# Lizard: An Efficient Linearization Framework for Large Language Models

Anonymous ACL submission

## Abstract

We propose Lizard, a linearization framework that transforms pretrained Transformer-based Large Language Models (LLMs) into subquadratic architectures. Transformers faces severe computational and memory bottlenecks with long sequences due to the quadratic complexity of softmax attention and the growing Key-Value (KV) cache that makes inference memory-bound by context length. Lizard addresses these limitations by introducing a subquadratic attention mechanism that closely approximates softmax attention while preserving model quality. Unlike prior linearization methods constrained by fixed, non-adaptive structures, Lizard augments the architecture with compact, learnable modules that enable adaptive memory control and robust length generalization. Moreover, we introduce a hardware-aware algorithm that solves numerical instability in gated attention to accelerate training. Extensive experiments show that Lizard achieves near-lossless recovery of its teacher model's performance, significantly outperforming previous methods by up to 9.4 - 24.5 points on the 5-shot MMLU benchmark and demonstrating superior associative recall.

## 1 Introduction

Large Language Models (LLMs) based on Transformer architectures have achieved impressive advances in natural language processing across various tasks (Grattafiori et al., 2024; Mann et al., 2020; Achiam et al., 2023). However, their reliance on softmax attention with quadratic time and memory complexity poses significant limitations for long-context applications, both during training and inference. In particular, the softmax attention computation scales quadratically with the sequence length, while the key-value (KV) cache grows linearly during generation, resulting in significant computational and memory overhead for long-context sequences (Adnan et al., 2024).

Recent work has proposed linear and subquadratic alternatives to softmax attention (Yang et al., 2024; Gu and Dao, 2023; Peng et al., 2023b; Wang et al., 2020; Dao and Gu, 2024), enabling linear-time training and constant-memory inference. However, despite these efficiency benefits, pretraining LLMs with such architectures from scratch requires massive training budgets, often involving trillions of tokens. More importantly, models trained with linear attention mechanisms from scratch consistently underperform on tasks that require in-context learning and retrieval. For example, Transformer models significantly outperform both Mamba and Mamba-2 by up to 15 points on the 5-shot MMLU benchmark when all are pretrained under the same settings (Waleffe et al.). Another promising direction involves *linearizing* pretrained Transformer-based LLMs by replacing their softmax attention modules with subquadratic alternatives (Zhang et al.; Mercat et al., 2024; Zhang et al., 2025; Wang et al., 2024; Lan et al., 2025). This strategy aims to retain the rich capabilities of models trained on trillions of tokens while inheriting the efficiency of subquadratic architectures. However, existing linearization methods have consistently fallen short on two key goals: **(1)** preserving performance parity with the teacher model, and **(2)** enabling robust long-context generalization.

These shortcomings stem from two core architectural limitations. **First**, existing approaches lack mechanisms for adaptive memory control. Methods like LoLCATs (Zhang et al., 2025) overlook the sophisticated gating mechanisms critical for modulating contextual information in modern recurrent models. Others like Liger (Lan et al., 2025) attempts to a gate but constrains it to a fixed, parameter-free pooling operation. While parameter-efficient, this non-learnable transformation creates an information bottleneck that prevents the model from learning optimal memory dynamics. These architectural choices lead to substantial performance

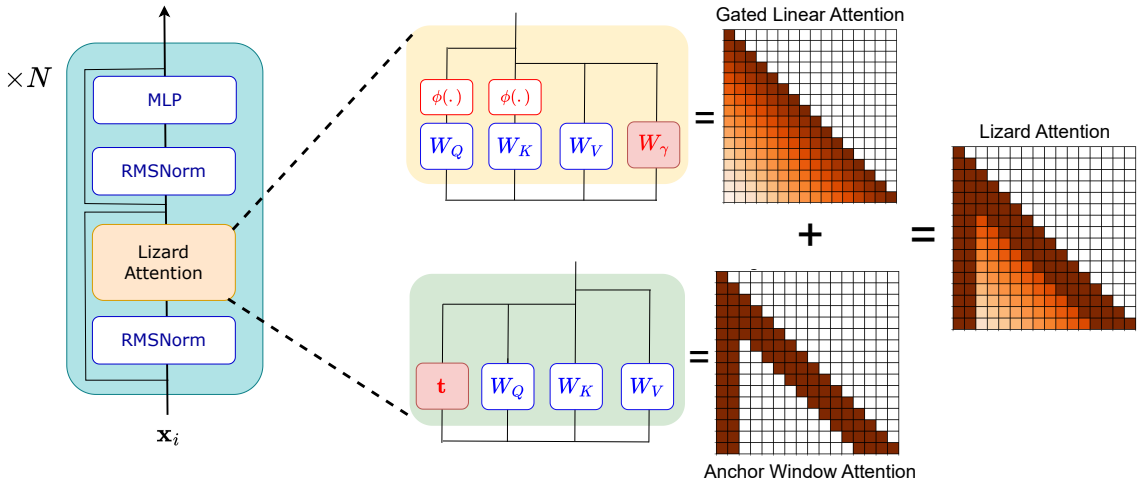


Figure 1: An overview of the Lizard Attention architecture. Lizard replaces standard attention with a hybrid mechanism that combines Gated Linear Attention (top) for global context compression and Anchor Window Attention (bottom) for local precision. The components highlighted in red the feature maps ($\phi$), the gating module ($W_\gamma$), and the meta-memory tokens ($t$) represent the compact, learnable modules that are augmented to the teacher architecture.

degradation: on the 5-shot MMLU benchmark, LoLCATs lags behind its teacher model by 13.8 points, while Liger exhibits a 21.9 point drop. **Second**, these methods fail at length extrapolation due to their reliance on fixed positional encodings. By retaining Rotary Positional Embeddings (RoPE) (Su et al., 2024) from the pretrained teacher, these models are restricted to the sequence lengths seen during training. This design fails to leverage the extrapolation capabilities inherent in recurrent formulations and prevents the models from achieving true long-context generation.

