LAWCAT: Efficient Distillation from Quadratic to Linear Attention with Convolution across Tokens for Long Context Modeling

Zeyu Liu[1], Souvik Kundu[2], Lianghao Jiang[1], Anni Li[1],
Srikanth Ronanki[3], Sravan Bodapati[3], Gourav Datta[4], Peter A. Beerel[1]
[1]University of Southern California, USA [2]Intel Labs, San Diego, USA
[3]Amazon AGI, USA, [4]Case Western Reserve University, USA
Correspondence: {liuzeyu, pabeerel}@usc.edu

Abstract

Although transformer architectures have achieved state-of-the-art performance across diverse domains, their quadratic computational complexity with respect to sequence length remains a significant bottleneck, particularly for latency-sensitive long-context applications. While recent linear-complexity alternatives are increasingly powerful, effectively training them from scratch is still resource-intensive. To overcome these limitations, we propose LAWCAT (Linear Attention with Convolution Across Time), a novel linearization framework designed to efficiently transfer the capabilities of pre-trained transformers into a performant linear attention architecture. LAWCAT integrates causal Conv1D layers to enhance local dependency modeling and employs normalized gated linear attention to improve generalization across varying context lengths. Our comprehensive evaluations demonstrate that, distilling Mistral-7B with only 1K-length sequences yields over 90% passkey retrieval accuracy up to 22K tokens, significantly extending its effective context window. Similarly, Llama3.2-1B LAWCAT variant achieves competitive performance on S-NIAH 1&2&3 tasks (1K-8K context length) and BABILong benchmark (QA2&QA3, 0K-16K context length), requiring less than 0.1% pre-training tokens compared with pre-training models. Furthermore, LAWCAT exhibits faster prefill speeds than FlashAttention-2 for sequences exceeding 8K tokens. LAWCAT thus provides an efficient pathway to high-performance, long-context linear models suitable for edge deployment, reducing reliance on extensive long-sequence training data and computational resources.

1 Introduction

The widespread adoption and significant success of transformer-based architectures have driven remarkable advancements across various natural language processing (NLP) (DeepSeek-AI, 2025; Grattafiori et al., 2024; Team, 2024), reasoning (Chen et al., 2025), and computer vision (CV) tasks (Yao et al., 2024; Ramachandran et al., 2025). Transformers leverage self-attention mechanisms to effectively model long-range dependencies, achieving state-of-the-art (SOTA) performance on diverse benchmarks (Fourrier et al., 2024). However, their quadratic computational complexity with respect to sequence length remains a critical bottleneck, limiting their applicability to latency-sensitive edge based applications that are constrained in terms of training data and cost.

To address this limitation, modern recurrent models have emerged as efficient alternatives, approximating or reformulating attention mechanisms to achieve linear complexity. Recent studies, such as Mamba-2 (Dao and Gu, 2024), Gated linear attention (GLA) (Yang et al., 2023), and Hierarchically gated linear RNN2 (HGRN2) (Qin et al., 2024), demonstrate the feasibility of state update mechanisms that substantially reduce computational costs while maintaining competitive performance. Despite these advances, training these models from scratch often requires substantial computational resources and extensive data, hindering rapid development and experimentation.

On the other hand, knowledge distillation, a powerful technique for transferring knowledge from a large, pretrained teacher model to a smaller and more efficient student model, offers an effective strategy to alleviate these challenges. Recently, some research like MambaInLlama (Wang et al., 2024) and LoLCATs (Zhang et al., 2025) directly distill transformer models into recurrent architectures. However, MambaInLlama still requires extensive training data (∼20B tokens) and retains 50% self-attention layers to achieve strong performance up to 38K tokens, and LoLCATs exhibits limited generalization to sequences significantly longer than those seen during distillation.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 20865–20881
November 4-9, 2025

To address this problem, we introduce LAWCAT (Linear Attention with Convolution Across Time), a novel distillation framework that efficiently transfers and enhances the rich representational capabilities of pre-trained transformer models into modern linear attention models. LAWCAT is particularly well-suited for edge applications that require long-context modeling under tight computational constraints with limited training data. LAWCAT integrates a depth-separable convolution module, consisting of a depth-wise convolution layer followed by a linear layer (similar to MobileNet (Howard, 2017), but we use causal Conv1D across tokens), and a gated linear attention with normalization to effectively bridge the gap between quadratic and linear attention mechanisms. The main contributions of this paper are summarized as follows:

- We proposed the integration of a causal Conv1D layer to enhance the capacity of linear attention mechanisms to model local dependencies and introduce the normalization for the gated linear attention to improve generalization to long contexts.
- We present LAWCAT, a framework leveraging these components to efficiently distill high-performing linear attention models. Specifically, we show that our Mistral v0.1 7B LAWCAT variant achieves over 90% accuracy on the passkey retrieval task with context lengths up to 22K tokens, significantly exceeding the original model's 8K limit. In addition, our Llama3.2 1B model LAWCAT variant achieves comparable performance on more complex retrieval tasks (NIAH1-3) and reasoning tasks (QA2&3 from BABILong benchmark) across various sequence lengths.
- We demonstrate the efficiency of our distillation approach, requiring significantly less data (less than 0.1% of the original pretraining tokens, and less than 1/3 of the original sequence length) to adapt a pretrained transformer into a high-performing linear attention model. Additionally, we show that the resulting LAWCAT model exhibits faster prefill-stage processing speeds compared to FlashAttention-2 for input sequences longer than 8K tokens.

2 Background and Related Work

Self-Attention The majority of the powerful LLM models like Llama 3 (Grattafiori et al., 2024), Mistral (Jiang et al., 2023), and Phi 4 (Abdin et al., 2024) adopt the multi-head scaled dot-product softmax attention (SDPA). Given a query $\mathbf{q}_t \in \mathbb{R}^{d_q}$, keys $\{\mathbf{k}_i\}_{i=1}^t \in \mathbb{R}^{d_k}$, and values $\{\mathbf{v}_i\}_{i=1}^t \in \mathbb{R}^{d_v}$ ($t \leq N$), a single-head SDPA computes output $\mathbf{o}_t$ as (omitting the scaling factor $\sqrt{d}$ for simplicity):

$$\mathbf{o}_t = \sum_{i=1}^{t} \frac{\exp\left(\mathbf{q}_t \mathbf{k}_i^T\right)}{\sum_{j=1}^{t} \exp\left(\mathbf{q}_t \mathbf{k}_j^T\right)} \mathbf{v}_i \tag{1}$$

This can be equivalently expressed as $\mathbf{o}_t = \sum_{i=1}^{t} w_{ti} \mathbf{v}_i$, where w_{ti} denotes the attention weight assigned to value $\mathbf{v}_i$ at output position t. The softmax in the numerator/denominator ensures context-dependent normalization – each weight w_{ti} depends not only on the query–key similarity $\mathbf{q}_t \mathbf{k}_i^T$ but also on all other similarities $\{\mathbf{q}_t \mathbf{k}_j^T\}_{j \leq t}$. This coupling gives softmax attention a powerful ability to focus on a few relevant tokens while normalizing out less relevant ones, but also yields the $O(N^2)$ computation complexity with respect to the sequence length N.

