MEMORY CACHING: RNNS WITH GROWING MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers have been established as the de-facto backbones for most recent advances in sequence modeling, mainly due to their growing memory capacity that scales with the context length. While plausible for retrieval tasks, it causes quadratic complexity and so has motivated recent studies to explore viable sub-quadratic recurrent alternatives. Despite showing promising preliminary results in diverse tasks, such recurrent architectures underperform Transformers in recall-intensive tasks, often attributed to their fixed-size memory. In this paper, we introduce Memory Caching (MC), a simple yet effective technique that enhances recurrent models by caching checkpoints of their memory states (a.k.a. hidden states). MC allows the effective memory capacity of RNNs to grow with sequence length, offering a flexible trade-off that interpolates between the fixed memory ($\mathcal{O}(L)$ complexity) of RNNs and the growing memory ($\mathcal{O}(L^2)$ complexity) of Transformers. We propose four variants of MC, including gated aggregation and sparse selective mechanisms, and discuss their implications on both linear and deep memory modules. Our experimental results on language modeling, and long-context understanding tasks show that MC enhances the performance of recurrent models, supporting its effectiveness. In in-context recall tasks, our results indicate that while Transformers still achieve the best performance, our MC variants show competitive performance, close the gap with Transformers, and performs better than state-of-the-art recurrent models.

1 INTRODUCTION

Transformers (Vaswani et al., 2017b) are foundational to recent advances across various domains (Jumper et al., 2021; Dosovitskiy et al., 2021; Comanici et al., 2025). Their success stems largely from the attention mechanism, which acts as an associative memory with growing capacity (Ramsauer et al., 2021; Bietti et al., 2024; Behrouz et al., 2025b). While effective for many retrieval tasks (Arora et al., 2024b; Behrouz et al., 2025a; Guo et al., 2025), this growing memory incurs quadratic complexity and high inference-time memory usage (KV-caching). This has motivated the development of sub-quadratic architectures that aim to improve efficiency while maintaining performance (Dai et al., 2019; Child et al., 2019; Poli et al., 2023).

In particular, recurrent neural networks that aim to compress the past data into their memory state, maintaining a fixed size over the entire input sequence, have regained attention in recent years (Katharopoulos et al., 2020; Irie et al., 2021; Sun et al., 2023; Behrouz et al., 2024). Despite showing promising preliminary results in diverse short-context language modeling and other sequence modeling tasks (Irie et al., 2022; Dalal et al., 2025), the fixed-memory state of such recurrent architectures is the bottleneck to unleash their actual power. The foundation of these architectures is based on recurrence and data compression, which, with careful design, can result in highly efficient and expressive learning algorithms (Merrill et al., 2024; Huang et al., 2024). However, their fixed capacity to compress a growing sequence forces them to forget past information, which is a critical bottleneck, specifically in recall-intensive and long-context tasks (Arora et al., 2024b; Kuratov et al., 2024; Sun et al., 2024).

Contributions. We introduce Memory Caching (MC), a general technique that allows the effective memory of recurrent models to grow with sequence length by caching checkpoints of the memory states. MC provides a flexible middle ground interpolating between standard recurrence and attention, offering a controllable complexity of $\mathcal{O}(NL)$. This allows for flexible interpolation between the $\mathcal{O}(L)$ complexity of RNNs and the $\mathcal{O}(L^2)$ complexity of Transformers. Our contributions are threefold:

- **The MC Framework:** We propose segmenting the sequence and caching the compressed memory state of each segment, allowing the model to directly access compressed information from the entire history.
- **Novel Aggregation Strategies:** We introduce four methods to utilize these cached memories: (i, ii) (Gated) Residual Memory, which uses residual connections and a novel context-aware gating mechanism; (iii) Memory Soup, inspired by weight souping, which averages the parameters of cached memory modules (distinct for non-linear memories); and (iv) Sparse Selective Caching (SSC), which uses a Mixture-of-Experts style router to select only the most contextually relevant cached memories for efficient aggregation.
- **Empirical Validation:** We demonstrate the effectiveness of MC on various architectures including the deep memory module Titans (Behrouz et al., 2024) and Deep Linear Attention (DLA) (Behrouz et al., 2025a)), across language modeling, long-context, and retrieval tasks, showing that MC enhances performance and extends the effective context length of RNNs.

2 Preliminaries and Background

In this section, we review necessary background and establish notations.

Notations. We use bold lowercase (resp. uppercase) letters for vectors (resp. matrices) and use subscript t to refer to the state of the entities correspond to time t. Throughout, we let $x \in \mathbb{R}^{L \times d_{\text{in}}}$ be the input, $\mathcal{M}_t$ be the state of memory $\mathcal{M}(\cdot)$ at time t, $\mathbf{K}$ be the keys, $\mathbf{V}$ be the values, $\mathbf{Q}$ be the query matrices, and L denote the sequence length. We focus on MLP-based architectures for memory with $\mathcal{L}_{\mathcal{M}} \geq 1$ layers. Notably, this formulation includes linear matrix-valued memory modules when $\mathcal{L}_{\mathcal{M}} = 1$. When it is needed, we parameterize the memory module $\mathcal{M}(\cdot)$ with $\boldsymbol{\theta}_{\mathcal{M}} := \{W_1, \dots, W_{\mathcal{L}_{\mathcal{M}}}, \dots\}$, which at least includes the parameters of linear layers in the MLP.

Attention. Attention Vaswani et al. (2017a) is the primary building block of Transformers that acts as their associative memory (Bietti et al., 2023; Sun et al., 2024; Behrouz et al., 2025b). Given input $x \in \mathbb{R}^{L \times d_{\text{in}}}$, causal attention computes output $\mathbf{y} \in \mathbb{R}^{L \times d_{\text{in}}}$ over input dependent key, value, and query matrices $\mathbf{Q} = x\mathbf{W}_{\mathbf{Q}}, \mathbf{K} = x\mathbf{W}_{\mathbf{K}}$, and $\mathbf{V} = x\mathbf{W}_{\mathbf{V}}$ as:

$$\mathbf{y}_i = \sum_{t=1}^{i} \frac{\exp\left(\mathbf{q}_i^\top \mathbf{k}_t\right) \mathbf{v}_t}{\sum_{\ell=1}^{i} \exp\left(\mathbf{q}_i^\top \mathbf{k}_\ell\right)} = \frac{1}{Z_i} \sum_{t=1}^{i} \exp\left(\mathbf{q}_i^\top \mathbf{k}_t\right) \mathbf{v}_t, \tag{1}$$

where $\mathbf{W}_{\mathbf{Q}}, \mathbf{W}_{\mathbf{K}}$, and $\mathbf{W}_{\mathbf{V}} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$ are learnable parameters, and $Z_i = \sum_{\ell=1}^{i} \exp\left(\mathbf{q}_i^\top \mathbf{k}_\ell\right)$ is the normalization term. Attention requires $O(L^2)$ operations due to the need to access all past tokens.