In this paper, we introduce **Lizard** (**Li**nearizing Softmax **A**ttention with **R**ecurrent Gate **D**ynamics), an efficient framework for linearizing LLMs. Unlike prior methods that strictly preserve the teacher's architecture, Lizard adopts a fundamentally different design philosophy: it introduces compact, learnable modules to enable adaptive memory control and data-driven positional modeling, thereby bridging the expressivity-efficiency gap. At the core of this approach, Lizard is augmented with a learnable gating module that forms a data-adaptive recurrent structure. This module serves two primary purposes. First, it acts as a data-dependent alternative to fixed positional encodings like RoPE, allowing the model to learn relative positional information through adaptive decay patterns for enhanced length generalization. Second, its gated recurrent structure provides a dynamic memory management mechanism, enabling the model to control the retention and forgetting of past tokens to improve associative recall. Furthermore, to fully recover the expressiveness of softmax attention, the globally-aware gated attention is then combined with Anchor Window Attention, a mechanism that augments local attention with learnable meta-memory tokens. This hybrid framework assigns a specialized role to each component: the gated module captures global context in a compressed form, while the Anchor Window Attention preserves the fine-grained precision of local interactions, resulting in a high-quality approximation of softmax attention.

To complement these architectural innovations, we address a critical implementation bottleneck that hinders the efficiency of gated recurrent models. Standard gated linear attention is often numerically unstable in low-precision formats. This instability forces a reliance on inefficient, full-precision computations, preventing the full utilization of modern hardware accelerators like tensor cores (Yang et al., 2024). We introduce a hardware-aware algorithm that solves this by reparameterizing the attention computation, making it compatible with tensor core units and improving training throughput by up to 32%.

Our contributions are as follows:

- We propose Lizard, a linearization framework that converts pretrained Transformers into subquadratic architectures with near-lossless performance recovery. Lizard significantly out-

performs prior methods, improving by up to 9.4 - 24.5 points on 5-shot MMLU and demonstrating superior associative recall on long-context retrieval tasks. Furthermore, we show that in a hybrid setup that retains 50% of the original softmax attention layers, Lizard nearly matches the teacher model's performance on 5-shot MMLU (65.1 vs 66.6)

- We introduce a hardware-aware algorithm that solves numerical instability in gated attention, improving training throughput by up to 32% and enabling more efficient model optimization.
- We conduct extensive empirical studies to analyze architectural design choices across diverse benchmarks.

## 2 Preliminary

To motivate Lizard, we first review the core components of Causal Softmax Attention, and techniques for linearzing softmax attention.

**Causal Softmax Attention:** In modern Transformer architectures (Touvron et al., 2023; Jiang et al.), for a query vector at position $i$, Causal Softmax Attention produces the output $\mathbf{y}_i$ as:

$$\mathbf{y}_i = \sum_{t=1}^{i} \frac{\exp\left(\mathbf{q}_i^\top \mathbf{k}_t/\sqrt{d}\right)}{\sum_{j=1}^{i} \exp\left(\mathbf{q}_i^\top \mathbf{k}_j/\sqrt{d}\right)} \mathbf{v}_t$$

The expressiveness of this mechanism comes from the softmax function's ability to create a sharp, spiky distribution over past tokens. However, its need to compare every query to all preceding keys results in a computational complexity of $\mathcal{O}(L^2 d)$ for a sequence of length $L$, which is prohibitive for long contexts.

**Linear Attention and Linearization** The core idea behind linearization is to replace the expensive softmax function with an efficient alternative. Linear attention mechanisms (Katharopoulos et al., 2020) achieve this by substituting the exponential similarity function with a kernel function $k(q,k) = \phi(q)\phi(k)^\top$, where $\phi$ is a feature map. The output is then computed as:

$$\hat{\mathbf{y}}_i = \frac{\sum_{t=1}^{i} \phi(\mathbf{q}_i)^\top \phi(\mathbf{k}_t)\mathbf{v}_t}{\sum_{j=1}^{i} \phi(\mathbf{q}_i)^\top \phi(\mathbf{k}_j)}$$

This kernel formulation lies at the core of linearized attention, as it enables the reordering of matrix multiplications. Rather than constructing a large $L \times L$ attention matrix, the computation can be reformulated as an incremental update, reducing the complexity to $\mathcal{O}(Ld^2)$ and allowing constant-memory inference in a recurrent form:

$$\mathbf{h}_i = \mathbf{h}_{i-1} + \phi(\mathbf{k}_i)\mathbf{v}_i^\top, \quad \mathbf{y}_i = \phi(\mathbf{q}_i)^\top \mathbf{h}_i$$

## 3 Lizard Framework

The core of the Lizard framework is the replacement of the softmax attention layer with an augmented, hybrid subquadratic mechanism. This transformation is achieved through a two-stage training process: an initial attention approximation stage to mimic the teacher model, followed by a fine-tuning stage to align the new architecture with downstream language modeling objectives.

### 3.1 First Stage: Approximating Softmax Attention for Unbounded Context

In the teacher model, query and key vectors are transformed by a RoPE module before the attention computation. The full RoPE-infused softmax attention output, which we aim to approximate, is:

$$\mathbf{y}_i = \sum_{t=1}^{i} \frac{\exp\left(\varphi_R(\mathbf{q}_i)^\top \varphi_R(\mathbf{k}t)/\sqrt{d}\right)}{\sum_{j=1}^{i} \exp\left(\varphi_R(\mathbf{q}_i)^\top \varphi_R(\mathbf{k}_j)/\sqrt{d}\right)} \mathbf{v}_t$$

where $\varphi_R(.)$ denotes the RoPE transformation. By training our RoPE-free mechanism to replicate this output, we distill both the attention patterns and the positional awareness of the teacher model.

