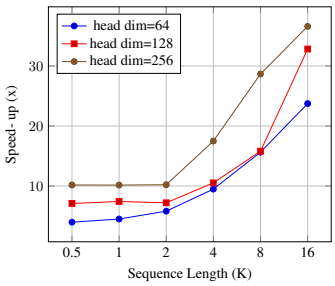

Figure 1: Speed-up of the chunkwise parallel form vs. the recurrent form.

Fully Parallel Form for DeltaNet. For completeness, we also discuss the fully parallel form of DeltaNet. While we use the concept of a "pseudo" value, it is possible to avoid modifying values. From Eq. 4, it is straightforward to compute the attention matrix $\mathbf{A}$: $\mathbf{A}_{ij} = \boldsymbol{k}_j^\intercal \mathbf{P}_{j+1}^i \boldsymbol{q}_i$ if $j \leq i$ and 0 otherwise. Notably, $\mathbf{A}$ has the matrix form $\mathbf{A} = \left(\mathbf{Q}\mathbf{K}^T \odot \mathbf{M}\right) \mathbf{T}$, obtained by combining Eq. 3 and 11. However, computing $\mathbf{T}$ requires a matrix inverse (Eq. 10), which scales cubically with sequence length without further algorithmic changes. Due to the above we avoid using the fully parallel form for training DeltaNet; however the "attention" matrix derived from this form could be of interest to the interpretability research community studying RNNs, as explored in Ali et al. [3] and Zimerman et al. [123].

[2] Note that our recurrent kernel is already 2× faster than the original CUDA kernel from Schlag et al. [99].

[3] So far we have been assuming a single head ($d_{\text{head}} = d$) for easier exposition. In practice we use multiple heads where the head dimension d_{head} is smaller than the model dimension d. We thus have $\mathbf{S}_t \in \mathbb{R}^{d\times d_{\text{head}}}$.

3.3 DeltaNet Transformer

We describe how the DeltaNet layer primitive is used to build up a transformer-like model using standard modules. We largely follow the LLaMA-architecture [Transformer++, 111] and simply replace the self-attention layer with the DeltaNet layer. We also apply normalization before output projection for stable training [84, 66]. As the additional parameters for computing scalar β_t terms are negligible, parameter allocation is roughly the same as in Transformer++, i.e., $4d^2$ for the DeltaNet layer and $8d^2$ for the SwiGLU FFN layer [101].

Feature map and normalization. Our key/query vectors are given by $\boldsymbol{k}_t = \frac{\mathrm{SiLU}(\mathbf{W}_K \boldsymbol{x}_t)}{\|\mathrm{SiLU}(\mathbf{W}_K \boldsymbol{x}_t)\|_2}$, $\boldsymbol{q}_t = \frac{\mathrm{SiLU}(\mathbf{W}_Q \boldsymbol{x}_t)}{\|\mathrm{SiLU}(\mathbf{W}_Q \boldsymbol{x}_t)\|_2}$. Schlag et al. [98] originally follow Katharopoulos et al. [47] and apply a "ELU + 1" [16] to nonlineary transform the key/query vectors. We instead use the SiLU activation [23], which was found to perform better [86, 18]. For stability, it is crucial to ensure that the norm of each eigenvalue of the transition matrices does not exceed one. The eigenvalues of $\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top$ are 1 with multiplicity $d-1$ and $1 - \beta_t \|\boldsymbol{k}_t\|_2$ with multiplicity 1. Schlag et al. [98] used the L_1 norm to normalize query/key vectors, ensuring that $0 \leq 1 - \beta_t \|\boldsymbol{k}_t\|_2 \leq 1$. We instead apply L_2 normalization, which we found to perform better and offers a more intuitive interpretation: when $\beta_t = 1$, $\mathbf{I} - \boldsymbol{k}_t \boldsymbol{k}_t^\top$ becomes a projection matrix, erasing information in one subspace while preserving the other $d-1$ subspaces. This is beneficial for retaining information while enabling more *targeted* forgetting.

3.4 Hybrid Models

Following recent work on combining subquadratic token-mixing layers with existing neural network primitives [6, 20, 53], we also experiment with hybridizing DeltaNet models.

Convolutional layers. Recent linear recurrent models typically incorporate a lightweight depthwise-separable convolution layer after the query/key/value projections [30, 9, 18]. This "short convolution" layer [81] generalizes the shift SSM [25], and is efficient in both number of parameters and computational cost. We also add a short convolution layer after the query/key/value projections.

Local sliding window and global attention. Linear attention largely uses a content-based addressing mechanism [28] and lacks positional information [120]. Arora et al. [6] also argue that linear attention lacks the ability to perform precise local token shifts and comparisons, thus facing difficulties on retrieval-intensive tasks. Motivated by this, we experiment with two different hybrid architectures that incorporate softmax attention. We first explore *sliding window attention* (SWA) which has been shown to significantly improve linear attention [84, 6, 55, 72]; we follow Griffin [20] and Samba [93] to interleave DeltaNet layers and SWA layers. We also experiment with *global attention*, which has been found to be helpful [50, 34] even if only few of the recurrent layers are replaced with global attention [53]. We follow Fu et al. [25] to replace only two layers with global attention: the second layer and the $(\frac{N}{2}+1)$-th layer, where N is total number of layers.

4 Empirical Study

We compare the DeltaNet against strong baselines in both synthetic and real-world language modeling settings. Our main baselines include: LLaMA-architecture Transformer++ [111]; RetNet [105], a linear attention Transformer with non-data-dependent exponential decay and large head dimension; GLA [116], a linear attention Transformer with data-dependent decay; and Mamba [30], a selective state-space model with data-dependent decay.

4.1 Synthetic Benchmarks

We evaluate on three synthetic benchmarks: Multi-query associative recall [MQAR; 4], Mechanistic Architecture Design [MAD; 82], and in-context language learning [RegBench; 2].

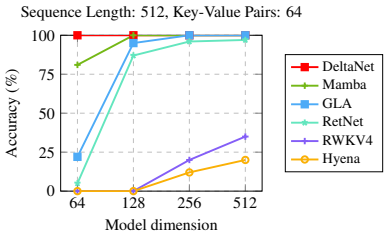

Figure 2: Accuracy (%) on MQAR.

MQAR evaluates language models' ability to (in-context) recall information within a context when faced with multiple recall queries. We use Arora et al. [4]'s training setting and for DeltaNet we use 2 heads. We do not use convolutions for these experiments. Figure 2 shows that DeltaNet performs perfectly (even without convolution) in the hardest setting and outperforms Mamba (which uses convolutions) in the low-dimension setting. Next, we consider the MAD benchmark [82], a suite of synthetic token

Model	Compress	Fuzzy Recall	In-Context Recall	Memorize	Noisy Recall	Selective Copy	Average
Transformer	51.6	29.8	94.1	85.2	86.8	99.6	74.5
Hyena [81]	45.2	7.9	81.7	89.5	78.8	93.1	66.0
Multihead Hyena [63]	44.8	14.4	99.0	89.4	98.6	93.0	73.2
Mamba [30]	52.7	6.7	90.4	89.5	90.1	86.3	69.3
GLA [116]	38.8	6.9	80.8	63.3	81.6	88.6	60.0
DeltaNet	42.2	35.7	100	52.8	100	100	71.8

Table 1: Results on the synthetic MAD benchmark. Results other than DeltaNet are directly borrowed from Poli et al. [82]. (Multi-head) Hyena, DeltaNet and Mamba make use of convolutions, whereas GLA does not.

