Empirical Evaluation of Knowledge Distillation from Transformers to Subquadratic Language Models

Anonymous ACL submission

Abstract

Knowledge distillation is a widely used technique for compressing large language models (LLMs), in which a smaller student model is trained to mimic a larger teacher model. Typically, both the teacher and student models are Transformer-based architectures, leveraging softmax attention for sequence modeling. However, the quadratic complexity of self-attention during inference remains a significant bottleneck, motivating the exploration of subquadratic alternatives such as structured state-space models (SSMs), linear attention, and recurrent architectures. In this work, we systematically evaluate the transferability of knowledge distillation from a Transformer teacher model to eight subquadratic student architectures. Our study investigates which subquadratic model can most effectively approximate the teacher model's learned representations through knowledge distillation, and how different architectural design choices influence the training dynamics. We further investigate the impact of initialization strategies, such as matrix mixing and query-key-value (QKV) copying, on the adaptation process. Our empirical results on multiple NLP benchmarks provide insights into the trade-offs between efficiency and performance, highlighting key factors for successful knowledge transfer to subquadratic architectures.

1 Introduction

The Transformer architecture (Vaswani et al., 2017) has led to significant advances in natural language processing (NLP) by enabling highly scalable and parallelizable training of language models (LMs). The core of its effectiveness is the self-attention mechanism, which produces contextualized token representations across long sequences. However, the quadratic computational complexity of self-attention, $\mathcal{O}(n^2)$ with respect to sequence length, leads to high inference costs for long sequences, posing challenges for resource-constrained applications.

Rise of linear complexity architectures. To address this limitation, alternative architectures have been proposed that reduce the complexity of self-attention. These models achieve subquadratic, and often linear, complexity with $\mathcal{O}(n)$. These include linear attention models (Katharopoulos et al., 2020), structured state-space models (SSMs) (Gu and Dao, 2024; Dao and Gu, 2024), and recurrent neural networks (RNNs) with improved gating mechanisms (Sun et al., 2023). These architectures aim to reduce computational overhead while maintaining competitive modeling capabilities.

While these architectures offer theoretical efficiency gains, pretraining them from scratch is prohibitively expensive and training-intensive. Moreover, their training dynamics remain less well understood than those of Transformers, making optimization more challenging. To avoid costly pretraining, we apply knowledge distillation (Hinton et al., 2015) from capable Transformer models into subquadratic architectures, aiming to retain their language modeling capabilities while significantly improving efficiency. Although knowledge distillation is typically applied between models of the same architecture, we adapt this paradigm to distill from a Transformer teacher into various subquadratic student models.

Contributions. To assess the feasibility of transferring knowledge from Transformer-based models into subquadratic architectures, we conduct a controlled empirical study involving eight distinct architectures (see Figure 1 for an overview of our approach). Our study aims to quantify the extent to which different architectures preserve the inductive biases and representations learned by attention-based Transformers, and to analyze the effect of various alignment strategies on downstream task performance.

Specifically, we incorporate several alignment

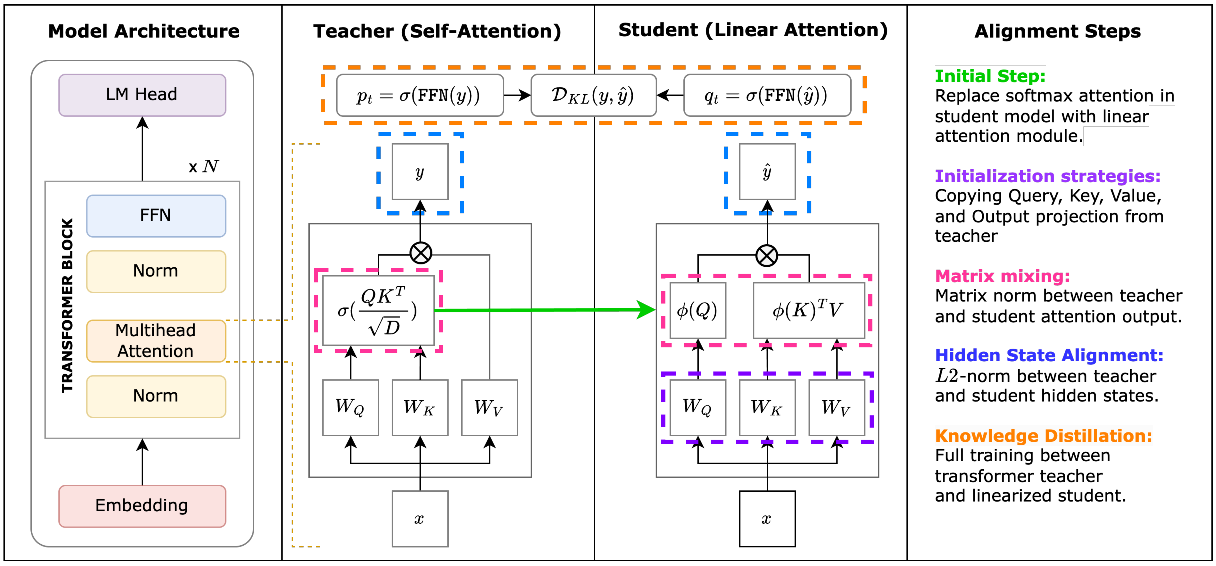

Figure 1: Overview of our knowledge distillation approach. We replace the softmax attention mechanism in transformer models with various subquadratic modules and train the resulting models using knowledge distillation and additional alignment techniques.

strategies to facilitate effective knowledge transfer, including *matrix mixing* (aligning the student's attention mechanism with the teacher's self-attention), *QKV copying* (initializing the student's query, key, and value projections with those learned by the teacher), and *hidden-state alignment* (minimizing the divergence between intermediate representations of the student and teacher models).

