ROBUST GATED LINEAR RNNS VIA A SINGLE LINEAR TIME-INVARIANT RECURRENT LAYER

Anonymous authors
Paper under double-blind review

ABSTRACT

Gated linear Recurrent Neural Networks (RNNs) incorporating input-dependent gate mechanisms have shown promising performance on sequence modeling. While input-dependent gating allows their dynamics to adapt flexibly to each input, it also makes the dynamics sensitive to task-irrelevant inputs. To characterize and address this issue, we directly analyze the gate mechanism and show that the gate has a *non-constant rate of change*: When the gate value is close to one, a decrease in the pre-activation causes a larger change in the gate than an increase of the same magnitude. This asymmetry can cause a downward shift in the gate values, making it difficult to preserve information over long sequences. Moreover, a larger learning rate can amplify this downward shift since the update of the pre-activation scales with the learning rate. This insight leads to our proposal: We replace a single input-dependent recurrent layer with a *linear time-invariant* (LTI) *recurrent layer* in a gated linear RNN while leaving the rest of the architecture unchanged, which we call the *single-LTI hybrid*. Since the state-transition matrix of the LTI layer is independent of the input, this replacement mitigates the effect of task-irrelevant inputs and large learning rates. Interestingly, replacing only a single layer is sufficient because residual connections commonly used in gated linear RNNs provide an identity path, allowing the features extracted by the LTI layer to be preserved through the subsequent layers. Our experiments show that the proposed single-LTI hybrid improves trainability on long-sequence modeling tasks, particularly with large learning rates, reduces undesirable sensitivity to inputs, and achieves competitive or better performance on language modeling.

1 INTRODUCTION

Transformer-based architectures have achieved strong performance on sequence modeling. However, their computational and memory costs grow quadratically with sequence length. Linear recurrent neural networks (linear RNNs) have emerged as efficient alternatives, and recent advances have incorporated *input-dependent gating mechanisms*. These gated linear RNNs, such as DeltaNet (Schlag et al., 2021; Yang et al., 2024b), Gated Linear Attention (Yang et al., 2024a), Mamba (Gu & Dao, 2024), Mamba-2 (Dao & Gu, 2024), and Gated DeltaNet (Yang et al., 2025), have shown strong empirical performance on long-sequence modeling tasks.

Prior work has studied the dynamics of the input-dependent gate in Mamba (Huang et al., 2025; Yu & Erichson, 2025). Huang et al. (2025) showed that the input-dependent gate can adaptively change its values depending on context, which helps preserve information. While the input-dependent gate mechanism allows the dynamics to adapt flexibly to each input, it also makes the dynamics sensitive to task-irrelevant inputs (Yu & Erichson, 2025). Yu & Erichson (2025) focused on the input-dependent write operation and showed that large inputs and long sequences can make training unstable. However, how parameter updates during training affect the input-dependent gate has not been thoroughly explored.

In this paper, we investigate how the sensitivity of the gate affects training dynamics and the ability to preserve information over long sequences. We characterize the gate mechanism and show that the gate has a *non-constant rate of change*: When the gate value is close to one, a decrease in the pre-activation causes a larger change in the gate than an increase of the same magnitude. This asymmetry can cause a downward shift in the gate values. Since capturing long-term dependencies requires

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

near-one gate values, this shift makes it difficult to preserve information over long sequences. We further show that a larger learning rate amplifies this effect. Since the update of the pre-activation scales with the learning rate, a larger learning rate increases the magnitude of these perturbations, which can further reduce the average gate value near one.

To mitigate the issue arising from the gate sensitivity, we propose a hybrid gated linear RNN, called *single-LTI hybrid*, in which a single input-dependent recurrent layer is replaced with a linear time-invariant (LTI) recurrent layer. This modification is applied to only a single layer, while the rest of the architecture remains unchanged. An LTI layer mitigates the effect of task-irrelevant inputs because its state-transition matrix does not depend on the input. Furthermore, an LTI layer mitigates the effect of gate-value changes amplified by a large learning rate since the state-transition matrix is not computed from the input through a gate. Notably, replacing a single layer with an LTI layer is sufficient because gated linear RNNs generally adopt residual connections. Since a residual connection provides an identity path at each layer, the feature extracted by the LTI layer is carried to all subsequent layers without passing through their input-dependent gates. Experimental results demonstrate that the proposed single-LTI hybrid can be trained with larger learning rates and generalizes better, achieving competitive or better performance on language modeling.

We summarize our contributions as follows:

- We show that the response of the gate to input changes has a non-constant rate of change, which leads to a downward shift in the gate values. This downward shift can cause training failure and make it difficult to preserve information over long sequences. We further show that a large learning rate amplifies this effect.
- We propose a simple hybrid approach for gated linear RNNs that replaces a single input-dependent recurrent layer with a linear time-invariant (LTI) recurrent layer.
- We empirically demonstrate that our single-LTI hybrid improves trainability on long-sequence modeling tasks with large learning rates, reduces sensitivity to inputs, and achieves competitive performance on language modeling.

2 PRELIMINARIES

Linear recurrent neural networks (linear RNNs) are sequence models whose recurrent update is linear with respect to the previous hidden state. Given an input sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$ with each $\mathbf{x}_t \in \mathbb{R}^d$, the recurrence of a linear RNN is defined as

$$\begin{aligned} \mathbf{H}_t &= \mathbf{A}_t \mathbf{H}_{t-1} + \mathbf{B}_t, \\ \mathbf{y}_t &= \mathbf{H}_t^\top \mathbf{q}_t, \end{aligned} \tag{1}$$

where $\mathbf{H}_0 \in \mathbb{R}^{n \times d}$ is the initial hidden state, $\mathbf{A}_t \in \mathbb{R}^{n \times n}$ is the state transition matrix, $\mathbf{B}_t \in \mathbb{R}^{n \times d}$ is the input-dependent matrix, and $\mathbf{q}_t \in \mathbb{R}^n$ is the readout vector. These $\mathbf{A}_t$, $\mathbf{B}_t$, and $\mathbf{q}_t$ depend on the current input $\mathbf{x}_t$.