Kernel-Based Linear Attention To mitigate this quadratic computation complexity, many linear attention methods aim to find a kernel function whose structure approximates attention scores through inner products of transformed query and key representations. Assuming the kernel κ is positive semi-definite (PSD), then there exists a feature mapping ϕ satisfying: $\kappa(\mathbf{q}, \mathbf{k}) = \phi(\mathbf{q})\phi(\mathbf{k})^T$ Thus, the attention output at position t can be computed as:

$$\begin{aligned} \mathbf{o}_t &= \sum_{i=1}^{t} \frac{\phi(\mathbf{q}_t)\phi(\mathbf{k}_i)^T}{\sum_{j=1}^{t} \phi(\mathbf{q}_t)\phi(\mathbf{k}_j)^T} \mathbf{v}_i \\ &= \frac{\phi(\mathbf{q}_t) \sum_{i=1}^{t} \phi(\mathbf{k}_i)^T \mathbf{v}_i}{\phi(\mathbf{q}_t) \sum_{j=1}^{t} \phi(\mathbf{k}_j)^T} \end{aligned} \tag{2}$$

Similar to the Eq. 1, we can simplify is as $\mathbf{o}_t = \sum_{i=1}^{t} \tilde{w}_{ti} \mathbf{v}_i$, and $\tilde{w}_{ti}$ can be regarded as the attention weights of the linear attention. This is analogous to softmax attention but uses $\phi(\mathbf{q})\phi(\mathbf{k})^T$ to approxiamte $\exp(\mathbf{q}\mathbf{k}^T)$ for all queries/keys, at least in expectation. If ϕ can perfectly realize $\exp$ as an inner product in a higher-dimensional feature space, then the linear attention exactly equals softmax attention. In practice, ϕ might be a simple nonlinear function such as $1 + \mathrm{ELU}$ (Katharopoulos et al., 2020), or a positive random feature map as in the Performer (Choromanski et al., 2020) that aims to fit the exponential. Such linear attention schemes run in $O(N)$ time, but still suffer a noticeable performance gap between transformer models.

Recent Modern Linear Attention Recent linear attention variants dispense with softmax normalization and utilize identity feature maps $\phi(x) = x$, yielding the formulation $\mathbf{o}_t = \mathbf{q}_t\mathbf{S}_t$, where the KV state $\mathbf{o}_t = \sum_{i=1}^{t} \mathbf{k}_i^T\mathbf{v}_i$ is computed recursively as $\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{k}_t^T\mathbf{v}_t$. Instead of directly approximating softmax function, these models focus on enriching the state update process itself. Most models propose state transitions that take the form

$$\mathbf{S}_t = g(\mathbf{S}_{t-1}, \mathbf{x}_t) + f(\mathbf{k}_t^T\mathbf{v}_t, \mathbf{x}_t) \tag{3}$$

The function $g(,)$ often implements input-dependent gating or forgetting, controlling the influence of the historical state $\mathbf{S}_{t-1}$ (Yang et al., 2023; Peng et al., 2024). The function $f(,)$ defines how the current key-value outer product contributes to the state update (Dao and Gu, 2024; Yang et al., 2024; Azizi et al., 2025; Ye et al., 2025). Both functions can accept the input $\mathbf{x}_t$ to make them input-dependent. These approaches leverage the efficiency of linear recurrence while introducing sophisticated mechanisms for state evolution and information flow, resulting in increasingly powerful sequence modeling capabilities. However, pre-training demands significant computational resources hindering widespread adoption.

Linearization of Attention To reduce the training costs, recent efforts try to leverage pre-trained transformer-based LLMs to obtain computationally efficient models with linear complexity. Hedgehog (Zhang et al., 2024) introduced a learned linear feature mapping (ϕ) to approximate softmax attention. Expanding on this approach, LoLCATs (Zhang et al., 2025) integrated sliding-window attention to improve performance further. However, these methods still struggle to maintain strong performance on tasks involving significantly longer contexts than seen during distillation.

Another approach, MambaInLlama (Wang et al., 2024), applied progressive distillation along with supervised fine-tuning (Kim and Rush, 2016; Ro et al., 2025) and directed preference optimization (Rafailov et al., 2023) to distill Mamba models from Llama3 models. Despite achieving strong performance at extended lengths up to 38K tokens, it still retains 50% of self-attention and requires substantial training data (∼20B tokens). SUPRA (Mercat et al., 2024) pursued a different strategy, modifying LLaMA3 models into architectures similar to RetNet (Sun et al., 2023), but still required extensive additional training (∼100B tokens) to extend effective context handling to 32K tokens. Thus, SUPRA does not fully resolve the substantial overhead associated with long-context model training.

3 Algorithmic Motivation

A fundamental distinction between softmax attention and its linear approximations lies in the normalization mechanism and the role of the exponential function. Softmax attention weights, $\{w_{ti} \propto \exp(\mathbf{q}_t\mathbf{k}_i^T)\}$, are normalized by a sum over all keys for a given query. This inherently creates a competitive dynamic: the exp function sharply amplifies scores for keys highly similar to the query relative to others, effectively focusing attention mass on the most relevant tokens (Vaswani et al., 2023). Although self-attention is renowned for modeling global dependencies, much of its effectiveness also stems from capturing local context (e.g., adjacent words or neighboring pixels). The softmax mechanism excels here as its exponential weighting amplifies locally coherent patterns (like increasing the relevance of "New York" when queried with "York"), effectively capturing correlations abundant in natural sequences. Empirical evidence (Han et al., 2024), including the disproportionate performance drop observed when masking local versus random tokens, further demonstrates that softmax models leverage local context more effectively than typical linear variants.

In contrast, linear attention often employs feature maps ϕ applied independently to queries and keys, yielding attention weights $\tilde{w}_{ti} \propto \phi(\mathbf{q}_t)\phi(\mathbf{k}_i)^T$. The normalization term $\sum_j \phi(\mathbf{q}_t)\phi(\mathbf{k}_j)^T$ simply aggregates transformed key features without the strong competitive effect of the softmax denominator. This can lead to limitations such as attention dilution or oversmoothing, where attention distributions become less peaked (Qin et al., 2022). More critically, this formulation struggles to replicate the strong local modeling capabilities inherent in softmax attention (Han et al., 2024). This weaker local modeling is a key drawback of naive linear attention. Meanwhile, the potentially low-rank nature of the $\phi(\mathbf{Q})\phi(\mathbf{K})^T$ kernel can make it insensitive to crucial local variations and contextual nuances. Two distinct queries q and q', even if originating from different local contexts, might yield similar $\phi(q) \approx \phi(q')$ and thus nearly identical attention patterns, a failure mode less likely in softmax due to its context-sensitive normalization.

4 Proposed Method

4.1 Integration of Causal Conv1D Layer

To address this deficiency in capturing local structure, we propose to integrate Conv1D layers to make the function ϕ more context-sensitive. In particular, as shown in Fig. 1, our LAWCAT adds a causal depthwise convolution layer with kernel size of $r+1$ (In practice, we set kernel size as 4) for the query and key, respectively, to introduce an inductive bias for locality into the attention approximation. Since this convolution layer is causal, it only uses the tokens before the current token to calculate the convolution. Similar to the KV cache mechanism, we can cache the query and key from the previous tokens, but the size of the cache is limited by the kernel size. After the Conv1D layer, similar to the LoLCATs (Zhang et al., 2025), we use a linear layer with nonlinearity to project the query and key but we make the query and key to share the same linear projection, which makes the approximation align with the Eq. 2 which use the same function ϕ for the query and value.

A causal Conv1D with kernel size of $r+1$ will compute each transformed query $\tilde{\mathbf{q}}_t = f_\theta\big([\mathbf{q}_{t-r}, ..., \mathbf{q}_t,]\big)$ as a function of a local window of the original queries (and similarly $\tilde{\mathbf{k}}_i$ from neighboring keys). The convolution is linear (a depthwise 1D convolution is essentially a weighted moving average of the inputs in a local window) and we do not include nonlinearities between this Conv1D layer and the subsequent linear layer. In the simplest case, let $\tilde{\mathbf{q}}_t = \sum_{\delta=0}^{r} w_\delta \, \mathbf{q}_{t-\delta}$ with some learnable weights $w_{0\ldots r}$ (and similarly $\tilde{\mathbf{k}}_i$). This operation mixes local token information into the queries and keys before they interact. Intuitively, the convolution can be seen as smoothing and enriching the representations of Q and K with their neighbors' features. As a result, the kernel function ϕ (now applied on $\tilde{\mathbf{q}}_t$ and $\tilde{\mathbf{k}}_i$) is no longer a purely pointwise function – it has implicit awareness of nearby tokens.