Model	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	Avg.	SWDE acc ↑	SQuAD acc ↑	FDA acc ↑	State exp.
340M params / 15B tokens													
Transformer++	28.39	42.69	31.0	63.3	34.0	50.4	44.5	24.2	41.2	42.2	22.1	21.4	N/A
RetNet (*w/o. conv*)	32.33	49.19	28.6	63.5	33.5	52.5	44.5	23.4	41.0	13.3	27.6	2.9	512x
Mamba (*w. conv*)	28.39	39.66	30.6	65.0	35.4	50.1	46.3	23.6	41.8	12.4	23.0	2.1	64x
GLA (*w/o. conv*)	28.65	43.35	30.3	64.8	34.5	51.4	45.1	22.7	41.5	18.6	27.2	8.1	128x
(*w. conv*)	29.47	45.53	31.3	65.1	33.8	51.6	44.4	24.6	41.8	24.0	24.7	7.3	128x
DeltaNet (*w/o. conv*)	29.08	50.87	30.0	63.6	33.6	51.7	46.0	23.0	41.3	24.6	26.9	4.5	128x
DeltaNet (*w. conv*)	28.24	37.37	32.1	64.8	34.3	52.2	45.8	23.5	42.1	26.4	28.9	12.8	128x
+ Sliding Attn	27.06	38.17	33.4	64.0	35.3	50.9	45.9	23.2	42.1	39.3	32.5	18.8	N/A
+ Global Attn (2 layers)	27.51	35.04	33.5	64.0	34.5	51.7	46.0	23.3	42.1	42.9	32.1	23.1	N/A
1.3B params / 100B tokens													
Transformer++	16.85	13.44	48.9	70.8	49.6	53.6	56.0	26.5	50.9	66.6	31.5	27.4	N/A
RetNet (*w/o. conv*)	18.64	17.27	43.3	70.0	47.3	52.5	54.8	25.6	48.9	42.8	34.7	14.3	512x
Mamba (*w. conv*)	17.06	13.89	46.2	72.2	40.1	54.1	59.0	28.2	50.0	41.4	35.2	6.2	64x
GLA (*w/o. conv*)	17.22	14.47	46.9	71.8	49.8	53.9	57.2	26.6	51.0	50.6	42.6	19.9	256x
(*w. conv*)	17.25	14.92	46.2	70.6	49.9	53.0	55.3	27.0	50.4	52.4	37.4	22.3	256x
DeltaNet (*w. conv*)	16.87	12.21	48.9	71.2	50.2	53.6	57.2	28.3	51.6	49.5	37.4	17.2	128x
+ Sliding Attn	16.56	11.74	49.2	71.8	51.1	52.8	58.9	28.8	52.1	53.3	43.3	22.3	N/A
+ Global Attn (2 layers)	16.55	12.40	48.8	70.8	50.7	54.2	58.4	28.1	51.8	71.0	43.0	29.8	N/A
DeltaNet Ablations (340M)													
w. L_1-norm & 1+ELU	31.12	55.96	26.3	63.9	33.0	50.9	44.3	21.8	40.1	14.5	23.9	6.2	128x
w. L_2-norm & 1+ELU	28.03	37.62	32.2	65.7	34.7	51.8	45.4	22.5	42.1	23.8	28.6	13.1	128x
w. L_2-norm & ReLU	28.75	43.53	30.2	64.0	33.9	48.9	45.6	22.8	40.9	27.2	26.7	9.0	128x

Table 2: Main language modeling results against Transformer++, RetNet [105], Mamba [30], and GLA [116]. All models are trained on the same subset of the SlimPajama dataset with the Mistral tokenizer. The Transformer++, RetNet, Mamba, GLA (*w/o. conv*) results are taking from Yang et al. [116]. For hybrid models, "Sliding Attn" interleaves a sliding window attention every other layer, and "Global Attn" uses full global attention on two layers. The 340M/1.3B models are trained for 15B/100B tokens respectively. All results are obtained through `lm-evaluation-harness` [26]. The last column denotes the expansion ratio of the recurrent state size relative to the product of the number of layers and model dimension (see Zhang et al. [122, App. C]).

manipulation tasks designed to probe capabilities of model architectures. The results are shown in Table 1. Compared with other architectures, including MHA, DeltaNet is better at recalling tasks, especially on Fuzzy Recall as expected, although it struggles on the "Memorize" task.

Finally, we consider RegBench [2], a synthetic data set designed to assess the in-context language learning capability of different model architectures. Each input sequence in this benchmark consists of 10 to 20 strings drawn from a distinct language defined by a probabilistic finite automaton (PFA), so that a model needs to infer the underlying language from the context on the fly. During testing, a model is evaluated on predicting the next token of testing sequences generated from held-out PFAs. Here again we find that DeltaNet performs strongly compared to baselines, as shown in Figure 3.

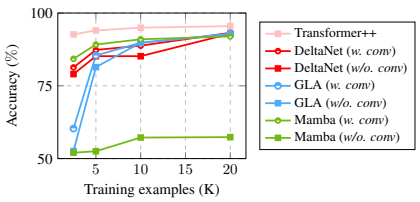

Figure 3: Accuracy (%) on RegBench.

4.2 Language Modeling

Experimental setup. Following prior work [30, 116], we evaluate on Wikitext perplexity and zero-shot common sense reasoning tasks, including LAMBADA [LMB.; 74], PiQA [12], HellaSwag [Hella.; 118], WinoGrande [Wino.; 96], ARC-easy (ARC-e) and ARC-challenge (Arc-c) [15]. Following Arora et al. [6], we also evaluate the models real-world recall-intensive tasks, including FDA [5], SWDE [58], and SQUAD [91]. Both SWDE and FDA focus on extracting structured information: SWDE from raw HTML to identify semi-structured relationships, and FDA from PDFs to retrieve key-value pairs. SQUAD evaluates language models on reading comprehension by providing a text passage and a related question. See §D for hyperparameter settings.

Model	ARC	HellaSwag	OBQA	PIQA	WinoGrande	MMLU	Average
Llama-3.2-3B [108]	59.1	73.6	43.4	77.5	69.2	54.1	62.8
PowerLM-3B [102]	60.5	74.6	43.6	79.9	70.0	45.0	62.3
DeltaNet-3B	60.4	72.8	41.0	78.5	65.7	40.7	59.8
RecurrentGemma-2B [29]	57.0	71.1	42.0	78.2	67.6	31.8	57.9
RWKV-6-3B [77]	49.5	68.6	40.6	76.8	65.4	28.4	54.9
Mamba-2.7B [30]	50.3	65.3	39.4	75.8	63.1	26.1	53.3

Table 3: Zero-shot model performance across selected benchmarks for 3B models. Llama-3.2-3B and PowerLM-3B are Transformer models, while the others are recurrent models.ARC results are averaged over accuracy and normalized accuracy across ARC-Easy and ARC-Challenge.

Results. Our main language modeling results are shown in Table 2. Since Mamba uses convolutions by default while GLA does not, we retrain the GLA with convolution, and also train DeltaNet without convolution. For the 1.3B setting we only train the DeltaNet with convolution due to limited compute resources. In general we find that DeltaNet outperforms the strong Mamba/GLA baselines in terms of both perplexity and downstream task performance. For recall-intensive tasks (i.e., SWDE, SQuAD, FDA), we find that under the same state size at the 340M scale, DeltaNet outperforms GLA, confirming the effectiveness of the delta rule. However, at the 1.3B scale, DeltaNet underperforms GLA due to its poorer state size scability (see §5.3), since state size plays an important role in recall-intensive tasks. Finally, we confirm the benefits of hybrid architectures [20, 53]: both the sliding window and global attention hybrids work well, outperforming the strong Transformer++ baselines.

We also scale DeltaNet to the 3B parameter scale trained with 1T tokens using the same settings as Shen et al. [102]. The results are shown in Table 3, where 3B DeltaNet slightly underperforms a Transformer architecture trained with the same setting (PowerLM-3B), but outperforms other RNN baselines in the 2B–3B range (though these are trained for a different number of tokens so are not exactly comparable).

Ablations. In Table 2 (bottom) we ablate the choice of feature map and normalization. We find that simply replacing the L_1-norm with the L_2-norm greatly increases performance. For the feature map, we experiment with $\{\texttt{ReLU}, \texttt{1 + ELU}, \texttt{SiLU}\}$ and find that `SiLU` performs the best, consistent with prior work [86].

Training throughput. Figure 4 compares the training throughputs of different 1.3B models in different training lengths and batch size settings. The training speed of DeltaNet is close to GLA and significantly faster than Mamba. All linear-time models outperform Transformers for longer-sequence training.