Linear Attention. Linear attention (Katharopoulos et al., 2020) and its variants (Sun et al., 2023; Peng et al., 2023; Schlag et al., 2021) improves efficiency of attention by replacing the $\exp(\cdot)$ operator in Equation 1 with a separable kernel $\phi(\cdot)$, resulting in an efficient recurrent formulation:

$$\mathbf{y}_i = \sum_{t=1}^{i} \frac{\phi\left(\mathbf{q}_i\right)^\top \phi\left(\mathbf{k}_t\right) \mathbf{v}_t}{\sum_{\ell=1}^{i} \phi\left(\mathbf{q}_i\right)^\top \phi\left(\mathbf{k}_\ell\right)} = \frac{1}{Z_i} \mathcal{M}_t \phi\left(\mathbf{q}_i\right) , \tag{2}$$

where $\mathcal{M}_t = \mathcal{M}_{t-1} + \mathbf{v}_t \phi\left(\mathbf{k}_t\right)^\top$ acts as the fixed-size memory (Katharopoulos et al., 2020).

Test-time Memorization Perspective. . A recent unifying framework interprets the forward pass of sequence models—including both attention and modern RNNs—as a dynamic learning process occurring at inference time Sun et al. (2024); Behrouz et al. (2025b); Wang et al. (2025). In this view, the model acts as an associative memory that actively learns the mapping between input tokens (keys and values). This memorization is achieved by optimizing an internal objective, often formalized as a regression problem Wang et al. (2025) or "attentional bias" Behrouz et al. (2025b). This perspective frames the memory state as a dynamic entity optimized during the forward pass. We leverage this view by introducing Memory Caching, where cached states serve as checkpoints of this optimization process, enhancing the model's ability to retrieve information across long sequences.

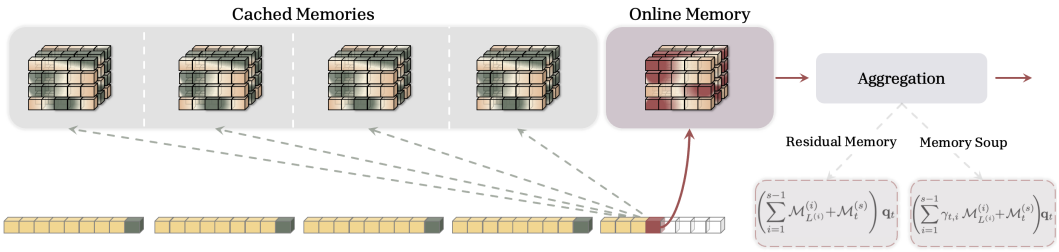

Figure 1: The Overall Memory Caching Method. Each token attends to its online memory as well as a set of cached memories from the past.

3 Recurrent Neural Networks with Memory Caching

RNNs maintain a fixed-size memory to compress the input sequence. As sequences grow long, this leads to memory overflow and performance degradation. Conversely, attention caches all past tokens, resulting in a growing memory but quadratic cost. We propose Memory Caching (MC) to cache intermediate memory states, providing a middle ground where the model's memory can grow with arbitrary scale. This allows computational costs to interpolate between $O(L)$ (RNNs) and $O(L^2)$ (Transformers). To this end, given a sequence of tokens $x \in \mathbb{R}^{L \times d_{\text{in}}}$, we split the sequence into segments $S^{(1)}, \dots, S^{(N)}$ with size $L^{(1)}, \dots, L^{(N)}$ and use memories $\mathcal{M}^{(1)}, \dots, \mathcal{M}^{(N)}$ to compress the segments. The memory update rule or the recurrence for memory corresponds to s-th segment is:

$$\boldsymbol{k}_t = x_t W_{\boldsymbol{k}}, \qquad \boldsymbol{v}_t = x_t W_{\boldsymbol{v}}, \qquad \boldsymbol{q}_t = x_t W_{\boldsymbol{q}}, \tag{3}$$

$$\mathcal{M}_t^{(s)} = f\left(\mathcal{M}_{t-1}^{(s)}; \boldsymbol{k}_t, \boldsymbol{v}_t\right), \qquad \text{where} \quad 1 \le t \le L^{(s)}, \tag{4}$$

where $f(\cdot)$ is the learning update rule (e.g., $f\left(\mathcal{M}_{t-1}^{(s)}; \boldsymbol{k}_t, \boldsymbol{v}_t\right) = \mathcal{M}_{t-1}^{(s)} + \boldsymbol{v}_t \boldsymbol{k}_t^\top$ for linear attention (Katharopoulos et al., 2020)). Using the above formulation, we update the memories in each segment and cache the last state of previous segments (i.e., $\{\mathcal{M}_{L^{(s)}}^{(s)}\}_{s=1}^{T}$ where T is the index of the current segment). Standard RNNs compute the output using only the current memory state: $\mathbf{y}_t = \mathcal{M}_t(\boldsymbol{q}_t)$. In contrast, our formulation uses all past cached memories alongside the current memory (online memory) to compute the output for query $\boldsymbol{q}_t$. Given an arbitrary aggregation function, $\texttt{Agg}(\cdot;\cdot;\cdot)$, the output is:

$$\mathbf{y}_t = \texttt{Agg}\left(\{\mathcal{M}_{L^{(1)}}^{(1)}(\cdot), \dots, \mathcal{M}_{L^{(s-1)}}^{(s-1)}(\cdot)\}; \mathcal{M}_t^{(s)}(\cdot); \mathbf{q}_t\right), \tag{5}$$

where s is the indices of the current segment. Note that for $1 \le i \le s$, the term $\mathcal{M}_{L^{(i)}}^{(i)}(\mathbf{q}_t)$ provides us with the corresponding information to query $\boldsymbol{q}_t$ in segment i. In the following sections, we present different effective choices of $\texttt{Agg}(\cdot;\cdot;\cdot)$ function to incorporate the past information into the computation of the current output and increasing the effective memory capacity of the model.

3.1 Residual Memory

We begin with the simplest $\texttt{Agg}(\cdot;\cdot;\cdot)$ operator: a summation, acting as a residual connection across memory states. In this case, given keys, values, and queries (see Equation 3) and segments $S^{(1)}, \dots, S^{(N)}$, we define the memory update and output computation at time t in segment s as:

$$\mathcal{M}_t^{(s)} = f\left(\mathcal{M}_{t-1}^{(s)}; \boldsymbol{k}_t, \boldsymbol{v}_t\right), \qquad \text{where} \quad 1 \le t \le L^{(s)}, \tag{6}$$

$$\mathbf{y}_t = \underbrace{\mathcal{M}_t^{(s)}(\mathbf{q}_t)}_{\text{Online Memory}} + \underbrace{\sum_{i=1}^{s-1} \mathcal{M}_{L^{(i)}}^{(i)}(\mathbf{q}_t)}_{\text{Cached Memories}}. \tag{7}$$

The critical change in memory caching is how the output is computed. In fact, for retrieval of the memory, the model uses forward passes over both the current memory (called online memory) and the cached memories for input query $\boldsymbol{q}_t$.