We consider the input-dependent gate mechanisms used in models such as Mamba, Mamba-2, and Gated DeltaNet. In this paper, we focus on analyzing the decay factors of the state transition matrix $\mathbf{A}_t$, which control how long information is preserved over long sequences. Given a pre-activation value z_t, these decay factors can be written in the following unified form:

$$f_{\alpha_i}(z_t) = \exp[-\alpha_i \, \mathrm{softplus}(-z_t)], \qquad \alpha_i > 0, \tag{2}$$

where α_i is a learnable parameter. The pre-activation is defined by a linear projection of the time-dependent representation as follows:

$$z_t = w^\top x_t + b. \tag{3}$$

For convenience of analysis, we use the opposite sign convention for z_t in Equation (2) from the original parameterizations of Mamba and Gated DeltaNet, so that a larger value corresponds to a larger gate value. For Mamba, α_i is the decay rate of the i-th state dimension in the discretization. For Mamba-2, $\alpha_i = \alpha$ is the scalar decay rate shared across state dimension. For Gated DeltaNet, $f_{\alpha_i}(z_t)$ corresponds to its scalar decay factor and $\alpha_i = \alpha$ is the scalar decay rate shared across state dimension. In the following section, we omit the subscript i and use α for α_i, since the analysis applies to each state dimension independently.

3 THEORETICAL ANALYSIS

In this section, we analyze how input-dependent gate mechanisms affect training dynamics and information preservation. We first investigate how the pre-activation is updated during training and how the learning rate affects these updates. We then analyze how the gate value responds to changes in the pre-activation. Finally, we analyze how these effects affect information preservation.

3.1 EFFECT OF PRE-ACTIVATION AND LEARNING RATE ON TRAINING DYNAMICS

We consider a single SGD step with learning rate η. Let $\mathcal{L}$ denote the loss function and $z_t = w^\top x_t + b$ be the pre-activation as in Equation (3). The gradients with respect to the parameters of the pre-activation are given as follows using the chain rule:

$$\frac{\partial \mathcal{L}}{\partial w} = \sum_{s=1}^{T} \frac{\partial \mathcal{L}}{\partial z_s} x_s, \quad \frac{\partial \mathcal{L}}{\partial b} = \sum_{s=1}^{T} \frac{\partial \mathcal{L}}{\partial z_s}. \tag{4}$$

After one SGD step, the shared parameters are updated as

$$w^+ = w - \eta \sum_{s=1}^{T} \frac{\partial \mathcal{L}}{\partial z_s} x_s, \quad b^+ = b - \eta \sum_{s=1}^{T} \frac{\partial \mathcal{L}}{\partial z_s}. \tag{5}$$

Evaluating the updated pre-activation at time t gives

$$z_t^+ = (w^+)^\top x_t + b^+ = z_t - \eta \sum_{s=1}^{T} \frac{\partial \mathcal{L}}{\partial z_s} \left(x_s^\top x_t + 1 \right). \tag{6}$$

Therefore, the change $\Delta z_t = z_t^+ - z_t$ is obtained as

$$\Delta z_t = -\eta \sum_{s=1}^{T} \frac{\partial \mathcal{L}}{\partial z_s} \left(x_s^\top x_t + 1 \right). \tag{7}$$

Thus, the pre-activation update depends on the similarity $x_s^\top x_t$ between representations at different time steps. Furthermore, a larger learning rate amplifies these input-dependent changes.

3.2 ANALYZING THE EFFECT OF GATE FOR CHANGING DYNAMICS OF PRE-ACTIVATIONS

Next, we show that the gate has a non-constant rate of change in the following theorem.

Theorem 1 (Asymmetric sensitivity on the gate dynamics)**.** *We consider the following function:*

$$f_\alpha(z) = \exp[-\alpha \operatorname{softplus}(-z)]. \tag{8}$$

For $\alpha > 0$, this function is monotonically increasing and concave under $z > \log \alpha$.

Thus, in the region $z > \log \alpha$, the amount of increase of $f_\alpha(z)$ becomes smaller as z increases. When the gate value is close to one, it still increases monotonically with the pre-activation, but the amount of increase becomes smaller as the pre-activation grows. As a result, a decrease in the pre-activation causes a larger change in the gate than an increase of the same magnitude. This asymmetry can cause a downward shift in the gate values, which is not desirable as gate values close to one are necessary to capture long-term dependencies.

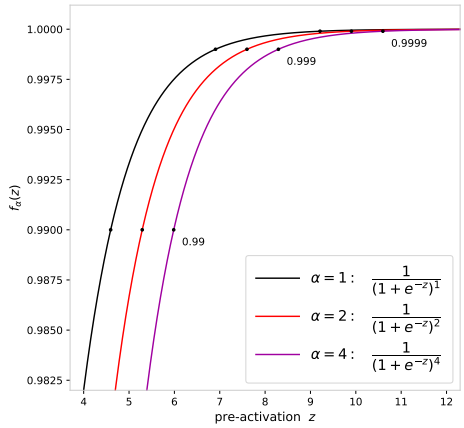

Figure 1: The gate dynamics for equal perturbation with varying α.

To illustrate the impact on the gate values, we consider the case $\alpha = 1$, 2, and 4. If the current gate value is $f_\alpha(z_t) = 0.999$ and a perturbation $\epsilon = \log 10$, then the same amount of shift of pre-activation moves approximately from 0.999 to 0.9999 in the increase direction, but from 0.999 to 0.99 in the reduce direction. Thus, as shown in Figure 1, although the two perturbations have the same amount of change in pre-activation space, the reduction of the gate is larger than the increase of the gate when the gate value is close to one.

