Scaling up the State Size of RNN LLMs for Long-Context Scenarios

Kai Liu[1,2] **and Jianfei Gao**[1] **and Kai Chen**[1]
[1]Shanghai AI Laboratory
[2]Tongji University
{liukai,gaojianfei,chenkai}@pjlab.org.cn

Abstract

The Transformer architecture has become the standard LLM architecture due to its powerful self-attention mechanism. However, it suffers from quadratic computational complexity and linear memory complexity. RNN-based LLMs have been proposed as alternatives. Yet, RNN models struggle in long-context scenarios, making it challenging to replace self-attention with RNNs. We identify the state size as a critical bottleneck, which is significantly smaller than that of Transformers with a basic context length of 2k. However, simply increasing the state size significantly raises the number of parameters and lowers training efficiency. In this paper, we propose an efficient scaling method to scale the state size of RNN models to match the 2k context length of Transformers, with small parameters overhead. Experimental results demonstrate that scaling the state size significantly enhances long-context understanding. Retrieval performance scales almost linearly with state size, with a 454M model featuring an expanded state achieving performance comparable to a 1.47B model on FDA, a recall-intensive task. These findings highlight state scaling as a promising approach for advancing RNN-based LLMs.

1 Introduction

Transformer-based Large Language Models (LLMs) (Achiam et al., 2023) have achieved state-of-the-art performance across various tasks and applications (Liu et al., 2023; Zhang et al., 2024). Their core self-attention mechanism (Vaswani, 2017) excels at capturing long-range dependencies and fine-grained past information. However, its quadratic computational complexity and linear memory requirements limit scalability. To overcome these challenges, RNN models (Peng et al., 2024; Gu and Dao) have been proposed as efficient alternatives, offering linear computational complexity and constant memory usage.

RNN models have recently made significant progress, achieving performance comparable to Transformers on tasks like language modeling and common-sense reasoning (Zuo et al.; Waleffe et al., 2024). This positions RNNs as a promising alternative to Transformers. However, they still struggle with long-context scenarios, particularly in recall-intensive tasks like needle-in-a-haystack (Hsieh et al., 2024), limiting their ability to replace self-attention. Improving long-context understanding remains a key challenge.

Recent RNN models have made notable advancements, particularly in state expansion (Peng et al., 2024; Qin et al.; Sun et al., 2023a). Early RNNs (Peng et al.; Qin et al., 2023) relied on vector states, which were severely limited in memory capacity. To overcome this limitation, newer models (Peng et al., 2024; Qin et al.) introduced matrix states, significantly enhancing memory capacity. Despite these improvements, the state sizes of RNNs remain much smaller than those of Transformers with a 2k context length, making state size a critical bottleneck for long-context understanding. Yet, simply increasing the state size significantly raises the number of parameters and lowers training efficiency, making it challenging to scale RNN models to match Transformers. We propose an efficient method to scale the state size of RNN models to match Transformers with a 2k context length, alleviating the challenges of increased parameter counts and reduced training efficiency.

We evaluate the performance of the scaled RNN models on various tasks, including language modeling, zero-shot common-sense reasoning, and long-context understanding tasks. The results demonstrate that scaling the state size of RNN models significantly enhances their long-context understanding ability. Additionally, we conduct experiments to investigate the scalability of RNN models by examining how increasing the state size affects the maximum recallable context length. Our findings

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 11516–11529

show that the maximum recallable context length scales nearly linearly with the RNN state size, providing a valuable reference for further scaling of RNN models. These results highlight scaling the state size of RNN models as a promising approach to improving their long-context understanding ability.

In conclusion, our contributions are as follows:

- We propose an efficient method to scale the state size of RNN models to match Transformers with a 2k context length, addressing the challenges of increased parameter counts and reduced training efficiency.
- We evaluate the performance of the scaled RNN models on various tasks, showing that scaling the state size significantly improves the long-context understanding ability of RNN models.
- We conduct experiments to investigate the scalability of RNN state size, providing a valuable reference for further scaling of RNN models.

2 Related Work

2.1 RNN LLMs

RNN LLMs have made significant progress in long-context understanding tasks. In this paper, we focus on the state expansion of RNN models. Early RNN models, such as RWKV4 (Peng et al.) and HGRN (Qin et al., 2023), utilized vector states, where states are updated at each step by a vector. However, this approach has limited memory capacity. Building on linear attention (Katharopoulos et al., 2020), matrix states have been widely adopted to replace vector states. In this approach, states are updated at each step by a matrix, which is the outer product of keys and values, significantly enhancing the memory capacity of RNN LLMs. Recent RNN models, such as RWKV5 (Peng et al., 2024), Mamba2 (Dao and Gu), and others, commonly employ matrix states. Nevertheless, the state size of RNN models remains much smaller than that of self-attention mechanisms with a basic context length of 2k. To the best of our knowledge, this is the first work to scale the state size of RNN LLMs to match Transformers with a 2k context length. We argue that there is considerable room to scale the state size of RNN models, which is crucial for advancing their performance in long-context understanding tasks.

In addition to state expansion, various other methods have been proposed to enhance the performance of RNN LLMs. Data-dependent gates (Yang et al.; Peng et al., 2024; Dao and Gu) enable models to selectively forget irrelevant information while retaining important information. The delta update rule (Yang et al., 2025, 2024) and test-time training (Sun et al.) reformulate recurrent updates as optimization problems, making state updates more efficient. Notably, our scaling method can be integrated with these techniques to further improve the performance of RNN LLMs.

Outside the scope of LLMs, some prior methods have attempted to upscale the state size of linear attention in a parameter-efficient manner. DPFP (Deterministic Parameter-Free Projection) (Schlag et al., 2021) leverages interactions between the features of queries and keys to expand their dimensions. While this approach avoids introducing additional parameters, it results in sparse features. Similarly, LFM (Learnable Feature Map) (Pramanik et al., 2024) proposes using the outer product to expand the state size, which is equivalent to repeating heads with different weights, but this also leads to reduced performance. We compare our method with these approaches in the experiments.

2.2 Scalability

In recent years, research has highlighted the importance of LLM scalability, with scaling laws widely used to analyze this property. (Kaplan et al., 2020; Hoffmann et al., 2022) conducted extensive experiments using dense curve fitting and regression analysis to examine the relationships among performance, model size, dataset size, and other factors, establishing their scaling law as a key reference for guiding LLM scaling. Broader scaling trend analyses, which do not rely on dense curves or regression analysis, have also been conducted to explore LLM scalability under different scenarios and targets, as seen in (Gu and Dao; Dao and Gu; Li et al., 2023).