The similarity between $\tilde{\mathbf{q}}_t$ and $\tilde{\mathbf{k}}_i$ in the conv-augmented space includes contributions from keys in the neighborhood of i as well. In effect, a key $\mathbf{k}_i$ will receive a high weight $\tilde{w}_{ti}$ not only if it aligns with the query, but also if its previous neighbors $\mathbf{k}_{i-1}, \mathbf{k}_{i-2}, \ldots$ align with the query. This property can help the linear attention mimic what softmax does: softmax will give a group of similar, adjacent keys a collectively high weight if the query matches that group. The Conv1D introduces an implicit group-aware effect as neighboring keys support each other's relevance. Conversely, if a key $\mathbf{k}_i$ is an outlier among irrelevant neighbors, $\tilde{\mathbf{k}}_i$ may be diluted by its neighbors (lowering $\phi(\tilde{\mathbf{k}}_i)$'s dot with $\phi(\tilde{\mathbf{q}}_t)$), which is analogous to how an isolated token might not stand out as strongly under softmax normalization if its similarity is not much larger than others. In summary, the convolution provides a localized smoothing and sharpening of attention scores: smoothing because it aggregates local information mitigating random fluctuations, and sharpening because a coherent local pattern (several similar tokens in a window) will amplify each token's effective feature.

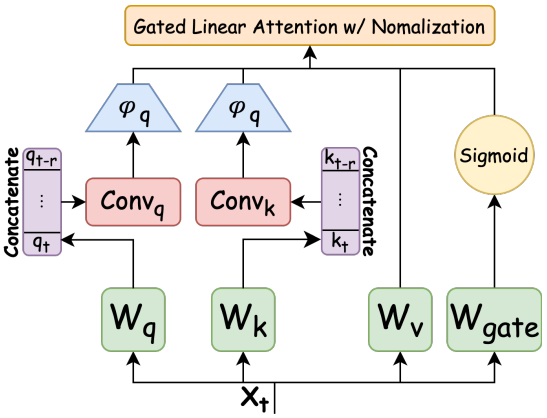

Figure 1: The overall structure of LAWCAT. We use Casual Conv1D with a kernel size of $r+1$ for visualization, and use $\mathbf{x}_t$ to represent the t-th token of input $\mathbf{x}$.

The advantages of the Conv1D layer over the sliding window attention On the other hand, LoLCATs (Zhang et al., 2025) proposed to use a sliding window attention to improve the overall performance. Specifically, for the tokens within the window, they will calculate the softmax attention score, and for the tokens outside the window, they will use the linear attention score, and use the weighted sum as the final attention score. However, this strategy does not attempt to improve the linear attention component's handling of local interactions, so retrieving information located far beyond the window relies solely on the capabilities of the linear attention component. If the linear attention part fails to accurately capture these long-range dependencies or lacks the necessary precision, the model's ability to retrieve distant information will be significantly impaired, leading to performance degradation as the distance to the target information increases.

In contrast, our LAWCAT produces one cohesive attention output that naturally balances local and global information which gets rid of the difficulties

of balancing the linear and softmax attention. Additionally, by blending neighboring tokens' features, LAWCAT can reduce the risk of query confusion (non-injectivity) and increase the effective rank of the attention, leading to a better approximation of full attention (Fan et al., 2024). Moreover, LAWCAT encodes a prior that nearby tokens are related, which can improve generalization on sequences with local structure helping combat the tendency of full attention to overfit noise.

4.2 GLA with Normalization

Another key design choice in our architecture is the adoption of Gated Linear Attention (GLA), which adds additional linear layers followed by a sigmoid activation to enable input-dependent gating. Unlike Yang et al. (2023), who omit the normalization term in Eq. 2, we find that retaining this normalization is critical for both training stability and final performance, particularly in the context of distillation from a standard Transformer model.

Let $\mathbf{G}_t$ denote the forget gate and $\mathbf{S}_t$ represent the key-value (KV) state of the linear attention mechanism at time step t. For notational simplicity, we denote the projected query and key of the i-th token, after Conv1D and linear projection, as $\dot{\mathbf{q}}_i$ and $\dot{\mathbf{k}}_i$, respectively. Then, the KV state update equation in GLA can be described as:

$$\mathbf{S}_t = \mathbf{G}_t \odot \mathbf{S}_{t-1} + \dot{\mathbf{k}}_t^T \mathbf{v}_t \tag{4}$$

where $\odot$ is the Hadamard product. Letting the $\mathbf{S}_0 = 0$, we can rewrite the above equation as:

$$\mathbf{S}_t = \sum_{i=1}^{t} \left[\left(\bigodot_{z=i+1}^{t} \mathbf{G}_z \right) \odot \left(\dot{\mathbf{k}}_i^T \mathbf{v}_i \right) \right] \tag{5}$$

and we use $\mathbf{J}_{i,t}$ to denote the consecutive Hadamard products when $i+1 \leq t$:

$$\mathbf{J}_{i,t} = \bigodot_{z=i+1}^{t} \mathbf{G}_z = \mathbf{G}_t \odot \mathbf{G}_{t-1} \odot \cdots \odot \mathbf{G}_{i+1} \tag{6}$$

otherwise, we set $\mathbf{J}_{i,t} = \mathbf{1}$. Finally, we get the expression of the KV state as Eq. 7 and the attention output as Eq. 8:

$$\mathbf{S}_t = \sum_{i=1}^{t} \left[\mathbf{J}_{i,t} \odot (\dot{\mathbf{k}}_i^T \mathbf{v}_i) \right] \tag{7}$$

$$\mathbf{o}_t = \frac{\dot{\mathbf{q}}_t \sum_{i=1}^{t} \left[\mathbf{J}_{i,t} \odot (\dot{\mathbf{k}}_i^T \mathbf{v}_i) \right]}{\dot{\mathbf{q}}_t \sum_{j=1}^{t} \left[\mathbf{J}_{j,t} \odot (\dot{\mathbf{k}}_j^T \mathbf{1}) \right]_{mean(1)}} \tag{8}$$

Note, since $\mathbf{G}_i \in \mathbb{R}^{d_k \times d_v}$, we also have $\mathbf{J}_{i,t} \in \mathbb{R}^{d_k \times d_v}$, but $\mathbf{k}_i \in \mathbb{R}^{1 \times d_k}$, it makes it not straightforward to calculate the normalization of the GLA. To address this, we let the $\mathbf{k}_i^T$ times $\mathbf{1}_{1 \times d_v}$ first, which actually repeat the $\mathbf{k}_i^T$ for d_v times along the second dimension, then follow the normal GLA operation. Before the multiplication with $\mathbf{q}_t$, we take the mean value of the $\mathbf{J}_{i+1,t} \odot (\mathbf{k}_i^T \mathbf{1})$ along the second dimension to reduce the shape from $d_k \times d_v$ to $d_k \times 1$. In this way, we normalize the GLA with a similar definition as in Eq. 2.

5 Experimental Results

5.1 Setup

Our training followed the two-stage methodology of LoLCATs (Zhang et al., 2025): an initial distillation stage employing only layer-wise MSE loss, followed by LoRA fine-tuning (Hu et al., 2022). In total, for Passkey Retrieval (Mohtashami and Jaggi, 2023), S-NIAH (Hsieh et al., 2024), and BABILong (Kuratov et al., 2024) tasks, we use 93M, 31M, and 103M tokens, respectively, with maximum input lengths of 1162, 1223, and 1300 tokens. More details regarding training datasets and training configurations are provided in Appendix A.1 and A.2. Standard pre-trained models typically leverage over 100B tokens to attain competitive performance. Compared with them, our approach requires less than 0.1% of such pre-training tokens.

5.2 Results on Passkey Retrieval Task

As shown in Fig. 2, the pre-trained LLaMA3 8B (Grattafiori et al., 2024) maintains perfect accuracy from 1K to 8K tokens, but drops to zero beyond 9K, exceeding its training context length. In contrast, our method extends the effective context window: even when distilled and fine-tuned solely on 1K-length sequences, the resulting model retains competitive performance up to 12K tokens. Similarly, for the pre-trained LLaMA3.2 1B model, which performs reliably up to 32K tokens, our approach preserves strong retrieval accuracy up to 7K tokens. Compared to the LoLCATs method, whose effectiveness is limited to 1K for LLaMA3 8B and 1–2K for LLaMA3.2 1B, our method demonstrates substantially superior preservation of long-context retrieval capabilities.