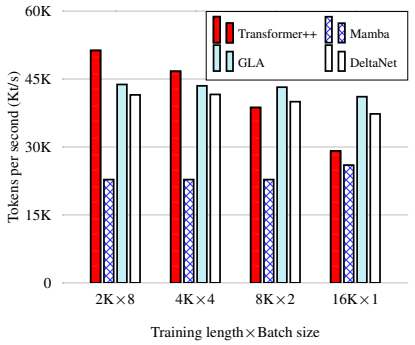

Figure 4: Training throughput of 1.3B models on a single H100.

5 Discussion and Related Work

5.1 DeltaNet vs. State Space Models / Linear RNNs

To discuss DeltaNet against existing linear RNNs (including state-space models) we first introduce a general class of associative RNNs with matrix-valued hidden states. Given a matrix-valued hidden state $\mathbf{S}_t \in \mathbb{R}^{d \times n}$ and current input $\boldsymbol{x}_t \in \mathbb{R}^d$, these models have the following form:

$$\mathbf{S}_t = \mathbf{S}_{t-1} \bullet \mathbf{M}_t + \boldsymbol{v}_t \boldsymbol{k}_t^\top, \qquad \text{(recurrence)}$$
$$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t, \qquad \text{(memory read-out)}$$

where $\bullet$ is an associative operator (e.g., Hadamard product, matrix multiplication, etc.). The matrix $\mathbf{M}_t$ and vectors $\boldsymbol{v}_t$, $\boldsymbol{k}_t$, $\boldsymbol{q}_t$ are (potentially non-linear) functions of the current input $\boldsymbol{x}_t$.

As is the case in vector-valued linear RNNs [62, 103], the use of an associative operator enables the use of parallel scan [13] to calculate $\mathbf{S}_1, \ldots, \mathbf{S}_L$ in $O(\log L)$ steps and $O(L)$ work (ignoring the terms associated with the associative operation) if the inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L$ are given (though see our discussion in footnote 1). Hence, as long as the associative operator is not too expensive, training can be efficient. However, parallel scan by itself is not sufficient for training language models at practical scale due to some associative operator's being too expensive. Recent models such as such

Model	Recurrence	Memory read-out
Linear Attention [47, 46]	$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
+ Kernel	$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{v}_t \phi(\boldsymbol{k}_t)^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \phi(\boldsymbol{q}_t)$
+ Normalization	$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{v}_t \phi(\boldsymbol{k}_t)^\top, \;\; \boldsymbol{z}_t = \boldsymbol{z}_{t-1} + \phi(\boldsymbol{k}_t)$	$\boldsymbol{o}_t = \mathbf{S}_t \phi(\boldsymbol{q}_t) / (\boldsymbol{z}_t^\top \phi(\boldsymbol{q}_t))$
DeltaNet [98]	$\mathbf{S}_t = \mathbf{S}_{t-1}(\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top) + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
Gated RFA [79]	$\mathbf{S}_t = g_t \mathbf{S}_{t-1} + (1-g_t) \boldsymbol{v}_t \boldsymbol{k}_t^\top, \;\; \boldsymbol{z}_t = g_t \boldsymbol{z}_{t-1} + (1-g_t) \boldsymbol{k}_t$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t / (\boldsymbol{z}_t^\top \boldsymbol{q}_t)$
S4 [31, 103]	$\mathbf{S}_t = \mathbf{S}_{t-1} \odot \exp(-(\boldsymbol{\alpha} \mathbf{1}^\top) \odot \exp(\boldsymbol{A})) + \boldsymbol{B} \odot (\boldsymbol{v}_t \mathbf{1}^\top)$	$\boldsymbol{o}_t = (\mathbf{S}_t \odot \boldsymbol{C}) \mathbf{1} + \boldsymbol{d} \odot \boldsymbol{v}_t$
ABC [80]	$\mathbf{S}_t^{\boldsymbol{k}} = \mathbf{S}_{t-1}^{\boldsymbol{k}} + \boldsymbol{k}_t \boldsymbol{\phi}_t^\top, \;\; \mathbf{S}_t^{\boldsymbol{v}} = \mathbf{S}_{t-1}^{\boldsymbol{v}} + \boldsymbol{v}_t \boldsymbol{\phi}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t^{\boldsymbol{v}} \operatorname{softmax}\left(\mathbf{S}_t^{\boldsymbol{k}} \boldsymbol{q}_t\right)$
DFW [61]	$\mathbf{S}_t = \mathbf{S}_{t-1} \odot (\boldsymbol{\beta}_t \boldsymbol{\alpha}_t^\top) + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
RetNet [105]	$\mathbf{S}_t = \gamma \mathbf{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
Mamba [30]	$\mathbf{S}_t = \mathbf{S}_{t-1} \odot \exp(-(\boldsymbol{\alpha}_t \mathbf{1}^\top) \odot \exp(\boldsymbol{A})) + (\boldsymbol{\alpha}_t \odot \boldsymbol{v}_t) \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t + \boldsymbol{d} \odot \boldsymbol{v}_t$
GLA [116]	$\mathbf{S}_t = \mathbf{S}_{t-1} \odot (\mathbf{1} \boldsymbol{\alpha}_t^\top) + \boldsymbol{v}_t \boldsymbol{k}_t^\top = \mathbf{S}_{t-1} \mathrm{Diag}(\boldsymbol{\alpha}_t) + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
RWKV-6 [77]	$\mathbf{S}_t = \mathbf{S}_{t-1} \mathrm{Diag}(\boldsymbol{\alpha}_t) + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = (\mathbf{S}_{t-1} + (\boldsymbol{d} \odot \boldsymbol{v}_t) \boldsymbol{k}_t^\top) \boldsymbol{q}_t$
HGRN-2 [89]	$\mathbf{S}_t = \mathbf{S}_{t-1} \mathrm{Diag}(\boldsymbol{\alpha}_t) + \boldsymbol{v}_t (\mathbf{1} - \boldsymbol{\alpha}_t)^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
mLSTM [9]	$\mathbf{S}_t = f_t \mathbf{S}_{t-1} + i_t \boldsymbol{v}_t \boldsymbol{k}_t^\top, \;\; \boldsymbol{z}_t = f_t \boldsymbol{z}_{t-1} + i_t \boldsymbol{k}_t$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t / \max\{1, \lvert \boldsymbol{z}_t^\top \boldsymbol{q}_t \rvert\}$
Mamba-2 [18]	$\mathbf{S}_t = \gamma_t \mathbf{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
GSA [122]	$\mathbf{S}_t^{\boldsymbol{k}} = \mathbf{S}_{t-1}^{\boldsymbol{k}} \mathrm{Diag}(\boldsymbol{\alpha}_t) + \boldsymbol{k}_t \boldsymbol{\phi}_t^\top, \;\; \mathbf{S}_t^{\boldsymbol{v}} = \mathbf{S}_{t-1}^{\boldsymbol{v}} \mathrm{Diag}(\boldsymbol{\alpha}_t) + \boldsymbol{v}_t \boldsymbol{\phi}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t^{\boldsymbol{v}} \operatorname{softmax}\left(\mathbf{S}_t^{\boldsymbol{k}} \boldsymbol{q}_t\right)$

Table 4: Overview of recent linear recurrent models that have been proposed and applied to autoregressive language modeling (ordered in rough chronological order). These works make use of a matrix-valued hidden state $\mathbf{S}_t \in \mathbb{R}^{d \times n}$ (or two matrix-valued hidden states $\mathbf{S}_t^{\boldsymbol{k}}, \mathbf{S}_t^{\boldsymbol{v}}$, e.g., [80, 122]) updated through an associative recurrence followed by an outer-product-based addition. Here $\odot$ is the Hadamard product. Some models make use of an additional linear RNN with hidden state vector $\boldsymbol{z}_t$, which used to normalized the query vector $\boldsymbol{q}_t$. Variables with the subscript t (e.g., $\boldsymbol{v}_t, \boldsymbol{\alpha}_t, f_t, \gamma_t$) are (potentially non-linear) functions of the current input $\boldsymbol{x}_t$. Non-time-varying parameters (e.g., $\boldsymbol{A}, \boldsymbol{d}, \gamma$) are denoted without subscripts; these parameters are either learned or set to fixed values. Matrices are denoted with bold upper case letters, vectors with bold lower case, and scalars with italic letters. Many models make use of a kernel ϕ (e.g., [98, 79]) but we subsume them into the key/value vectors to reduce notational clutter.

as Mamba [30] and gated linear attention Transformers [105, 116, 89, 77, 9] thus make use of cheap element-wise recurrence updates, in particular the Hadamard product, i.e., $\bullet = \odot$. See Table 4 for how recent models can be cast into this form.