In this paper, we focus on the scalability of state size in RNN LLMs, a dimension less explored than model size. We assess scalability through coarse scaling trends, given real-world limitations in obtaining precise scaling laws for RNN LLMs, as detailed in Section 6. Consistent with (Ye et al., 2024), we observe that scaling different model components affects performance in distinct ways, and increasing the state size of RNNs is particularly effective for improving performance on long-context understanding tasks.

3 Preliminaries

3.1 Attention

Attention has become the cornerstone of large language models, leveraging queries and keys to compute attention scores, which are then used to generate the output.

The output is computed using the following formula:

$$\mathbf{O} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T + \mathbf{M}}{\sqrt{d_{qk}}}\right)\mathbf{V} \quad (1)$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ represent the queries, keys, and values, respectively. d_{qk} denotes the dimensionality of the queries and keys, and $\mathbf{M}$ is the causal attention mask.

During the generation process, key and value caches/states are maintained continuously, causing the state size to grow proportionally with the context length. This mechanism allows the attention model to recall information from earlier contexts effectively. For a context length of L, the average state size per batch during inference is $L \times H \times (d_{qk} + d_v)/2$, where H is the number of attention heads, and d_v is the dimensionality of the values.

3.2 Linear Attention

To address the quadratic complexity of traditional attention mechanisms, linear attention (Katharopoulos et al., 2020) was proposed. It eliminates the softmax operation and uses feature maps to process $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, approximating the attention score. The computation takes the following form:

$$\mathbf{O} = (\mathbf{Q}\mathbf{K}^T) \odot \mathbf{M}\mathbf{V} \quad (2)$$

This can also be reformulated in an RNN-like structure:

$$\begin{aligned} \mathbf{S}_t &= \mathbf{S}_{t-1} + \mathbf{k}_t^T\mathbf{v}_t = \sum_0^t \mathbf{k}_i^T\mathbf{v}_i, \\ \mathbf{o}_t &= \mathbf{q}_t \times \mathbf{S}_t, \quad \mathbf{S}_t \in \mathbb{R}^{[H \times d_{qk} \times d_v]} \end{aligned} \quad (3)$$

Here, $\mathbf{S}$ represents the state matrix, while $\mathbf{k}_i$ and $\mathbf{v}_i$ denote the key vector and value vector, respectively. Linear attention reduces the quadratic complexity of self-attention to linear complexity and replaces the growing state size with a fixed-sized state. As a result, linear attention is more efficient than self-attention in terms of both inference speed and memory usage. The state size for linear attention per batch is $H \times d_{qk} \times d_v$.

Assuming $d_{qk} = d_v = d$, the ratio of the state sizes between self-attention and linear attention can be expressed as:

$$\text{Ratio} = \frac{L \times H \times (d_{qk} + d_v)/2}{H \times d_{qk} \times d_v} = \frac{L}{d} \quad (4)$$

For $d = 128$ and $L = 2048$, the ratio is 16. This indicates that the state size of linear attention is 16 times smaller than that of self-attention, potentially making the reduced state size a bottleneck for long-context understanding tasks.

3.3 Recent Linear Attention Variants

In recent years, several linear attention variants have been proposed to improve the memory efficiency of linear attention. Among them, Gated Linear Attention (GLA) (Yang et al.) and Gated DeltaNet (Yang et al., 2024) are two notable examples with SOTA performance.

GLA introduces data-dependent gating mechanisms, similar to the forget gate used in LSTMs (Graves and Graves, 2012), to enhance the memory capabilities of linear attention. Its formulation is as follows:

$$\mathbf{S}_t = \text{diag}(\mathbf{g}_t) \odot \mathbf{S}_{t-1} + \mathbf{k}_t^T\mathbf{v}_t \quad (5)$$

where $\mathbf{g}_t$ acts as a gate that enables the model to selectively forget irrelevant information and retain important information, thereby improving the memory retention of linear attention.

Gated DeltaNet builds upon this idea by incorporating the delta rule (Schlag et al.) to update the state matrix. The update rule is expressed as:

$$\begin{aligned} \mathbf{S}_t &= \alpha_t\mathbf{S}_{t-1} + \beta_t\mathbf{k}_t^T(\mathbf{v}_t - \alpha_t\mathbf{k}_t\mathbf{S}_{t-1}) \\ &= \mathbf{S}_{t-1}(\alpha_t(\mathbf{I} - \beta_t\mathbf{k}_t^T\mathbf{k}_t)) + \beta_t\mathbf{k}_t^T\mathbf{v}_t \end{aligned} \quad (6)$$

From a test-time training perspective, the state $\mathbf{S}$ is treated as a weight matrix optimized via online stochastic gradient descent (SGD) with the objective $Loss(\mathbf{S}_t) = \frac{1}{2}\|\mathbf{k}_t\mathbf{S}_t - \mathbf{v}_t\|^2$. In this context, α_t functions as an adaptive weight decay term, while β_t serves as a learning rate. This delta-based update mechanism further strengthens the memory capabilities of linear attention.

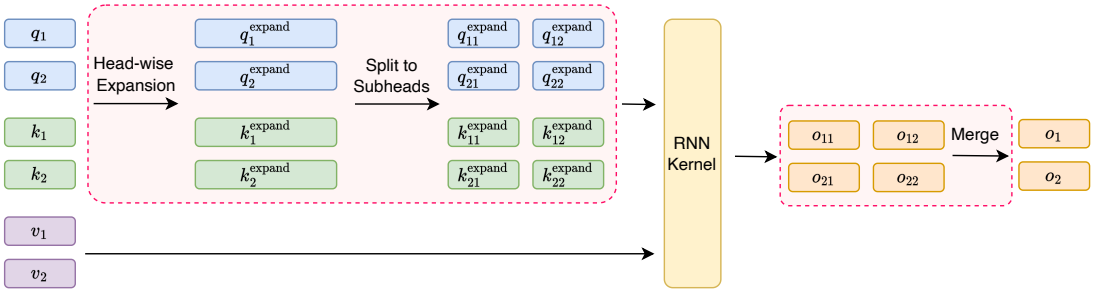

Figure 1: Our method for scaling the state size of RNN models includes head-wise expansion and the division of heads into subheads, highlighted by the red dashed-line rectangles. This figure demonstrates the expansion process with an expansion factor of 2 applied to 2 heads.