We also evaluate our approach on the Mistral v0.1 and v0.3 7B models (Jiang et al., 2023), which fail to achieve 100% accuracy even within 8K, likely due to their limited optimization for retrieval

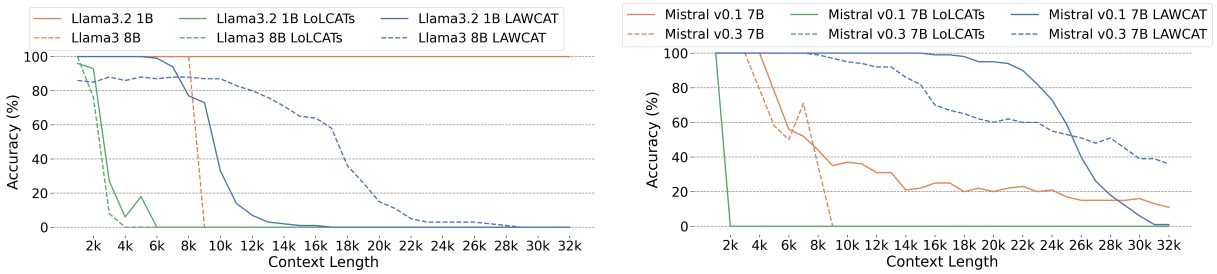

Figure 2: The comparison between the pre-trained transformer model and our converted linear attention model.

	S-NIAH-1 (pass-key retrieval)				S-NIAH-2 (number in haystack)				S-NIAH-3 (uuid in haystack)		
	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K
Llama3.2 1B Instruct	100	100	100	100	100	100	96	100	100	100	100
DeltaNet-1.3B	97.4	96.8	99.0	**98.8**	98.4	45.6	18.6	14.4	85.2	47.0	22.4
Mamba2-1.3B	99.2	98.8	65.4	30.4	99.4	98.8	56.2	17.0	64.4	47.6	4.6
Gated DeltaNet-1.3B	98.4	88.4	91.4	91.8	100	**99.8**	**92.2**	29.6	**86.6**	84.2	27.6
Distilled Model	Pre-trained Model: Llama3.2-1B-Instruct										
LoLCATs	100	84	0	0	84	44	0	0	72	24	0
Ours	**100**	**100**	**100**	80	**100**	96	88	**48**	56	44	24
	Pre-trained Model: Llama3-8B-Instruct										
LoLCATs	100	4	0	0	92	4	0	0	84	24	0
Ours	100	100	96	80	100	92	84	32	80	**88**	**60**
	Pre-trained Model: Mistral v0.1 7B										
LoLCATs	100	100	100	100	100	96	64	0	72	72	16
	Pre-trained Model: Mistral v0.3 7B										
LoLCATs	100	100	100	100	100	88	60	16	68	24	12

Table 1: Performance comparison across different models on S-NIAH 1, 2, and 3 tasks.

tasks. Despite this, our linearized Mistral v0.3 7B model retains good accuracy up to 15K tokens, and the linearized Mistral v0.1 7B model extends this further, achieving over 90% accuracy up to 22K tokens. In contrast, the LoLCATs-distilled Mistral models reach peak performance at only 1K tokens and rapidly degrade to zero beyond 2K tokens. These results highlight the robustness and generalizability of our method in preserving long-context capabilities across various model architectures.

5.3 Results on S-NIAH Benchmark

Beyond the Passkey Retrieval task, we evaluate our method on the more challenging S-NIAH 1-3 task from the RULER benchmark (Hsieh et al., 2024). For a comprehensive comparison, we include results from several SOTA pre-trained recurrent models, DeltaNet (Yang et al., 2024), Mamba2 (Dao and Gu, 2024), and Gated DeltaNet (Yang et al., 2025). All pre-trained models are trained on 100B tokens, and the training length is 4K. And the results are sourced from Yang et al. (2025).

As shown in Table 1, on the S-NIAH 1 task, our model achieves 100% accuracy from 1K to 4K tokens, surpassing all listed SOTA recurrent models. Even at 8K, it remains competitive, despite being trained on sequences more than 3× shorter (1223 vs. 4000 tokens) and using only 0.03% of the data (31M vs. 100B tokens). On the S-NIAH 2 task, LAWCAT achieves comparable performance up to 4K and maintains a strong 48% accuracy at 8K—substantially outperforming other models and demonstrating robust generalization to longer contexts. While performance on the more complex S-NIAH 3 task lags slightly behind others, we hypothesize this is due to the limited diversity in our training data. LoLCATs only maintain good results around 1K for all S-NIAH tasks, which is consistent with the results on the Passkey Retrieval task.

The results related to the Llama3 8B and Mistral series models demonstrate similar trends; our model demonstrates superior robustness to increasing input lengths, with notably smaller performance drop compared to LoLCATs and other SOTA pre-trained recurrent models.

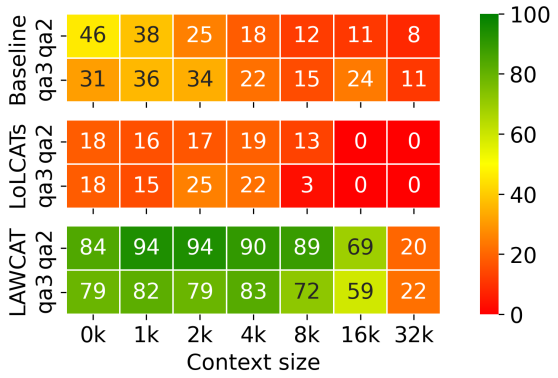

Figure 3: Comparison of accuracy on QA2&3 from BABILong benchmark between Llama 3.2 1B (top), LoLCATs (middle), and LAWCAT (bottom).

5.4 Results on BABILong benchmark

Besides retrieval capabilities, we assessed LAWCAT's performance on complex reasoning tasks using the BABILong benchmark (Kuratov et al., 2024), specifically focusing on multi-hop reasoning tasks, QA2 (two supporting facts) and QA3 (three supporting facts). The results, presented in Figure 3, demonstrate a significant advantage for our distilled and fine-tuned LAWCAT variant compared with the original Llama3.2 1B model. Across the evaluated context lengths (0K to 32K tokens), LAWCAT consistently maintains accuracy levels more than double those of the baseline Llama3.2 1B model for both QA2 and QA3 tasks.

This notable enhancement in multi-hop reasoning performance underscores the robustness of the LAWCAT framework, particularly highlighting its capability to overcome inherent limitations observed in the foundational pretrained model. Moreover, compared with LoLCATs, which failed in this task, the results demonstrate LAWCAT's capacity not merely for transferring existing knowledge but also for significantly enhancing complex reasoning skills within extensive contexts. We also evaluate the Llama3 8B and Mistral series models on the BABILong benchmark, more details are shown in Appendix A.5.

5.5 Efficiency Comparison

To evaluate computational performance, we benchmarked the pre-fill latency of our Llama3.2 1B LAWCAT model against a Llama3.2 1B model incorporating FlashAttention-2 (FA2) (Dao, 2024) and the LoLCATs variant, using one NVIDIA RTX A6000 GPU. For a comprehensive comparison, we also include results from SOTA recurrent models, specifically GLA 1.3B and Mamba 1 1.3B.

As illustrated in Fig. 4, LAWCAT incurs slightly higher prefill latency than LLaMA3.2 1B with FA2 for sequences shorter than 8K tokens. Beyond this point, however, LAWCAT achieves significantly lower latency, with the gap widening as sequence length increases. In contrast, the naive implementation of LoLCATs (i.e., LoLCATs without the ThunderKittens kernel (HazyResearch, 2024)) exhibits substantially higher latency and greater GPU memory consumption.

When compared to the GLA 1.3B model, LAWCAT's inclusion of a Conv1D and linear layer, and GLA normalization contributes to a slightly higher latency. Notably, LAWCAT consistently outperforms the Mamba 1 1.3B model, which also incorporates a Conv1D layer, across all evaluated sequence lengths.