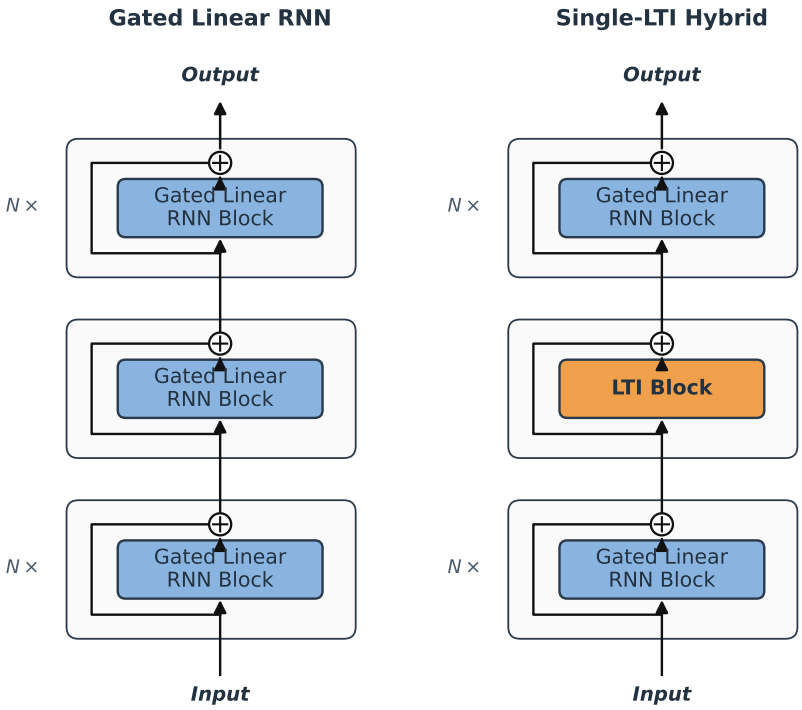

Figure 2: Proposed architecture.

3.3 Effect of Information Preservation

To isolate the effect of this decay gate on information preservation, we consider the recurrence of the decay factor as follows:

$$h_t = f_\alpha(z_t)h_{t-1} + u_t, \tag{9}$$

where u_t denotes the input-dependent write term. Assuming $h_0 = 0$ and unrolling the recurrence, we obtain

$$h_t = \sum_{s=1}^{t} \left(\prod_{r=s+1}^{t} f_\alpha(z_r) \right) u_s, \tag{10}$$

where the empty product for $s = t$ is defined as one.

This equation shows that information written at time s is retained through the cumulative product of the subsequent decay gates. Since the gate is repeatedly applied over T time steps, even small reductions in the gate values can accumulate multiplicatively. For example, we consider a gate value of 0.999 over 1000 steps. Suppose that symmetric perturbations in the pre-activation move the gate to 0.99 for 500 steps and to 0.9999 for the 500 steps. Then, $0.999^{1000} \approx 3.68 \times 10^{-1}, 0.99^{500} \times 0.9999^{500} \approx 6.25 \times 10^{-3}$. Thus, even when positive and negative pre-activation perturbations have similar magnitudes, their asymmetric effects on gates close to one can make it more difficult to preserve previous information over long sequences.

4 Hybrid Gated Linear RNNs with a Single LTI Recurrent Layer

To mitigate the issue arising from input sensitivity of the gates, we introduce a hybrid gated linear RNN, called *single-LTI hybrid*, in which a single input-dependent recurrent layer is replaced with an LTI recurrent layer. Our method modifies only a single layer while leaving the remaining architecture unchanged.

The transition matrix of the LTI layer remains shared across all time steps after parameter updates because the LTI layer does not have the input-dependent gate mechanism. Since the transition matrix

is the same at each time step, the LTI layer avoids the downward shift caused by input sensitivity, thereby preventing a larger learning rate from amplifying this effect.

We first introduce the single-LTI hybrid. We then analyze how the LTI layer mitigates the issue caused by gate sensitivity. Finally, we explain why replacing only one layer can be sufficient.

The single-LTI hybrid replaces the k-th input-dependent layer of a gated linear RNN with an LTI layer. We use a standard discrete-time LTI state-space layer defined as follows:

$$\mathbf{h}_t^{(k)} = \mathbf{A}\mathbf{h}_{t-1}^{(k)} + \mathbf{B}\mathbf{u}_t^{(k)}, \qquad \mathbf{y}_t^{(k)} = \mathbf{C}\mathbf{h}_t^{(k)} \tag{11}$$

where $\mathbf{u}_t^{(k)} \in \mathbb{R}^d$ and $\mathbf{y}_t^{(k)} \in \mathbb{R}^d$ are the input and output of the k-th layer, $\mathbf{h}_t^{(k)} \in \mathbb{R}^n$ is the hidden state, and $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times d}$, and $\mathbf{C} \in \mathbb{R}^{d\times n}$ are the state, input, and output matrices, which are shared across all time steps. Following Gu et al. (2022a), we omit the term $\mathbf{D}\mathbf{u}_t^{(k)}$ from the output, since it can be viewed as a skip connection. In our experiments, we place the LTI layer at the middle of the stack, $k = \lceil (L+1)/2 \rceil$. We examine the effect of the placement k in Section 5.5.

The state-transition matrix of an LTI layer, such as S4 (Gu et al., 2022a) and S4D (Gu et al., 2022b), does not depend on the current input. The state-transition matrix is shared across all time steps; that is, $\mathbf{A}_t = \mathbf{A}$ for all t. After one SGD update, the shared transition matrix becomes

$$\mathbf{A}^+ = \mathbf{A} - \eta \nabla_{\mathbf{A}} \mathcal{L}. \tag{12}$$

Since the updated matrix is shared across all time steps, the updated transition matrix also satisfies $\mathbf{A}_t^+ = \mathbf{A}^+$ for all t. Therefore, the downward shift does not occur in an LTI Layer by the input-dependent gate mechanism and cannot be amplified by a larger learning rate.