Standard matrix multiplications (i.e., $\mathbf{S}_{t-1} \bullet \mathbf{M}_t = \mathbf{S}_{t-1} \mathbf{M}_t$) on the other hand can model richer interactions that go beyond elementwise recurrence. Without any structural assumptions on $\mathbf{M}_t$ however, these operations would take $O(dn^2)$ for each update (as opposed to $O(dn)$ for elementwise products), which would be prohibitively expensive. Hence, DeltaNet's use of $\mathbf{M}_t = \mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top$ can be seen as exploiting structured matrices to efficiently model interactions beyond elementwise recurrences. Our chunkwise algorithm could generalize to a broader class of matrices in the Diagonal-Plus-Low-Rank (DPLR) form $\mathbf{M}_t = \mathbf{D} - \boldsymbol{a}_t \boldsymbol{b}_t^\top$, which has been explored in S4 [31], although their DPLR transition matrices are data-independent. We adopt DeltaNet's parameterization in this work (i.e., $\mathbf{D} = \mathbf{I}, \boldsymbol{a}_t = \beta_t \boldsymbol{k}_t, \boldsymbol{b}_t = \boldsymbol{k}_t$) as we are primarily interested in improving recall (through DeltaNet's key-value update rule) while maintaining parameter efficiency. We leave the exploration of more generalized parameterizations for future work.

5.2 Towards a Unifying Framework for Efficient Autoregressive Sequence Transformations

While the above class of models makes it possible to unify recent models, we do not claim that it is the "right" level at which view (autoregressive) sequence transformations of the form $\{\boldsymbol{x}_t\}_{t=1}^L \mapsto \{\boldsymbol{o}_t\}_{t=1}^L$, where $\boldsymbol{o}_t$ cannot depend on any $\boldsymbol{x}_j$ if $j > t$. For example, this framing makes it difficult to (neatly) capture other subquadratic models that have been shown to be effective [117, 48, 95, 81]. An alternative unifying framework might be to view the above sequence transformations as a discretization of a continuous state space model [31, 103, 30], or as a matrix multiplication with a masked structured matrix [73, 85, 45, 18]. What does seem important, however, is that a framework should ideally expose efficient algorithms for training, and the algorithm should be hardware-efficient, which, in the case of modern GPUs, means that it should be rich in matrix multiplications. From this perspective, the state-space duality (SSD) framework recently proposed by Dao and Gu [18], which provides a connection between SSM-based sequence transformations and structured matrix multiplications with a semiseparable matrix, seems a promising candidate. However, this framework may not capture an important class of models, e.g., models where the associative recurrence involves matrix multiplication with an unstructured matrix, or models that make use of more exotic associative operators (e.g., in Peng et al. [78]).

Finally, we observe that there have been many recent linear-time models that have been proposed which purportedly match or outperform classic transformers. As can be seen in Table 4, the "sequence mixing" component of these works are closely related to one another. However, the way in which the token-mixing primitive is used to build up a transformer-like model varies widely. For example, while most recent works make use of depthwise-separable convolution layers (not shown in Table 4) [30, 76, 90, 9, 18], earlier works generally do not [47, 99, 79]. There are also differences in the parameterizations of the feedforward layers used for the "channel mixing" component. Such variations should be taken into account before declaring a particular model layer superior to another.

5.3 Limitations and Future Work

Our work has several limitations. First, in terms of computation, although we propose a new hardware-efficient algorithm, the training speed still lags behind that of GLA. This is due to the overhead caused by modeling state-to-state dependencies as described above, which requires "marginalizing" over the head dimension inside the kernel, similar to the case of softmax attention. However, for GLA since there are no intra-state dependencies (everything is elementwise), and thus it is easy to use tiling to support arbitrary size of head dimension, as implemented in Yang and Zhang [115]. This limitation would potentially limit DeltaNet's memory size, consequently lowering the recall-intensive task performance as we observed in §4.2. However, it may be feasible to adopt block diagonal generalized Householder transition matrices with block sizes fitting GPU SRAM (e.g., 128) while maintaining a overall large head dimension (and thus a large recurrent state size).

We also found that the length generalization of DeltaNet was limited,[4] while GLA and RetNet (and Mamba to an extent) have been found to be able to extrapolate beyond the training length [116]. We speculate that this is because DeltaNet lacks explicit decay factors. This could be improved through incorporating a gating term in the recurrence, which we leave for future work.

6 Related Work

We briefly discuss related work here and give an extended discussion in Appendix C.

Linear transformers can be seen as a type of iterated Hopfield networks [69], and this connection can provide perspectives on the limitations and improvements of linear attention transformers. For example, vanilla linear transformers use a Hebbian-like update rule, which has been shown to have limited memory capacity [65]. Later works in Hopfield networks use higher-order polynomials [21] and exponential kernels [92, 49] to enhance the memory capacity, which is also related to attention with polynomial kernels explored in PolysketchFormer [44] and Based Linear Attention [6, 1]. On the other hand, the delta rule has been shown to have better memory capacity [27, 83, 54, 97]. In this sense, given the fixed size recurrent state, using the delta rule is able to achieve a better frontier of the recall-memory tradeoff curve [6], and has recently been applied to enhance real-world retrieval tasks [71, 94]. Moreover, it outperforms the additive rule used in vanilla linear transformers across multiple domains [98, 36, 39, 35, 41].

Despite these advantages, Irie et al. [41] revealed theoretical limitations of the delta update rule in terms of expressiveness. Recurrent enhancements of DeltaNet, such as Recurrent DeltaNet [37] and the Modern Self-Referential Weight Matrix [40], were proposed and found to be superior in Irie et al. [41]. However, these models extend beyond linear RNNs and cannot be parallelized across sequence length. This suggests a fundamental trade-off between parallelism and expressiveness [67]. How to further enhance DeltaNet without sacrificing parallelism remains an open question, and the hybrid cross-chunk nonlinear and intra-chunk linear strategy used in TTT [107] might provide a suitable middle ground. Finally, we remark that delta rule is closely related to meta or online learning via gradient descent [70, 38], which has been revisited in recent works like Longhorn [56] and TTT [107].

7 Conclusion

We describe an algorithm that parallelizes DeltaNet training across the sequence length dimension, achieving significant speed-ups against existing implementations on modern hardware. This makes it possible to scale up DeltaNet to moderate-scale language modeling settings, where we find that it performs well compared to recent linear-recurrent baselines.

[4] However we found the DeltaNet + local sliding-window attention hybrid to generalize well, which could provide an appealing middle ground.

Acknowledgements

This study was supported by funding from the MIT-IBM Watson AI Lab. We are grateful to Mayank Mishra for assistance with training and evaluating the 3B models, to Kazuki Irie for valuable feedback on the draft, and to Simran Arora, Liliang Ren and Eric Alcaide for their insightful discussions. We also thank Michael Poli and Armin Thomas for sharing the raw results from the MAD benchmark experiment.

References

[1] Y. Aksenov, N. Balagansky, S. M. L. C. Vaina, B. Shaposhnikov, A. Gorbatovski, and D. Gavrilov. Linear Transformers with Learnable Kernel Functions are Better In-Context Models, June 2024. URL http://arxiv.org/abs/2402.10644. arXiv:2402.10644 [cs].

[2] E. Akyürek, B. Wang, Y. Kim, and J. Andreas. In-context language learning: Arhitectures and algorithms. *arXiv preprint arXiv:2401.12973*, 2024.