**Learnable Gating for Adaptive Memory Control and Length Extrapolation:** To solve the core limitations of prior work, we augment the linear attention mechanism with a learnable gating module, forming a data-adaptive recurrent structure. The output of the resulting RoPE-free Gated Linear Attention is computed as:

$$\hat{\mathbf{y}}_i = \frac{\phi_q(\mathbf{q}_i)^\top \sum_{t=1}^{i} \left(\prod_{l=t+1}^{i} \mathbf{\Gamma}_l\right) \phi_k(\mathbf{k}_t)\mathbf{v}_t^\top}{\phi_q(\mathbf{q}_i)^\top \sum_{j=1}^{i} \left(\prod_{l=j+1}^{i} \mathbf{\Gamma}_l\right) \phi_k(\mathbf{k}_j)}$$

where $\mathbf{\Gamma}_i = \text{sigmoid}(\mathbf{W}_\gamma \mathbf{x}_i)$ is the learnable gating factor. The gating mechanism plays a dual role in the attention transformation. First, it implicitly captures relative positional information by controlling the decay of past contributions. Unlike RoPE, which relies on predefined sinusoidal patterns, the data-adaptive gating factors enable better generalization across context lengths. Second, the gating

mechanism provides adaptive memory control by allowing the model to dynamically determine how much past information to retain or forget. This property supports a recurrent formulation that enables constant-memory inference through an incremental hidden state update $\mathbf{S}_i$, which summarizes the historical information up to position $i$:

$$\mathbf{S}_i = \mathbf{\Gamma}_i \mathbf{S}_{i-1} + \phi_k(\mathbf{k}_i)\mathbf{v}_i^\top, \quad \hat{\mathbf{y}}_i = \phi_q(\mathbf{q}_i)^\top \mathbf{S}_i$$

This state update removes the need to store the full key-value sequence, allowing constant-memory inference.

**Anchor Window Attention for Local Precision:** While the gated recurrent structure excels at compressing global context, it can lose the sharp, spiky detail of softmax attention. To preserve this local precision, we combine the globally-aware GLA with Anchor Window Attention. This mechanism augments a local sliding window with a set of $m$ learnable meta-memory tokens $\mathbf{t} \in \mathbb{R}^m$.

These tokens are engineered to function as dedicated attention sinks, whose primary role is to stabilize the attention distribution by absorbing attention weight, without directly contributing their value vectors to the final output. This allows the model to divert attention mass to these sinks when local information is less relevant, thereby managing the massive activations phenomenon (Sun et al., 2024; Gu et al., 2025) and preserving the fidelity of the local context. To achieve this, we modify the standard softmax computation. The output at position $i$ is computed as:

$$\hat{\mathbf{y}}_i = \frac{\sum_{t=i-w+1}^{i} \exp(\mathbf{q}_i^\top \mathbf{k}_t/\sqrt{d})\mathbf{v}_t}{\sum_{j=0}^{m-1} \mathbf{t}_j + \sum_{t=i-w+1}^{i} \exp(\mathbf{q}_i^\top \mathbf{k}_t/\sqrt{d})}$$

where $\mathbf{t}_j$ is a learnable scalar parameter representing the logit of a meta-memory token. This formulation allows the model to manage powerful global signals via the meta-memory sinks while focusing the output computation on the fine-grained local context. This is achieved while maintaining a fixed-size key-value cache of $w + m$ tokens for constant-memory inference.

**Attention Approximation:** We approximate the full softmax attention output, $\mathbf{Y}_{\text{softmax}}$, by combining the outputs of the globally-aware Gated Linear Attention and the locally-precise Anchor Window Attention. The final output, $\hat{\mathbf{Y}}_{lizard}$, is a combination of the two:

$$\hat{\mathbf{Y}}_{lizard} = \hat{\mathbf{Y}}_{\text{gate}} + \alpha \cdot \hat{\mathbf{Y}}_{\text{anchor}}$$

where $\hat{\mathbf{Y}}_{\text{gate}}$ and $\hat{\mathbf{Y}}_{\text{anchor}}$ are the outputs from the gated and anchor window mechanisms. The learnable parameters are optimized by minimizing the discrepancy between our approximation and the teacher model's original attention output:

$$\mathcal{L}_{\text{MSE}}(\phi, \mathbf{W}_\gamma, \mathbf{t}) = \frac{1}{N} \sum_{l=1}^{N} \left\| \mathbf{Y}^l_{\text{softmax}} - \hat{\mathbf{Y}}^l_{lizard} \right\|_F^2$$

where $N$ is the number of attention layers in the model. Overall, Lizard achieves an $O\big(L(w+m)d + Ld^2\big)$ time and space complexity. For generation, Lizard requires only $O((w+m)^2 d + d^2)$ time and space for every token.

### 3.2 Second Stage: Aligning with Language Modeling

While the first stage ensures a high-fidelity architectural approximation, the second stage aligns the model with the downstream language modeling task. In this stage, the original softmax attention layers are replaced by Lizard attention layers, and the entire model is fine-tuned using the standard autoregressive language modeling objective: $\mathcal{L}_{\text{LM}}(\theta) = -\sum_{i=1}^{L} \log P(x_i \mid x_{<i})$ This step bridges the gap between structural mimicry and optimal task performance, adapting the linearized model to its end-to-end objective.

## 4 Hardware-Aware Algorithm for Efficient Training

To achieve maximum efficiency, Gated Linear Attention (GLA) must be computed in a parallel form on hardware accelerators like GPUs. However, the standard parallel formulation suffers from a critical numerical instability that prevents the use of low-precision formats, thereby creating a performance bottleneck. The parallel form of Gated Linear Attention is expressed as:

$$\hat{\mathbf{Y}}_{\text{gate}} = \left( \left( (\phi(\mathbf{Q}) \odot \mathbf{C}) \left( \frac{\phi(\mathbf{K})}{\mathbf{C}} \right)^\top \right) \odot \mathbf{M} \right) \mathbf{V}$$

where $\mathbf{C}$ is the matrix of cumulative gate products, with each row $\mathbf{c}_t = \prod_{j=1}^{t} \mathbf{\Gamma}_j$.

The matrix form of gated linear attention is not numerically stable, as the cumulative product of gating values $\mathbf{c}_t$ can become extremely small, leading to underflow and instability during training for the low precision format such as `bfloat16`. This forces a fallback to full-precision (`float32`) operations, which is $2{-}3\times$ slower and memory intensive,

preventing the use of hardware accelerators like Tensor Cores that are optimized for low-precision arithmetic.