Our empirical results reveal significant performance disparities across different subquadratic architectures, with xLSTM (Beck et al., 2024) achieving the highest average performance. Additionally, leveraging all advanced alignment techniques combined yields notable improvements. We summarize our contributions as follows:

- We present a systematic empirical evaluation of knowledge distillation into subquadratic models, comparing alignment techniques and downstream task performance.
- We analyze the effectiveness of various alignment strategies, such as hidden-state alignment, and direct and indirect token mixer alignment, providing insights into the role of structural compatibility in student-teacher adaption
- We release our code and models to facilitate further research on linearizing attention-based Transformer models.

2 Preliminaries and Related Work

With the introduction of Transformers (Vaswani et al., 2017), the softmax attention mechanism became the *de facto* standard for language modeling. However, it has a computational complexity of $\mathcal{O}(n^2 d)$, where n is the sequence length and d the hidden dimension of the model.

Parallel form of softmax attention. Given an input sequence $\boldsymbol{x} \in \mathbb{R}^{n \times d}$, the model computes projected "query," "key," and "value" representations as $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} = \boldsymbol{x}\boldsymbol{W}_Q, \boldsymbol{x}\boldsymbol{W}_K, \boldsymbol{x}\boldsymbol{W}_V$, where $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{d \times d}$ are learnable weight matrices. The output $\boldsymbol{y} \in \mathbb{R}^{n \times d}$ of softmax attention is computed as:

$$\boldsymbol{y} = softmax((\boldsymbol{Q}\boldsymbol{K}^\top) \odot \boldsymbol{M})\boldsymbol{V}, \qquad (1)$$

where $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ is a causal mask to prevent the model from attending to future tokens. Thus, softmax attention allows each token to attend to all tokens in the sequence by computing similarity scores between queries and keys, and using these scores to compute a weighted sum of value vectors.

Recurrent form for inference. While self-attention can be computed in parallel during training (Equation (1)), which is efficient on GPUs, inference requires sequential computation. At each decoding step, a newly generated token $\boldsymbol{x}_t \in \mathbb{R}^{1 \times d}$ attends to all previous tokens. Thus, the recurrent formulation of softmax attention is given by

$$\boldsymbol{y}_t = \frac{\sum_{i=1}^{t} exp(\boldsymbol{q}_t \boldsymbol{k}_i^\top)\boldsymbol{v}_i}{\sum_{i=1}^{t} exp(\boldsymbol{q}_t \boldsymbol{k}_i^\top)}, \qquad (2)$$

Architecture	Recurrence	Decay Term
mLSTM (Beck et al., 2024)	$\boldsymbol{S}_t = f_t\boldsymbol{S}_{t-1} + i_t\boldsymbol{v}_t\boldsymbol{k}_t^\top$	dynamic
GLA (Yang et al., 2024)	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}\text{Diag}(\boldsymbol{\alpha}_t) + \boldsymbol{v}_t\boldsymbol{k}_t^\top$	dynamic
RetNet (Sun et al., 2023)	$\boldsymbol{S}_t = \gamma\boldsymbol{S}_{t-1} + \boldsymbol{v}_t\boldsymbol{k}_t^\top$	static
MetaLA (Chou et al., 2024)	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}\text{Diag}(\boldsymbol{\alpha}_t) + \boldsymbol{v}_t(1-\alpha_t)^\top$	dynamic
DeltaNet (Yang et al., 2025)	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}(\alpha(\text{I} - \beta_t\boldsymbol{k}_t\boldsymbol{k}_t^\top)) + \beta\boldsymbol{v}_t\boldsymbol{k}_t^\top$	dynamic
Linear Attention	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \boldsymbol{v}_t\phi(\boldsymbol{k}_t)^\top$	-
+ Vanilla (Choromanski et al., 2022)	where $\phi(x) = elu(x) + 1$	-
+ ReBased (Aksenov et al., 2024)	where $\phi(x) = (\gamma \cdot norm(x) + \beta)^2$	-
+ Hedgehog (Zhang et al., 2024b)	where $\phi(x) = \exp(Wx + b)$	-

Table 1: Overview of all architectures and their recurrent form under evaluation. $\boldsymbol{S}_t \in \mathbb{R}^{d\times n}$

where $\boldsymbol{q}_t, \boldsymbol{k}_t, \boldsymbol{v}_t = \boldsymbol{x}_t\boldsymbol{W}_Q, \boldsymbol{x}_t\boldsymbol{W}_K, \boldsymbol{x}_t\boldsymbol{W}_V$. As a result, autoregressive inference incurs growing memory and computational costs, since each new token must recompute attention over a ever-expanding set of keys and values $\{\boldsymbol{k}_i, \boldsymbol{v}_i\}_{i=1}^{t-1}$.