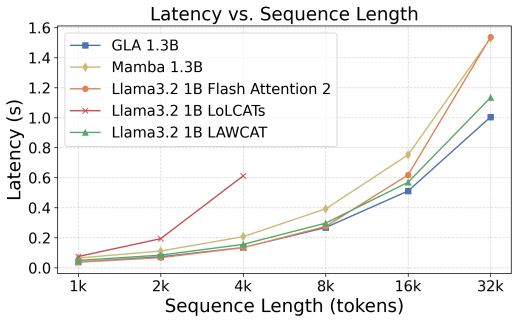

Figure 4: Comparison of prefill-stage latency among 5 different models. Note, LoLCATs runs out of memory for sequence lengths exceeding 8K tokens.

5.6 Ablation Study and Discussion

We conducted a comprehensive ablation study to systematically evaluate the contribution of each component in the proposed LAWCAT framework and discuss the effect of prevalent techniques such as Rotary Position Embeddings (RoPE) and Stochastic Weight Averaging (SWA). Unless otherwise specified, all experiments are based on models distilled and fine-tuned from the pre-trained LLaMA3.2-1B using training data with a sequence length of $\sim$1K tokens. The ablation analysis is structured around the following key questions:

Do we need the GLA normalization and Conv1D across tokens? To assess the contribution of key components within our proposed LAWCAT framework, we conduct an ablation study focusing on the GLA normalization and Conv1D layer.

	1K	2K ∼ 6K	7K	8K	9K	10K	11K
NN	0.9	0	0	0	0	0	0
NC	1.0	1.0	1.0	0.8	0.9	0.9	0.4
O	1.0	1.0	0.9	0.9	0.8	0.5	0.2

Table 2: Accuracy on the passkey retrieval task from 1K to 11K. "NN" means removing the GLA normalization from LAWCAT, "NC" means removing the Conv1D layer, and "O" means the complete LAWCAT model.

As presented in Table 2, removing GLA normalization significantly degrades performance on passkey retrieval tasks involving longer contexts, although the model maintains reasonable accuracy at the 1K-length scale. This finding supports our hypothesis that normalization is crucial for generalization. By constraining the GLA attention output to remain a normalized weighted sum of the values ($\mathbf{v}_i$) during layer-wise distillation (via MSE loss minimization against self-attention outputs), the normalization term mitigates overfitting to shorter sequence lengths and enhances the model's ability to preserve performance as context length increases.

On the other hand, the results in Table 2 suggest that the Conv1D layer offers limited benefit for the relatively simpler passkey retrieval task. We attribute this to the nature of the task, where much of the context contains redundant or useless information; applying Conv1D over such sequences does not necessarily facilitate the extraction of more salient features for identifying and memorizing the passkey. However, its importance becomes evident in more complex scenarios. Table 3 reveals a substantial performance gap on the NIAH task between the full LAWCAT model and its variant without the Conv1D layer. Notably, without Conv1D, LAWCAT fails to retrieve the correct UUID in all tested cases. This stark difference underscores the critical role of the Conv1D layer in capturing dependencies and memorizing information across tokens, particularly when dealing with longer sequences and more intricate information retrieval demands.

	S-NIAH-1				S-NIAH-2				S-NIAH-3		
	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K
NC	96	96	96	8	80	80	80	36	0	0	0
O	100	100	100	80	100	96	88	48	56	44	24

Table 3: Accuracy on S-NIAH 1&2&3 tasks. "NC" means removing the Conv1D layer from LAWCAT, and "O" means the complete LAWCAT model.

Do we need the RoPE and SWA? The necessity of explicit positional encodings like RoPE in linear attention is an open question, with varied adoption across models. Our ablation study (Table 5 in Appendix A.3) on LoLCATs and LAWCAT models addresses this. For LoLCATs, removing RoPE decreased performance on shorter contexts (1K-3K tokens) but slightly improved it on longer ones (4K-8K). This effect was more stark for our LAWCAT: the variant without RoPE maintained high accuracy up to 8K tokens, whereas the RoPE-equipped variant's performance collapsed beyond 3K tokens. These findings suggest that RoPE, while potentially aiding short-sequence modeling, introduces biases detrimental to long-context generalization, supporting the idea that recurrent formulations can inherently capture sequence order.

We also investigated the role of SWA (Table 7 in Appendix A.3). On long-context passkey retrieval, removing SWA significantly impaired LoLCATs, confirming its reliance on this mechanism. Conversely, LAWCAT performed better without SWA, especially at longer contexts; we hypothesize this is due to the difficulty in balancing contributions from the GLA and SWA, particularly when generalizing from shorter training to longer evaluation sequences. Additional results and analysis on the impact of SWA on standard short-context benchmarks (LM Eval, Table 7) are presented in Appendix A.3.

6 Conclusions

We introduced LAWCAT (Linear Attention with Convolution Across Time), a novel and efficient distillation framework that strategically integrates a causal Conv1D layer to bolster local dependency modeling and employs a normalized gated linear attention mechanism for robust generalization across varying sequence lengths. Our comprehensive experiments validate the effectiveness of LAWCAT. Distilling models like Mistral v0.1 7B and Llama3.2 1B yielded linear attention variants capable of remarkable long-context performance, achieving over 90% accuracy on passkey retrieval tasks up to 22K tokens and matching SOTA recurrent models on NIAH benchmarks, all while using minimal training data and shorter sequence lengths during distillation. Furthermore, the resulting LAWCAT models demonstrate practical efficiency gains, exhibiting faster prefill latency than highly optimized FlashAttention-2 implemen-

tations for sequences exceeding 8K tokens.

LAWCAT offers a practical and resource-efficient pathway for developing and deploying long-context models by leveraging existing pre-trained transformers. This approach helps democratize access to moderately long-sequence modeling capabilities, particularly beneficial for resource-constrained edge environments or rapid prototyping. Furthermore, due to its causal and linear nature, LAWCAT is particularly attractive for streaming inference, opening new opportunities for low-latency, on-device reasoning.

Limitations

While LAWCAT demonstrates strong long-context performance, on some complex benchmarks like S-NIAH-3 (1K) as shown in Table 1 and LM Evaluation Harness tasks (Gao et al., 2024) as shown in Table 7, purely distilled linear attention models may not fully match the performance of similarly sized transformer models or the linear attention model with sliding window attention (SWA). Although incorporating the SWA can bridge this gap on shorter sequences, as observed in our experiments, it potentially compromises the model's effectiveness on tasks requiring very long context generalization. This presents an interesting avenue for future research: developing dynamic mechanisms that can adaptively balance the global reach of linear attention with the local strength often captured by windowed approaches, perhaps conditioned on input characteristics or sequence length. Alternatively, exploring the integration of more sophisticated linear attention designs like Mixture-of-Memories (Du et al., 2025) might further mitigate the performance gap on standard benchmarks, although this could introduce new optimization challenges. Addressing this trade-off between benchmark performance and long-context fidelity remains a key direction for advancing efficient attention mechanisms.

Acknowledgments

This work was supported in part by Amazon through the 2024-2025 Amazon ML fellowship.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, and 1 others. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.

Seyedarmin Azizi, Souvik Kundu, Mohammad Erfan Sadeghi, and Massoud Pedram. 2025. Mambaextend: A training-free approach to improve long context extension of mamba. In *The Thirteenth International Conference on Learning Representations*.

Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. 2025. Seal: Steerable reasoning calibration of large language models for free. *Conference on Language Modeling*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, and 1 others. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

Tri Dao. 2024. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*.

Tri Dao and Albert Gu. 2024. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning (ICML)*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, and Yu Cheng. 2025. Mom: Linear sequence modeling with mixture-of-memories. *arXiv preprint arXiv:2502.13685*.

Qihang Fan, Huaibo Huang, and Ran He. 2024. Breaking the low-rank dilemma of linear attention. *arXiv preprint arXiv:2411.07635*.

Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. 2024. Open llm leaderboard v2. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard.

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, and 5 others. 2024. The language model evaluation harness.

Saurav Goel. 2023. Paul graham essay collection dataset. https://huggingface.co/datasets/sgoel9/paul_graham_essays. Accessed: 2025-05-05.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Dongchen Han, Yifan Pu, Zhuofan Xia, Yizeng Han, Xuran Pan, Xiu Li, Jiwen Lu, Shiji Song, and Gao Huang. 2024. Bridging the divide: Reconsidering softmax and linear attention. *Advances in Neural Information Processing Systems*, 37:79221–79245.