[3] A. Ali, I. Zimerman, and L. Wolf. The hidden attention of mamba models, 2024.

[4] S. Arora, S. Eyuboglu, A. Timalsina, I. Johnson, M. Poli, J. Zou, A. Rudra, and C. Ré. Zoology: Measuring and improving recall in efficient language models. *CoRR*, abs/2312.04927, 2023.

[5] S. Arora, B. Yang, S. Eyuboglu, A. Narayan, A. Hojel, I. Trummer, and C. Ré. Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes, Apr. 2023. arXiv:2304.09433 [cs].

[6] S. Arora, S. Eyuboglu, M. Zhang, A. Timalsina, S. Alberti, D. Zinsley, J. Zou, A. Rudra, and C. Ré. Simple linear attention language models balance the recall-throughput tradeoff. *CoRR*, abs/2402.18668, 2024. arXiv: 2402.18668.

[7] S. Arora, A. Timalsina, A. Singhal, B. Spector, S. Eyuboglu, X. Zhao, A. Rao, A. Rudra, and C. Ré. Just read twice: closing the recall gap for recurrent language models, 2024. URL https://arxiv.org/abs/2407.05483.

[8] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[9] M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter. xlstm: Extended long short-term memory. *arXiv preprint arXiv:2405.04517*, 2024.

[10] R. v. d. Berg, L. Hasenclever, J. M. Tomczak, and M. Welling. Sylvester Normalizing Flows for Variational Inference, Feb. 2019. URL http://arxiv.org/abs/1803.05649. arXiv:1803.05649 [cs, stat].

[11] C. H. Bischof and C. V. 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.

[12] Y. Bisk, R. Zellers, J. Gao, Y. Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

[13] G. E. Blelloch. Prefix sums and their applications. 1990.

[14] W. Brandon, A. Nrusimha, K. Qian, Z. Ankner, T. Jin, Z. Song, and J. Ragan-Kelley. Striped Attention: Faster Ring Attention for Causal Transformers. *ArXiv*, abs/2311.09431, 2023.

[15] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[16] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs), Feb. 2016. URL http://arxiv.org/abs/1511.07289. arXiv:1511.07289 [cs].

[17] T. Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. *CoRR*, abs/2307.08691, 2023. doi: 10.48550/ARXIV.2307.08691. arXiv: 2307.08691.

[18] T. Dao and A. Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv: 2405.21060*, 2024.

[19] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *NeurIPS*, 2022.

[20] S. De, S. L. Smith, A. Fernando, A. Botev, G. Cristian-Muraru, A. Gu, R. Haroun, L. Berrada, Y. Chen, S. Srinivasan, G. Desjardins, A. Doucet, D. Budden, Y. W. Teh, R. Pascanu, N. De Freitas, and C. Gulcehre. Griffin: Mixing Gated Linear Recurrences with Local Attention for Efficient Language Models, Feb. 2024. URL http://arxiv.org/abs/2402.19427. arXiv:2402.19427 [cs].

[21] M. Demircigil, J. Heusel, M. Löwe, S. Upgang, and F. Vermet. On a model of associative memory with huge storage capacity. *Journal of Statistical Physics*, 168(2):288–299, July 2017. ISSN 0022-4715, 1572-9613. doi: 10.1007/s10955-017-1806-y. URL http://arxiv.org/abs/1702.01929. arXiv:1702.01929 [math].

[22] A. E. T. Dominguez and E. S. Q. Orti. Fast blocking of householder reflectors on graphics processors. *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 385–393, 2018. URL https://api.semanticscholar.org/CorpusID:46960439.

[23] S. Elfwing, E. Uchibe, and K. Doya. Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning, Nov. 2017. URL http://arxiv.org/abs/1702.03118. arXiv:1702.03118 [cs].

[24] M. Fathi, J. Pilault, P.-L. Bacon, C. Pal, O. Firat, and R. Goroshin. Block-state transformer. *arXiv preprint arXiv:2306.09539*, 2023.

[25] D. Y. Fu, T. Dao, K. K. Saab, A. W. Thomas, A. Rudra, and C. Ré. Hungry Hungry Hippos: Towards Language Modeling with State Space Models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.

[26] L. Gao, J. Tow, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, K. McDonell, N. Muennighoff, J. Phang, L. Reynolds, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou. A framework for few-shot language model evaluation, Sept. 2021.

[27] E. Gardner. The space of interactions in neural network models. *Journal of Physics A*, 21: 257–270, 1988.

[28] A. Graves, G. Wayne, and I. Danihelka. Neural Turing Machines, Dec. 2014. URL http://arxiv.org/abs/1410.5401. arXiv:1410.5401 [cs].

[29] R. Griffin and G. Teams. Recurrentgemma: Moving past transformers for efficient open language models. *ArXiv*, abs/2404.07839, 2024.

[30] A. Gu and T. Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. In *Proceedings of COLM*, 2023.

[31] A. Gu, K. Goel, and C. 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.

[32] K. Helfrich, D. Willmott, and Q. Ye. Orthogonal recurrent neural networks with scaled cayley transform. In *International Conference on Machine Learning*, pages 1969–1978. PMLR, 2018.

[33] W. Hua, Z. Dai, H. Liu, and Q. V. Le. Transformer Quality in Linear Time. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 9099–9117. PMLR, 2022.

[34] F. Huang, K. Lu, C. Yuxi, Z. Qin, Y. Fang, G. Tian, and G. Li. Encoding recurrence into transformers. In *The Eleventh International Conference on Learning Representations*, 2022.

[35] K. Irie and J. 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/pdf?id=ddad0PNUvV.

[36] K. Irie, I. Schlag, R. Csord'as, and J. Schmidhuber. Going beyond linear transformers with recurrent fast weight programmers. *ArXiv*, abs/2106.06295, 2021. URL https://api.semanticscholar.org/CorpusID:235417174.

[37] K. Irie, I. Schlag, R. Csordás, and J. Schmidhuber. Going beyond linear transformers with recurrent fast weight programmers. *Advances in Neural Information Processing Systems*, 34: 7703–7717, 2021.

[38] K. Irie, R. Csordás, and J. Schmidhuber. The dual form of neural networks revisited: Connecting test time predictions to training patterns via spotlights of attention. In *Proc. Int. Conf. on Machine Learning (ICML)*, Baltimore, MD, USA, July 2022.

[39] K. Irie, F. Faccio, and J. Schmidhuber. Neural differential equations for learning to program neural nets through continuous learning rules. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/fc09b26b85ab3abb2832bd555a2e4215-Abstract-Conference.html.

[40] K. Irie, I. Schlag, R. Csord'as, and J. Schmidhuber. A modern self-referential weight matrix that learns to modify itself. In *International Conference on Machine Learning*, 2022.

[41] K. Irie, R. Csordás, and J. Schmidhuber. Practical computational power of linear transformers and their recurrent and self-referential extensions. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9455–9465, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.588. URL https://aclanthology.org/2023.emnlp-main.588.

[42] L. Jing, C. Gulcehre, J. Peurifoy, Y. Shen, M. Tegmark, M. Soljacic, and Y. Bengio. Gated Orthogonal Recurrent Units: On Learning to Forget. *Neural Computation*, 31(4):765–783, Apr. 2019. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco_a_01174. URL https://direct.mit.edu/neco/article/31/4/765-783/8458.

[43] T. Joffrain, T. M. Low, E. S. Quintana-Ortí, R. A. van de Geijn, and F. G. V. Zee. Accumulating householder transformations, revisited. *ACM Trans. Math. Softw.*, 32:169–179, 2006. URL https://api.semanticscholar.org/CorpusID:15723171.

[44] P. Kacham, V. Mirrokni, and P. Zhong. Polysketchformer: Fast transformers via sketches for polynomial kernels. *arXiv preprint arXiv:2310.01655*, 2023.

[45] Y. Kang, G. Tran, and H. De Sterck. Fast multipole attention: A divide-and-conquer attention mechanism for long sequences. *arXiv preprint arXiv:2310.11960*, 2023.