Linear complexity with kernelized feature maps. Katharopoulos et al. (2020) introduce a kernel-based approximation of the softmax attention by applying a feature map $\phi(\cdot)$, such that:

$$softmax(\boldsymbol{Q}\boldsymbol{K}^\intercal) \approx \phi(\boldsymbol{Q})\phi(\boldsymbol{K})^\intercal. \tag{3}$$

Leveraging the associative property of matrix multiplication, we can rewrite the recurrent form of attention:

$$\boldsymbol{y}_t = \frac{\sum_{i=1}^t \phi(\boldsymbol{q}_t)\phi(\boldsymbol{k}_i)^\intercal\boldsymbol{v}_i}{\sum_{i=1}^t \phi(\boldsymbol{q}_t)\phi(\boldsymbol{k}_i)^\intercal} \tag{4}$$

$$= \frac{\phi(\boldsymbol{q}_t)\sum_{i=1}^t \phi(\boldsymbol{k}_i)^\intercal\boldsymbol{v}_i}{\phi(\boldsymbol{q}_t)\sum_{i=1}^t \phi(\boldsymbol{k}_i)^\intercal}. \tag{5}$$

Unlike the standard softmax formulation (cf. Equation (2)), which scales with $\mathcal{O}(n^2d)$, the kernelized approximation (cf. Equation (5)) reduces the complexity to $\mathcal{O}(nd^2)$.

Existing Linear Attention Models. Several feature map strategies have been proposed to address issues such as negative attention weights and training instabilities. **TransNormer** (Qin et al., 2022) and Retention Networks (**RetNet**) (Sun et al., 2023) identify instabilities in the normalization term of linear attention and replace classical normalization with GroupNorm (Wu and He, 2018). **ReBased** (Aksenov et al., 2024) introduces a learnable polynomial kernel that adapts during training, mitigating the limitations of fixed feature maps. Similarly, **Hedgehog** (Zhang et al., 2024b) extends this idea by learning feature maps using single-layer networks, which preserve low-entropy attention weights and enforce monotonicity of query-key dot products. **DeltaNet** (Yang et al., 2025) introduces a delta update rule designed to improve memory efficiency and recall.

Beyond kernel-based methods, recent work incorporates recurrent structures into linear attention models. This includes Linear Recurrent Unit (**LRU**) (Orvieto et al., 2023) and Receptance Weighted Key Value (**RWKV**) (Peng et al., 2023, 2024), which both model sequence information through gated recurrence. Several works explore alternative gating parameterizations to improve selective information flow. Examples include Gated Linear Attention (**GLA**) (Yang et al., 2024), Hierarchically Gated Recurrent Neural Networks (**HGRN/HGRN2**) (Qin et al., 2023, 2024), **Griffin** (De et al., 2024), and **mLSTM** (Beck et al., 2024). **Mamba2** (Dao and Gu, 2024) proposes a variant of linear attention based on state-space models from control theory, where sequence dynamics are modeled using latent state variables. Other approaches, such as Meta Linear Attention (**MetaLA**) (Chou et al., 2024) and Zimerman et al. (2024), present unified theoretical frameworks that improve the approximation of softmax attention while reducing parameter redundancy.

Linearizing softmax attention in pretrained LMs. Rather than training linear models from scratch, several approaches (Kasai et al., 2021; Mao, 2022) replace softmax attention with linear attention blocks in pretrained Transformers and apply knowledge distillation (Hinton et al., 2015). More recent work refines this paradigm with increasingly targeted strategies. SUPRA (Mercat et al., 2024) introduces a scalable uptraining framework to convert pretrained Transformers into recurrent architectures. LoLCATs (Zhang et al., 2024a) combines low-rank adaptation (Hu et al., 2021) with attention transfer to efficiently approximate softmax attention.. MOHAWK (Bick et al., 2024) employs a staged distillation pipeline that progressively aligns the student with its Transformer teacher. Further extensions include Mamba-LLaMA (Wang et al., 2025), which applies progressive distillation with instruction tuning, and LIGER (Lan et al., 2025), which reuses Transformer weights to construct gating modules for a range of subquadratic models, incorporating sliding-window attention. Finally, Yueyu et al. (2025) linearize Qwen-2.5 using RWKV-7 blocks, combining hidden-state alignment with word-level distillation. As Mamba has already become a common target for such distillation efforts, we focus our analysis on alternative subquadratic architectures.

3 Methodology

The first step in linearizing softmax attention-based language models involves replacing the attention block with a linear attention module (see Table 1). The common approach for training such linearized language models is to apply knowledge distillation (KD) from a softmax attention-based teacher model to a student model, thereby avoiding the need for expensive pretraining. The student model is trained using two objectives: *(1)* cross-entropy loss for next-token prediction and *(2)* the Kullback-Leibler (KL) divergence between output distributions of the teacher and the student. The total distillation loss $\mathcal{L}_{\text{KD}}$ is defined as:

$$\mathcal{L}_{\text{KD}} = \mathcal{L}_{\text{CE}} + \lambda \cdot \mathcal{L}_{\text{KL}}, \tag{6}$$

where $\mathcal{L}_{\text{CE}}$ is the cross-entropy loss and $\mathcal{L}_{\text{KL}}$ is the KL divergence loss. λ is a scaling factor controlling the contribution of each term. The KL divergence loss is given by:

$$\mathcal{L}_{KL} = \frac{1}{N}\sum_{i=1}^{N} \text{KL}\big(p_T^{(i)} \| p_S^{(i)}\big), \tag{7}$$

where N is the number of tokens, KL denotes the Kullback-Leibler divergence, and $p_T^{(i)}$ and $p_S^{(i)}$ are the output probability distributions of the teacher and student models, respectively, for the i-th token. We provide a conceptual overview of these two steps in Figure 1 and introduce additional alignment techniques in the following sections. As a preliminary verification, we confirm that knowledge distillation significantly improves student model performance and that parameter copying (e.g., copying the teacher's MLP layers, embeddings, and language modeling head) provides an effective starting point, consistent with prior findings (Appendix A).