HazyResearch. 2024. ThunderKittens: HazyResearch/ThunderKittens. https://github.com/HazyResearch/ThunderKittens. Accessed: 2025-05-12.

Andrew G Howard. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

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? *arXiv preprint arXiv:2404.06654*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

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

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. *Preprint*, arXiv:1606.07947.

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2021. Booksum: A collection of datasets for long-form narrative summarization.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. 2024. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. In *Advances in Neural Information Processing Systems*, volume 37, pages 106519–106554. Curran Associates, Inc.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Jean Mercat, Igor Vasiljevic, Sedrick Scott Keh, Kushal Arora, Achal Dave, Adrien Gaidon, and Thomas Kollar. 2024. Linearizing large language models. In *First Conference on Language Modeling*.

Amirkeivan Mohtashami and Martin Jaggi. 2023. Random-access infinite context length for transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Bo Peng, Daniel Goldstein, Quentin Gregory Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Teddy Ferdinan, Kranthi Kiran GV, Haowen Hou, Satyapriya Krishna, Ronald McClelland Jr., Niklas Muennighoff, Fares Obeid, Atsushi Saito, Guangyu Song, Haoqin Tu, Ruichong Zhang, Bingchen Zhao, and 3 others. 2024. Eagle and finch: RWKV with matrix-valued states and dynamic recurrence. In *First Conference on Language Modeling*.

Zhen Qin, Xiaodong Han, Weixuan Sun, Dongxu Li, Lingpeng Kong, Nick Barnes, and Yiran Zhong. 2022. The devil in linear transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7025–7041, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zhen Qin, Dong Li, Weigao Sun, Weixuan Sun, Xuyang Shen, Xiaodong Han, Yunshen Wei, Baohong Lv, Xiao Luo, Yu Qiao, and 1 others. 2023. Transnormerllm: A faster and better large language model with improved transnormer. *arXiv preprint arXiv:2307.14995*.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741.

Akshat Ramachandran, Mingyu Lee, Huan Xu, Souvik Kundu, and Tushar Krishna. 2025. Ouromamba: A data-free quantization framework for vision mamba models. *International Conference on Computer Vision*.

Yeonju Ro, Zhenyu Zhang, Souvik Kundu, Zhangyang Wang, and Aditya Akella. 2025. On-the-fly adaptive distillation of transformer to dual-state linear attention for long-context llm serving. In *Forty-second International Conference on Machine Learning*.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

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

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander M Rush, and Tri Dao. 2024. The mamba in the llama: Distilling and accelerating hybrid models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. 2025. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. 2023. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. 2024. Parallelizing linear transformers with the delta rule over sequence length. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, Qianyu Chen, Huarong Zhou, Zhensheng Zou, Haoye Zhang, Shengding Hu, Zhi Zheng, Jie Zhou, Jie Cai, Xu Han, and 4 others. 2024. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint 2408.01800*.

Zhifan Ye, Zheng Wang, Kejing Xia, Jihoon Hong, Leshu Li, Lexington Whalen, Cheng Wan, Yonggan Fu, Yingyan Celine Lin, and Souvik Kundu. 2025. Lamb: A training-free method to enhance the long-context understanding of ssms via attention-guided token filtering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1200–1209.

Michael Zhang, Simran Arora, Rahul Chalamala, Benjamin Frederick Spector, Alan Wu, Krithik Ramesh, Aaryan Singhal, and Christopher Re. 2025. LoLCATs: On low-rank linearizing of large language models. In *The Thirteenth International Conference on Learning Representations*.

Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Re. 2024. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. In *The Twelfth International Conference on Learning Representations*.

A Appendix

A.1 Training Dataset

All datasets are in English. **Passkey Retrieval** We use the same format as (Mohtashami and Jaggi, 2023) to generate 10K samples to build the training dataset and 1K samples for validation. For the tokenizer of the Llama3.2 model, the length of each sample is 1162 tokens, and for the tokenizer of the Mistral model, the length is 1171 tokens.

S-NIAH 1&2&3 We use the same format as (Hsieh et al., 2024) to generate 8456 samples to build the training dataset and 1K samples for validation. We used the BookSum dataset (Kryściński et al., 2021) as the contextual 'haystack', different from the Paul Graham Essay Collection Dataset (Goel, 2023) in evaluation. For the tokenizer of the Llama3.2 model, the maximum length is 1231, the minimum length is 675, the average length is 785.50, the mode length is 768, and the std is 49. We also found that adding additional 1K data with the S-NIAH-1 format can improve the overall performance. For these 1K data, the maximum length is 1223, the minimum length is 1222, the average length is 1222.92, the mode length is 1223, and the std is 0.27.

BABILong QA2&QA3 For the BABILong benchmark (Kuratov et al., 2024), training and validation data were generated on-the-fly per the original setup. Specifically, there are 10K samples for the training data of QA2 and 9868 samples for the training data of QA3. Different from the curriculum learning strategy in the original paper, we simplified the training by fine-tuning solely on sequences of length 1300 with a mixed format of QA2 and QA3.

A.2 Training Configuration

All experiments were implemented using the Flash Linear Attention framework (Yang and Zhang, 2024). For the Passkey Retrieval task, we distilled and fine-tuned the LLaMA3.2 1B model over 4 epochs each, totaling $1162 \times 10,000 \times 4 \times 2 = 92.96M$ training tokens. For larger models (LLaMA3 8B, Mistral v0.1 and v0.3 7B), distillation and fine-tuning were performed for 2 epochs each, resulting in $1162 \times 10,000 \times 2 \times 2 = 92.96M$ tokens for LLaMA and $1171 \times 10,000 \times 2 \times 2 = 93.68M$ tokens for Mistral. Similarly, for the NIAH tasks, we distilled and fine-tuned for 2 epochs, yielding approximately $(785.5 \times 8456 + 1222.9 \times 1000) \times 2 \times 2 \approx 31.46M$ tokens. For the BABILong tasks (QA2 and QA3), datasets were generated dynamically; thus, exact token counts varied. With a maximum sequence length of 1300 tokens, the total training tokens were under $1300 \times (10,000 + 9,868) \times 2 \times 2 \approx 103.31M$.

For the Passkey Retrieval task with Llama3.2 1B model, we use the cosine learning rate schedule, and for other models and other tasks, we all use the ReduceLROnPlateau, which reduces the learning rate when the validation loss has stopped reducing. For all experiments, we use the Adamw optimizer (Loshchilov and Hutter, 2019) with an initial learning rate of 0.1 for distillation, 0.0001 for fine-tuning. For the LoRA fine-tuning, the rank is 8 and the alpha is 16, and we only add LoRA module for the linear layer for the query, key, value and output.

We keep the similar training hyperparameters as Zhang et al. (2025). Specifically, for the Passkey Retrieval task with the LLaMA3.2-1B model, we adopt a cosine learning rate schedule, while for all other models and tasks, we employ the `ReduceLROnPlateau` scheduler, which decreases the learning rate when the validation loss plateaus. All experiments use the AdamW optimizer (Loshchilov and Hutter, 2019), with an initial learning rate of 0.1 during distillation and 1e-4 for fine-tuning. During LoRA fine-tuning, we set the rank to 8 and the scaling factor (α) to 16, applying LoRA modules exclusively to the linear projections of the query, key, value, and output layers in the attention block. For all experiments, we run 3 times using different random seeds and report the results from the run achieving the best overall performance.

A.3 Other Ablation Study

Do we need to share the linear and Conv layers? In Table 4, we analyze four distinct configurations regarding parameter sharing between linear and convolutional layers. Our empirical results demonstrate that optimal performance is achieved when allowing Q and K to independently process contextual information through separate convolutional layers while sharing the subsequent linear projection. This architecture aligns with the theoretical foundation in Equation 2, where both query and key utilize the same transformation function $\phi(\cdot)$. The separation of convolutional processing is intuitively justified, as queries and keys serve distinct roles in attention mechanisms and therefore benefit from specialized feature

extraction pathways when incorporating previous token information. However, the shared linear projection enforces consistency in the embedding space where similarity is computed, maintaining the mathematical elegance of the linear attention formulation.