Gated linear RNNs generally adopt residual blocks. Under a pre-normalization architecture, the ℓ-th block is written as follows:

$$\begin{aligned} \mathbf{u}_t^{(\ell)} &= \mathcal{N}\left(\mathbf{o}_t^{(\ell-1)}\right), \\ \mathbf{y}_t^{(\ell)} &= \mathcal{F}_\ell\left(\mathbf{u}_t^{(\ell)}\right), \\ \mathbf{o}_t^{(\ell)} &= \mathbf{o}_t^{(\ell-1)} + \mathbf{y}_t^{(\ell)}, \qquad \ell = 1, \dots, L, \end{aligned} \tag{13}$$

where $\mathbf{o}_t^{(0)} = \mathbf{x}_t$, L is the number of layers, $\mathcal{F}_\ell$ is either a gated linear RNN layer or an LTI layer, and $\mathcal{N}$ denotes normalization. In the single-LTI hybrid, $\mathcal{F}_k = \mathcal{F}_{\mathrm{LTI},k}$ is the LTI layer defined in Equation (11), while all other layers remain unchanged.

Then, the output of the k-th block is

$$\mathbf{o}_t^{(k)} = \mathbf{o}_t^{(k-1)} + \mathcal{F}_{\mathrm{LTI},k}\left(\mathcal{N}\left(\mathbf{o}_t^{(k-1)}\right)\right). \tag{14}$$

Recursively applying Equation (13) from the k-th block to the final layer, we have

$$\mathbf{o}_t^{(L)} = \mathbf{o}_t^{(k-1)} + \mathcal{F}_{\mathrm{LTI},k}\left(\mathcal{N}\left(\mathbf{o}_t^{(k-1)}\right)\right) + \sum_{\ell=k+1}^{L} \mathcal{F}_\ell\left(\mathcal{N}\left(\mathbf{o}_t^{(\ell-1)}\right)\right). \tag{15}$$

Consequently, the feature produced by the LTI layer has a direct additive path through all subsequent layers.

In contrast, without residual connections, we obtain the recurrent update as follows:

$$\mathbf{o}_t^{(\ell)} = \mathcal{F}_\ell\left(\mathcal{N}\left(\mathbf{o}_t^{(\ell-1)}\right)\right), \qquad \ell = 1, \dots, L. \tag{16}$$

Recursively substituting this update gives

$$\mathbf{o}_t^{(L)} = (\mathcal{F}_L \circ \mathcal{N}) \circ \cdots \circ (\mathcal{F}_{k+1} \circ \mathcal{N})\left(\mathcal{F}_{\mathrm{LTI},k}\left(\mathcal{N}\left(\mathbf{o}_t^{(k-1)}\right)\right)\right). \tag{17}$$

Thus, the feature produced by the LTI layer must be processed by the input-dependent gates of every subsequent layer without an identity path.

This contrast explains why replacing a single layer with an LTI layer can be sufficient in a residual architecture. We empirically examine this mechanism through a residual-connection ablation in Section 5.4.

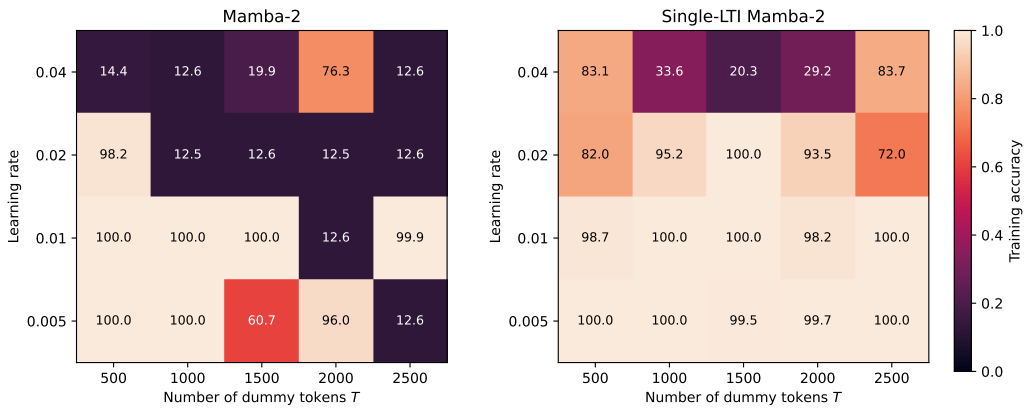

Figure 3: Training accuracy across different learning rate and sequence length on the copying task for Mamba-2 (left) and single-LTI hybrid Mamba-2 (right).

5 Experiments

We empirically validate our theoretical analysis and evaluate the proposed single-LTI hybrid. We first examine the effects of learning rate and sequence length on trainability using a synthetic task. Next, we investigate the effects of learning rate on trainability and generalization on Pathfinder which is a task from the Long Range Arena (LRA) benchmark. Finally, we evaluate the proposed model on language modeling. Throughout these experiments, we use Mamba-2 and Gated DeltaNet as the base gated linear RNNs and use the LTI layer using S4D. Implementation details and the full experimental setup are provided in Appendix C.

5.1 Trainability across learning rates and sequence lengths

We first investigate the effects of learning rate and sequence length on trainability using the copying task. The copying task is a standard benchmark that evaluates the ability to memorize a sequence of tokens and reproduce it after T dummy tokens. Instead of using a fixed dummy token, as in the standard copying task, we sample each dummy token at random different from the memorization tokens, so that the model receives task-irrelevant inputs. Further details are provided in Appendix C.2.

Figures 3 and 4 illustrate the training accuracy for the time lag from 500 to 2500 and from learning rate from 5×10^{-3} to 4×10^{-2} with 10,000 training steps. For short time step lags and small learning rates, models can learn the task. In contrast, longer time step lags and larger learning rates make training more difficult. Our proposed hybrid models can reduce the region in which training fails and achieve high training accuracy over wider region.

5.2 Effect of learning rate on generalization

The results in Figures 3 and 4 raise the question of whether a smaller learning rate is sufficient to avoid the training failure. We therefore investigate how reducing the learning rate affects trainability and generalization on the Pathfinder task.