3.1 Additional Alignment Improvements

In the following section, we present refined alignment techniques to improve the distillation process between the transformer teacher model and the linearized student.

Attention matrix alignment. This approach aims to align the teacher's self-attention matrix with that of the linearized student model. However, this is non-trivial, since linear attention models do not explicitly compute full attention matrices. Prior work reconstructs approximate attention matrices from linear counterparts to enable alignment (Zhang et al., 2024b,a). In particular, the MOHAWK framework (Bick et al., 2024) proposes a method based on minimizing the Frobenius norm between the teacher's self-attention matrix and the student's materialized matrix at each layer, referred to as "matrix mixing."

We extend this approach empirically to all eight linear architectures listed in Table 1. The matrix mixing loss is defined as:

$$\mathcal{L}_{\text{MM}} = \frac{1}{L}\sum_{i=1}^{L} \|\text{AttnMat}_{\text{T}}^{(i)} - \text{AttnMat}_{\text{S}}^{(i)}\|_F, \tag{8}$$

where L is the number of layers, $\text{AttnMat}_{\text{T}}^{(i)}$ is the teacher's self-attention matrix at layer i, and $\text{AttnMat}_{\text{S}}^{(i)}$ is the materialized attention matrix of the student at the corresponding layer.

Hidden state alignment. An additional alignment strategy introduced in the MOHAWK framework is hidden state alignment, which encourages the student model's hidden representations to remain close to those of the teacher. This is achieved by minimizing the L_2-norm between corresponding hidden states at each layer. The hidden state alignment loss is defined as:

$$\mathcal{L}_{\text{H2H}} = \frac{1}{L}\sum_{i=1}^{L} \|h_T^{(i)} - h_S^{(i)}\|_2^2, \tag{9}$$

where L is the number of layers, $h_T^{(i)}$ is the hidden state of the teacher model at layer i, and $h_S^{(i)}$ is the corresponding hidden state of the student model. This loss encourages the student model to preserve intermediate representations of the teacher, thereby improving structural alignment between the models.

4 Experimental Setup

For our empirical evaluation, we consider eight subquadratic architectures as student models, listed in Table 1. We use SmolLM-360M (Allal et al., 2025) as our softmax attention-based teacher model, which is built on the Llama architecture (Touvron et al., 2023). To construct a linearized student model, we retain the teacher's normalization layers, MLP blocks, embedding layers, and language modeling head while replacing the self-attention mechanism with the corresponding linearized attention module (see Table 1). We show the exact parameter counts for each model in Appendix C.

We then train the student model using knowledge distillation, with additional alignment techniques progressively incorporated as described in Section 3. After training, we evaluate the student model's performance on various downstream tasks.

4.1 Training Dataset and Evaluation

All student models are trained on a 3B-token subset of the FineWeb dataset (Penedo et al., 2024), a cleaned and deduplicated English web corpus. Text is concatenated and chunked into fixed-length sequences of 512 tokens. We allocate fixed budgets for alignment objectives: 80M tokens for matrix mixing and 160M for hidden-state alignment, following the MOHAWK setup (Bick et al., 2024). For evaluation, we follow LM-Eval-Harness (Gao et al., 2023) to assess six zero-shot tasks: LAMBADA (Paperno et al., 2016), WinoGrande (Sakaguchi et al., 2019), ARC (easy/challenge) (Clark et al., 2018), PIQA (Bisk et al., 2019), and HellaSwag (Zellers et al., 2019). LAMBADA is reported as the mean of its Standard and OpenAI variants. To evaluate long-context capabilities, we include five subsets from LongBench (Bai et al., 2024): WikiMQA, MultiFieldQA, NarrativeQA, TREC, and TriviaQA. Inputs exceeding the context window are left-truncated.

4.2 Training Details

We largely follow the training setup proposed in MOHAWK, using the Adam (Kingma and Ba, 2017) optimizer for matrix mixing, hidden state alignment and end-to-end training. For learning rate scheduling, we apply a stable decay schedule with warmup during matrix mixing phase and a linear schedule for end-to-end training, which we found to yield more stable results across all model variants. The maximum learning rate was set to 1×10^{-3}, with a batch size of 48. We note that MOHAWK uses only the KL divergence as its final loss, whereas we additionally optimize with a cross-entropy loss term (see Equation (6)), as it is widely adopted in distillation setups and aligns with its use in many practical implementations (Sanh et al., 2019; Jiao et al., 2020; Haller et al., 2024). We primarily use FLA (Yang and Zhang, 2024) for model implementations, PyTorch (Paszke et al., 2019) along with the Hugging Face Transformers and Datasets libraries (Wolf et al., 2020; Lhoest et al., 2021) for model training, inference, and dataset management. We also compared the use of Frobenius norm vs. mean squared error (MSE) loss for matrix mixing and found both losses to perform similarly (Appendix A). Based on this observation, we opted for Frobenius norm alignment in our experiments due to its conceptual alignment with prior approaches (Bick et al., 2024).

5 Experiments and Results

5.1 Experiment 1: Downstream Evaluation

Our first experiment aims to answer which subquadratic architectures are best suited for knowledge distillation from a Transformer-based teacher. To this end, we compare 8 architectures under different applications of the three phases of the MOHAWK framework: Stage 3 represents a full finetuning of the architecture and is always applied. Stages 1 and 2 correspond to attention matrix alignment and hidden state alignment, respectively. Applying all three phases constitutes to the full MOHAWK setup.