C	L	1K	2K	3K	4K	5K	6K	7K	8K	9K
✓	✓	1.0	1.0	1.0	0.9	0.8	0.5	0.1	0.1	0
✓	×	0.9	1.0	0.3	0	0.2	0	0	0	0
×	×	1.0	1.0	1.0	0.8	0.8	0.6	0.3	0.1	0
×	✓	1.0	1.0	1.0	1.0	1.0	1.0	0.9	0.9	0.8

Table 4: Accuracy on the passkey retrieval task from 1K to 9K. "C" and "L" mean if sharing the Conv1D or linear projection between query and key, "✓" means yes, and "×" means no.

Do we need to add RoPE to linear attention? Modern linear attention models exhibit diverse strategies for handling positional information. Some architectures, like GLA (Yang et al., 2023) and Gated DeltaNet (Yang et al., 2025), often omit explicit position embeddings, relying on their recurrent formulation to implicitly capture sequence order. Conversely, others, including TransNormerLLM (Qin et al., 2023) and Retentive Networks (Sun et al., 2023), explicitly integrate mechanisms like rotary position embeddings (RoPE) (Su et al., 2024) or its variants. This divergence raises a question regarding knowledge distillation: when transferring knowledge from standard Transformers (which typically use position embeddings) to linear attention models, is it beneficial to retain these explicit positional signals? Ablation study presented in Table 5 addresses this for the LoLCATs and LAWCAT. The results consistently indicate that incorporating RoPE actually impairs generalization performance on long-context tasks.

For LoLCATs, removing RoPE decreases performance on shorter contexts (1K-3K tokens) but slightly improves performance on longer contexts (4K-8K tokens). This effect is more pronounced in our LAWCAT architecture, where the variant without RoPE maintains high accuracy form 1K to 8K context lengths, while the RoPE variant's performance drops to zero beyond 3K tokens.

These results indicate that position embeddings constitute a critical factor limiting generalization to extended context lengths. The fixed positional encoding scheme optimized for training context appears to introduce inappropriate biases when applied to longer sequences. This insight aligns with recent research suggesting that recurrent computation inherently captures sequential dependencies without requiring explicit positional information (Gu and Dao, 2023).

M	PE	1K	2K	3K	4K	5K	6K	7K	8K
LoLCATs	✓	0.9	0.8	0.3	0	0	0	0	0
	×	0.4	0.2	0.1	0.2	0.2	0.1	0.2	0.2
LAWCAT	✓	1	0.8	0	0	0	0	0	0
	×	1.0	1.0	1.0	1.0	1.0	1.0	0.9	0.9

Table 5: The test accuracy on the passkey retrieval task from 1K to 8K. "M" means model name, and "PE" means if using the RoPE for linear attention. "✓" means yes, and "×" means no.

Do we need sliding window attention? Prior work posited SWA as a key component for enhancing linear attention model performance (Zhang et al., 2025). To rigorously evaluate this claim, we conducted ablation studies on both the original LoLCATs model and our proposed LAWCAT architecture, examining the impact of SWA across different task types and context lengths.

Our investigation first focused on long-context retrieval tasks. As shown in Table 5, row 3, removing SWA significantly degrades the performance of the LoLCATs model, suggesting its reliance on SWA for effective passkey retrieval. Conversely, incorporating SWA into our LAWCAT model adversely affected performance, particularly as context length increased. We hypothesize that this stems from the difficulty in optimally balancing the contributions of GLA and SWA. Furthermore, achieving a balance that generalizes from shorter training sequences to longer evaluation sequences appears challenging. This suggests it is difficult to balance the weights between linear attention and softmax attention, and even if the model can make their combination fit well on training data, it may be hard to generalize it to longer context lengths. For example, the ideal weights of LA and SWA for 1K-length contexts may not be good for 8K-length contexts.

We further assessed the effect of SWA on standard short-context benchmarks using LM Evaluation Harness tasks (LM Eval) (Gao et al., 2024) (Table 7). When SWA is removed, LAWCAT substantially outperforms LoLCATs, consistent with our findings on long-context tasks. However, adding SWA (window size 64) significantly boosts per-

M	SWA	1K	2K	3K	4K	5K	6K	7K	8K
LoL-CATs	✓	0.9	0.8	0.3	0	0	0	0	0
	×	0.1	0.0	0	0	0	0	0	0
LAW-CAT	✓	0.5	0.5	0.2	0.1	0	0	0	0
	×	1.0	1.0	1.0	1.0	1.0	1.0	0.9	0.9

Table 6: The test accuracy on the passkey retrieval task from 1K to 8K. "M" means model name, and "SWA" means if using the sliding window attention. "✓" means yes, and "×" means no.

Model	**PI**	**AE**	**AC**	**HS**	**WG**	**Avg.**
Pre-trained Transformer Model						
Llama 3.2 1B	74.4	65.5	35.8	63.7	60.5	60.0
Llama 3.2 1B*	73.2	64.5	35.6	40.3	59.9	54.7
Pre-trained Recurrent Model						
DeltaNet 1.3B	71.2	57.2	28.3	50.2	53.6	52.1
GLA 1.3B	71.8	57.2	26.6	49.8	53.9	51.9
Mamba 2 1.3B	73.2	64.3	33.3	59.9	60.9	58.3
Distilled Model						
LoLCATs	59.4	41.2	23.8	31.4	50.2	41.2
LoLCATs*	74.6	63.0	35.1	**63.7**	61.5	59.6
Ours	71.9	65.0	**36.3**	53.1	52.8	55.8
Ours*	**75.3**	**65.2**	35.1	62.0	**62.0**	**59.9**

Table 7: Comparison of various models on 5 tasks from LM EVAL benchmark. Model name with * indicates the model add sliding window attention with size of 64. Specially, Llama 3.2 1B * means the Llama 3.2 1B model only uses SWA.

formance for both models, bringing them to comparable levels (59.6 vs. 59.9 average score). For context, we include results from relevant baselines: the standard Llama 3.2 1B transformer and several SOTA recurrent models. Notably, a modified Llama 3.2 1B using only SWA (denoted Llama 3.2 1B*) achieves competitive performance (54.7 avg.) compared to the original (60.0 avg.), albeit with a marked drop on HellaSwag. This suggests that SAW alone possesses considerable capability. The strong results of LoLCATs may therefore be primarily attributed to the inherent effectiveness of SWA, with the linear component potentially offering further gains on specific tasks like HellaSwag.

In conclusion, our ablation studies reveal a nuanced role for SWA. While its integration can substantially enhance performance on short-context tasks, potentially contingent on successful optimization of the hybrid attention mechanism, it appears detrimental to generalization capability in long-context scenarios.

Ablation study on the rank of the gate function in GLA GLA models (Yang et al., 2023) utilize 2 consecutive low-rank linear layers to implement the gate function for efficiency, with the original models adopting a rank of 16. We investigate the impact of this rank hyperparameter in our LAWCAT models, distilled from Llama3.2 1B Instruct (Grattafiori et al., 2024), on complex S-NIAH retrieval tasks: S-NIAH-1 (pass-key), S-NIAH-2 (number), and the particularly demanding S-NIAH-3 (UUID retrieval).

As shown in Table 8, LAWCAT models with rank 16 performed well on S-NIAH-1 and S-NIAH-2 but struggled significantly on S-NIAH-3 (e.g., 12% at 1K, 0% at 2K). This indicates that a rank of 16 limits the forget gate's capacity for recalling long, complex items like UUIDs. Increasing the rank to 32 substantially improved S-NIAH-3 performance (e.g., to 56% at 1K and 44% at 2K) while maintaining strong performance on S-NIAH-1 and S-NIAH-2. This suggests rank 32 offers a better balance of capacity for complex retrieval.