Gated Residual Memory (GRM). When the memory module is strictly linear (i.e., $\mathcal{M}$ is a matrix), the Residual Memory formulation (equation 7) mathematically collapses into a standard fixed-size memory, as the cached memories can be pre-summed (c.f. equation 12 below). However, in practice, our experimental results show that even this simple formulation can enhance the power of recurrent models (see Section 4). A further limitation of the residual approach is that it treats all cached memories equally, ignoring their relevance to the query q_t. To enable selective retrieval, we introduce input-dependent gating. Given input x_t in segment s, we define parameters $0 \leq \gamma_t^{(1)}, \dots, \gamma_t^{(s)} \leq 1$ be input-dependent parameters and reformulate the output as:

$$\mathcal{M}_t^{(s)} = f\left(\mathcal{M}_{t-1}^{(s)}; \boldsymbol{k}_t, \boldsymbol{v}_t\right), \text{ for } 1 \leq t \leq L^{(s)}, \quad \mathbf{y}_t = \gamma_t^{(s)} \mathcal{M}_t^{(s)}(\mathbf{q}_t) \; + \; \sum_{i=1}^{s-1} \gamma_t^{(i)} \mathcal{M}_{L^{(i)}}^{(i)}(\mathbf{q}_t) \quad (8)$$

Here, parameters $\gamma_t^{(i)}$ modulate the contribution of each segment to the output. When $\gamma_t^{(i)} \to 1$ (resp. $\gamma_t^{(i)} \to 0$), i-th segment has more (resp. less) contribution to the output. Due to these input dependent parameters, the above formulation cannot be pre-computed before this token and also cannot be reused for next tokens/segments. Therefore, contrary to the previous variant, it does not collapse into the fixed-size memory case (even in the linear memory case) and so requires to be recomputed for every token and needs caching memory states. A simple choice of parametrization for $\gamma_t^{(i)}$s is to define them as linear projection of input x_t (similar to projections for keys, values, and queries). With this parametrization, however, $\gamma_t^{(i)}$ acts as a position-based filtering/focus, meaning that the context of x_t only determines how much the i-th segment's memory (based on the position) contributes, no matter what its context is. To overcome this issue, we suggests making $\gamma_t^{(i)}$ as a function of both x_t and i-th segment $S^{(i)}$, incorporating both of their contexts and how similar they are. To this end, we introduce a connector parameter $\boldsymbol{u}_t$ as the linear projection of input, and define $\gamma_t^{(i)}$ as the similarity of $\boldsymbol{u}_t$ and i-th segment $S^{(i)}$:

$$\gamma_t^{(i)} = \langle \boldsymbol{u}_t, \texttt{MeanPooling}(S^{(i)}) \rangle \qquad \text{where } \boldsymbol{u}_t = x_t W_{\boldsymbol{u}} \; . \quad (9)$$

Here, $\texttt{MeanPooling}(\cdot)$ provides a simple representation of segment's context as the mean of all tokens. It, however, can be replaced by any other pooling process. As an alternative parameterization, we can use $\boldsymbol{u}_t = \boldsymbol{q}_t$. When $\gamma_t^{(i)} = 1$, then GRM is equivalent to residual memory variant.

Example. To better illustrate the above formulations and as an illustrative example, we let $f\left(\mathcal{M}_{t-1}^{(s)}; \boldsymbol{k}_t, \boldsymbol{v}_t\right) = \mathcal{M}_{t-1}^{(s)} - \nabla \langle \mathcal{M}_{t-1}^{(s)}(\boldsymbol{k}_t), \boldsymbol{v}_t \rangle$, where memory $\mathcal{M}(\cdot)$ is an arbitrary feedforward layer (e.g., MLP or gated MLP layers). This general form is equivalent to Deep Linear Attention (DLA) (Behrouz et al., 2025a) and when the memory is a matrix (i.e., MLP with one layer) it is equivalent to the linear attention (Katharopoulos et al., 2020). Using residual memory caching on DLA results in a model with update and retrieval rules of:

$$\mathcal{M}_t^{(s)} = \mathcal{M}_{t-1}^{(s)} - \nabla \langle \mathcal{M}_{t-1}^{(s)}(\boldsymbol{k}_t), \boldsymbol{v}_t \rangle, \qquad \mathbf{y}_t = \mathcal{M}_t^{(s)}(\mathbf{q}_t) \; + \; \sum_{i=1}^{s-1} \mathcal{M}_{L^{(i)}}^{(i)}(\mathbf{q}_t) \; . \quad (10)$$

When using a linear matrix-valued memory (i.e., linear attention), Equation 10 can be simplified to:

$$\mathcal{M}_t^{(s)} = \mathcal{M}_{t-1}^{(s)} + \boldsymbol{v}_t \boldsymbol{k}_t^\top, \quad (11)$$

$$\mathbf{y}_t = \mathcal{M}_t^{(s)} \mathbf{q}_t \; + \; \sum_{i=1}^{s-1} \mathcal{M}_{L^{(i)}}^{(i)} \mathbf{q}_t = \left(\mathcal{M}_t^{(s)} \; + \; \sum_{i=1}^{s-1} \mathcal{M}_{L^{(i)}}^{(i)} \right) \mathbf{q}_t \; . \quad (12)$$

Memory Complexity. In the retrieval process, we use the current memory (online memory) and the cached memories of all previous segments and so given a fixed training sequence length, the number of cached memories is a function of segment lengths. While the memory update process has not changed and so requires $\mathcal{O}(L)$ operation, the retrieval process requires forward pass over all cached memory and so needs $\mathcal{O}(N)$ operations per token. This brings the complexity of the model to $\mathcal{O}(NL)$, where $1 \leq N \leq L$. Note that when $N = 1$ (only one segment), we do not need to cache any memory state, resulting in a simple recurrent memory model. When $N = L$, it means that each token is treated as a separate segment and so the memory state for all past tokens are cached. This closely matches the intuition behind the power of attention. In fact, attention by caching all past tokens, provide a direct access to each part of the sequence, enhancing the retrieval ability.

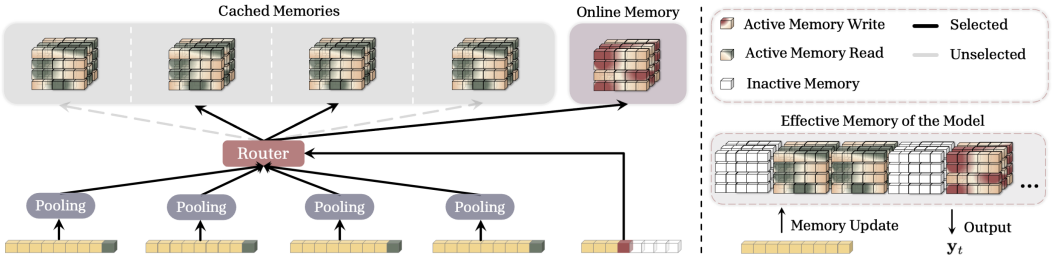

Figure 2: Sparse Selective Caching (SSC) of Memories. A router measures the contextual similarity of each token to its past segments and chooses a subset of past cached memory for better efficiency.

3.2 Memory Soup

Viewing the recurrence as a meta-learning process where memory states are checkpoints, we introduce the Memory Soup variant, inspired by Wortsman et al. (2022). The core idea is to combine the memory states (parameters) into a single data-dependent memory for retrieval. Similar to the previous variant, we use $\mathcal{M}^{(i)}_{L^{(i)}}$ to refer to the cached memory corresponds to i-th segment and parameterize it with $\boldsymbol{\theta}_{\mathcal{M}^{(i)}_{L^{(i)}}} := \{W^{(i)}_1, \dots, W^{(i)}_c\}$. Note that the architecture of memory is unchanged and so c (the number of parameters) is the same for all memory states. Accordingly, the memory update and retrieval process for memory caching is defined as:

$$\mathcal{M}^{(s)}_t = f\left(\mathcal{M}^{(s)}_{t-1}; \boldsymbol{k}_t, \boldsymbol{v}_t\right), \text{ for } 1 \leq t \leq L^{(s)}, \quad \mathbf{y}_t = \mathcal{M}^{*}_t(\boldsymbol{q}_t), \tag{13}$$

where $\mathcal{M}^{*}_t$ is parametrized as: $\boldsymbol{\theta}_{\mathcal{M}^{*}_t} := \{\sum_{i=1}^{s} \gamma^{(i)}_t W^{(i)}_1, \dots, \sum_{i=1}^{s} \gamma^{(i)}_t W^{(i)}_c\}$. Therefore, each token has its own memory for retrieval that also depends on the input-data and can change. In fact, one can interpret the above process as *a memory system that each token, builds its own memory to retrieve corresponding information from*. Note that here $\gamma^{(i)}_t$ parameters are defined with the same process as Equation 9.