Figure 5 shows the mean accuracies and standard deviation over three random seeds on the Pathfinder task. As shown in Figure 5, Mamba-2 trained with the larger learning rate of 5×10^{-4} fails to learn the task, with its test accuracy remaining at about 0.5 accuracy. Reducing the learning rate to 1×10^{-4} enables Mamba-2 to learn the task. However, it can cause overfitting since the test accuracy does not reach near optimal accuracy. In contrast, the single-LTI hybrid successfully learns at the larger learning rate, achieving near optimal test accuracy.

We observe a similar trend for Gated DeltaNet. At the larger learning rate of 1×10^{-3}, the Gated DeltaNet baseline remains close to 50% for most of training. Although reducing the learning rate to 5×10^{-4} enables the baseline to learn the task, but its test accuracy varies substantially across runs. By contrast, the single-LTI hybrid trained at 1×10^{-3} achieves near optimal test accuracy.

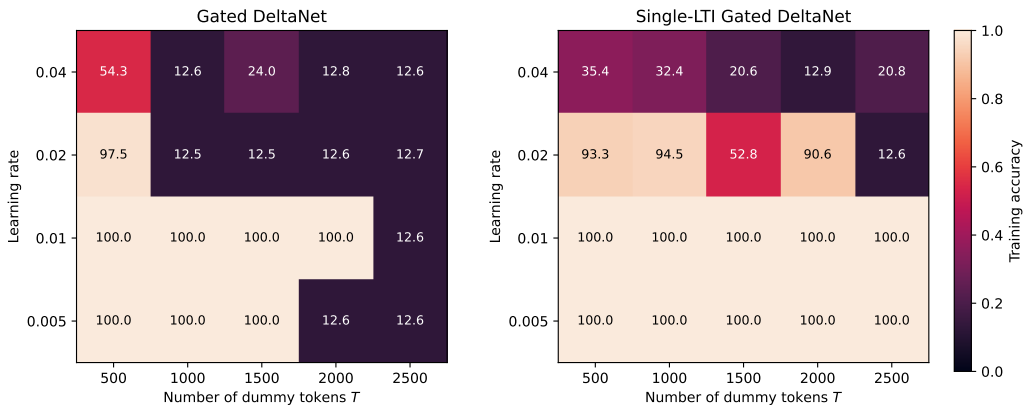

Figure 4: Training accuracy across different learning rate and sequence length on the copying task for Gated DeltaNet (left) and single-LTI hybrid Gated DeltaNet (right).

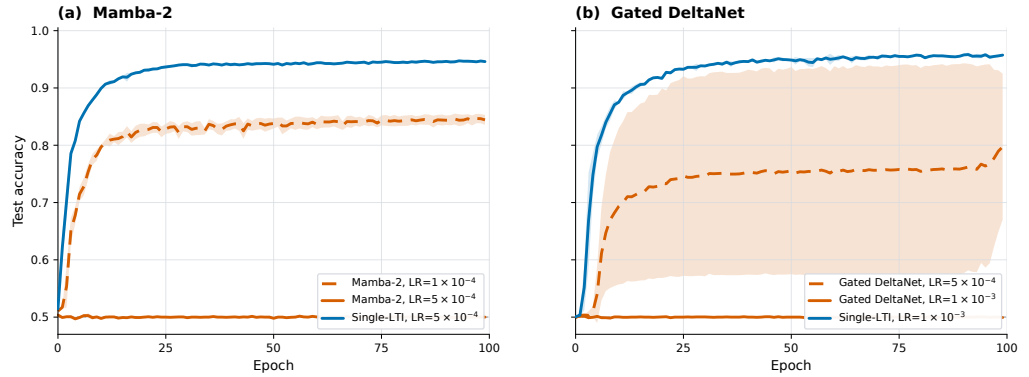

Figure 5: Performance of Mamba-2 (left) and Gated DeltaNet (right) on the Pathfinder task.

These improvements are consistent with our analysis that inserting a single input-independent S4D layer helps preserve information over long sequences, even when input-dependent gate mechanisms are difficult to optimize at a larger learning rate.

5.3 Language Modeling

We evaluate the single-LTI hybrid against pure gated linear RNNs on language modeling using PG-19. All models have 12 layers and approximately 110M parameters, and are trained on approximately 3B tokens with context lengths from 4K to 32K. Further experimental details are provided in the Appendix.

As shown in Table 1, the single-LTI hybrid achieves lower (better) perplexity than the baseline in all configurations except Mamba-2 at 16K. At 32K, the Mamba-2 baseline degrades sharply compared to 16K, whereas the single-LTI hybrid degrades much less. This observation is consistent with the copying results in Section 5.1 and with our analysis that input-dependent gates can cause training failure at long sequence lengths, whereas the LTI layer mitigates this effect by providing an input-independent gate. Our results show that replacing a single input-dependent recurrent layer with an LTI layer generally improves language-modeling performance and training stability, without requiring LTI layers at every depth (Section 4).

Table 1: Perplexity on PG-19 under the same training budget. Lower is better.

Backbone	Model	4K	8K	16K	32K	Avg.
Gated DeltaNet	Base	17.60	17.04	16.92	17.46	17.26
	Single-LTI	**16.65**	**15.81**	**16.65**	**16.99**	**16.52**
Mamba-2	Base	18.22	18.06	**15.73**	20.45	18.11
	Single-LTI	**17.19**	**16.99**	16.03	**19.21**	**17.36**