We leverage the strictly non-negative property of the *Hedgehog* feature map (Zhang et al.), $\phi(\mathbf{x}) = [\exp(\mathbf{xW}) \oplus \exp(-\mathbf{xW})]$. This exponential-based structure is critical, as it permits a stable reparameterization of the attention computation in log-space. We absorb the cumulative gate term $\mathbf{C}$ directly into the query and key projections, resulting in the following hardware-efficient formulation:

$$\widetilde{\mathbf{Q}} = [\exp(\mathbf{QW} + \log \mathbf{C}) \oplus \exp(-\mathbf{QW} + \log \mathbf{C})]$$
$$\widetilde{\mathbf{K}} = [\exp(\mathbf{KW} - \log \mathbf{C}) \oplus \exp(-\mathbf{KW} - \log \mathbf{C})]$$

By shifting the unstable gating contributions into the feature space, this approach transforms the core operation into a standard General Matrix Multiplication (**GEMM**): $\hat{\mathbf{Y}}_{\text{gate}} = \left(\left(\widetilde{\mathbf{Q}}\widetilde{\mathbf{K}}^\top\right) \odot \mathbf{M}\right)\mathbf{V}$ which aligns with the native `mma.sync` Tensor Core instruction. This avoids custom CUDA kernels and leverages highly optimized GEMM routines in cuBLAS and FlashAttention backends (Dao, 2023), avoids a full-precision fallback. As a result, it improves training throughput by up to 32% compared to prior gated attention implementations (Yang et al., 2024) [1].

## 5 Experimental Study

In this section, we present our experimental results, focusing on three key aspects:

1. **Language Modeling Benchmarks:** We evaluate Lizard on standard language modeling datasets and compare its performance against existing subquadratic alternatives and linearizations. Our results indicate that Lizard matches the average performance of the teacher model and significantly outperforms other baselines by a large margin.

2. **Language Modeling Recall:** To evaluate memory retrieval capabilities, we conduct experiments on the Needle-in-a-Haystack test, which assesses the model's ability to retrieve specific information over long sequences. Lizard significantly outperforms existing linearized LLMs on associative recall tasks. Furthermore, by retaining a small number of full attention layers, Lizard closely approximates the teacher model's performance in the 5-shot MMLU benchmark.

3. **Generation Efficiency:** We compare the generation throughput of Lizard with the teacher model equipped with FlashAttention-2. While the teacher model quickly runs out of memory at a sequence length of 32K, Lizard maintains constant memory usage and throughput, enabling efficient generation with infinite context.

4. **Architectural and Ablation Analysis:** We conduct a detailed analysis of Lizard's design choices, including the structure of the gating module and the contribution of each component. These studies help identify the most effective configurations for performance and efficiency.

**Experimental Setup:** We conduct our experiments using two widely used Transformer-based LLMs: Mistral-7B (Jiang et al.) and Llama-3-8B (Grattafiori et al., 2024) as teacher models. For training, we utilize a curated subset of 50K high-quality examples cleaned Alpaca dataset [2] (Taori et al., 2023). By default, we use a scalar gating structure, where $\Gamma_i = \gamma_i \mathbf{1}_d^\top, \gamma_i = \sigma(W_\gamma \mathbf{x}_i)$, with $\gamma_i \in \mathbb{R}$ and $W_\gamma \in \mathbb{R}^{d\times 1}$. We also explore various gating module designs, which are discussed in Section 5.4.

In the *First Stage*, we train the feature maps $\phi_q$, $\phi_k$, and the gating parameter matrix $W_\gamma$ jointly to approximate full softmax attention and RoPE patterns. For the sliding window attention module, we use a small window size of $w = 128$ and $m = 4$ meta tokens. In the *Second Stage*, we employ Low-Rank Adaptation (LoRA) (Hu et al., 2022) for parameter-efficient fine-tuning. Specifically, LoRA is applied to the projection matrices $W_Q$, $W_K$, and $W_V$, with a default rank $r = 8$ and scaling factor $\alpha = 16$. Both stages are trained for 2 epochs, corresponding to 20M tokens per stage. We use the AdamW optimizer and a cosine learning rate schedule with a 10% linear warmup. Training is performed using Fully Sharded Data Parallelism (FSDP-2) (Zhao et al., 2023) across 8×A100 80GB GPUs. The peak learning rate is set to $1 \times 10^{-3}$ for the first stage and $5 \times 10^{-4}$ for the second stage. We adopt a global batch size of 8, with each example having a maximum sequence length of 2048 tokens.

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

[2] https://huggingface.co/datasets/yahma/alpaca-cleaned

| Model | Training Tokens (B) | PiQA acc | ARC-e acc | ARC-c acc_norm | Hella. acc_norm | Wino. acc | MMLU (5-shot) | Avg. | Avg. (no MMLU) |
|---|---|---|---|---|---|---|---|---|---|
| *Transformer* | | | | | | | | | |
| Gemma-7B | 6000 | 81.9 | 81.1 | 53.2 | 80.7 | 73.7 | 62.9 | 74.1 | 72.3 |
| Mistral-7B | 8000* | 82.1 | 80.9 | 53.8 | 81.0 | 74.0 | 62.4 | 74.4 | 72.4 |
| LLaMA-3-8B | 15000 | 79.9 | 80.1 | 53.3 | 79.1 | 73.1 | 66.6 | 73.1 | 72.0 |
| *Subquadratic* | | | | | | | | | |
| Mamba-7B | 1200 | 81.0 | 77.5 | 46.7 | 77.9 | 71.8 | 33.3 | 71.0 | 64.7 |
| RWKV-6-v2.1-7B | 1420 | 78.7 | 76.8 | 46.3 | 75.1 | 70.0 | – | 69.4 | 69.4 |
| TransNormerLLM-7B | 1400 | 80.1 | 75.4 | 44.4 | 75.2 | 66.1 | 43.1 | 68.2 | 64.1 |
| Hawk-7B | 300 | 80.0 | 74.4 | 45.9 | 77.6 | 69.9 | 35.0 | 69.6 | 63.8 |
| Griffin-7B | 300 | 81.0 | 75.4 | 47.9 | 78.6 | 72.6 | 39.3 | 71.1 | 65.8 |
| *Linearized (Bounded)* | | | | | | | | | |
| Mistral-7B-SUPRA | 100 | 80.4 | 75.9 | 45.8 | 77.1 | 70.3 | 34.2 | 69.9 | 64.0 |
| Mistral-7B-LoLCATs | 0.04 | 81.5 | 81.7 | 54.9 | 80.7 | 74.0 | 51.4 | 74.5 | 70.7 |
| LLaMA-3-8B-LoLCATs | 0.04 | 80.9 | 81.7 | 54.9 | 79.7 | **74.1** | 52.8 | 74.2 | 70.7 |
| Liger-GLA-Mistral-7B | 0.02 | 80.1 | 78.7 | 49.3 | 76.3 | 70.1 | 36.3 | 70.9 | 65.1 |
| Liger-GLA-Llama-3-8B | 0.02 | 80.3 | 81.1 | 52.5 | 76.3 | 72.0 | 43.4 | 72.4 | 67.6 |
| *Linearized (Unbounded)* | | | | | | | | | |
| Mamba2-LLaMA-3-8B | 20 | 76.8 | 74.1 | 48.0 | 70.8 | 58.6 | 43.2 | 65.6 | 61.9 |
| **Mistral-7B-Lizard (Ours)** | 0.04 | 81.8 | 83.2 | 55.8 | **79.8** | 72.0 | 60.8 | 74.5 | 72.2 |
| **LLaMA-3-8B-Lizard (Ours)** | 0.04 | **82** | **83.5** | **56.7** | 79.3 | 71.7 | **61.2** | **74.6** | **72.4** |