As a point of reference, we include two configurations where the student is also based on the LLama architecture: one where a newly initialized LLama-based student is trained from the teacher (Llama→$\text{Llama}_{student}$) and a sanity check in which the full teacher model is copied into the student and then continuously fine-tuned (Llama→$\text{Llama}_{fullcopy}$). Table 2 shows the results of this comparison. We make the following observations:

Recoverage of linearized models. Among all student architectures, xLSTM, GLA, and MetaLA consistently achieve the highest recoverage scores

Model	Stages	Lamb. acc.	WinoG. acc.	Arc-E acc. norm.	Arc-C acc. norm.	PIQA acc. norm	HellaS. acc. norm.	Avg.↑	Rec.
SmolLM-360M (Teacher)	-	41.33	56.51	63.72	36.01	71.49	53.37	53.73	-
Llama→Llama$_{fullcopy}$	3	40.88	56.04	63.01	36.35	71.44	53.59	53.55	-
Llama→Llama$_{student}$	3	33.58	53.20	58.38	32.08	70.57	47.36	49.19	-
Llama→Llama$_{student}$	$2+3$	40.75	56.99	63.43	36.26	71.60	53.10	**53.68**	99.90%
Llama→Llama$_{student}$	$1+2+3$	40.89	56.69	63.30	36.18	70.95	53.03	53.50	-
Llama→xLSTM	3	32.06	54.54	59.30	31.83	70.67	48.34	49.45	-
Llama→xLSTM	$2+3$	34.44	54.46	59.72	32.68	71.49	49.89	50.44	-
Llama→xLSTM	$1+2+3$	35.71	56.43	60.40	32.51	70.95	50.37	**51.06**	95.03%
Llama→MetaLA	3	32.17	53.83	58.04	31.66	70.95	47.99	49.10	-
Llama→MetaLA	$2+3$	36.60	54.70	60.56	32.51	70.67	50.40	50.90	-
Llama→MetaLA	$1+2+3$	36.39	54.22	61.07	32.68	71.22	50.21	**50.95**	94.82%
Llama→GLA	3	32.74	53.59	57.95	31.66	70.95	48.40	49.21	-
Llama→GLA	$2+3$	34.52	53.75	61.20	32.25	70.57	50.15	50.40	-
Llama→GLA	$1+2+3$	35.05	53.67	60.94	32.42	70.35	50.17	**50.43**	93.85%
Llama→RetNet	3	30.01	53.04	57.41	32.17	69.86	46.45	48.15	-
Llama→RetNet	$2+3$	32.32	55.33	59.13	31.23	70.51	48.47	**49.49**	92.10%
Llama→RetNet	$1+2+3$	31.54	53.83	59.97	32.00	70.35	48.47	49.35	-
Llama→DeltaNet	3	32.44	53.51	58.84	31.74	71.55	47.81	49.31	-
Llama→DeltaNet	$2+3$	28.28	52.49	57.32	31.74	70.46	46.38	**47.77**	88.90%
Llama→DeltaNet	$1+2+3$	28.38	52.01	56.86	31.83	70.18	45.98	47.54	-
Llama→VanillaLA	3	19.03	50.20	51.01	27.65	67.68	38.53	42.53	-
Llama→VanillaLA	$2+3$	31.74	53.91	56.90	31.83	69.75	46.99	**48.52**	90.30%
Llama→VanillaLA	$1+2+3$	30.94	53.75	55.68	31.48	70.02	46.33	48.03	-
Llama→Rebased	3	20.76	50.51	50.55	27.99	68.12	39.29	42.80	-
Llama→Rebased	$2+3$	31.77	53.35	58.25	30.97	69.80	47.60	48.62	-
Llama→Rebased	$1+2+3$	34.41	52.80	57.83	32.42	69.75	48.60	**49.30**	91.75%
Llama→Hedgehog	3	20.57	51.07	52.06	28.58	68.66	39.43	43.95	-
Llama→Hedgehog	$2+3$	30.94	53.83	56.94	31.14	69.75	46.45	**48.17**	89.65%
Llama→Hedgehog	$1+2+3$	30.72	53.99	56.99	30.38	70.57	46.18	48.13	-

Table 2: Results on Zero-Shot LM downstream benchmarks. All models, except the teacher model SmolLM-360M, were trained for 3B tokens of the FineWeb dataset. We provide two Llama-Llama results as upper bounds of transfer within the same architecture: (1) Llama→Llama$_{student}$, where a new transformer model is distilled from a teacher. (2) Llama→Llama$_{fullcopy}$, a sanity check where the teacher is fully copied into the student. We find that several subquadratic architectures, such as xLSTM and MetaLA, outperform the Llama→Llama$_{student}$ baseline.

across all training stage combinations, recovering up to 95% of the teacher model's performance. In contrast, models lacking dynamic decay mechanisms, like those with static or no decay terms, consistently underperform. This trend highlights the importance of explicit memory dynamics in preserving the inductive biases of the teacher during distillation.

Subquadratic architectures without decay term consistently underperform. Kernel-based attention models such as VanillaLA, Rebased, and Hedgehog fail to match the performance of recurrent or gated architectures, even when trained with advanced alignment strategies. Although Hedgehog incorporates learnable feature maps to approximate softmax attention, it does not outperform simpler baselines, indicating that capturing softmax-like properties alone is insufficient. These results highlight the importance of explicit memory mechanisms, such as decay or gating, for effectively transferring the teacher model's sequential reasoning capabilities.