When the memory module $\mathcal{M}$ is linear, Memory Soup is mathematically equivalent to GRM (Equation 8). This is because souping the weights and then applying the query is identical to applying the query to individual memories and then ensembling the outputs, due to the linearity of the operation. The distinction becomes crucial when using deep or non-linear memory modules (e.g., DLA or Titans). In these cases, the equivalence breaks down. Memory Soup constructs a new, input-dependent memory module $\mathcal{M}^{*}_t$ by interpolating the parameters themselves, effectively creating a specialized non-linear retrieval function tailored for that specific timestep.

3.3 Sparse Selective Caching (SSC) of Memories

The previous variants attend to all past cached memories, which can cause significant memory overhead for ultra-long sequences. We introduce Sparse Selective Caching (SSC), where each token contextually *selects* a subset of cached memories, improving efficiency. To this end, inspired by Mixture of Experts (MoEs) (Shazeer et al., 2017), we use a router that based on the token and its similarity to the context of each segment choose a subset of cached memories. For each segment $S^{(i)}$, following Equation 9, we let $\texttt{MeanPooling}(S^{(i)}) = \sum_{j \in S^{(i)}} \boldsymbol{k}_j$, and define the relevance score of each segment $S^{(i)}$ to query x_t as:

$$\mathbf{r}^{(i)}_t = \langle \boldsymbol{u}_t, \texttt{MeanPooling}(S^{(i)}) \rangle, \qquad \text{where } \boldsymbol{u}_t = x_t W_{\boldsymbol{u}}. \tag{14}$$

Given relevance scores, the router chooses k of the cached memories with highest relevance, i.e., $\mathcal{R}_t = \arg \texttt{Top-k}(\{\mathbf{r}^{(i)}_t\}^{s-1}_{i=1})$, as well as the current online memory for retrieval. Given selected memories, the retrieval process is the same as previous variants but using only selected memories:

$$\begin{aligned} &\mathcal{M}^{(s)}_t = f\left(\mathcal{M}^{(s)}_{t-1}; \boldsymbol{k}_t, \boldsymbol{v}_t\right), \qquad \text{where} \quad 1 \leq t \leq L^{(s)}, \\ &\mathbf{y}_t = \gamma^{(s)}_t \mathcal{M}^{(s)}_t(\mathbf{q}_t) \;+\; \sum_{i \in \mathcal{R}_t} \gamma^{(i)}_t \mathcal{M}^{(i)}_{L^{(i)}}(\mathbf{q}_t). \end{aligned} \tag{15}$$

Table 1: Performance of models on language modeling and common-sense reasoning tasks.

Model	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	SIQA acc ↑	BoolQ acc ↑	Avg. ↑
760M params / 30B tokens											
Transformer++	24.18	24.27	36.3	67.2	41.8	52.0	65.6	33.4	39.1	61.7	49.64
RetNet	25.77	24.19	34.5	66.8	41.2	51.9	63.6	32.5	38.8	56.2	48.19
DeltaNet	24.52	24.38	36.8	67.3	44.5	51.8	64.2	32.7	39.6	60.1	49.63
Miras (Memora)	22.28	22.31	38.2	67.8	49.3	53.3	63.6	36.1	40.9	63.0	51.53
Samba*	21.07	22.85	39.2	68.9	47.8	53.1	65.8	34.9	38.9	63.1	51.46
DLA	23.12	22.09	36.1	68.0	47.9	52.7	65.8	34.6	39.1	59.6	50.48
+ Log-Linear	23.08	21.15	36.8	68.1	47.7	53.0	65.6	35.1	39.2	59.3	50.60
+ GRM	22.91	20.10	37.5	69.2	48.7	52.8	66.1	36.8	40.3	59.9	51.41
+ Memory Soup	22.78	20.49	37.2	69.6	48.3	53.4	65.8	36.5	39.6	60.2	51.33
+ SSC	23.14	20.86	37.0	68.4	47.7	52.7	66.0	35.2	39.7	60.1	50.85
Titans (LMM)	20.04	21.96	37.4	69.3	48.5	52.3	66.3	35.8	40.1	62.8	51.56
+ Log-Linear	19.79	20.62	37.8	70.1	48.0	52.5	66.8	35.6	40.3	62.8	51.74
+ GRM	19.14	20.21	38.3	70.6	48.4	54.0	67.5	36.4	41.7	63.5	52.55
+ Memory Soup	19.52	20.38	38.0	71.4	48.6	53.7	67.1	35.4	41.3	63.1	52.33
+ SSC	19.39	20.46	37.7	70.9	48.7	53.5	66.9	36.3	41.2	63.1	52.29
1.3B params / 100B tokens											
Transformer++	17.92	17.73	42.6	71.4	51.3	54.1	69.9	36.0	41.8	58.4	53.19
RetNet	18.91	17.04	41.2	71.3	49.1	55.2	67.5	34.1	41.4	61.0	52.60
DeltaNet	18.62	17.10	41.6	70.1	49.4	52.7	67.6	35.2	39.7	54.8	51.39
Miras (Memora)	15.90	12.04	48.7	73.1	56.0	57.4	71.5	37.9	40.2	61.3	55.76
Samba*	16.15	13.21	45.2	71.5	53.8	55.8	69.1	36.7	40.6	63.0	54.46
DLA	16.31	12.29	44.5	70.6	53.9	54.2	69.6	36.0	40.8	60.2	53.72
+ Log-Linear	16.22	12.25	44.9	71.1	54.5	54.8	70.0	36.6	41.3	60.7	54.24
+ GRM	16.08	12.10	45.8	72.5	55.9	55.8	71.5	41.2	42.8	62.2	55.96
+ Memory Soup	16.16	12.17	45.6	71.9	55.4	55.6	70.9	37.7	42.0	61.5	55.08
+ SSC	16.20	12.19	45.3	71.7	54.8	55.3	70.4	37.1	41.4	61.1	54.64
Titans (LMM)	15.60	11.41	49.1	73.1	56.3	59.8	72.4	40.8	42.1	61.0	56.82
+ Log-Linear	15.49	11.38	49.4	73.6	56.5	60.3	72.8	41.1	42.5	61.3	57.19
+ GRM	15.37	11.29	50.4	74.5	57.4	61.5	73.8	42.6	43.9	62.5	58.33
+ Memory Soup	15.42	11.31	49.9	74.2	57.3	60.8	73.5	42.2	43.4	62.0	57.91
+ SSC	15.44	11.35	49.6	73.8	57.0	60.6	73.1	41.9	42.8	61.8	57.58

In this formulation, $\texttt{MeanPooling}(S^{(i)})$ of each segment can be pre-computed and so computing the relevance score as well as choosing $\texttt{Top-k}$ segments for each token are simply parallelizable. Also, such computations do not require to store the state of the cached memories in the accelerators (i.e., GPUs, TPUs, etc.). Therefore, this process only requires loading the *"selected"* memories for each token and so can enhance memory consumption during both training and inference.