Further rank increases to 64 yielded mixed results; while improving some S-NIAH-1/2 scores at longer contexts (e.g., S-NIAH-1 8K at 92%; S-NIAH-2 8K at 64%), it did not consistently improve S-NIAH-3 over rank 32 (36% vs 56% at 1K). Ranks 128 and Full Rank (FR) generally led to performance degradation across tasks, particularly at longer contexts (e.g., rank 128 scored 36% on S-NIAH-1 8K). We hypothesize this non-monotonic trend stems from higher ranks introducing optimization challenges due to increased parameters and a greater risk of overfitting, which can impair the forget gate's ability to learn generalizable retention patterns.

In conclusion, these findings indicate an optimal gate function rank—around 32 for our 1B parameter LAWCAT models—that balances model capacity for complex retrieval against optimization difficulties and overfitting.

A.4 The visualization of Attention Scores

To qualitatively assess the attention mechanisms, we visualize attention score heatmaps from a representative head on an example from the S-NIAH-3 task.

Figure 5 illustrates an attention head primarily focused on local contextual information. A comparative analysis of the heatmaps reveals that while the Transformer, LoLCATs, and LAWCAT models

	S-NIAH-1 (pass-key retrieval)				S-NIAH-2 (number in haystack)				S-NIAH-3 (uuid in haystack)		
Rank	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K
16	100	100	96	80	100	**96**	92	52	12	0	4
32	100	100	100	80	100	**96**	88	48	**56**	**44**	**24**
64	100	92	100	**92**	100	**96**	**96**	**64**	36	32	20
128	100	92	88	36	92	72	72	4	20	24	12
FR	96	96	80	12	96	72	36	4	24	28	12

Table 8: Performance comparison across different models on S-NIAH 1, 2, and 3 tasks.

exhibit broadly similar local attention patterns, notable differences emerge. The LoLCATs model displays a tendency for its attention to be diffused by more distant preceding tokens, potentially diluting its focus. In contrast, LAWCAT maintains a sharper concentration on the immediately relevant local context, particularly the current token, mirroring the behavior of the standard softmax Transformer more closely. This suggests that the integrated Conv1D layer in LAWCAT effectively strengthens local bias. By design, the convolutional operation aggregates information from adjacent tokens, allowing the model to consolidate the importance of the local neighborhood onto the current token's representation, thereby achieving a more focused attention distribution akin to softmax attention.

Further insights are provided by Figure 6, which depicts an attention head responsible for identifying the relationship between an answer and the "needle" (the target information to be retrieved). In this scenario, LoLCATs tends to allocate attention more broadly across the sequence, a characteristic that can hinder precise identification of the crucial answer-needle relationship. LAWCAT, however, mitigates this diffusion, exhibiting a more localized attention pattern that closely resembles the Transformer's focused engagement on the relevant segments. This closer alignment in attention scores between LAWCAT and the softmax Transformer not only correlates with LAWCAT's superior performance on retrieval tasks but also empirically validates the efficacy of the Conv1D layer in fostering a beneficial local inductive bias, enabling more precise attention allocation.

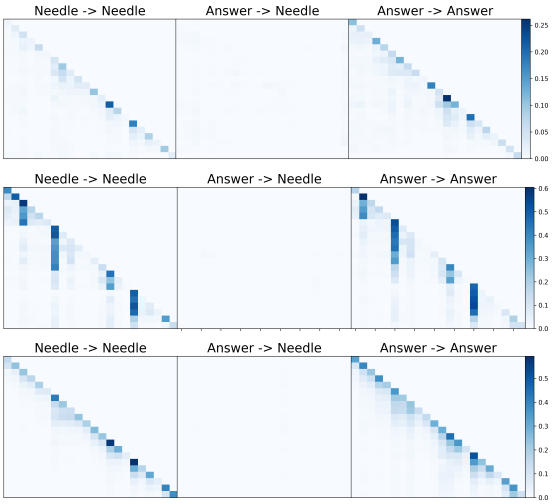

Figure 5: Visualization of attention scores from layer 15, head 5 across three models: Transformer (top), LoLCATs (middle), and LAWCAT (bottom). Each row presents three attention maps: needle-to-needle (left), answer-to-needle (center), and answer-to-answer (right)

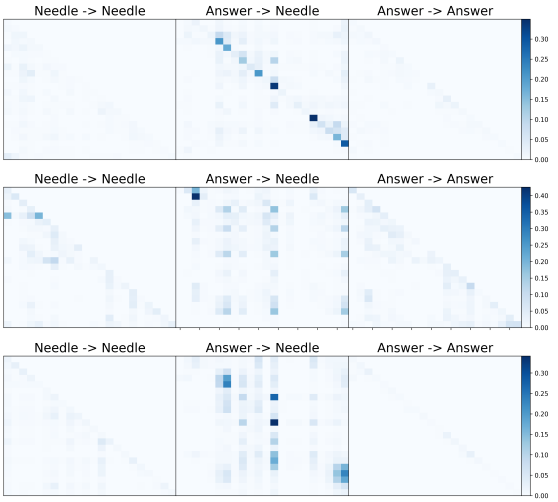

Figure 6: Visualization of attention scores from layer 15, head 22 across three models: Transformer (top), LoLCATs (middle), and LAWCAT (bottom). Each row presents three attention maps: needle-to-needle (left), answer-to-needle (center), and answer-to-answer (right)

A.5 More Resutls on BABILong

We also added the rustles on BABILong of the Llama 3 8B Instruct and the LAWCAT variants, as shown in Fig. 7, our LAWCAT variants consistently outperform the baselines, particularly at longer context lengths, mirroring the findings from experi-

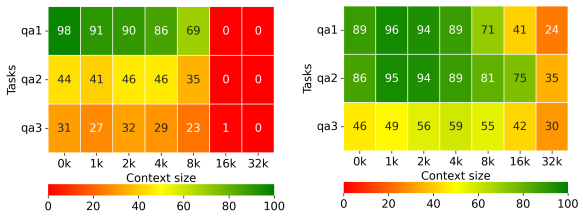

Figure 7: Comparison of accuracy on QA1&QA2&QA3 from BABILong benchmark between Llama 3 8B Instruct (left) and LAWCAT (right).

Figure 8: Comparison of accuracy on QA1&QA2&QA3 from BABILong benchmark between Mistral v0.3 7B (left) and LAWCAT (right).

ments with the smaller LLaMA 3.2 1B model. As shown in Fig. 8, compared to the original model, the distilled Mistral v0.3 7B model exhibits significantly improved performance on inputs longer than 8k tokens. However, it suffers from noticeable performance degradation on shorter context lengths, especially the 0k setting, which does not contain any "haystack" content. We hypothesize that this issue arises because Mistral v0.3 is a pretrained base model that is highly sensitive to prompt formatting, particularly after limited fine-tuning. We observed that, instead of producing incorrect answers, the model often outputs nothing for 0k inputs.

Task	0K	1K	2K	4K	8K	16K	32K
Pre-trained Transformer Model							
QA2	44	52	47	35	14	5	2
QA3	36	34	35	23	27	23	16
LAWCAT							
QA2	94	97	95	90	82	32	8
QA3	71	73	71	68	44	18	9

Table 9: Results of Mistral v0.1 Instruct 7B on BABILong benchmark (QA2&QA3)

Therefore, we added the results of Mistral v0.1 Instruct 7B and Mistral v0.3 Instruct 7B models on the QA2 and QA3 tasks of the BABILong benchmark, as shown in Table 9 and Table 10 respectively. As expected, applying LAWCAT to the models after instruction tuning will result in more stable performance, and the results of the Mistral models are consistent with those of the Llama model (although there are slight accuracy drops for the Mistral v0.1 7B Instruct LAWCAT variant model on QA3 when the context length is longer than 16k).

Task	0K	1K	2K	4K	8K	16K	32K
Pre-trained Transformer Model							
QA2	48	41	31	19	3	2	1
QA3	36	32	36	25	22	18	19
LAWCAT							
QA2	87	92	95	96	96	72	43
QA3	59	55	66	72	69	64	43

Table 10: Results of Mistral v0.3 Instruct 7B on BABILong benchmark (QA2&QA3)

Overall, we believe these results further validate the effectiveness of our LAWCAT on various model architectures and tasks.