4 Method

As discussed in the previous section, linear attention employs a fixed-size state matrix to store information from the past, achieving greater computational and memory efficiency compared to self-attention. However, this efficiency comes at a cost: accumulating information over time makes it more challenging for RNN-based LLMs to retrieve past information, limiting their long-context understanding ability. From Equation (3), assuming all keys are normalized, when $\mathbf{q}_j = \mathbf{k}_j$ is used to retrieve $\mathbf{v}_j$, the output becomes:

$$\begin{aligned}\mathbf{o}_j &= \mathbf{q}_j^T \mathbf{S}_t = \sum_{i=0}^{t} (\mathbf{q}_j \mathbf{k}_i^T)\mathbf{v}_i \\ &= \mathbf{v}_j + \sum_{i=0, i\neq j}^{t} (\mathbf{q}_j \mathbf{k}_i^T)\mathbf{v}_i \end{aligned} \tag{7}$$

Without the softmax function to polarize attention scores, linear attention suffers from *attention dilution* (Qin et al., 2022). Ideally, all keys should be orthogonal to one another, allowing the attention score to focus entirely on the target key. One straightforward way to improve the model's retrieval ability is to increase the dimensionality of the keys, which makes it more likely for keys to become orthogonal.

Furthermore, as previously mentioned, there is significant room to scale the state size, which is proportional to d_{qk}. Therefore, scaling the state size with d_{qk} is a promising direction to enhance the long-context retrieval ability of RNN-based models. However, scaling the state size introduces two key challenges: an increased parameter count and reduced training efficiency. In the following sections, we propose our method to alleviate these challenges effectively. Our entire expansion method is summarized in Figure 1.

4.1 Head-wise Expansion for Parameter Efficiency

Directly scaling d_{qk} significantly increases the number of parameters. Specifically, $\mathbf{W}_q, \mathbf{W}_k \in \mathbb{R}^{D \times H d_{qk}}$, and scaling d_{qk} by a factor of E results in a proportional E-fold increase in parameters. This approach is not parameter-efficient. For instance, in a 400M-parameter model, scaling d_{qk} by a factor of 8 would nearly double the total number of parameters.

To address this inefficiency, we propose a head-wise expansion method. Instead of naively scaling d_{qk} across all heads, we expand d_{qk} within each head individually. Queries and keys are first generated using the original $\mathbf{W}_q$ and $\mathbf{W}_k$, and then each head is expanded separately. For example, the query expansion process of the ith head is as follows:

$$\begin{aligned}&\mathbf{q}_i = \mathbf{x} \times \mathbf{W}_{q_i} \in \mathbb{R}^{1 \times d_{qk}}, \\ &\mathbf{q}_i^{\text{expand}} = \mathbf{q}_i \times \mathbf{W}_{q_i}^{\text{expand}} \in \mathbb{R}^{1 \times E d_{qk}}, \\ &\mathbf{W}_{q_i}^{\text{expand}} \in \mathbb{R}^{d_{qk} \times E d_{qk}}, \\ &\mathbf{W}_{q_i} \in \mathbb{R}^{d_{\text{model}} \times d_{qk}}, \quad i \in [0, H] \end{aligned} \tag{8}$$

We summarize the parameter comparison in Table 1. Naive scaling introduces E-fold more parameters, whereas the head-wise expansion method introduces only $(1 + \frac{E}{H})$-fold more parameters, making it significantly more parameter-efficient. When $E = H$, the additional $\mathbf{W}_q^{\text{expand}}$ has the same size as $\mathbf{W}_q$. Since $\mathbf{W}_q$ and $\mathbf{W}_k$ constitute only a small portion of the total parameters, the overhead is limited—approximately 10%, which is acceptable. For example, in a 400M-parameter model with $H = 8$,

	Number of Parameters	Scaling Ratio
Baseline	$d_{\text{model}} \times (d_{qk} \times H)$	1 ×
Naive	$d_{\text{model}} \times (d_{qk} \times H) \times E$	E ×
Head-wise	$d_{\text{model}} \times (d_{qk} \times (H + E))$	$(1 + \frac{E}{H})\times$

Table 1: Parameter comparison: The number of parameters for queries or keys using different methods. The parameter count is for a single layer of the model. Our head-wise expansion method is significantly more parameter-efficient than naive scaling.

Method	Device	E=8	E=16
Naive	H800	9569	2933
Subhead	H800	15722	6554
Naive	RTX-3090	OOM	OOM
Subhead	RTX-3090	4829	2658

Table 2: Training Efficiency Comparison. We compare the training throughput (tokens per second) of the naive and subhead division on two different GPUs. The subhead division significantly improves training efficiency when the state size is large. OOM indicates that the model runs out of shared memory. The speed is measured with 400M-parameter models, with training lengths of 4096 and 1024 on H800 and RTX-3090, respectively.

scaling d_{qk} by a factor of 8 results in an increase to 454M parameters, representing a manageable overhead.

4.2 Subhead Division for Training Efficiency

In existing RNN kernels (Yang et al., 2024), a single CUDA block is assigned to process a linear attention head and stores the states in shared memory instead of global memory. This approach significantly improves training efficiency. However, each CUDA block has access to only limited shared memory and computational resources. When the expansion factor E becomes large, the model runs slowly or may even exhaust the shared memory.

To overcome this limitation, we propose splitting a head into multiple subheads, running on multiple CUDA blocks, and then merging them back into a single head. Specifically, an expanded query or key head is divided into E subheads, with each subhead retaining the same size as the original head. Additionally, RNN-based LLMs commonly employ convolutional layers and feature maps to process queries and keys, which we denote as f. We ensure that f operates on each subhead independently. The values ($\mathbf{v}_i$) are shared across all subheads within the same head. The dimensions of other components, such as gates, α, and β, are expanded to align with the subheads, as their associated parameters are relatively small. We find that the expanded gates, α, and β provide finer control over the state update, which is important for the model's performance. The output for each subhead is computed independently, and the final output of the head is obtained by summing the outputs of all subheads:

$$\mathbf{q}_{ij}^{\text{expand}} = \mathbf{q}_{i\,[:,j\cdot d_{qk}:(j+1)\cdot d_{qk}]}^{\text{expand}} \in \mathbb{R}^{1\times d_{qk}} \quad (9)$$

$$\mathbf{o}_i = \sum_{j=0}^{E} \text{RNN}(f(\mathbf{q}_{ij}^{\text{expand}}), f(\mathbf{k}_{ij}^{\text{expand}}), \mathbf{v}_i) \quad (10)$$

We compare our subhead division with the naive method in terms of training efficiency, as shown in Table 2. Our subhead division doubles the training throughput on H800 when $E = 16$. Additionally, it enables the model to run efficiently on less resource-intensive GPUs, such as the RTX-3090.