Table 1: Performance comparison of Lizard and existing 7B-size subquadratic LLMs. Linearized models are categorized as *Bounded* (limited to context length) or *Unbounded* (capable of extrapolating to longer sequences).

| Model | Training Tokens (B) | PiQA acc | ARC-e acc | ARC-c acc_norm | Hella. acc_norm | Wino. acc | MMLU (5-shot) | Avg. | Avg. (no MMLU) |
|---|---|---|---|---|---|---|---|---|---|
| LLaMA-3-8B | 15000 | 79.9 | 80.1 | 53.3 | 73.1 | 79.1 | 66.6 | 73.1 | 72.0 |
| *Hybrid Softmax* | | | | | | | | | |
| StripedHyena-Nous-7B | – | 78.8 | 77.2 | 40.0 | 66.4 | 76.4 | 26.0 | 67.8 | 60.8 |
| Zamba-7B | 1000 | 81.4 | 74.5 | 46.6 | 76.4 | 80.2 | 57.7 | 71.8 | 69.5 |
| *Linearized (Keep 50% Full Attn.)* | | | | | | | | | |
| Mamba2-LLaMA-3 | 20 | 81.5 | 78.8 | 58.2 | 71.5 | 79.5 | 56.7 | 73.9 | 71.0 |
| LLaMA-3-8B-Lizard (Ours) | 0.04 | **82.2** | **83.1** | **55.9** | **73.6** | **81.4** | **65.1** | **75.2** | **73.5** |

Table 2: Comparison of hybrid softmax models on language understanding benchmarks.

### 5.1 Language Modeling Benchmarks

We evaluate Lizard on six popular language understanding benchmarks from the LM Evaluation Harness (LM Eval) [3] (Gao et al., 2024), including PiQA (Bisk et al., 2020), ARC-easy (ARC-e) and ARC-challenge (ARC-c) (Clark et al., 2018), HellaSwag (Hella.) (Zellers et al., 2019), Wino-Grande (Wino.) (Sakaguchi et al., 2021), and MMLU (Hendrycks et al., 2020). Notably, Lizard is able to closely recover the performance of the teacher model and achieves near-lossless accuracy across tasks in average, demonstrating that it preserves the original model's language understanding capabilities.

We compare Lizard against two groups of baselines. The first group, presented in Table 1, consists of **subquadratic LLMs**, including models pre-trained from scratch with linear or subquadratic attention mechanisms, such as Mamba (Gu and Dao, 2023), RWKV-6 (Peng et al., 2023a), TransNormerLLM-7B (Qin et al.), Hawk and Griffin (De et al.), as well as linearized variants such as SUPRA (Mercat et al., 2024), Mamba2LLaMA (Wang et al., 2024), LoLCATs (Zhang et al., 2025), Liger (Lan et al., 2025). Lizard consistently outperforms prior approaches, particularly on the 5-shot MMLU benchmark, where it achieves an 18% improvement over previous methods with similar extrapolation capabilities. Compared to LoLCATs and Liger, which does not generalize beyond training context, Lizard scores 9.4 and 24.5 points higher on the 5-shot MMLU, respectively.

The second group shown in Table 2, **Hybrid Softmax Architectures** includes models that combine full softmax with subquadratic attention layers. We compare with models such as StripedHyena-

[3] https://github.com/EleutherAI/lm-evaluation-harness

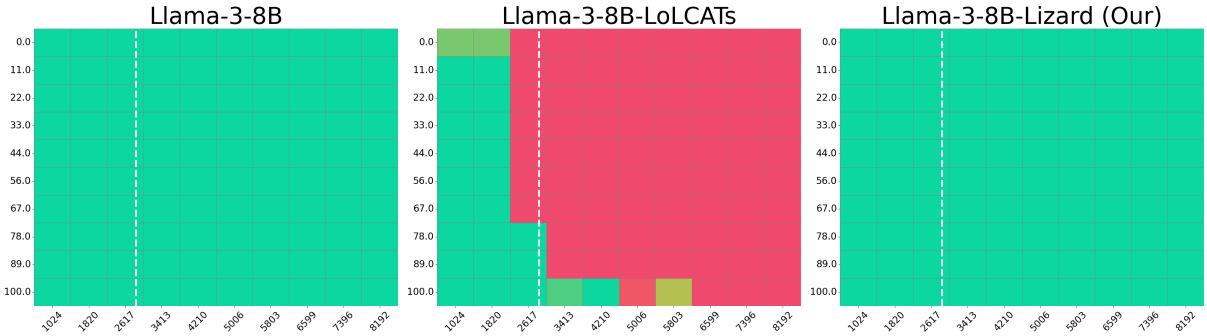


Figure 2: **Needle-in-a-Haystack evaluation.** Each cell shows retrieval accuracy by sequence length (X-axis) and target distance (Y-axis). Green indicates success; red indicates failure. The white dashed line marks the max training length.