Effective Memory. One interesting interpretation of SSC is to see it as a sparse unified memory module. We illustrate this in Figure 2 (Right). One can see SSC as a model with growing memory, where for each token activates a subset of parameters for memory write operation (storing the token), and a larger subset of parameters for retrieval. This formulation allows the memory to (1) store information without any interfering with past memories, and (2) efficiently and adaptively retrieve information. The segment size, here, determines the size of the blocks in the unified memory that become active together.

4 Experiments

Next, we evaluate the effectiveness of memory caching in improving the performance of models on language modeling, commonsense reasoning, needle in haystack, and in-context recall tasks.

Experimental Setup. We train our models with training context window of size {1K, 2K, 4K, 8K, 16K, 32K} and segment lengths ranging from {16, 32, 64, 128, 256, 512} tokens using FineWeb dataset (Penedo et al., 2024). Unless stated otherwise, the default context length is 4K with 256 segment length. We use model size of 760M, and 1.3B parameters and train them on 30B and 100B tokens sampled from the dataset. Perplexity is measured on held-out validation data. As for the downstream tasks, we evaluate trained models on Wikitext (Merity et al., 2017), LMB (Paperno et al., 2016), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), Wino-Grande (Sakaguchi et al., 2021), ARC-easy (ARC-e) and ARC-challenge (ARC-c) (Clark et al.,

Table 2: NIAH experiments, including three single-needle tasks—S-NIAH-1 (passkey retrieval), S-NIAH-2 (numerical needle), and S-NIAH-3 (UUID-based needle).

	S-NIAH-1 (pass-key retrieval)			**S-NIAH-2** (number in haystack)			**S-NIAH-3** (uuid in haystack)		
Model	4K	8K	16K	4K	8K	16K	4K	8K	16K
Transformer	88.6	76.4	79.8	100	98.8	94.2	78.0	69.2	40.8
DLA	96.4	71.2	44.0	79.6	42.6	28.2	18.2	8.8	4.0
+ Log-Linear	100	96.2	70.4	87.6	70.4	18.0	28.8	20.4	6.0
+ GRM	100	100	82.4	94.6	82.8	54.8	48.2	34.4	18.2
+ Memory Soup	100	100	78.2	91.8	77.2	40.4	43.0	32.8	14.8
+ SSC	100	98.2	76.8	89.2	74.8	37.6	34.0	28.6	11.2
Titans (LMM)	100	100	100	99.6	84.6	75.4	74.2	42.8	21.2
+ Log-Linear	100	100	100	95.6	88.4	74.8	76.0	48.4	24.2
+ GRM	100	100	100	99.8	96.6	88.2	89.4	69.0	32.2
+ Memory Soup	100	100	100	98.8	92.2	83.0	84.2	61.8	28.6
+ SSC	100	100	100	98.6	90.4	79.6	81.0	54.2	27.0

2018), SIQA (Sap et al., 2019), and BoolQ (Clark et al., 2019). Additional details about the experimental setups and other used datasets are in Appendix B.

4.1 Language Modeling

We start with common and academic-scale language modeling. The results of DLA, and Titans with and without memory caching are reported in Table 1. There are four observations: (1) Comparing DLA, Titans, and LA with their enhanced version with memory caching, we observe that all memory caching variants provides consistent improvements on different downstream tasks, and also on average over their baseline. This shows the importance of memory caching to further enhance memory bounded models. (2) As discussed earlier, memory caching can be seen as a hybrid of sparse attention with recurrent model. Comparing memory caching enhanced models and two of the state-of-the-art hybrid models, memory caching provides a more powerful solution to the problem of limited memory in recurrent models. Particularly, Titans + MC and DLA + MC achieves +0.8% performance gain over the Titans. (3) MC enhanced variants show better performance compared to simple recurrent models. We also compare against the Log-Linear attention approach Guo et al. (2025), which utilizes a hybrid caching strategy. (4) Comparing among MC variants and Log-Linear method, we observed that our three variants provide better results. Furthermore, GRM and then SSC achieves better results among our provided methods. We attribute this performance gain to larger effective memory size that MC provides for the model.

4.2 Needle-In-A-Haystack Tasks

We evaluate the impact of MC on long-context retrieval using Needle-in-a-Haystack (NIAH) tasks (Table 2). MC-enhanced DLA and Titans consistently outperform the base models. Furthermore, MC variants outperform the Log-Linear approach, especially at longer contexts. Log-Linear struggles because it forces a single memory to compress very large initial segments (e.g., 8K tokens in a 16K sequence), whereas MC distributes the compression load more effectively.

4.3 In-context Retrieval Tasks

In-context recall tasks are among the most challenging benchmarks for recurrent neural networks. In this section, we follow Arora et al. (2024b) and perform experiments on SWDE (Lockard et al., 2019), NQ (Kwiatkowski et al., 2019), DROP (Dua et al., 2019), FDA (Arora et al., 2023), SQUAD (Rajpurkar et al., 2016), and TQA (Kembhavi et al., 2017) to evaluate and compare the performance of MC-enhanced variants with baselines and Transformers. The results are reported in Table 3. While Transformers still achieve the best results in in-context recall tasks, our MC variants show competitive performance, close the gap with Transformers, and performs better than state-of-the-art recurrent models. We again attribute this performance to larger memory capacity that scales with sequence length.

Table 3: Accuracy on retrieval tasks w/ input truncated to different lengths.

	SWDE				SQuAD				FDA			
Model	512	1024	2048	16k	512	1024	2048	16k	512	1024	2048	16k
Transformer	46.2	43.7	44.4	44.0	33.1	33.3	33.6	33.4	71.0	69.5	71.6	71.0
Titans (MAL)	51.9	48.6	48.3	48.5	28.3	29.2	29.1	28.8	71.1	73.9	72.1	71.7
DLA	44.5	39.9	32.7	32.5	23.8	24.0	23.8	24.1	55.6	40.2	25.9	23.3
+ Log-Linear	43.7	37.7	30.4	30.6	27.8	27.8	27.9	28.3	55.1	39.6	22.3	18.9
+ GRM	52.4	48.9	48.7	48.5	29.5	30.7	30.7	30.1	63.3	51.6	48.9	41.5
+ Memory Soup	49.5	45.0	38.0	37.7	28.4	28.6	28.5	29.1	60.5	48.4	37.2	34.6
+ SSC	47.0	42.5	35.5	35.3	26.0	28.1	27.1	28.8	58.0	46.0	28.8	29.4
Titans (LMM)	43.2	34.4	29.2	29.7	25.7	26.2	26.3	25.6	59.3	45.5	35.4	32.5
+ Log-Linear	48.0	41.4	37.2	37.0	27.2	27.3	27.2	27.1	67.0	55.5	41.2	32.4
+ GRM	52.6	49.3	49.5	50.1	29.7	30.4	31.5	32.0	72.9	68.7	61.1	52.6
+ Memory Soup	50.3	46.7	44.8	45.4	29.2	29.7	29.8	30.3	70.3	63.8	55.7	45.8
+ SSC	48.6	44.2	41.0	41.4	28.3	28.8	28.5	28.8	68.2	59.4	47.6	38.9