4.3 Enhancing Nonlinear Capacity

Building on the head-wise expansion method, we further enhance the model's nonlinear capacity by incorporating a SiLU activation function (Elfwing et al., 2017) prior to the head-wise expansion. Our experiments demonstrate that the SiLU activation function improves the model's retrieval ability, enabling it to capture more complex patterns.

5 Experiment

5.1 Experimental Setting

We scale the state size of two RNN models, Gated-DeltaNet (Yang et al., 2024) and GLA (Yang et al.), under two configurations: approximately 400M and 1.5B parameters. Both models are derived from Transformer++ (Touvron et al., 2023), replacing the self-attention mechanism with linear attention variants. All base models consist of 24 layers. For the 400M-parameter models, the model dimension is set to 1024 with $d_{qk} = d_v = 128$, while for the 1.5B-parameter models, the model dimension is 2048 with $d_{qk} = d_v = 256$.

For the 400M-parameter models, we use expansion factors of 2, 4, 8, and 16, corresponding to state sizes equivalent to those of a Transformer with context lengths of 256, 512, 1024, and 2048, respectively. For the 1.5B-parameter models, the expansion factors are 2, 4, and 8, corresponding to state sizes equivalent to those of a Transformer with context lengths of 512, 1024, and 2048, respectively. Additionally, we compare the effects of scaling the

number of layers versus scaling the state size while maintaining the same total parameter count. In the tables below, "-L" indicates scaling the number of layers, while "-E" indicates scaling the state size. We provide more details in Appendix A.3.

5.2 Common Sense Reasoning

First, we evaluate the performance of scaled RNN models and Transformer++ (Touvron et al., 2023) on language modeling and zero-shot common-sense reasoning tasks. While improving performance on these tasks is not our primary objective, increasing the state size of RNNs yields modest gains. With a comparable number of parameters, scaling the state size achieves performance similar to scaling the number of layers. At the 400M parameter scale, Gated DeltaNet-E8 outperforms Gated DeltaNet-L by an average of 0.2%.

5.3 Real-World Recall-Intensive Tasks

We evaluate the performance of scaled RNN models and Transformer++ on real-world retrieval tasks, including FDA (Arora et al., 2023), SWDE (Lockard et al., 2019), and SQuAD (Rajpurkar et al., 2018), with maximum context lengths of 2723, 2222, and 928, respectively.

Finding 1: Scaling the state size of RNNs significantly enhances recall ability, especially for long-context tasks. Our results demonstrate that increasing the state size of RNNs substantially improves recall performance, particularly for tasks with longer contexts. Expanding the state size of RNN models from $2\times$ to $8\times$ consistently yields performance gains. At the 1.5B scale, Gated DeltaNet-E8 outperforms the base Gated DeltaNet by an average of 11.73%, highlighting the effectiveness of state size scaling in enhancing long-context retrieval ability.

In contrast, scaling the number of layers has a more limited impact. At the same 1.5B scale, Gated DeltaNet-L achieves only a 3.8% improvement on average compared to the baseline model. Gated DeltaNet-E8 with 454M parameters achieves better performance than Gated DeltaNet with 1.47B parameters on FDA. Furthermore, the performance gains across the three tasks follow the order FDA > SWDE > SQuAD, indicating that scaled RNN models are particularly effective for tasks involving longer contexts.

Finding 2: The update rule plays a critical role in recall ability scaling. At the 400M scale, Gated DeltaNet-E8 achieves a 9.36% improvement, whereas GLA achieves only a 3.46% average gain. This underscores the importance of the update rule in scaling retrieval ability. Specifically, the delta rule employed by Gated DeltaNet proves more effective than the gated linear attention mechanism used in GLA.

5.4 Long Context Understanding

To further assess the performance of scaled RNN models, we evaluate Gated DeltaNet with 1.5B parameters on LongBench (Bai et al., 2024), a specialized benchmark designed for long-context understanding. The maximum context length in LongBench is up to 64k tokens. The results of the scaled RNN models are presented in Table 5. Our findings indicate that expanding the state size of RNN models consistently enhances performance across tasks. Specifically, Gated DeltaNet-E8 achieves an average improvement of 5.74% compared to the baseline model and outperforms Gated DeltaNet-L by 3.16% on average. Furthermore, due to the limited extrapolation capabilities of Transformer++, its performance on LongBench is inferior to that of Gated DeltaNet. We further compare with Transformer++-2k, which is a Transformer++ model using sliding window attention with a window size of 2048. The results show that Gated DeltaNet-E8 outperforms Transformer++-2k by 2.79% on average with a smaller state size[1].

5.5 State-Recall Context Scaling Trend

To quantify the scalability of RNNs on long-context tasks, we propose the state-recall context scaling trend. The maximum recall context length is defined as the longest context length at which RNNs maintain recall accuracy above a specified threshold. To evaluate this, we test our models on S-NIAH (Single Needle-In-A-Haystack) (Hsieh et al., 2024) with UUIDs as needle values. We measure accuracy with progressively increasing haystack context lengths, using a step size of 512. Thresholds are set between 0.5 and 0.8, incremented by 0.1. The resulting maximum recall context lengths for various state sizes are presented in Figure 2.

Finding 3: The maximum recall context length of RNNs scales linearly with state size. The results in the figure reveal a near-linear relationship between the maximum recall context length and the state size, indicating that the recall ability of RNNs can be significantly improved by scaling the

[1]The computation of the state size for Transformer++ with sliding window attention is detailed in Appendix A.4.