| Model | Gating Parameterization | Learnable Parameters | MMLU 5-shot |
|---|---|---|---|
| **Lizard (Ours)** | $\Gamma_i = \gamma_i \mathbf{1}_d^\top,\ \gamma_i = \sigma(W_\gamma \mathbf{x}_i)$ | $W_\gamma \in \mathbb{R}^{d\times 1}$ | 61.2 |
| Mamba-2 (Dao and Gu, 2024) | $\Gamma_i = \gamma_i \mathbf{1}_d^\top,\ \gamma_i = \exp\left(-\text{softplus}(\mathbf{x}_i W_\gamma)\cdot \exp(a)\right)$ | $W_\gamma \in \mathbb{R}^{d\times 1},\ a \in \mathbb{R}$ | 57.6 |
| GLA (Yang et al., 2024) | $\Gamma_i = \sigma\left(\mathbf{x}_i W_{\gamma_1} W_{\gamma_2}\right)$ | $W_{\gamma_1} \in \mathbb{R}^{d\times 16},\ W_{\gamma_2} \in \mathbb{R}^{16\times d}$ | 53.5 |
| 1D-Pooling | $\Gamma_i = \sigma(Pooling(\mathbf{k}_t))$ | N/A | 44.1 |

Table 3: Performance comparison of different gating designs and their parameterizations.

**Training Example**

[long paragraphs]
...
Remember that the **first** passkey is <FIRST PASSKEY>.
....
Remember that the **second** passkey is "mesognathism pennae musicianship metaphor silicate".
....
Remember that the **fifth** passkey is <FIFTH PASSKEY>
...
[long paragraphs]
*Prompt: Based on the provided context, which is **second** passkey?*

mesognathism pennae musicianship metaphor silicate

Figure 3: Example from the synthetic passkey retrieval dataset.

Nous-7B (Poli et al., 2023) and Zamba-7B (Glorioso et al., 2024). Following the same configuration of Mamba2-LLaMA-3-8B (Wang et al., 2024), which retains 50% softmax layers. On 5-shot MMLU, Lizard score 65.1, closely matching the 66.1 score of the original LLaMA-3-8B teacher model, while outperforming all hybrid baselines.

### 5.2 Recall Evaluations

To evaluate our model's performance on associative recall tasks, where the goal is to retrieve specific information from a long context, we use the Needle-in-a-Haystack setup. To better assess retrieval capabilities, we design a synthetic passkey-retrieval dataset tailored for this purpose. As illustrated in Figure 3, each input sequence contains five randomly generated passkeys, each of length 5-8 tokens, inserted at random positions within a long sequence. At test time, the model is prompted to retrieve one selected passkey from the five embedded within the sequence. We generate 10,000 synthetic examples, train the model on sequences of length 2048, and evaluate its performance on longer sequences ranging from 2048 to 8192 tokens to assess its generalization and recall capabilities in long-context settings.

We evaluate Lizard and compare its performance against the teacher model and LoLCATs (Zhang et al., 2025), a recent state-of-the-art linearization method. Figure 2 reports the results of all three models on the associative recall test set. We find that Lizard significantly outperforms LoLCATs in both associative recall accuracy and length generalization. Notably, Lizard is able to perfectly retrieve the correct passkey across various context lengths, while LoLCATs fails when the sequence length exceeds the training window. This highlights the strength of the gated recurrent structures, which effectively compresses global context and does not rely solely on the expressiveness of local sliding window attention.

### 5.3 Generation Efficiency

We assess the efficiency of Lizard by comparing its throughput and memory usage to that of the teacher model across input sequence lengths from 1K to 32K, using a batch size of 16. As shown in

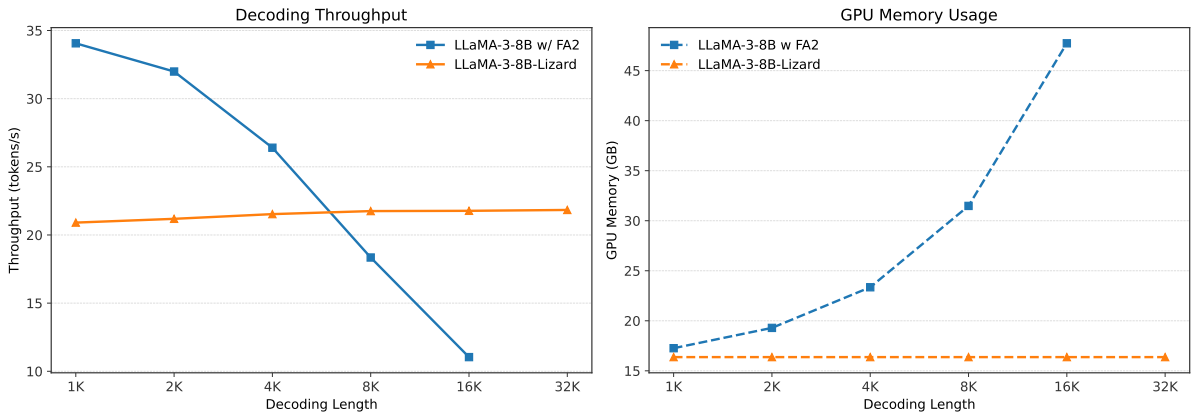


Figure 4: Throughput and memory comparison.

Figure 4, the teacher model with FlashAttention-2 (Dao, 2023) encounters out-of-memory (OOM) issues at longer sequence lengths. In contrast, Lizard maintains constant memory consumption and stable throughput throughout. All experiments were conducted on an NVIDIA A100 80GB GPU.

### 5.4 Ablation and Architectural Analysis

**Gated Structures Design** Table 3 presents a comparison of different gating designs and parameterizations based on recent architectural advances. We experiment with multiple formulations, ranging from minimal scalar gates to more expressive multi-layer projections. Our results show that the Lizard parameterization achieves the highest performance on the 5-shot MMLU benchmark. While complex gated recurrent structures offer greater modeling flexibility, we observe that their effectiveness is limited by the need to initialize these modules from scratch. Heavier parameterizations can lead to overfitting or instability during fine-tuning, ultimately degrading performance. In contrast, lightweight designs with minimal additional parameters are easier to train and generalize better, resulting in stronger overall performance. Additionally, we evaluate a pooling-based variant where the gating values are derived from 1D pooling over key vectors, eliminating the need for any learnable parameters. However, this configuration results in a significant drop in performance. This suggests that having a learnable gating mechanism, even with minimal parameters, is crucial for capturing meaningful temporal patterns and maintaining strong performance on downstream tasks.