Hidden state alignment substantially boosts performance, especially on tasks requiring long-range reasoning. We observe that hidden-state alignment and end-to-end training (Stages 2+3) yields consistent improvements across all architectures compared to full fine-tuning alone (Stage 3), with average gains of 1–3 points. These improvements are particularly pronounced on LAMBADA, a benchmark designed to test long-range dependency modeling. For example, MetaLA improves from

MODEL	STAGES	LAMB. acc.	WINOG. acc.	ARC-E acc. norm.	ARC-C acc. norm.	PIQA acc_norm	HELLAS. acc. norm.	AVG.↑
Llama→xLSTM	3	32.06	54.54	59.30	31.83	70.67	48.34	49.45
Llama→xLSTM$_{qkv}$	3	32.04	52.72	59.34	32.59	70.13	48.37	49.19
Llama→GLA	3	32.74	53.59	57.95	31.66	70.95	48.40	49.21
Llama→GLA$_{qkv}$	3	30.67	53.83	59.86	31.91	70.13	48.24	49.10
Llama→RetNet	3	30.01	53.04	57.41	32.17	69.86	46.45	48.15
Llama→RetNet$_{qkv}$	3	27.63	54.70	57.73	32.08	70.08	46.06	48.04
Llama→DeltaNet	3	32.44	53.51	58.84	31.74	71.55	47.81	49.31
Llama→DeltaNet$_{qkv}$	3	26.75	51.54	55.18	31.14	70.24	44.99	46.64
Llama→MetaLA	3	32.17	53.83	58.04	31.66	70.95	47.99	49.10
Llama→MetaLA$_{qkv}$	3	30.10	54.14	58.21	31.83	69.64	47.48	48.56
Llama→LA	3	19.03	50.20	51.01	27.65	67.68	38.53	42.53
Llama→LA$_{qkv}$	3	19.53	49.72	51.22	27.56	67.46	39.73	42.53
Llama→Rebased	3	20.76	50.51	50.55	27.99	68.12	39.29	42.80
Llama→Rebased$_{qkv}$	3	19.57	49.80	51.22	26.79	66.97	38.35	42.11
Llama→Hedgehog	3	20.57	51.07	52.06	28.58	68.66	39.43	43.95
Llama→Hedgehog$_{qkv}$	3	23.99	49.72	53.75	29.78	69.59	42.41	44.87

Table 3: Effect of copying query, key, value, and output projections from the teacher compared to random initialization.

30.10 to 36.60 accuracy, and Rebased from 19.57 to 31.77.

Attention matrix alignment only provides marginal improvements. Extending training to include attention matrix alignment (Stages 1+2+3) provides only marginal improvements over hidden state alignment alone (Stages 2+3), and primarily for architectures that already provide a strong baseline. For most architectures, this phase has negligible or even negative impact, indicating that attention matrix alignment is only beneficial when the student model is structurally capable of representing softmax-style interactions.

For full details on the convergence behavior across training stages, we provide per-stage plots in Appendix D.

5.2 Experiment 2: Impact of QKV Copying

We conduct an ablation experiment to investigate whether copying the query, key, and value and output projections from the teacher model provides a good initialization for more effective alignment. To this end, we train each model both with and without copying all projections from the Transformer teacher. The results are shown in Table 3. We find that, while copying each projection offers a helpful initialization, it is insufficient for effective knowledge transfer on its own. Only for Llama→Hedgehog do we observe a noticeable improvement. This suggests that additional alignment stages are necessary to address structural mismatches and enable effective distillation.

5.3 Experiment 3: Explicit vs. Implicit Approximation of Self-Attention

In this experiment, we investigate whether directly approximating the attention weights leads to better performance than aligning the attention hidden state. We compare two setups: In the first, we only train the parameters necessary to reconstruct the attention weights for a given linear attention model (taken from Experiment 2). In the second, we apply an implicit approximation by aligning the attention hidden state, which involves performing a whole forward pass of the token mixer. The results are depicted in Table 4. We observe that implicit approximation via hidden-state alignment slightly outperforms direct attention weight reconstruction in most cases, particularly for MetaLA and GLA. This suggests that fully engaging the token mixer during training allows the student to better internalize the teacher's inductive biases. However, the differences remain small, indicating that both strategies can support alignment, provided the model has sufficient structural capacity. Overall, implicit methods appear more robust across architectures.