Model	State	Params	Wiki ↓	LMB ↓	LMB ↑	PIQA ↑	Hella ↑	Wino ↑	ARC-e ↑	ARC-c ↑	SIQA ↑	BoolQ ↑	Avg ↑
Transformer++	/	400M	24.24	32.49	34.64	64.69	30.97	52.33	45.79	18.94	36.23	55.29	42.36
Transformer++	/	1.47B	14.31	9.74	52.53	72.36	43.55	59.12	60.14	26.79	40.43	61.41	52.04
GLA	1×	400M	26.32	36.60	31.54	63.66	30.85	49.41	45.12	19.62	37.67	59.91	42.22
GLA-L	1.2×	455M	**25.41**	32.99	33.98	65.67	31.45	**52.64**	**47.22**	**20.39**	**37.72**	60.21	**43.66**
GLA-E2	2×	412M	26.34	35.56	33.28	**64.69**	31.07	51.30	44.95	19.20	36.39	58.78	42.46
GLA-E4	4×	427M	26.07	**32.53**	**34.43**	64.42	31.04	49.96	45.54	19.28	37.31	**61.07**	42.88
GLA-E8	8×	454M	25.99	34.74	32.89	63.93	**31.52**	51.30	45.54	18.77	36.80	59.08	42.48
Gated DeltaNet	1×	400M	25.25	36.20	31.40	64.31	31.20	51.85	45.03	18.43	36.90	57.61	42.09
Gated DeltaNet-L	1.2×	455M	24.17	33.60	32.97	64.80	32.17	51.93	46.84	**19.88**	**38.02**	58.62	43.16
Gated DeltaNet-E2	2×	413M	24.38	31.93	34.76	**66.21**	31.73	50.12	**48.40**	19.03	37.21	58.72	43.27
Gated DeltaNet-E4	4×	427M	23.96	32.74	33.53	65.61	32.24	**52.49**	47.56	19.11	36.64	**58.96**	43.27
Gated DeltaNet-E8	8×	454M	**23.61**	**29.50**	**36.85**	65.29	**32.30**	50.36	46.46	19.28	37.82	58.53	**43.36**
Gated DeltaNet	1×	1.47B	17.27	14.57	44.89	71.06	40.28	56.75	56.31	23.81	**40.38**	61.22	49.34
Gated DeltaNet-L	1.2×	1.68B	16.42	**10.37**	**50.75**	72.63	41.95	57.77	58.63	**26.71**	39.71	60.49	**51.08**
Gated DeltaNet-E2	2×	1.52B	16.45	10.65	**50.75**	**72.91**	41.72	57.46	59.34	26.11	38.69	**61.62**	51.07
Gated DeltaNet-E4	4×	1.57B	15.99	10.80	50.16	72.20	41.54	58.09	59.47	24.57	40.28	60.52	50.85
Gated DeltaNet-E8	8×	1.68B	**15.66**	10.87	50.63	72.52	**42.03**	**58.96**	**59.64**	25.34	39.82	59.05	51.00

Table 3: Performance Comparison on Language Modeling and Zero-shot Common-sense Reasoning. "L" denotes scaling the number of layers, and "E" denotes scaling the state size. "State" indicates the state size with respect to corresponding base models. The upward arrow (↑) indicates that higher values are better, while the downward arrow (↓) indicates that lower values are better.

Model	State	Params	FDA ↑	SWDE ↑	SQuAD ↑	Avg ↑
Transformer++	/	400M	72.96	64.00	36.96	57.97
Transformer++	/	1.47B	77.77	79.93	48.26	68.65
GLA	1×	400M	11.52	31.95	28.32	23.93
GLA-L	1.2×	455M	12.34	31.68	**29.93**	24.65
GLA-E2	2×	412M	15.52	28.98	29.12	24.54
GLA-E4	4×	427M	17.06	32.94	28.28	26.09
GLA-E8	8×	450M	**19.06**	**34.47**	28.65	**27.39**
Gated DeltaNet	1×	400M	22.50	32.94	28.65	28.03
Gated DeltaNet-L	1.2×	455M	20.69	36.27	28.35	28.44
Gated DeltaNet-E2	2×	413M	21.14	38.34	29.79	29.76
Gated DeltaNet-E4	4×	427M	34.85	41.49	**31.23**	35.86
Gated DeltaNet-E8	8×	454M	**38.93**	**43.02**	30.23	**37.39**
Gated DeltaNet	1×	1.47B	37.66	53.47	37.60	42.91
Gated DeltaNet-L	1.2×	1.69B	41.74	60.31	38.07	46.71
Gated DeltaNet-E2	2×	1.52B	46.01	60.40	**39.71**	48.71
Gated DeltaNet-E4	4×	1.57B	52.63	61.48	39.51	51.21
Gated DeltaNet-E8	8×	1.68B	**58.53**	**66.16**	39.24	**54.64**

Table 4: Performance Comparison on Real-World Retrieval Tasks: FDA, SWDE, and SQuAD.

state size. Furthermore, compared to GLA, Gated DeltaNet exhibits a steeper slope, highlighting its greater potential for state size scaling. We further evaluate the scaling trend on 1.5B-parameter models in Figure 3, which is consistent with our findings on 400M-parameter models.

5.6 Comparison with Transformer under Similar State Sizes

To investigate whether scaling the state size of RNNs can make them competitive with Transformer models, we compare the performance of RNNs and Transformer++ with similar state sizes. Since the state size of a Transformer is proportional to its context length, we control the Transformer's state size by adjusting the context length in the tasks. For this comparison, we use a more challenging benchmark, M-NIAH (Multi-keys Needle-In-A-Haystack) with numbers as needle values (Hsieh et al., 2024), with context lengths of 512, 1024, and 2048. Specifically, we evaluate the performance of Gated DeltaNet and Transformer++ at the 1.5B parameter scale. Gated DeltaNet, with expansion factors of 2, 4, and 8, has state sizes comparable to those of a Transformer with context lengths of 512, 1024, and 2048, respectively. The results are provided in Table 6.

Finding 4: Gated DeltaNet remains less com-

Model	State	Single-Doc QA ↑			Multi-Doc QA ↑			Summarization ↑			Few-shot ↑			Code ↑		Avg ↑
		NQA	QQA	MFQ	HQA	2WM	Mus	GvR	QMS	MNs	TRC	TQA	SSM	LCC	RBP	
Transformer++	/	0.35	6.77	12.58	1.36	5.66	0.38	8.14	3.17	11.72	17.50	12.49	7.77	45.00	19.52	10.89
Transformer++-2k	16×	5.38	6.05	12.74	7.70	9.46	3.21	9.02	14.47	6.71	35.00	**44.62**	27.69	**52.05**	**48.37**	20.18
Gated DeltaNet	1×	2.46	5.78	13.67	5.81	7.70	3.71	7.06	16.71	12.16	28.00	32.16	26.97	40.83	38.20	17.23
Gated DeltaNet-L	1.2×	6.18	6.65	13.53	6.92	9.13	2.76	10.27	16.38	14.81	40.50	36.84	31.86	43.48	37.98	19.81
Gated DeltaNet-E2	2×	**7.29**	6.80	13.66	6.40	8.96	3.88	11.09	**16.85**	10.20	48.50	30.60	31.11	41.61	40.49	19.82
Gated DeltaNet-E4	4×	6.90	6.45	13.79	7.84	9.97	**4.45**	**13.67**	16.40	10.16	61.00	23.69	30.78	44.68	41.42	20.80
Gated DeltaNet-E8	8×	4.11	**6.94**	**16.71**	**7.98**	**10.61**	3.78	11.07	16.47	**18.11**	**67.50**	41.88	**33.95**	43.09	39.39	**22.97**

Table 5: Performance Comparison on 14 Tasks from LongBench (Bai et al., 2024). Transformer++-2k is a Transformer++ model using sliding window attention with a window size of 2048.