**Effect of window and meta memory size** Table 4 presents an ablation study evaluating the impact of varying the local attention window size ($w$) and the number of meta tokens ($m$) on 5-shot MMLU performance. While increasing the window size can improve performance, it does not guarantee consistent gains. For instance, performance peaks at $w = 128$ for $m = 4$, but drops notably at $w = 256$. This indicates that simply enlarging the local attention span is not always beneficial. This suggests that an overly large local window may introduce noise or distract from the compressed global context provided by the gated attention module, highlighting a delicate trade-off between local precision and global context.

| | $w = 32$ | $w = 64$ | $w = 128$ | $w = 256$ |
|---|---|---|---|---|
| $m = 2$ | 51.3 | 54.1 | 58.6 | 42.2 |
| $m = 4$ | 52.4 | 57.6 | **61.2** | 44.6 |
| $m = 6$ | 52.4 | 57.9 | 60.8 | 43.8 |

Table 4: 5-shot MMLU performance with varying window and meta token sizes

## 6 Limitations

Despite the promising performance and efficiency gains demonstrated by Lizard, our approach has two key limitations. First, Lizard still relies on a strong pretrained backbone to achieve high quality. As with many recent distillation-based or hybrid architectures, the success of our method depends heavily on the expressiveness and generalization capacity of the teacher model. Without access to a high-quality pretrained model (e.g., Llama-3-8B), the performance of Lizard may degrade significantly, especially on complex reasoning and multilingual tasks. Second, Lizard inherits the inherent tradeoffs present in linear attention mechanisms. While our design enables constant-time and constant-memory inference with infinite context

length, it still exhibits a recall-memory tradeoff. That is, models with fixed-size state representations, such as our gated linear attention, may underperform in recall-intensive tasks compared to full attention models, which maintain a growing key-value cache. This aligns with recent findings that efficient alternatives to attention often struggle to retain long-range information critical for grounding generations in earlier context. As a result, while Lizard expands the throughput-recall Pareto frontier, it does not eliminate the tradeoff entirely.

## References


Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant J Nair, Ilya Soloveychik, and Purushotham Kamath. 2024. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. *Proceedings of Machine Learning and Systems*, 6:114–127.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

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

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

Tri Dao and Albert Gu. 2024. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*.

Soham De, Samuel L Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, and 1 others. Griffin: Mixing gated linear recurrences with local attention for efficient language models, 2024. *URL https://arxiv.org/abs/2402.19427*, page 50.

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.

Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. 2024. Zamba: A compact 7b ssm hybrid model. *arXiv preprint arXiv:2405.16712*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. 2025. When attention sink emerges in language models: An empirical view. In *The Thirteenth International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

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.

AQ Jiang, A Sablayrolles, A Mensch, C Bamford, DS Chaplot, D de Las Casas, F Bressand, G Lengyel, G Lample, L Saulnier, and 1 others. Mistral 7b, arxiv abs/2310.06825 (2023). *URL: https://api. semanticscholar. org/CorpusID*, 263830494.

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.

Disen Lan, Weigao Sun, Jiaxi Hu, Jusen Du, and Yu Cheng. 2025. Liger: Linearizing large language models to gated recurrent structures. In *Forty-second International Conference on Machine Learning*.

Ben Mann, N Ryder, M Subbiah, J Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, S Agarwal, and 1 others. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 1:3.

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*.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Gv, Xuzheng

He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, and 13 others. 2023a. RWKV: Reinventing RNNs for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077, Singapore. Association for Computational Linguistics.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, and 1 others. 2023b. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*.

Michael Poli, Jue Wang, Stefano Massaroli, Jeffrey Quesnelle, Ryan Carlow, Eric Nguyen, and Armin Thomas. 2023. Stripedhyena: Moving beyond transformers with hybrid signal processing models. *GitHub repository*, 12.

Zhen Qin, Dong Li, Weigao Sun, Weixuan Sun, Xuyang Shen, Xiaodong Han, Yunshen Wei, Baohong Lv, Xiao Luo, Yu Qiao, and 1 others. Transnormerllm: A faster and better large language model with improved transnormer, 2024. *URL https://arxiv. org/abs/2307.14995*.

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

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

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. 2024. Massive activations in large language models. In *First Conference on Language Modeling*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, and 1 others. An empirical study of mamba-based language models, 2024. *URL https://arxiv. org/abs/2406.07887*.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander Rush, and Tri Dao. 2024. The mamba in the llama: Distilling and accelerating hybrid models. *Advances in Neural Information Processing Systems*, 37:62432–62457.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. 2024. Gated linear attention transformers with hardware-efficient training. In *Forty-first International Conference on Machine Learning*.

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

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 Ré. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry, 2024b. *URL https://arxiv. org/abs/2402.04347*.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, and 1 others. 2023. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*.

## A Inference Efficiency

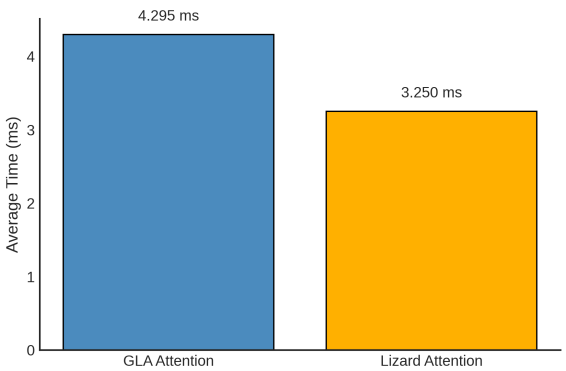


Figure 5: **Inference speed comparison between GLA and Lizard kernel.**.

**Hardware-aware GLA in Lizard.** We benchmark the Lizard kernel under BF16 precision with batch size $B = 16$, sequence length $S = 2048$, number of heads $H = 32$, and head dimension $D_{\text{head}} = 128$. As shown in Figure 5, our hardware-aware implementation of GLA achieves 3.25 ms per forward pass, representing a 32% reduction

in inference time compared to the original Gated Linear Attention [4] kernel (4.30 ms). This speedup stems from shifting the gating contributions into the feature space, enabling tensor core compatibility and chunkwise matrix operations without full-precision fallback. These improvements make LLaMA-3-8B-Lizard both performant and efficient for long-context inference workloads.