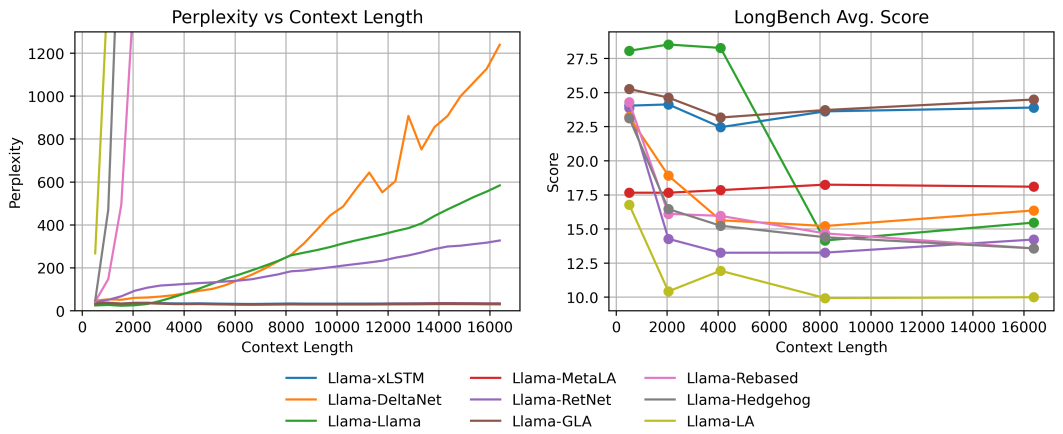

Figure 2: Long-context evaluation. Left: Perplexity over increasing context lengths. Right: LongBench scores. Models with dynamic decay terms (xLSTM, GLA, MetaLA) retain performance across increasing context lengths, while others show degradation.

MODEL	EXPLICIT	IMPLICIT
Llama→xLSTM	51.06	50.84
Llama→GLA	50.43	50.80
Llama→RetNet	49.35	49.64
Llama→MetaLA	50.95	51.00
Llama→DeltaNet	47.54	46.80
Llama→LA	47.88	48.03
Llama→Rebased	49.30	48.95
Llama→Hedgehog	48.13	48.01

Table 4: Final average performance across downstream benchmarks for each model and alignment variant. Full results are listed in Appendix E.

5.4 Experiment 4: Long-Context Evaluation

To assess the generalization ability of distilled models beyond standard sequence lengths, we evaluate them under long-context scenarios. First, we conduct controlled perplexity measurements on progressively longer input sequences to analyze each model's capacity to integrate and retain information over extended contexts. Second, we evaluate downstream performance using a subset of tasks from the LongBench benchmark, which reflects realistic, context-heavy applications. For inputs exceeding a model's maximum context length, we apply left-truncation. As shown in Figure 2, models with dynamic decay terms, like xLSTM, GLA, and MetaLA, maintain stable performance across longer sequences. In contrast, models without such mechanisms (e.g., DeltaNet, RetNet, LA) exhibit significant degradation, indicating limited long-range generalization.

6 Conclusion

Our study evaluates the effectiveness of distilling Transformer-based language models into a range of subquadratic architectures, focusing on alignment techniques such as QKV copying, attention-, and hidden-to-hidden alignment. We find that models with dynamic decay mechanisms consistently achieve the highest performance and recover well across training stages. In contrast, models without explicit memory dynamics - such as VanillaLA, Rebased, and Hedgehog - struggle to match the teacher, even with advanced alignment strategies. While QKV copying serves as a convenient initialization, it is insufficient alone, highlighting the importance of progressive alignment.

Among the evaluated techniques, hidden-to-hidden alignment emerges as the most reliable strategy for guiding student models toward the teacher's representations. Attention alignment can further support this process, though its benefits are more architecture-dependent. Notably, several subquadratic models, such as xLSTM, GLA, and MetaLA, achieve strong downstream performance while preserving the efficiency advantages of linearized attention.

As an outlook, preliminary results with scaled variants of xLSTM (Table 10) suggest promising gains with increased model capacity. Future work may explore scaling and adapting hidden-state alignment for larger models.

We release our training pipelines, architectures, and evaluation framework to support continued research on efficient model design and cross-architecture distillation.

Limitations

While our findings offer meaningful contributions, several limitations should be considered:

Lack of qualitative analysis. While we provide a broad empirical evaluation across diverse sub-quadratic backbones, we do not examine how the models' inductive biases manifest during the approximation of attention weights. A deeper analysis of the resulting attention patterns—e.g., spikiness, focus distribution, or alignment dynamics—could offer valuable insights into why certain architectures align better than others and inform future improvements to the distillation process.

Limited training data. The experiments were conducted with a constrained dataset, limiting our ability to assess the full generalization potential of the proposed techniques. Larger-scale training could reveal additional insights into model adaptation across diverse benchmarks.

Scaling to larger models. Our study primarily focuses on mid-sized models (350M to 500M parameters), and it remains an open question how well these techniques generalize to larger architectures. We hypothesize that matrix mixing may be more effective for larger models due to their increased hidden state dimensionality and greater representational capacity, allowing for a closer approximation of the teacher's attention matrix.

Despite these limitations, our findings provide a foundation for future work exploring more effective alignment techniques, improved compatibility layers, and novel training methodologies for efficient language models. Further research into alternative architectures and task-specific adaptations will be essential for advancing the deployment of subquadratic models in real-world applications.

References

Yaroslav Aksenov, Nikita Balagansky, Sofia Maria Lo Cicero Vaina, Boris Shaposhnikov, Alexey Gorbatovski, and Daniil Gavrilov. 2024. Linear transformers with learnable kernel functions are better in-context models. *Preprint*, arXiv:2402.10644.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2025. Smollm2: When smol goes big – data-centric training of a small language model. *Preprint*, arXiv:2502.02737.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. 2024. xlstm: Extended long short-term memory. In *Thirty-eighth Conference on Neural Information Processing Systems*.

Aviv Bick, Kevin Y. Li, Eric P. Xing, J. Zico Kolter, and Albert Gu. 2024. Transformers to ssms: Distilling quadratic knowledge to subquadratic models. *Preprint*, arXiv:2408.10189.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. *Preprint*, arXiv:1911.11641.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. 2022. Rethinking attention with performers. *Preprint*, arXiv:2009.14794.