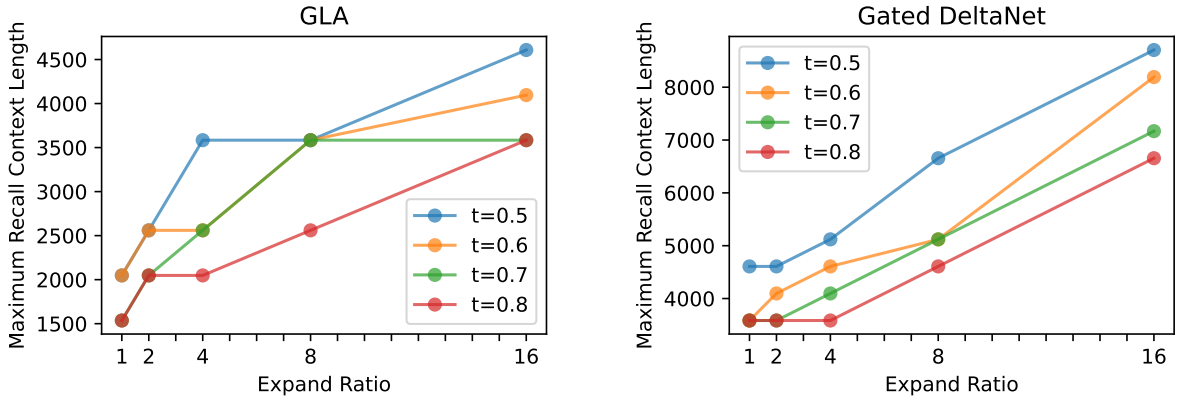

Figure 2: Scaling Trend of Maximum Recall Context Length vs. State Size on S-NIAH. The X-axis represents the state size expansion factor, while the Y-axis denotes the maximum recall context length at which the model maintains recall accuracy above a specified threshold t.

Model	512 ↑	1024 ↑	2048 ↑	AVG ↑
Transformer++	100.00	99.60	99.20	99.60
Gated DeltaNet-E2	<u>68.20</u>	24.40	4.00	32.20
Gated DeltaNet-E4	65.00	<u>30.20</u>	7.20	34.13
Gated DeltaNet-E8	94.20	54.80	<u>20.80</u>	56.60

Table 6: Comparison of Transformer and Gated DeltaNet on M-NIAH: We compare Transformer++ and Gated DeltaNet models with similar state sizes. For the Transformer, the state size is controlled by adjusting the context length ("512", "1024", "2048"). Underlined values indicate that the Gated DeltaNet has the same state size as the Transformer for the corresponding context length.

petitive than Transformer models in terms of recall ability, even with similar state sizes. The results demonstrate that Gated DeltaNet, even when matched in state size, underperforms compared to Transformer models. This highlights the need for further advancements in memory efficiency and the development of new mechanisms to narrow the retrieval ability gap between RNNs and Transformers.

5.7 Ablation Study

We conduct an ablation study to evaluate the contributions of the key components in our proposed methods. 1) We replace the head-wise expansion with other efficient alternatives, including low-rank expansion, DPFP (Schlag et al., 2021), and LFM (Pramanik et al., 2024). 2) We evaluate the impact of removing the subhead mechanism. 3) We assess the effect of replacing the SiLU activation function with ReLU, 1+ELU, or identity.

The results are summarized in Table 7. The results indicate that the head-wise expansion mechanism preserves more information compared to other methods. The subhead division enhances performance by providing finer control over state updates. Additionally, the SiLU activation function improves the model's performance by introducing greater nonlinear capacity.

Additionally, we compare our method with a naive approach of directly increasing the number of heads, as shown in Table 8. Given the same state size, increasing the number of heads does improve performance, but it also significantly increases the number of parameters, thereby constraining further

Model	Wiki ↓	Recall ↑
Gated DeltaNet-E8	**23.61**	**37.39**
Head-wise expand -> Low Rank	23.93	36.71
Head-wise expand -> DPFP	24.41	33.93
Head-wise expand -> LFM	24.59	34.52
w/o Subhead	25.16	29.84
SiLU -> ReLU	23.61	36.48
SiLU -> 1+ELU	24.06	26.86
SiLU -> Identity	23.71	36.29

Table 7: Ablation Study on Gated DeltaNet-E8: The table summarizes the impact of replacing the head-wise expansion, subhead division, and SiLU activation function with alternatives. "Recall" represents the average performance across FDA, SWDE, and SQuAD.

Model	State	Param	FDA ↑	SWDE ↑	SQuAD ↑	Avg ↑
Gated DeltaNet-H2	2×	526M	29.76	39.69	**31.57**	33.67
Gated DeltaNet-E2	2×	413M	21.14	38.34	29.79	29.76
Gated DeltaNet-E8	8×	454M	**38.93**	**43.02**	30.23	**37.39**

Table 8: Comparison with vanilla increasing the number of heads. "H2" indicates that the number of heads is directly increased by 2 times.

scalability. In contrast, our Gated DeltaNet-E8 achieves better performance with fewer parameters than simply increasing the number of heads. We further compare our models with additional RNN variants, including Mamba, RetNet, and DeltaNet in Appendix A.2. This highlights the superiority of our scaled RNN models in terms of recall ability.

6 Conclusion

To address the limitations of RNNs in handling long-context tasks, we propose an efficient method to scale the state size of RNNs, aiming to match the state size of Transformers with a 2k context length. Our results reveal that the recall ability of RNNs scales nearly linearly with state size, indicating that enlarging the state size of RNNs is a promising approach to improving their long-context understanding ability. We hope this work inspires further research into developing more powerful and efficient RNN models, particularly by scaling their state size effectively.

Limitations

Although we propose subhead division to improve training efficiency, our expanded models still train more slowly than the baseline models. However, as discussed in Appendix A.1, this limitation can be mitigated by developing more efficient RNN kernels, as the FLOPs of our scaled RNN models are comparable to those of self-attention. We leave this exploration for future work.