	TriviaQA				Drop				NQ			Avg.
Model	512	1024	2048	16k	512	1024	2048	16k	512	1024	2048	
Transformer	47.5	48.5	47.4	47.6	21.8	22.0	21.5	21.4	23.6	23.1	23.7	41.00
Titans (MAL)	44.8	45.1	44.6	44.8	20.6	20.5	20.8	20.9	22.1	22.4	22.5	40.46
DLA	43.3	44.2	43.5	43.2	20.1	19.9	20.6	20.0	19.7	18.4	18.5	30.51
+ Log-Linear	43.7	44.8	43.6	43.8	20.3	20.2	20.8	20.2	19.9	18.8	21.0	30.75
+ GRM	50.1	47.3	44.8	50.0	21.9	21.8	22.0	21.7	23.5	23.3	23.4	38.03
+ Memory Soup	48.0	46.4	44.2	48.7	21.5	21.3	21.7	21.2	22.8	22.4	22.5	35.05
+ SSC	45.8	45.5	43.9	46.1	20.9	20.7	21.2	20.6	21.4	20.6	21.8	33.09
Titans (LMM)	44.2	44.7	43.9	44.5	20.2	20.1	20.3	20.6	20.1	19.5	19.1	31.75
+ Log-Linear	44.5	44.9	44.1	44.7	20.4	20.4	20.5	20.7	21.5	19.8	20.4	34.37
+ GRM	50.2	47.5	45.3	50.9	21.7	21.8	21.9	21.5	23.7	23.4	23.3	40.50
+ Memory Soup	48.3	46.6	44.8	49.4	21.3	21.4	21.7	21.1	22.9	22.2	22.5	38.43
+ SSC	46.1	45.7	44.3	46.9	20.8	20.7	21.2	20.9	21.9	20.4	21.5	36.27

Table 4: Accuracy on LongBench tasks (Bai et al., 2024): NarrativeQA, QasperQA, MultiFieldQA, HotpotQA, 2WikiMultiQA, Musique, GovReport, QMSum, MultiNews, TREC, TriviaQA, SamSum, LCC, and RepoBench-P.

	Single-Doc QA			Multi-Doc QA			Summarization			Few-shot			Code	
Model	NQA	QQA	MFQ	HQA	2WM	Mus	GvR	QMS	MNs	TRC	TQA	SSM	LCC	RBP
Transformer	11.5	9.6	19.1	21.5	28.9	6.5	13.0	9.2	3.1	27.2	27.9	15.1	22.9	29.1
DLA	9.4	17.5	12.1	11.8	22.3	4.8	9.5	7.4	5.1	4.8	23.5	9.7	38.4	34.9
+ Log-Linear	10.1	10.2	17.1	12.4	23.3	5.5	6.6	12.7	5.8	18.6	24.7	16.2	31.6	31.0
+ GRM	11.6	10.3	19.8	18.2	26.9	6.4	13.5	14.1	6.9	25.7	28.2	18.3	32.7	33.9
+ Memory Soup	11.2	10.3	19.5	16.7	25.1	6.3	11.2	13.8	6.2	22.5	26.9	17.7	32.3	33.5
+ SSC	10.7	10.2	18.8	14.2	24.8	5.9	8.4	12.9	6.1	20.5	25.7	16.8	31.9	32.6
Titans (LMM)	8.7	12.5	18.4	15.6	26.1	6.7	10.5	12.6	11.8	37.1	26.2	24.5	31.3	31.4
+ Log-Linear	9.6	8.9	19.3	18.7	26.9	6.8	6.7	12.9	2.8	11.2	42.7	25.0	29.5	29.7
+ GRM	11.8	9.4	19.9	21.4	29.1	7.2	8.4	13.3	3.1	14.8	49.7	25.5	31.0	32.8
+ Memory Soup	10.7	9.2	19.6	20.2	28.2	7.1	7.8	13.1	3.0	13.7	47.1	25.3	30.8	31.4
+ SSC	9.9	9.1	19.4	19.8	27.5	6.9	7.1	13.0	2.8	12.5	44.8	25.2	29.9	30.8

4.4 Long Context Understanding Tasks

We evaluate long-context understanding on LongBench (Bai et al., 2024) (Table 4). All MC-enhanced variants provide performance gains compared to their base RNNs, again attributed to their increased memory capacity.

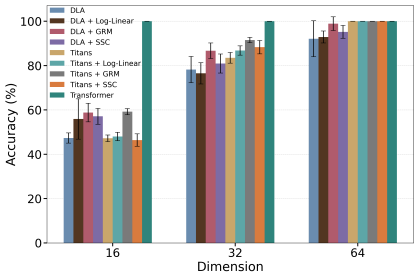

Figure 3: Average accuracies (mean ± std) on MQAR over 5 seeds.

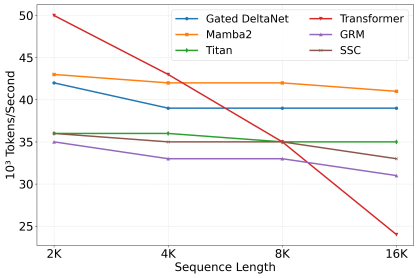

Figure 4: Training throughput comparison of memory caching variants and baselines.

4.5 Multi-Query Associative Recall (MQAR)

In this section, we evaluate the performance of MC-enhanced variants in Multi-Query Associative Recall (MQAR) task (Arora et al., 2024a). The results are reported in Figure 3. Our models show good performance compared to their base RNNs also the state-of-the-art recurrent models, achieving the best performance per dimension value compared to state-of-the-art models such as Atlas (Behrouz et al., 2025a).

4.6 Ablation Studies

Figure 5: Ablation Study on MC. All design choices of MC are positively contributing to its effectiveness.

Model	Language Modeling ppl ↓	C.S. Reasoning acc ↑	Retrieval acc ↑
Titans (GRM)	13.3	58.3	40.5
- Context-dependent	13.4	57.4	33.0
- Gating	13.5	56.9	32.4
- Linear Memory	13.7	56.3	34.5
Titans (SSC)	13.4	57.6	36.3
- Context-dependent	13.4	57.1	32.6
- Gating	13.5	56.8	31.9
- Linear Memory	13.8	56.8	33.4

In this section, we evaluate the effect of design choices in the MC framework. The first choice is wether γ should be the function of only input or also the context of blocks. The results are reported in Figure 5. This design choice has shown significant improvement on average. The second design is to remove the gating. Note that without gating, the design collapses into residual memory. The results show even this simple design can enhance the performance of the models. Finally, in the third design, we use a linear memory module. Surprisingly, using memory caching results in more robustness of the performance with respect to the memory architecture and expressivity.

4.7 Efficiency

Finally, we evaluate the training throughput of our variants with baselines. The results are reported in Figure 4. Our MC variants provide a middle ground between Transformers and RNNs, and they become extremely efficient compared to Transformers, when increasing the context length. These results indicate that our SSC variant has the best of both worlds and while performs on par or better compared to other variants in the diverse downstream tasks that we discussed earlier, they also add minimal overhead compare to their original base RNN variant. Furthermore, they show significantly better efficiency in longer sequences.