[46] J. Kasai, H. Peng, Y. Zhang, D. Yogatama, G. Ilharco, N. Pappas, Y. Mao, W. Chen, and N. A. Smith. Finetuning Pretrained Transformers into RNNs. In M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 10630–10643. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.830.

[47] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[48] N. Kitaev, Ł. Kaiser, and A. Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[49] D. Krotov and J. Hopfield. Large Associative Memory Problem in Neurobiology and Machine Learning, Apr. 2021. URL http://arxiv.org/abs/2008.06996. arXiv:2008.06996 [cond-mat, q-bio, stat].

[50] T. Lei. When Attention Meets Fast Recurrence: Training Language Models with Reduced Compute. In M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7633–7648, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.602. URL https://aclanthology.org/2021.emnlp-main.602.

[51] T. Lei, R. Tian, J. Bastings, and A. P. Parikh. Simple recurrence improves masked language models. *arXiv preprint arXiv:2205.11588*, 2022.

[52] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, Apr. 2021. URL http://arxiv.org/abs/2005.11401. arXiv:2005.11401 [cs].

[53] O. Lieber, B. Lenz, H. Bata, G. Cohen, J. Osin, I. Dalmedigos, E. Safahi, S. Meirom, Y. Belinkov, S. Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

[54] K. C. Lingashetty. Delta learning rule for the active sites model. *arXiv preprint arXiv:1007.0417*, 2010.

[55] L. D. Lingle. Transformer-vq: Linear-time transformers via vector quantization. *arXiv preprint arXiv:2309.16354*, 2023.

[56] B. Liu, R. Wang, L. Wu, Y. Feng, P. Stone, and Q. Liu. Longhorn: State space models are amortized online learners. *ArXiv*, abs/2407.14207, 2024.

[57] H. Liu, M. Zaharia, and P. Abbeel. Ring Attention with Blockwise Transformers for Near-Infinite Context. *ArXiv*, abs/2310.01889, 2023.

[58] C. Lockard, P. Shiralkar, and X. L. Dong. OpenCeres: When Open Information Extraction Meets the Semi-Structured Web. In J. Burstein, C. Doran, and T. Solorio, editors, *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)*, pages 3047–3056, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1309. URL https://aclanthology.org/N19-1309.

[59] X. Ma, C. Zhou, X. Kong, J. He, L. Gui, G. Neubig, J. May, and L. Zettlemoyer. Mega: moving average equipped gated attention. *arXiv preprint arXiv:2209.10655*, 2022.

[60] X. Ma, X. Yang, W. Xiong, B. Chen, L. Yu, H. Zhang, J. May, L. Zettlemoyer, O. Levy, and C. Zhou. Megalodon: Efficient llm pretraining and inference with unlimited context length. *arXiv preprint arXiv:2404.08801*, 2024.

[61] H. H. Mao. Fine-Tuning Pre-trained Transformers into Decaying Fast Weights. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10236–10242, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.697.

[62] E. Martin and C. 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.

[63] S. Massaroli, M. Poli, D. Y. Fu, H. Kumbong, R. N. Parnichkun, A. Timalsina, D. W. Romero, Q. McIntyre, B. Chen, A. Rudra, C. Zhang, C. Ré, S. Ermon, and Y. Bengio. Laughing hyena distillery: Extracting compact recurrences from convolutions. *ArXiv*, abs/2310.18780, 2023. URL https://api.semanticscholar.org/CorpusID:264590326.

[64] A. Mathiasen, F. Hvilshøj, J. R. Jørgensen, A. Nasery, and D. Mottin. Faster orthogonal parameterization with householder matrices. In *ICML, Workshop Proceedings*, 2020.

[65] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh. The capacity of the hopfield associative memory. *IEEE Trans. Inf. Theory*, 33:461–482, 1987.

[66] J. Mercat, I. Vasiljevic, S. Keh, K. Arora, A. Dave, A. Gaidon, and T. Kollar. Linearizing large language models. *arXiv preprint arXiv:2405.06640*, 2024.

[67] W. Merrill, J. Petty, and A. Sabharwal. The Illusion of State in State-Space Models, Apr. 2024. URL http://arxiv.org/abs/2404.08819. arXiv:2404.08819 [cs].

[68] Z. Mhammedi, A. Hellicar, A. Rahman, and J. Bailey. Efficient Orthogonal Parametrisation of Recurrent Neural Networks Using Householder Reflections, June 2017. URL http://arxiv.org/abs/1612.00188. arXiv:1612.00188 [cs].

[69] B. Millidge. Linear Attention as Iterated Hopfield Networks. URL http://www.beren.io/2024-03-03-Linear-Attention-as-Iterated-Hopfield-Networks/.

[70] T. Munkhdalai, A. Sordoni, T. Wang, and A. Trischler. Metalearned Neural Memory. *ArXiv*, July 2019. URL https://www.semanticscholar.org/paper/a513bb6e1967f5a31ad4f38954e66d4169b613e5.

[71] T. Munkhdalai, M. Faruqui, and S. Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 2024.

[72] Y. Nahshan, J. Kampeas, and E. Haleva. Linear Log-Normal Attention with Unbiased Concentration, Feb. 2024. URL http://arxiv.org/abs/2311.13541. arXiv:2311.13541 [cs].

[73] T. Nguyen, V. Suliafu, S. Osher, L. Chen, and B. Wang. Fmmformer: Efficient and flexible transformer via decomposed near-field and far-field attention. *Advances in neural information processing systems*, 34:29449–29463, 2021.

[74] D. Paperno, G. Kruszewski, A. Lazaridou, Q. N. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context, June 2016. URL http://arxiv.org/abs/1606.06031. arXiv:1606.06031 [cs].

[75] J. Park, J. Park, Z. Xiong, N. Lee, J. Cho, S. Oymak, K. Lee, and D. Papailiopoulos. Can mamba learn how to learn? a comparative study on in-context learning tasks. *arXiv preprint arXiv:2402.04248*, 2024.

[76] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, H. Cao, X. Cheng, M. Chung, M. Grella, K. K. G. V, X. He, H. Hou, P. Kazienko, J. Kocon, J. Kong, B. Koptyra, H. Lau, K. S. I. Mantri, F. Mom, A. Saito, X. Tang, B. Wang, J. S. Wind, S. Wozniak, R. Zhang, Z. Zhang, Q. Zhao, P. Zhou, J. Zhu, and R.-J. Zhu. RWKV: Reinventing RNNs for the Transformer Era. *CoRR*, abs/2305.13048, 2023. doi: 10.48550/ARXIV.2305.13048. arXiv: 2305.13048.

[77] B. Peng, D. Goldstein, Q. Anthony, A. Albalak, E. Alcaide, S. Biderman, E. Cheah, X. Du, T. Ferdinan, H. Hou, P. Kazienko, K. K. GV, J. Kocoń, B. Koptyra, S. Krishna, R. McClelland Jr., N. Muennighoff, F. Obeid, A. Saito, G. Song, H. Tu, S. Woźniak, R. Zhang, B. Zhao, Q. Zhao, P. Zhou, J. Zhu, and R.-J. Zhu. Eagle and Finch: RWKV with Matrix-Valued States and Dynamic Recurrence, Apr. 2024. URL http://arxiv.org/abs/2404.05892. arXiv:2404.05892 [cs].

[78] H. Peng, R. Schwartz, S. Thomson, and N. A. Smith. Rational recurrences. *ArXiv*, abs/1808.09357, 2018.

[79] H. Peng, N. Pappas, D. Yogatama, R. Schwartz, N. A. Smith, and L. Kong. Random feature attention. *arXiv preprint arXiv:2103.02143*, 2021.

[80] H. Peng, J. Kasai, N. Pappas, D. Yogatama, Z. Wu, L. Kong, R. Schwartz, and N. A. Smith. ABC: Attention with Bounded-memory Control. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[81] M. Poli, S. Massaroli, E. Nguyen, D. Y. Fu, T. Dao, S. Baccus, Y. Bengio, S. Ermon, and C. Ré. Hyena Hierarchy: Towards Larger Convolutional Language Models. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 28043–28078. PMLR, 2023.