We do not conduct a precise scaling law analysis in this work, instead focusing on a coarse scaling trend. There are several reasons. First, our coarse scaling trend sufficiently demonstrates the effectiveness of increasing RNN state size for long-context understanding tasks, which meets the needs of this paper. Second, evaluating long-context abilities is sensitive to factors such as evaluation methods, training data, and prompts, making it difficult to obtain a smooth scaling curve comparable to those seen in loss-related scaling laws. Third, training RNN LLMs with large state sizes is slow, limiting the number of experiments we can perform. It's hard to solve these issues within the scope of this work. Therefore, we leave a precise scaling law analysis for future work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. 2023. Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes.

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. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. *Preprint*, arXiv:2308.14508.

Yonatan Bisk, Rowan Zellers, Ronan LeBras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. 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), pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

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

XTuner Contributors. 2023. Xtuner: A toolkit for efficiently fine-tuning llm. https://github.com/InternLM/xtuner.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *Preprint*, arXiv:2307.08691.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality. *Preprint*, arXiv:2405.21060.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. 2017. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Preprint*, arXiv:1702.03118.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation.

Alex Graves and Alex Graves. 2012. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45.

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *Preprint*, arXiv:2312.00752.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models. *Preprint*, arXiv:2203.15556.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? *Preprint*, arXiv:2404.06654.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. *Preprint*, arXiv:2006.16236.

Xianhang Li, Zeyu Wang, and Cihang Xie. 2023. An inverse scaling law for clip training. *Preprint*, arXiv:2305.07017.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Preprint*, arXiv:2304.08485.

Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. 2019. 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)*, pages 3047–3056, Minneapolis, Minnesota. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

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

Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Xingjian Du, Teddy Ferdinan, Haowen Hou, Przemysław Kazienko, Kranthi Kiran GV, Jan Kocoń, Bartłomiej Koptyra, Satyapriya Krishna, Ronald McClelland Jr., Jiaju Lin, Niklas Muennighoff, Fares Obeid, Atsushi Saito, Guangyu Song, Haoqin Tu, Cahya Wirawan, Stanisław Woźniak, Ruichong Zhang, Bingchen Zhao, Qihang Zhao, Peng Zhou, Jian Zhu, and Rui-Jie Zhu. 2024. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence. *Preprint*, arXiv:2404.05892.

Subhojeet Pramanik, Esraa Elelimy, Marlos C. Machado, and Adam White. 2024. Agalite: Approxi-

mate gated linear transformers for online reinforcement learning. *Preprint*, arXiv:2310.15719.

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

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. HGRN2: Gated Linear RNNs with State Expansion. *Preprint*, arXiv:2404.07904.

Zhen Qin, Songlin Yang, and Yiran Zhong. 2023. Hierarchically gated recurrent neural network for sequence modeling. *Preprint*, arXiv:2311.04823.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. 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)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740. AAAI Press.

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

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear Transformers Are Secretly Fast Weight Programmers. *Preprint*, arXiv:2102.11174.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. 2021. Linear transformers are secretly fast weight programmers. *Preprint*, arXiv:2102.11174.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (Learn at Test Time): RNNs with Expressive Hidden States.

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

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

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

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

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, Garvit Kulshreshtha, Vartika Singh, Jared Casper, Jan Kautz, Mohammad Shoeybi, and Bryan Catanzaro. 2024. An Empirical Study of Mamba-based Language Models. Publisher: arXiv Version Number: 1.

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

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated Linear Attention Transformers with Hardware-Efficient Training.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. 2025. Parallelizing linear transformers with the delta rule over sequence length. *Preprint*, arXiv:2406.06484.

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

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. 2024. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. *Preprint*, arXiv:2407.20311.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. 2024. Building cooperative embodied agents modularly with large language models. *Preprint*, arXiv:2307.02485.

Jingwei Zuo, Maksim Velikanov, Dhia Eddine Rhaiem, Ilyas Chahed, Younes Belkada, Guillaume Kunsch, and Hakim Hacid. Falcon Mamba: The First Competitive Attention-free 7B Language Model.

A Appendix

A.1 Training Efficiency Discussion

We report the training throughput of our scaled RNN models in Table 9. The results show that the training throughput of Gated DeltaNet-E8 is only 0.21 times that of Gated DeltaNet-E1, indicating a significant decrease in training efficiency as the state size increases. This reduction is primarily attributed to two factors: the RNN kernel and the expansion of queries and keys.

Starting with the RNN kernels, we compare the FLOPs of GLA, Gated DeltaNet, and self-attention in Table 10. As shown, current RNN kernels require significantly fewer FLOPs than self-attention. For example, when $L = 4096$ and $D = 128$, self-attention consumes 10.7 times more FLOPs than GLA and 8 times more FLOPs than Gated DeltaNet. Consequently, our scaled RNN models have similar FLOPs to self-attention and should exhibit comparable training efficiency. However, we identify the bottleneck within the RNN kernel itself. Despite having a similar number of FLOPs as self-attention, the RNN kernels run significantly slower than FlashAttention-2 (Dao, 2023). Specifically, the kernel TFLOPS of Gated DeltaNet and GLA are approximately one-tenth that of FlashAttention-2, indicating substantial room for improving the kernel efficiency of RNNs. FlashAttention-2 benefits from a highly optimized CUDA implementation, whereas the kernels of GLA and Gated DeltaNet rely on less optimized Triton implementations (Tillet et al., 2019). This highlights the potential to enhance RNN kernel efficiency by adopting CUDA-based optimizations.

The second factor is the expansion of queries and keys, which involves multiple operations with fewer FLOPs, such as head-wise matrix multiplication, SiLU activation, convolutional layers, and normalization. Although these operations require relatively few FLOPs, they incur higher memory access costs. Fortunately, as these operations are head-wise parallel, they can be fused into the RNN kernels, saving most memory operations, which should ideally make their overhead negligible.

As a result, the training efficiency of our scaled RNN models has the potential to match that of self-attention. We leave this as future work.

	State	TGS	Ratio
Gated DeltaNet-E1	1×	24462	1.00
Gated DeltaNet-E2	2×	13998	0.57
Gated DeltaNet-E4	4×	9736	0.40
Gated DeltaNet-E8	8×	5172	0.21

Table 9: Training Throughput Comparison of Scaled Gated DeltaNets. "TGS" refers to training tokens per GPU per second.

	FLOPs	Ratio	Kernel TFLOPS
Self-Attention	$2HDL^2$	1×	213
GLA	$6HLD^2$	$\frac{L}{3D}\times$	21
Gated DeltaNet	$8HLD^2$	$\frac{L}{4D}\times$	25

Table 10: FLOPs and Kernel Comparison of Self-Attention, GLA, and Gated DeltaNet. "FLOPs" refers to floating-point operations, while "TFLOPS" denotes the number of tera floating-point operations per second. H, D, and L represent the number of heads, head dimension, and sequence length, respectively. "Kernel TFLOPS" is measured with $B = 4$, $H = 8$, $D = 128$, and $L = 4096$ on an H800 GPU. The self-attention kernel is implemented using FlashAttention-2, whereas the kernels for GLA and Gated DeltaNet are derived from FLA.