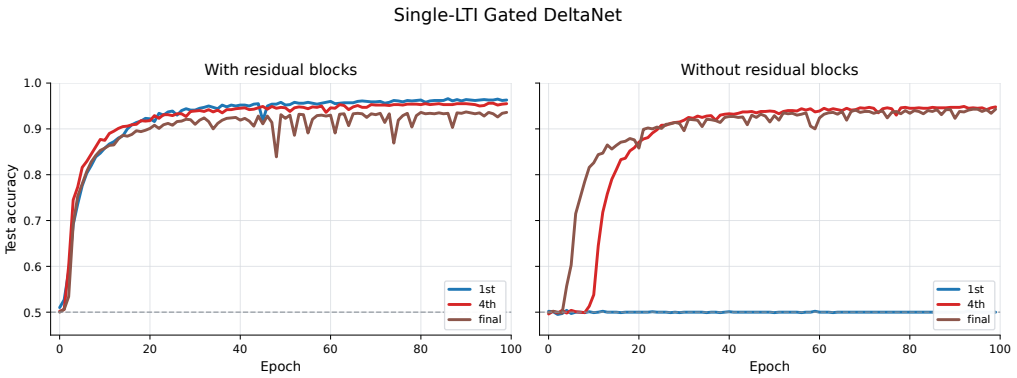

Figure 6: Performance of the single-LTI hybrid Gated DeltaNet with residual connections (left) and without Residual connections (right) on the Pathfinder task.

5.4 Ablation study on residual connections

In this section, we examine how residual connections affect the performance of the single LTI hybrid model. Figure 6 shows the test accuracy on the Pathfinder task. As shown in the left panel of Figure 6, the single-LTI hybrid Gated DeltaNet with residual connections reaches near-optimal test accuracy for all three placements of the LTI layer. In contrast, as shown in the right panel of Figure 6 , removing the residual connections makes the performance more sensitive to the placement of the LTI layer. With the LTI layer at the first position, the model fails to learn the task and remains at about 0.5 accuracy, whereas the fourth and final placements still learn the task.

This result is consistent with our findings that residual connections preserve the features produced by S4D as discussed in Section 4. Without a residual connection, the S4D features must pass through all subsequent Gated DeltaNet layer, making it more difficult for the model to capture long-term dependencies.

5.5 Sensitivity to the placement of the LTI layer

To examine how the placement of the LTI layer affects performance, we train the single-LTI hybrid of Gated DeltaNet on PG-19 with the LTI layer replaced at different positions. We train all models using the same training budget and evaluate them at context lengths ranging from 4K to 32K.

Figure 7 shows test perplexity at each context length. As shown in Figure 7, all placements other than the first layer improve on the Gated DeltaNet baseline at every context length. The first-layer replacement improves on the baseline at 4K, underperforms it at 8K, and is comparable at 16K and 32K. One possible explanation for this instability of the first-layer replacement is that, with the first layer replaced, every gate is unable to use the raw input directly. When the first Gated DeltaNet layer is replaced with an S4D layer, the raw input is carried to the next layer only in combination with the output of the S4D layer, even with residual connections. Thus, subsequent Gated DeltaNet layers compute their gates from this mixed representation rather than from the raw input itself. This difference may explain why the first-layer placement is less consistently effective.

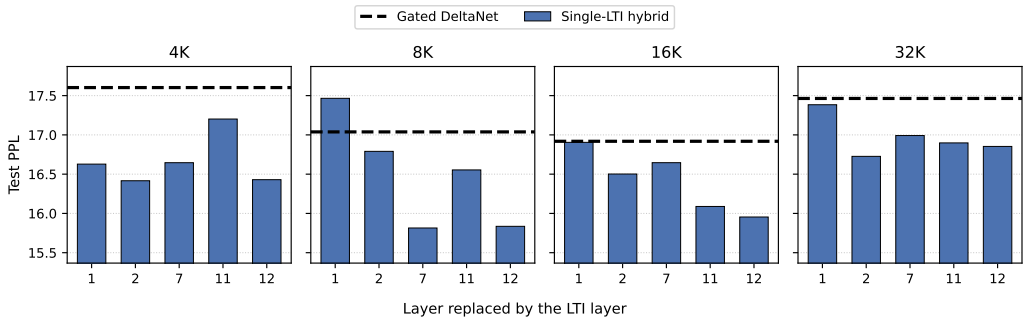

Figure 7: Placement of the LTI layer in the 12-layer stack with varying context length in Gated DeltaNet on PG-19.

Table 2: Sensitivity to the number of LTI layers at 8K context length. Lower PPL is better.

# LTI layers	Replaced layers	PPL ↓
Gated DeltaNet (baseline)	–	17.04
1	7	<u>15.81</u>
3	3, 7, 11	**15.77**
5	3, 5, 7, 9, 11	16.41
7	2, 3, 5, 7, 9, 11, 12	16.41
11	2–12	18.92
12 (all)	1–12	21.00

5.6 Sensitivity analysis of increasing the number of LTI layers

To investigate how the number of LTI layers affects model performance, we conduct language modeling experiments with Gated DeltaNet while varying the number of input-dependent recurrent layers replaced by LTI layers. Table 2 shows the test perplexity at an 8K context length. The three-LTI variant improves perplexity over the baseline to a similar degree as the single-LTI hybrid. Increasing the number of LTI layers to five or seven retains a smaller improvement over the baseline, and performance falls below the baseline only when eleven or all twelve layers are replaced. These results indicate that a single LTI layer is sufficient to capture most of the benefit, whereas replacing most of the input-dependent layers can reduce the model's expressive capacity.

6 Conclusion

In this work, we studied how the dynamics of input-dependent gates in gated linear RNNs affect trainability and information preservation. We showed that the gate has a *non-constant rate of change* with respect to its pre-activation. When the gate value is close to one, a decrease in the pre-activation causes a larger change in the gate than an equal-magnitude increase. This asymmetry can cause a downward shift in the gate values, making it difficult to preserve information over long sequences. We further showed that a larger learning rate can amplify this effect.

Based on these insights, we proposed the *single-LTI hybrid*, a simple approach that replaces a single input-dependent recurrent layer of a gated linear RNN with a *linear time-invariant* (LTI) *recurrent layer* while leaving the rest of the architecture unchanged. Replacing only a single layer can be sufficient because residual connections carry the features extracted by the LTI layer to all subsequent layers. Experimental results showed that the single-LTI hybrid improves trainability and generalization on long-sequence modeling tasks, and achieves competitive or better performance on language modeling.