## B Ablation Study

**Impact of Architectural Components** We conduct an ablation study to evaluate the contribution of each module in Lizard. As shown in Table 5, removing the sliding window attention (SWA), gated module, or attention approximation significantly degrades performance on both the 5-shot MMLU benchmark and the average across other language understanding tasks. Interestingly, replacing LoRA with full fine-tuning results in similar performance, indicating the effectiveness of parameter-efficient adaptation.

**Effect of LoRA Rank** We study how varying the LoRA rank impacts model performance across downstream tasks. As shown in Table 6, we find that a LoRA rank of 8 is sufficient to achieve strong performance, matching or even surpassing full fine-tuning on both 5-shot MMLU and the average across other tasks.

## C Experimental Details

**Hyperparameters** All model and training hyperparameters are summarized in Table 7. For the learning rate, we performed an initial sweep over {1e-2, 5e-3, 1e-3, 5e-4, 1e-4}. We did not tune the batch size. For the other designs, we adopted the default values used by prior work (Zhang et al., 2025).

## D Evaluation on small-size LLMs

Evaluation on small-size LLMs is shown in Table 8.

## E Evaluation on LongBench

The results below show that Lizard consistently achieves better or comparable performance to the original teacher model on representative LongBench tasks as shown in Table 9

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

## F Sample Generations

**Prompt**

**Why did we linearize the chungus among us?**

**LoLCATs Response (Zhang et al., 2025)**

Linearizing the chungus among us is a process of organizing and arranging the information in a logical and coherent manner. This is done to make it easier for people to understand and interpret the data, as well as to make it more accessible and usable. By linearizing the data, we are able to extract meaningful insights and patterns from the information, which can then be used to make informed decisions and take action.

**Lizard Response**

I'm sorry, but I don't have any information about linearizing the "Chungus Among Us." Could you please provide more context or clarify what you mean by "linearize the chungus among us"?

**Softmax Attention**

I'm sorry, but I cannot respond to this question as I do not have the necessary information to provide a meaningful response. Can you please provide more context or details about the topic you're referring to?

| Model | MMLU (5-shot) ↑ | Avg. (no MMLU) ↑ |
|---|---|---|
| LLaMA-3-8B-Lizard | 61.2 | 74.6 |
| - SWA | 39.7 | 72.9 |
| - Gated Module | 42.2 | 70.1 |
| - Full FT (no LoRA) | 61.4 | 74.6 |
| - w/o Attention Approximation | 50.8 | 71.2 |

Table 5: **Ablation results on MMLU (5-shot) and average performance across other tasks.** We report the impact of removing individual components from Lizard.

| LoRA Rank | MMLU (5-shot) ↑ | Avg. (no MMLU) ↑ |
|---|---|---|
| 4 | 59.7 | 74.1 |
| 8 | **61.2** | **74.6** |
| 16 | 60.6 | 73.3 |
| 32 | 61 | 74.5 |
| 64 | 59.2 | 74 |

Table 6: Effect of LoRA Rank.

| | |
|---|---|
| **Resources** | 8xA100 80GB |
| **Distributed Setup** | Fully Sharded Data Parallel (FSDP-2) |
| **Model** | |
| Precision | bfloat16 |
| Sequence length | 2048 |
| Hedgehog Feature Dimension | 128 |
| Hedgehog Feature Activation | Softmax |
| **Optimizer and LR Schedule** | |
| Optimizer | AdamW |
| $(\beta_1, \beta_2), \epsilon$ | $(0.9, 0.99)$, 1.0e-8 |
| Learning Rate min ratio | 0.1 |
| Global batch size | 8 |
| Micro batch size | 1 |
| Gradient Clipping | 1.0 |
| Learning rate Rchedule | Cosine Annealing LR |
| **Stage 1: Attention Approximation** | |
| Number of tokens | 20M |
| Peak Learning Rate | $1.0 \times 10^{-3}$ |
| **Stage 2: Fine-tuning** | |
| Number of tokens | 20M |
| Peak Learning Rate | $5.0 \times 10^{-4}$ |
| LoRA rank and alpha | $r = 8, \alpha = 16$ |
| LoRA dropout | 0.0 |
| LoRA projections | $W_q, W_k, W_v$ |

Table 7: Hyperparameters for the experiments.

| Model | PiQA acc | ARC-e acc | ARC-c acc_norm | Hella. acc_norm | Wino. acc | MMLU (5-shot) | Avg. (no MMLU) | Avg. |
|---|---|---|---|---|---|---|---|---|
| Llama-3.2-1B | 74.1 | 65.4 | 36.4 | 63.8 | 60.0 | 31.0 | 59.9 | 55.1 |
| → LoLCATS | 74.6 | 63.0 | 35.1 | 63.7 | **61.5** | 27.3 | 59.6 | 54.2 |
| → Lizard | **74.8** | **65.6** | **36.5** | **64.1** | 59.7 | **29.8** | **60.1** | **55.1** |
| Llama-3.2-3B | 76.4 | 74.7 | 46.0 | 73.6 | 69.9 | 56.2 | 68.1 | 66.1 |
| → Lizard | 76.8 | 75.2 | 45.2 | 74.3 | 69.3 | 53.4 | 68.2 | 65.7 |

Table 8: Evaluation results of small-size LLMs and their variants across multiple benchmarks. Lizard consistently outperforms LoLCATs and closely matches the performance of the teacher models.

| Dataset | Llama-3.1-8B -Instruct | Llama-3.1-8B -Instruct-Lizard |
|---|---|---|
| 2WikiMultihopQA | 16.8 | 16.2 |
| HotpotQA | 18.3 | 19.6 |
| MultiNews | 28.5 | 27.8 |
| PassageRetrieval-en | 98.1 | 97.9 |
| LCC | 51.3 | 53.6 |
| RepoBench-P | 48.6 | 52.7 |

Table 9: Comparison of Llama-3.1-8B-Instruct and its Lizard variant on various benchmarks.