A.2 Comparison with Additional RNN Models

We compare our scaled Gated DeltaNet with other RNN models in Table 11. As shown, Gated DeltaNet outperforms other RNN models, including RetNet (Sun et al., 2023b), Mamba (Gu and Dao), and DeltaNet (Yang et al., 2025). Furthermore, our scaled Gated DeltaNet-E8 achieves even greater performance improvements.

A.3 Implementation Details

A.3.1 Architecture

The architectural details of the two basic configurations are summarized in Table 12, serving as the foundation for all our models. Both GLA (Yang et al.) and Gated DeltaNet (Yang et al., 2024) are implemented based on FLA (Yang and Zhang, 2024), incorporating short convolutions and output gates in their RNN layers. Since Transformer++

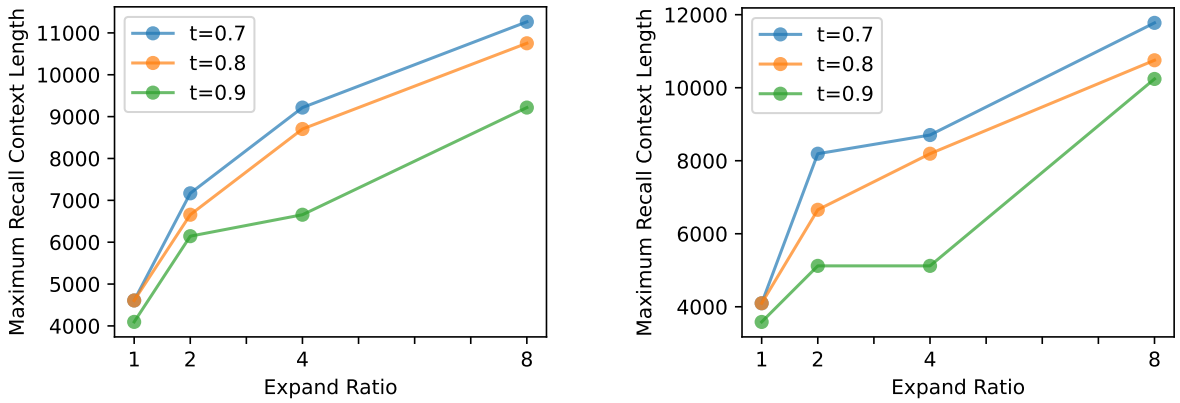

Figure 3: Scaling Trend of Maximum Recall Context Length vs. State Size on S-NIAH. We use numbers and UUIDs as needle values, respectively, on 1.5B Scale

Model	FDA ↑	SWDE ↑	SQuAD ↑	Avg ↑
RetNet	14.3	42.8	34.7	30.6
Mamba	6.2	41.4	35.2	27.6
DeltaNet	17.2	49.5	37.4	34.7
Gated DeltaNet	21.7	53.6	37.8	37.7
Gated DeltaNet-E8	**27.0**	**66.2**	**39.2**	**44.1**

Table 11: Comparison with Additional RNN Models on FDA, SWDE, and SQuAD. Following (Yang et al., 2025), we truncate the context length to 2048 for our models. The results of RetNet, Mamba, and DeltaNet are from (Yang et al., 2025).

Hyperparameter	400M	1.5B
Model Dim	1024	2048
Num Heads	8	8
Head Dim $d_{qk} = d_v$	128	256
Num Layers	24	24
Intermediate Dim	2816	5632

Table 12: Architecture Details

has fewer parameters per layer compared to GLA and Gated DeltaNet, we increase its number of layers to achieve a comparable parameter count.

A.3.2 Training Details

All models are trained on the SlimPajama dataset (Soboleva et al., 2023). The 400M-parameter models are trained on 15B tokens, while the 1.5B-parameter models are trained on 100B tokens. Across all experiments, we employ the AdamW optimizer with a weight decay of 0.1 and gradient clipping of 1.0. We use cosine learning rate scheduling with a peak learning rate of 3e-4. The warm-up steps are set to 1000 and 2000 for the 400M and 1.5B models, respectively. The batch size is set to 0.5M tokens, with a sequence length of 4096 tokens during training. All models use the Mistral tokenizer with a vocabulary size of 32,000.

A.3.3 Evaluation Details

Following (Yang et al., 2024), our common-sense reasoning evaluation includes Wikitext (Merity et al., 2016), LAMBADA (Paperno et al., 2016), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020), ARC-easy (ARC-e) and ARC-challenge (ARC-c) (Clark et al., 2018), SIQA (Sap et al., 2019), and BoolQ (Clark et al., 2019). Real-world retrieval tasks include FDA (Arora et al., 2023), SWDE (Lockard et al., 2019), and SQuAD (Rajpurkar et al., 2018). The above tasks are evaluated using lm-eval (Gao et al., 2024). We also evaluate on 14 English tasks in LongBench (Bai et al., 2024), including NarrativeQA, QasperQA, MultiField QA, HotpotQA, 2WikiMulti QA, Musique, GovReport, QMSum, MultiNews, TRec, Trivia QA, SamSum, LCC, and RepoBench-P.

A.3.4 Implementation

Our experiments are primarily conducted using publicly available libraries, including PyTorch, XTuner (Contributors, 2023), and FLA (Yang and Zhang, 2024), with training performed on H100 GPUs. This paper has been refined with the assistance of a chatbot.

A.4 State Size of Transformer++ with Sliding Window Attention

There are two scenarios regarding the state size of Transformer++ with sliding window attention. First, when the context length is smaller than the window size, the state size is the same as that

of the original Transformer++, i.e., $L \times H \times (d_{qk} + d_v)/2$. Second, when the window size is smaller than the context length, as is the case in LongBench where the context length can be up to 64k tokens—much larger than the window size of 2048 in Transformer++-2k—the state size of Transformer++-2k with sliding window attention is approximately $W \times H \times (d_{qk} + d_v)$, where W denotes the window size.

A.5 Ethical Considerations

Our work focuses on improving the long-context understanding ability of RNNs. The potential applications of our work include enhancing the performance of RNN models in long-context tasks. We believe that our research can contribute to the development of more powerful and efficient RNN models, which can benefit society by improving the performance of AI systems in various applications. Our work is not associated with any ethical concerns.