AI USE STATEMENT

In this work, we used generative AI tools for polishing the text and formatting latex code and figures, modifying experiment code and searching information. We have not used generative AI tools for any important component of the research methodology or design experiments. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary Evolution Recurrent Neural Networks. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

Aaron Blakeman, Aarti Basant, Abhinav Khattar, Adithya Renduchintala, Akhiad Bercovich, Aleksander Ficek, Alexis Bjorlin, Ali Taghibakhshi, Amala Sanjay Deshmukh, Ameya Sunil Mahabaleshwarkar, et al. Nemotron-H: A Family of Accurate and Efficient Hybrid Mamba-Transformer Models, 2025.

John Cooper, Ilias Diakonikolas, Mingchen Ma, and Frederic Sala. Expressivity-Efficiency Tradeoffs for Hybrid Sequence Models. In *Proceedings of The 43rd International Conference on Machine Learning*, 2026.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*. PMLR, 2024.

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

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. In *Conference on Language Modeling*, 2024.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently Modeling Long Sequences with Structured State Spaces. In *International Conference on Learning Representations*, 2022a.

Albert Gu, Ankit Gupta, Karan Goel, and Christopher Ré. On the Parameterization and Initialization of Diagonal State Space Models. In *Advances in Neural Information Processing Systems*, 2022b.

Ningyuan Huang, Miguel Sarabia, Abhinav Moudgil, Pau Rodriguez, Luca Zappella, and Federico Danieli. Understanding Input Selectivity in Mamba: Impact on Approximation Power, Memorization, and Associative Recall Capacity. In *Proceedings of The 42rd International Conference on Machine Learning*, 2025.

Aakash Lahoti, Kevin Y. Li, Berlin Chen, Caitlin Wang, Aviv Bick, J. Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved Sequence Modeling using State Space Principles. In *International Conference on Learning Representations*, 2026.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A Hybrid Transformer-Mamba Language Model, 2024.

Antonio Orvieto, Samuel L. Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting Recurrent Neural Networks for Long Sequences. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Teddy Ferdinan, Haowen Hou, Przemysław Kazienko, Kranthi Kiran GV, Satyapriya Krishna, Ronald McClelland Jr., Niklas Muennighoff, Fares Obeid, Atsushi Saito, Guangyu Song, Haoqin Tu, Ruichong Zhang, Bingchen Zhao, Qihang Zhao, Jian Zhu, and Rui-Jie Zhu. Eagle and Finch: RWKV with Matrix-Valued States and Dynamic Recurrence. In *Conference on Language Modeling*, 2024.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, Nathan Wilce, Johan S. Wind, Tianyi Wu, Daniel Wuttke, and Christian Zhou-Zheng. RWKV-7 "Goose" with Expressive Dynamic State Evolution. In *Conference on Language Modeling*, 2025.

Zhen Qin, Songlin Yang, and Yiran Zhong. Hierarchically Gated Recurrent Neural Network for Sequence Modeling. In *Advances in Neural Information Processing Systems*, 2023.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. HGRN2: Gated Linear RNNs with State Expansion. In *Conference on Language Modeling*, 2024.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear Transformers Are Secretly Fast Weight Programmers. In *Proceedings of The 38rd International Conference on Machine Learning*, 2021.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated Linear Attention Transformers with Hardware-Efficient Training. In *Proceedings of the 41st International Conference on Machine Learning*, 2024a.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing Linear Transformers with the Delta Rule over Sequence Length. In *Advances in Neural Information Processing Systems*, 2024b.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated Delta Networks: Improving Mamba2 with Delta Rule. In *International Conference on Learning Representations*, 2025.

Annan Yu and N. Benjamin Erichson. Block-Biased Mamba for Long-Range Sequence Processing. In *Advances in Neural Information Processing Systems*, 2025.

A Related work

A.1 Linear recurrent models

Linear recurrent models have attracted attention as an efficient alternative to Transformers. Recent linear recurrent architectures such as S4 (Gu et al., 2022a), S4D (Gu et al., 2022b), and LRU (Orvieto et al., 2023) have demonstrated improvements on long-sequence modeling tasks. More recently, several linear RNNs have incorporated input-dependent mechanisms to improve expressivity, such as DeltaNet (Schlag et al., 2021; Yang et al., 2024b), Gated Linear Attention (Yang et al., 2024a), Griffin (De et al., 2024), Mamba (Gu & Dao, 2024), Mamba-2 (Dao & Gu, 2024), Mamba-3 (Lahoti et al., 2026), HGRN (Qin et al., 2023; 2024), Gated DeltaNet (Yang et al., 2025), and RWKV (Peng et al., 2024; 2025). While input-dependent gate mechanisms improve expressivity, the gate mechanisms make training unstable in Mamba for large inputs and long sequences (Yu & Erichson, 2025). Yu & Erichson (2025) analyzed this instability under the assumption that the decay factor does not depend on the input. In this work, we investigate how input-dependent gating affects long-sequence modeling when the decay factor is input-dependent.

A.2 Hybrid Models

Recent hybrid architectures combine Transformers with state-space models (SSMs) to retain efficient computation while compensating for the limited expressivity of pure SSMs. Jamba (Lieber et al., 2024) uses one attention layer for every seven Mamba layers, and Nemotron-H (Blakeman et al., 2025) uses attention layers in only approximately 8% of its layers. These hybrid models achieve strong performance with only a small number of complementary layers, showing empirically that a complementary layer need not be replaced at every layer. Recent work has studied the expressivity and efficiency of hybrid models combining Transformers with SSMs (Cooper et al., 2026). However, why a small number of complementary layers can compensate for the limitations of pure models remains insufficiently understood. We study whether a single LTI layer can compensate for the limitations of input-dependent recurrent layers in gated linear RNNs.