[82] M. Poli, A. W. Thomas, E. Nguyen, P. Ponnusamy, B. Deiseroth, K. Kersting, T. Suzuki, B. Hie, S. Ermon, C. Ré, C. Zhang, and S. Massaroli. Mechanistic Design and Scaling of Hybrid Architectures, Mar. 2024. arXiv:2403.17844 [cs].

[83] D. Prados and S. Kak. Neural network capacity using delta rule. *Electronics Letters*, 3(25): 197–199, 1989.

[84] Z. Qin, X. Han, W. Sun, D. Li, L. Kong, N. Barnes, and Y. Zhong. The devil in linear transformer. *arXiv preprint arXiv:2210.10340*, 2022.

[85] Z. Qin, X. Han, W. Sun, B. He, D. Li, D. Li, Y. Dai, L. Kong, and Y. Zhong. Toeplitz neural network for sequence modeling. *arXiv preprint arXiv:2305.04749*, 2023.

[86] Z. Qin, D. Li, W. Sun, W. Sun, X. Shen, X. Han, Y. Wei, B. Lv, F. Yuan, X. Luo, et al. Scaling transnormer to 175 billion parameters. *arXiv preprint arXiv:2307.14995*, 2023.

[87] Z. Qin, W. Sun, K. Lu, H. Deng, D. Li, X. Han, Y. Dai, L. Kong, and Y. Zhong. Linearized Relative Positional Encoding, July 2023. URL http://arxiv.org/abs/2307.09270. arXiv:2307.09270 [cs].

[88] Z. Qin, W. Sun, D. Li, X. Shen, W. Sun, and Y. Zhong. Lightning attention-2: A free lunch for handling unlimited sequence lengths in large language models. 2024.

[89] Z. Qin, S. Yang, W. Sun, X. Shen, D. Li, W. Sun, and Y. Zhong. HGRN2: Gated Linear RNNs with State Expansion. 2024. URL https://api.semanticscholar.org/CorpusID:269043328.

[90] Z. Qin, S. Yang, and Y. Zhong. Hierarchically gated recurrent neural network for sequence modeling. *Advances in Neural Information Processing Systems*, 36, 2024.

[91] P. Rajpurkar, R. Jia, and P. 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.

[92] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, T. Adler, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, V. Greiff, D. Kreil, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter. Hopfield Networks is All You Need, Apr. 2021. URL http://arxiv.org/abs/2008.02217. arXiv:2008.02217 [cs, stat].

[93] L. Ren, Y. Liu, Y. Lu, Y. Shen, C. Liang, and W. Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. *arXiv preprint arXiv:2406.07522*, 2024.

[94] I. Rodkin, Y. Kuratov, A. Bulatov, and M. Burtsev. Associative recurrent memory transformer. *ArXiv*, abs/2407.04841, 2024.

[95] A. Roy, M. Saffar, A. Vaswani, and D. Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9: 53–68, 2021.

[96] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[97] I. Schlag, K. Irie, and J. Schmidhuber. Linear transformers are secretly fast weight programmers. In M. Meila and T. Zhang, editors, *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*, pages 9355–9366. PMLR, 2021.

[98] I. Schlag, K. Irie, and J. Schmidhuber. Linear Transformers Are Secretly Fast Weight Programmers. In M. Meila and T. Zhang, editors, *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*, pages 9355–9366. PMLR, 2021.

[99] I. Schlag, T. Munkhdalai, and J. Schmidhuber. Learning Associative Inference Using Fast Weight Memory, Feb. 2021. URL http://arxiv.org/abs/2011.07831. arXiv:2011.07831 [cs].

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

[101] N. Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[102] Y. Shen, M. Stallone, M. Mishra, G. Zhang, S. Tan, A. Prasad, A. M. Soria, D. D. Cox, and R. Panda. Power scheduler: A batch size and token number agnostic learning rate scheduler. *ArXiv*, abs/2408.13359, 2024.

[103] J. T. H. Smith, A. Warrington, and S. 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.

[104] W. Sun, Z. Qin, D. Li, X. Shen, Y. Qiao, and Y. Zhong. Linear attention sequence parallelism. *arXiv preprint arXiv:2404.02882*, 2024.

[105] Y. Sun, L. Dong, S. Huang, S. Ma, Y. Xia, J. Xue, J. Wang, and F. Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

[106] Y. Sun, L. Dong, Y. Zhu, S. Huang, W. Wang, S. Ma, Q. Zhang, J. Wang, and F. Wei. You only cache once: Decoder-decoder architectures for language models. *arXiv preprint arXiv:2405.05254*, 2024.

[107] Y. Sun, X. Li, K. Dalal, J. Xu, A. Vikram, G. Zhang, Y. Dubois, X. Chen, X. Wang, O. Koyejo, T. Hashimoto, and C. Guestrin. Learning to (learn at test time): Rnns with expressive hidden states. *ArXiv*, abs/2407.04620, 2024. URL https://api.semanticscholar.org/CorpusID:271039606.

[108] L. Team. The llama 3 herd of models. *ArXiv*, abs/2407.21783, 2024.

[109] P. Tillet, H. Kung, and D. D. 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, MAPL@PLDI 2019*, pages 10–19. ACM, 2019. doi: 10.1145/3315508.3329973.

[110] J. M. Tomczak and M. Welling. Improving Variational Auto-Encoders using Householder Flow, Jan. 2017. arXiv:1611.09630 [cs, stat].

[111] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[112] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[113] E. Vorontsov, C. Trabelsi, S. Kadoury, and C. Pal. On orthogonality and learning recurrent networks with long term dependencies. In *International Conference on Machine Learning*, pages 3570–3578. PMLR, 2017.

[114] B. Widrow, M. E. Hoff, et al. Adaptive switching circuits. In *IRE WESCON convention record*, volume 4, pages 96–104. New York, 1960.

[115] S. Yang and Y. Zhang. FLA: A Triton-Based Library for Hardware-Efficient Implementations of Linear Attention Mechanism, Jan. 2024. URL https://github.com/sustcsonglin/flash-linear-attention. original-date: 2023-12-20T06:50:18Z.

[116] S. Yang, B. Wang, Y. Shen, R. Panda, and Y. Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

[117] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

[118] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[119] J. Zhang, Q. Lei, and I. S. Dhillon. Stabilizing Gradients for Deep Neural Networks via Efficient SVD Parameterization, Mar. 2018. arXiv:1803.09327 [cs, stat].

[120] J. Zhang, S. Jiang, J. Feng, L. Zheng, and L. Kong. Linear Attention via Orthogonal Memory, 2023. arXiv:2312.11135.

[121] Q. Zhang, D. Ram, C. Hawkins, S. Zha, and T. Zhao. Efficient long-range transformers: You need to attend more, but not necessarily at every layer. In H. Bouamor, J. Pino, and K. Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

[122] Y. Zhang, S. Yang, R. Zhu, Y. Zhang, L. Cui, Y. Wang, B. Wang, F. Shi, B. Wang, W. Bi, P. Zhou, and G. Fu. Gated slot attention for efficient linear-time sequence modeling. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[123] I. Zimerman, A. Ali, and L. Wolf. A unified implicit attention formulation for gated-linear recurrent sequence models. *CoRR*, abs/2405.16504, 2024.

A Derivation of WY representation

To reduce notational clutter, we only discuss the first chunk here.