5 Conclusion

In this paper, we present Memory Caching (MC), a simple technique applicable to all recurrent neural networks, that caches a subset of memory state, allowing subsequent tokens directly attend to its past relevant tokens. Our experiments show improvements over a subset of baselines. A lot of choices in this paper have been made to keep the resulting model as simple as possible, better showing the effect of memory caching idea. In future work, more expressive pooling or routing mechanism can be used to further enhance the performance.

REFERENCES

Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language models enable simple systems for generating structured views of heterogeneous data lakes. *arXiv preprint arXiv:2304.09433*, 2023.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Re. Zoology: Measuring and improving recall in efficient language models. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=LY3ukUANko.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Re. Simple linear attention language models balance the recall-throughput tradeoff. In *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=e93ffDcpH3.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding. In *ACL (1)*, pp. 3119–3137, 2024. URL https://aclanthology.org/2024.acl-long.172.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Atlas: Learning to optimally memorize the context at test time. *arXiv preprint arXiv:2505.23735*, 2025a.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. It's all connected: A journey through test-time memorization, attentional bias, retention, and online optimization. *arXiv preprint arXiv:2504.13173*, 2025b.

Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. Birth of a transformer: A memory viewpoint. *Advances in Neural Information Processing Systems*, 36: 1560–1588, 2023.

Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. Birth of a transformer: A memory viewpoint. *Advances in Neural Information Processing Systems*, 36, 2024.

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

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

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

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

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Róbert Csordás, Christopher Potts, Christopher D Manning, and Atticus Geiger. Recurrent neural networks learn to store and generate sequences using non-linear representations. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 248–262, 2024.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *ACL (1)*, pp. 2978–2988. Association for Computational Linguistics, 2019. ISBN 978-1-950737-48-2.

Karan Dalal, Daniel Koceja, Gashon Hussein, Jiarui Xu, Yue Zhao, Youjin Song, Shihao Han, Ka Chun Cheung, Jan Kautz, Carlos Guestrin, et al. One-minute video generation with test-time training. *arXiv preprint arXiv:2504.05298*, 2025.

Tri Dao, Albert Gu, Matthew Eichhorn, Atri Rudra, and Christopher Ré. Learning fast algorithms for linear transforms using butterfly factorizations. In *International conference on machine learning*, pp. 1517–1527. PMLR, 2019.

Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate training. In *International Conference on Machine Learning*, pp. 4690–4721. PMLR, 2022.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*, 2019.

Xavier Gonzalez, Andrew Warrington, Jimmy Smith, and Scott Linderman. Towards scalable and stable parallelization of nonlinear rnns. *Advances in Neural Information Processing Systems*, 37: 5817–5849, 2024.

Han Guo, Songlin Yang, Tarushii Goel, Eric P Xing, Tri Dao, and Yoon Kim. Log-linear attention. *arXiv preprint arXiv:2506.04761*, 2025.

Ramin Hasani, Mathias Lechner, Tsun-Hsuan Wang, Makram Chahine, Alexander Amini, and Daniela Rus. Liquid structural state-space models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=g4OTKRKfS7R.

Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology press, 2005.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Jerry Yao-Chieh Hu, Dennis Wu, and Han Liu. Provably optimal memory capacity for modern hopfield models: Transformer-compatible dense associative memories as spherical codes. *arXiv preprint arXiv:2410.23126*, 2024.

Yuzhen Huang, Jinghan Zhang, Zifei Shan, and Junxian He. Compression represents intelligence linearly. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=SHMj84U5SH.

Kazuki Irie, Imanol Schlag, Robert Csordas, and Juergen Schmidhuber. Going beyond linear transformers with recurrent fast weight programmers. *Advances in neural information processing systems*, 34:7703–7717, 2021.

Kazuki Irie, Imanol Schlag, Róbert Csordás, and Juergen Schmidhuber. A modern self-referential weight matrix that learns to modify itself. In *International Conference on Machine Learning*, pp. 9660–9677. PMLR, 2022.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

Yanming Kang, Giang Tran, and Hans De Sterck. Fast multipole attention: A divide-and-conquer attention mechanism for long sequences. *arXiv preprint arXiv:2310.11960*, 2023.

M. Karami and V. Mirrokni. Lattice: Learning to efficiently compress the memory, 2025.

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

Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern recognition*, pp. 4999–5007, 2017.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Dmitry Krotov. Hierarchical associative memory. *arXiv preprint arXiv:2107.06446*, 2021.

Dmitry Krotov and John J Hopfield. Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29, 2016.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Igorevich Sorokin, Artyom Sorokin, and Mikhail Burtsev. BABILong: Testing the limits of LLMs with long context reasoning-in-a-haystack. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=u7m2CG84BQ.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

Xiaoyu Li, Yuanpeng Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. On the expressive power of modern hopfield networks. *arXiv preprint arXiv:2412.05562*, 2024.

Yi Heng Lim, Qi Zhu, Joshua Selfridge, and Muhammad Firmansyah Kasim. Parallelizing non-linear sequential models over the sequence length. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=E34AlVLN0v.

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. Longhorn: State space models are amortized online learners. *arXiv preprint arXiv:2407.14207*, 2024.

Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. Openceres: When open information extraction meets the semi-structured web. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3047–3056, 2019.

Carlo Lucibello and Marc Mézard. Exponential capacity of dense associative memories. *Physical Review Letters*, 132(7):077301, 2024.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=QZgo9JZpLq.

Tsendsuren Munkhdalai and Hong Yu. Neural semantic encoders. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 1, pp. 397. NIH Public Access, 2017.

Tsendsuren Munkhdalai, Alessandro Sordoni, Tong Wang, and Adam Trischler. Metalearned neural memory. *Advances in Neural Information Processing Systems*, 32, 2019.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 2024.

Tan Nguyen, Vai Suliafu, Stanley Osher, Long Chen, and Bao Wang. Fmmformer: Efficient and flexible transformer via decomposed near-field and far-field attention. *Advances in neural information processing systems*, 34:29449–29463, 2021.

Denis Paperno, German Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Bo Peng, Eric Alcaide, Quentin Gregory Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Nguyen Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan S. Wind, Stanisław Wozniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the transformer era. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=7SaXczaBpG.

Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Xingjian Du, Teddy Ferdinan, Haowen Hou, et al. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence. *arXiv preprint arXiv:2404.05892*, 2024.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Haowen Hou, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, et al. Rwkv-7" goose" with expressive dynamic state evolution. *arXiv preprint arXiv:2503.14456*, 2025.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pp. 28043–28078. PMLR, 2023.

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

Shikai Qiu, Andres Potapczynski, Marc Finzi, Micah Goldblum, and Andrew Gordon Wilson. Compute better spent: Replacing dense layers with structured matrices. *arXiv preprint arXiv:2406.06248*, 2024.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Thomas Adler, David Kreil, Michael K Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=tL89RnzIiCd.

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

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454/.

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

Juergen Schmidhuber. Learning to control fast-weight memories: An alternative to recurrent nets. accepted for publication in. *Neural Computation*, 1992.

Juergen Schmidhuber. Reducing the ratio between learning complexity and number of time varying variables in fully recurrent nets. In *ICANN'93: Proceedings of the International Conference on Artificial Neural Networks Amsterdam, The Netherlands 13–16 September 1993 3*, pp. 460–463. Springer, 1993.

Mark Schöne, Babak Rahmani, Heiner Kremer, Fabian Falck, Hitesh Ballani, and Jannes Gladrow. Implicit language models are rnns: Balancing parallelization and expressivity. *arXiv preprint arXiv:2502.07827*, 2025.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. Deltaproduct: Increasing the expressivity of deltanet through products of householders. *arXiv preprint arXiv:2502.10297*, 2025.

Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Ai8Hw3AXqks.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

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

Matteo Tiezzi, Michele Casoni, Alessandro Betti, Tommaso Guidi, Marco Gori, and Stefano Melacci. On the resurgence of recurrent models for long sequences: Survey and research opportunities in the transformer era. *arXiv preprint arXiv:2402.08132*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017a. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017b. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Johannes Von Oswald, Maximilian Schlegel, Alexander Meulemans, Seijin Kobayashi, Eyvind Niklasson, Nicolas Zucchet, Nino Scherrer, Nolan Miller, Mark Sandler, Max Vladymyrov, et al. Uncovering mesa-optimization algorithms in transformers. *arXiv preprint arXiv:2309.05858*, 2023.

Ke Alexander Wang, Jiaxin Shi, and Emily B Fox. Test-time regression: a unifying framework for designing sequence models with associative memory. *arXiv preprint arXiv:2501.12352*, 2025.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. PMLR, 2022.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024a.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in Neural Information Processing Systems*, 37:115491–115522, 2024b.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluis Marquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Zhanpeng Zeng, Sourav Pal, Jeffery Kline, Glenn M Fung, and Vikas Singh. Multi resolution analysis (mra) for approximate self-attention. In *International conference on machine learning*, pp. 25955–25972. PMLR, 2022.

Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T Freeman, and Hao Tan. Test-time training done right. *arXiv preprint arXiv:2505.23884*, 2025.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

A Related Work

Modern Linear Recurrent Neural Networks. Recent efforts have focused on alleviating the quadratic complexity and context-length limitations of Transformers, motivating the development of efficient recurrent alternatives that offer faster inference and training (Tiezzi et al., 2024). Early models such as RetNet (Sun et al., 2023), RWKV (Peng et al., 2023), and S5 (Smith et al., 2023) relied on data-independent transition matrices with Hebbian-like updates. A subsequent wave of work introduced input-dependent parameters into linear RNN architectures (e.g., linear RNNs (Hasani et al., 2023; Smith et al., 2023), RWKV6 (Peng et al., 2024)), alongside more expressive update mechanisms, including variants of the delta rule (Peng et al., 2025; Schlag et al., 2021; Yang et al., 2024b;a; Liu et al., 2024). This line of research has further extended to deeper architectures, incorporating delta-rule-like updates (Sun et al., 2024) or data-dependent momentum-based rules with forget gating (Behrouz et al., 2024). More recently, Siems et al. (2025) enhanced delta-rule models by applying multiple gradient descent updates per token, yielding more expressive state-tracking capabilities. Beyond linear recurrent models, several works investigate RNNs with non-linear recurrence (Behrouz et al., 2025b; Csordás et al., 2024; Merrill et al., 2024; Lim et al., 2024; Schöne et al., 2025; Karami & Mirrokni, 2025; Von Oswald et al., 2023; Gonzalez et al., 2024), with emphasis on accelerating their training (Gonzalez et al., 2024; Lim et al., 2024; Schöne et al., 2025).

Efficient Attention Mechanisms. In addition to recurrent architectures, recent work has proposed using structured matrices to improve the efficiency of token and channel mixing layers. For example, Butterfly matrices (Dao et al., 2019), Monarch matrices (Dao et al., 2022), and Block Tensor-Train matrices (Qiu et al., 2024) provide compact yet expressive parameterizations that reduce the computational burden of dense projections. Other approaches design sparse or hybrid attention mechanisms, such as sliding-window attention or models that combine localized recurrence with selective long-range connections (Nguyen et al., 2021; Arora et al., 2024b; Munkhdalai et al., 2024). Another family of approaches reduces the quadratic complexity of attention to nearly log-linear time. Classical examples include Reformer (Kitaev et al., 2020), which uses locality-sensitive hashing to cluster queries and keys, and LogSparse Transformer (Li et al., 2019) and Informer (Zhou et al., 2021), which rely on structured sparsity patterns for efficiency in long-sequence and time-series tasks. Subsequent work has introduced more elaborate designs, such as multi-resolution attention (Zeng et al., 2022), which progressively refines attention scores from coarse to fine levels, and Fast Multipole Attention (Kang et al., 2023), which adapts the fast multipole method for scalable long-range interactions. Recently, Guo et al. (2025) introduce Log-Linear Attention, a framework that augments linear attention with a logarithmically growing set of hidden states organized via Fenwick tree partitioning. This design achieves $O(T \log T)$ training complexity and $O(\log T)$ decoding memory, while preserving hardware-efficient parallelization.

Fast Weight Programs and Meta Learning. The view of linear layers as key-value associative memory systems dates back to Hopfield networks (Hopfield, 1982). This idea was later extended through the development of fast weight programmers, in which dynamic fast programs are integrated into recurrent neural networks to function as writable memory stores (Schlag et al., 2021; Schmidhuber, 1992; 1993). Among the learning paradigms for such systems, Hebbian learning (Hebb, 2005) and the delta rule (Prados & Kak, 1989) have been most prominent. Both rules have been extensively studied in the literature (Munkhdalai & Yu, 2017; Schmidhuber, 1992; Munkhdalai et al., 2019; Schlag et al., 2021; Irie et al., 2021; Yang et al., 2024b;a).

Hopfield Networks. Our formulation builds on the broad concept of associative memory, where the goal is to learn mappings between keys and values. Seminal work by Hopfield (1982) introduced Hopfield Networks as one of the earliest neural architectures explicitly based on associative memory, formalized through the minimization of an energy function for storing key-value pairs. While classical Hopfield networks have seen reduced applicability due to limitations in vector-valued memory capacity and the structure of their energy function, recent studies have sought to enhance their capacity through various approaches (Krotov, 2021; Li et al., 2024; Krotov & Hopfield, 2016). In particular, extensions of their energy functions with exponential kernels have been explored (Krotov & Hopfield, 2016; Lucibello & Mézard, 2024). Moreover, connections between modern Hopfield networks and Transformer architectures have been actively investigated (Ramsauer et al., 2021; Hu et al., 2024).

Table 5: Architectural Details.

Model	Block	Dim	Head	Peak LR	Token
760M	24	1536	16	1.25e-3	30B
1.3B	18	2048	8	7e-4	100B

B Experimental Details

In our experimental setup we follow recent studies on recurrent models (Yang et al., 2024a; Behrouz et al., 2024; 2025b; Zhang et al., 2025; Guo et al., 2025), we use Wikitext (Merity et al., 2017), LMB (Paperno et al., 2016), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-easy (ARC-e) and ARC-challenge (ARC-c) (Clark et al., 2018), SIQA (Sap et al., 2019), and BoolQ (Clark et al., 2019). Also, the baselines results are from Behrouz et al. (2025b; 2024). In the training, we use a vocabulary size of 32K and use training length of 4K tokens (2K for SWA). We employ AdamW optimizer with learning rate of $4e$-4 with cosine annealing schedule with batch size of 0.5M tokens, and weight decay of 0.1.

For the memory architecture, unless state otherwise, we use an MLP with 2 layers with expansion factor of 4 and GELU activation function (Hendrycks & Gimpel, 2016). We also use residual connections and layer norm at the end of each chunk: $\mathcal{M}(x) = x + W_1\sigma(W_2x)$.