B Proof of Theorem 1

Proof. From Equation (2) and the definition of the softplus function,

$$f_\alpha(z) = (1 + e^{-z})^{-\alpha}. \tag{18}$$

Differentiating with respect to z, we have

$$f'_\alpha(z) = -\alpha(1 + e^{-z})^{-\alpha-1}(-e^{-z}) \tag{19}$$

$$= \frac{\alpha e^{-z}}{1 + e^{-z}}(1 + e^{-z})^{-\alpha} \tag{20}$$

$$= \frac{\alpha e^{-z}}{1 + e^{-z}} f_\alpha(z) > 0. \tag{21}$$

Therefore, $f_\alpha(z)$ is monotonically increasing when $\alpha > 0$.

Since $f_\alpha(z)$ is twice differentiable, it suffices to show that $f''_\alpha(z) < 0$. For the second derivative, differentiating Equation (21) gives

$$f''_\alpha(z) = \alpha \left[-\frac{e^{-z}}{(1 + e^{-z})^2} f_\alpha(z) + \frac{e^{-z}}{1 + e^{-z}} f'_\alpha(z) \right]. \tag{22}$$

Substituting Equation (21),

$$f''_\alpha(z) = \alpha \left[-\frac{e^{-z}}{(1 + e^{-z})^2} f_\alpha(z) + \frac{\alpha e^{-2z}}{(1 + e^{-z})^2} f_\alpha(z) \right] \tag{23}$$

$$= \alpha f_\alpha(z) \left[-\frac{e^{-z}}{(1 + e^{-z})^2} + \frac{\alpha e^{-2z}}{(1 + e^{-z})^2} \right] \tag{24}$$

$$= \alpha f_\alpha(z) \frac{e^{-z}}{(1 + e^{-z})^2} \left(\alpha e^{-z} - 1 \right). \tag{25}$$

If $z > \log \alpha$, we have

$$f''_\alpha(z) < 0. \tag{26}$$

Therefore, $f_\alpha(z)$ is concave under $z > \log \alpha$. □

C Experiments

C.1 Implementations

Our implementation is based on the official S4 repository (Gu et al., 2022a),[1] with Mamba-2 and Gated DeltaNet implementations adapted from FLA.[2].

C.2 Experimental settings

Copying task. For the copying task (Arjovsky et al., 2016), the input sequence consists of M memorization tokens, T dummy tokens, and M output tokens. In our experiment, we use randomly sampled dummy tokens instead of the fixed dummy token used in the standard copying task. The model must memorize the first M tokens and reproduce them at the final M positions. We use a vocabulary of $K = 18$ tokens $\{0, 1, \ldots, K-1\}$. The memorization tokens are sampled independently and uniformly from $\{1, \ldots, K/2-1\}$, the dummy tokens are sampled independently and uniformly from $\{K/2, \ldots, K-2\}$, and the final M input tokens are set to the output token $K-1$. Thus, the memorization and dummy token sets are disjoint. Token 0 is not used in the input sequences. We set $M = 10$ and vary the time lag $T \in \{500, 1000, 1500, 2000, 2500\}$, resulting in a total sequence length of $2M + T$.

All models have three recurrent blocks with hidden dimension and embedding dimension are set to 64. We train models with the AdamW optimizer with batch size 64, $\Delta_{\max} = 0.1$, $\Delta_{\min} = 0.001$, and a constant learning rate without warmup or gradient clipping for 10,000 optimizer steps, sweeping the learning rate over $\{0.005, 0.01, 0.02, 0.04\}$. The single-LTI hybrid replaces the second block with an S4D layer ($k = 2$).

[1] https://github.com/state-spaces/s4

[2] https://github.com/fla-org/flash-linear-attention

Pathfinder task. Pathfinder is an LRA task in which a 32×32 image is flattened into a sequence of length 1024. This task requires predicting whether two dots are connected by a path. We use the `curv_contour_length_9` and `curv_contour_length_14` difficulty levels.

All models have six recurrent layers with hidden dimension 256 and dropout 0.0. We train each model for 100 epochs using AdamW with betas $(0.9, 0.999)$, weight decay 0.01, $\Delta_{\min} = 0.001$, and batch size 32. The learning rate follows a cosine schedule with $200{,}000$ warmup steps. We evaluate peak learning rates of 1×10^{-4} and 5×10^{-4} for Mamba-2 and 5×10^{-4} and 1×10^{-3} for Gated DeltaNet. The single-LTI hybrids are trained using the same learning rate of each backbone and replace the fourth block with an S4D layer ($k = 4$).

Language modeling. PG-19 is a widely used benchmark for long-context language modeling. We train and evaluate our models on PG-19 using context lengths ranging from 4K to 32K tokens. We tokenize the dataset using the GPT-2 tokenizer and append an EOS token to each document. Validation and test windows do not cross document boundaries.

All models have 12 recurrent blocks, a hidden dimension of 768, no dropout, and approximately 110M trainable parameters. Each mixer is followed by a two-layer FFN with an expansion factor of 2 and a GELU activation. Both the mixer and FFN use pre-norm residual connections. We use LayerNorm for the Gated DeltaNet-based models and RMSNorm for the Mamba-2 based models. We train each model for 5,848 optimizer steps, corresponding to approximately one epoch, using AdamW with betas $(0.9, 0.999)$, weight decay 0.1, and gradient clipping at 1.0. The learning rate follows a cosine schedule with $1,000$ warmup steps. We use peak learning rates of 1×10^{-2} then decayed with a cosine schedule to 3×10^{-5}. For the S4D layer, we use $d_{\text{state}} = 64$, two output channels, and one shared SSM. We set $\Delta_{\max} = 0.1$ and use $\Delta_{\min} = 10^{-3}, 10^{-4}, 3 \times 10^{-5}$, and 10^{-5} for context lengths of 4K, 8K, 16K, and 32K, respectively. The single-LTI hybrid replaces the seventh block with an S4D layer ($k = 7$). Across all experiments, the effective training batch size is fixed at approximately 0.5M tokens.