We first show $\mathbf{P}_n = \mathbf{I} - \sum_{t=1}^{n} \boldsymbol{w}_t \boldsymbol{k}_t^\intercal$ by induction,

$$
\begin{aligned}
\mathbf{P}_n &= \prod_{t=1}^{n} (\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\intercal) \\
&= \mathbf{P}_{n-1}(\mathbf{I} - \beta_n \boldsymbol{k}_n \boldsymbol{k}_n^\intercal) \\
&= (\mathbf{I} - \sum_{t=1}^{n-1} \boldsymbol{w}_t \boldsymbol{k}_t^\intercal)(\mathbf{I} - \beta_n \boldsymbol{k}_n \boldsymbol{k}_n^\intercal) \\
&= \mathbf{I} - \sum_{t=1}^{n-1} \boldsymbol{w}_t \boldsymbol{k}_t^\intercal - \beta_n \boldsymbol{k}_n \boldsymbol{k}_n^\intercal + (\sum_{t=1}^{n-1} \boldsymbol{w}_t \boldsymbol{k}_t^\intercal) \beta_n \boldsymbol{k}_n \boldsymbol{k}_n^\intercal \\
&= \mathbf{I} - \sum_{t=1}^{n-1} \boldsymbol{w}_t \boldsymbol{k}_t^\intercal - \underbrace{\left(\beta_n \boldsymbol{k}_n - \beta_n \sum_{t=1}^{n-1} \left(\boldsymbol{w}_t (\boldsymbol{k}_t^\intercal \boldsymbol{k}_n) \right) \right)}_{\boldsymbol{w}_n} \boldsymbol{k}_n^\intercal \\
&= \mathbf{I} - \sum_{t=1}^{n} \boldsymbol{w}_t \boldsymbol{k}_t^\intercal
\end{aligned}
$$

Similarly, we show $\mathbf{S}_n = \sum_{t=1}^{n} \boldsymbol{u}_t \boldsymbol{k}_n^\intercal$ by induction,

$$
\begin{aligned}
\mathbf{S}_n &= \mathbf{S}_{n-1}(\mathbf{I} - \beta_n \boldsymbol{k}_n \boldsymbol{k}_n^\intercal) + \beta_n \boldsymbol{v}_n \boldsymbol{k}_n^\intercal \\
&= \left(\sum_{t=1}^{n-1} \boldsymbol{u}_t \boldsymbol{k}_t^\intercal \right) (\mathbf{I} - \beta_n \boldsymbol{k}_n \boldsymbol{k}_n^\intercal) + \beta_n \boldsymbol{v}_n \boldsymbol{k}_n^\intercal \\
&= \sum_{t=1}^{n-1} \boldsymbol{u}_t \boldsymbol{k}_t^\intercal - \left(\sum_{t=1}^{n-1} \boldsymbol{u}_t \boldsymbol{k}_t^\intercal \right) \beta_n \boldsymbol{k}_n \boldsymbol{k}_n^\intercal + \beta_n \boldsymbol{v}_n \boldsymbol{k}_n^\intercal \\
&= \sum_{t=1}^{n-1} \boldsymbol{u}_t \boldsymbol{k}_t^\intercal + \underbrace{\left(\beta_n \boldsymbol{v}_n - \beta_n \sum_{t=1}^{n-1} \boldsymbol{u}_t \left(\boldsymbol{k}_t^\intercal \boldsymbol{k}_n \right) \right)}_{\boldsymbol{u}_n} \boldsymbol{k}_n^\intercal \\
&= \sum_{t=1}^{n} \boldsymbol{u}_t \boldsymbol{k}_n^\intercal
\end{aligned}
$$

B Pseudo code

```
def chunk_delta_rule_forward(Q, K, V, beta, C):
    '''
    Q/K/V: query, key, value of shape [L, d]
    beta: beta of shape [L]
    C: chunk size
    '''
    # L: sequence length, d: head dimension
    L, d = Q.shape

    # chunking
    Q, K, V = map(lambda x:  x.reshape(-1,C,d), [Q, K, V])
    beta = beta.reshape(-1, C)
    K_beta = K * beta.unsqueeze(-1)
    V_beta = V * beta.unsqueeze(-1)

    # compute Eq. 10
    mask = torch.triu(torch.ones(C, C), diagonal=0).bool()
    T = -(K_beta @ K.t()).masked_fill_(mask, 0)
    # vectorized forword substitution.
    for i in range(1, C):
        T[i, :i] = T[i, :i] + (T[i, :, None] * T[:, :i]).sum(-2)
    T += torch.eye(C)
    # compute Eq. 11
    W = T @ K_beta
    U = T @ V_beta
    # chunkwise parallel. Eq. 8-9
    S = torch.zeros(d, d)
    O = torch.empty_like(V)
    mask = torch.triu(torch.ones(C, C), diagonal=1).bool()
    for i in range(L//C):
        q_i, k_i, w_i = Q[i], K[i], W[i]
        u_i = U[i] - w_i @ S
        o_inter = q_i @ S
        A_i = (q_i @ k_i.t()).masked_fill_(mask, 0)
        o_intra = A_i @ u_i
        S += k_i.t() @ u_i
        O[i] = o_intra + o_inter
    return O.reshape(L, d)
```

Listing 1: Pytorch-like code snippet of the forward pass of our chunkwise algorithm for training DeltaNet. We omit the dimensions of batch size and number of heads for clarity.

C Related Work Continued

Chunkwise linear attention. Hua et al. [33] first proposed chunkwise form for linear attention; however, they used a hybrid linear and nonlinear attention model similar to Munkhdalai et al. [71]. It is possible to adapt their algorithm to compute the *exact* output of the pure linear attention, as shown in Sun et al. [105] and Yang et al. [116]. The chunkwise linear attention algorithm has also been independently discovered in several works [105, 44, 18]. Yang et al. [116] and Qin et al. [88] discuss I/O-aware hardware optimization for chunkwise linear attention and Sun et al. [104] make generalization to multi-node distributed training. Inspired by the chunkwise form, we propose a new algorithm for hardware-efficient DeltaNet training, significantly improving the training efficiency and allowing for large-scale experiments.

Hybrid models. There has been much recent work on developing hybrid models by combining linear recurrent layers (state-space models, linear recurrent Transformers, linear RNNs) with local chunk attention [59, 121, 24, 60, 71] or sliding window attention [121, 6, 20, 93] or global attention [50, 51, 34, 25, 53, 75, 106]. Poli et al. [82] systematically study the scaling law of hybrid models. We similarly show that combining DeltaNet with classic attention is an effective strategy.

Householder matrices. Householder matrices, known for preserving norms, are a type of orthogonal matrix extensively used in machine learning [64, 68, 119, 110, 87, 10]. These matrices allow for efficient computation of inverses and their Jacobian determinant of one, making them particularly suitable for applications in normalizing flows [64, 10]. Notably, Mathiasen et al. [64] developed a chunkwise fast algorithm for computing the cumulative product of Householder matrices for normalizing flows, leveraging the WY representation. Our approach, while sharing the same high-level concept, tackles a different problem and is arguably more general.

There has also been significant interest in using orthogonal matrices to parameterize the transition matrices of RNNs [68, 42, 113, 32] for mitigating vanishing gradients. Mhammedi et al. [68] use the WY representation to reduce the memory footprint when training nonlinear RNNs with Householder transition matrices.

D Hyperparameters

We used 8 H100 GPUs for 340M and 1.3B language modeling experiments. Each model uses AdamW for optimization, with a peak learning rate of 3×10^{-4}. The 340M models are trained using 15 billion tokens and a batch size of 0.5M tokens, while the 1.3B models are trained with 100 billion tokens and a batch size of 2M tokens. We use a cosine learning rate schedule, starting with a warm-up phase of 0.5 billion tokens for the 340M models and 1 billion tokens for the 1.3B models. Both configurations have initial and final learning rates set at 3×10^{-5}. We apply a weight decay of 0.01 and use gradient clipping at a maximum of 1.0. The head dimension of DeltaNet is set to 128, and the kernel size for convolution layers is set at 4.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: This paper's contributions and scope are reflected in abstract and introduction part clearly.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of this work in §5.3.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This paper does not include theoretical results that require a full proof.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides sufficient details on hyperparameters and training procedures in §D to reproduce the results supporting its main conclusions.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Our code is publicly available at https://github.com/sustcsonglin/flash-linear-attention. Our primary training corpus is Slimpajama, an open-source dataset.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have detailed all the training and evaluation settings before the main results in the experimental part.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We do not have enough resources to obtain error bars as running the experiments multiple times is computationally expensive due to the large model size.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide information of GPU type and number of GPUs used for running our experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work follows the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: We foresee no potential societal impact of this work.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We foresee no such risks posed by this work.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All of the datasets we use are publicly available at huggingface site, and we have properly cited all the training and evaluation datasets we used.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This work does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.