Yuhong Chou, Man Yao, Kexin Wang, Yuqi Pan, Ruijie Zhu, Yiran Zhong, Yu Qiao, Jibin Wu, Bo Xu, and Guoqi Li. 2024. Metala: Unified optimal linear approximation to softmax attention map. *Preprint*, arXiv:2411.10741.

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. *Preprint*, arXiv:1803.05457.

Tri Dao and Albert Gu. 2024. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *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, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando De Freitas, and Caglar Gulcehre. 2024. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *Preprint*, arXiv:2402.19427.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. *Preprint*, arXiv:2312.00752.

Patrick Haller, Jonas Golde, and Alan Akbik. 2024. BabyHGRN: Exploring RNNs for sample-efficient language modeling. In *The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning*, pages 82–94, Miami, FL, USA. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *Preprint*, arXiv:1503.02531.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A. Smith. 2021. Fine-tuning pretrained transformers into rnns. *Preprint*, arXiv:2103.13076.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

Disen Lan, Weigao Sun, Jiaxi Hu, Jusen Du, and Yu Cheng. 2025. Liger: Linearizing large language models to gated recurrent structures. *Preprint*, arXiv:2503.01496.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander M. Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. *Preprint*, arXiv:2109.02846.

Huanru Henry Mao. 2022. Fine-tuning pre-trained transformers into decaying fast weights. *Preprint*, arXiv:2210.04243.

Jean-Pierre Mercat, Igor Vasiljevic, Sedrick Scott Keh, Kushal Arora, Achal Dave, Adrien Gaidon, and Thomas Kollar. 2024. Linearizing large language models. *ArXiv*, abs/2405.06640.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. 2023. Resurrecting recurrent neural networks for long sequences. *Preprint*, arXiv:2303.06349.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *Preprint*, arXiv:1606.06031.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. *Preprint*, arXiv:1912.01703.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Jiaju Lin, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Bolun Wang, Johan S. Wind, Stanislaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. 2023. Rwkv: Reinventing rnns for the transformer era. *Preprint*, arXiv:2305.13048.

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

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, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. 2024. Hgrn2: Gated linear rnns with state expansion. *Preprint*, arXiv:2404.07904.

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

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *Preprint*, arXiv:1907.10641.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

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. *Preprint*, arXiv:2307.08621.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander M. Rush, and Tri Dao. 2025. The mamba in the llama: Distilling and accelerating hybrid models. *Preprint*, arXiv:2408.15237.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

Yuxin Wu and Kaiming He. 2018. Group normalization. *Preprint*, arXiv:1803.08494.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. 2024. Gated linear attention transformers with hardware-efficient training. *Preprint*, arXiv:2312.06635.

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

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

Lin Yueyu, Li Zhiyuan, Peter Yue, and Liu Xiao. 2025. Arwkv: Pretrain is not what we need, an rnn-attention-based language model born from transformer. *Preprint*, arXiv:2501.15570.

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

Michael Zhang, Simran Arora, Rahul Chalamala, Alan Wu, Benjamin Spector, Aaryan Singhal, Krithik Ramesh, and Christopher Ré. 2024a. Lolcats: On low-rank linearizing of large language models. *Preprint*, arXiv:2410.10254.

Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Ré. 2024b. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. *Preprint*, arXiv:2402.04347.

Itamar Zimerman, Ameen Ali, and Lior Wolf. 2024. Explaining modern gated-linear rnns via a unified implicit attention formulation. *Preprint*, arXiv:2405.16504.

A Preliminary Experiments

To validate our approach before full-scale training, we conducted preliminary experiments comparing standard training (without parameter copying) against parameter-initialized training on a next-token prediction task. Our goal was to assess whether initializing student models with parameters from a pre-trained Transformer teacher could provide a more effective starting point.

Additionally, we explored the effect of Frobenius norm vs. MSE loss for Attention Alignment, finding both to yield similar performance.

Model	Initialization Method	Lamb.	WinoG.	Arc-E	Arc-C	PIQA	HellaS.	Avg.↑
SmolLM-360M		49.26	59.35	70.24	36.65	71.65	43.11	55.04
Preliminary Standard Training								
xLSTM		10.36	51.38	36.70	20.05	61.81	20.07	33.39
Llama→xLSTM		22.09	53.20	52.03	25.09	67.95	35.36	42.62
Frobenius vs. MSE								
Llama→$\text{xLSTM}_{\text{Frobenius}}$	+ *QKV* + *Matrix Mixing*	34.13	55.17	66.40	29.01	70.62	38.54	48.98
Llama→$\text{xLSTM}_{\text{MSE}}$	+ *QKV* + *Matrix Mixing*	33.76	55.41	65.43	29.35	70.24	38.55	48.79

Table 5: Preliminary experiments conducted on 1B tokens.

B Attention Matrix Approximation

Table 6 summarizes all models under evaluation and how each attention matrix equivalent is constructed. We furthermore include references to the original definition.

We define $\boldsymbol{CM}$ as the causal mask, where

$$\boldsymbol{CM}_{ij} = \begin{cases} 0, & \text{if } j \leq i \\ -\infty, & \text{if } j > i \end{cases} \tag{10}$$

C Model Parameter Counts

Table 7 lists the number of parameters for each model after replacing the attention layer with the corresponding linear attention backbone.

Model	#Params
Llama	361M
Llama→xLSTM	478M
Llama→GLA	478M
Llama→RetNet	477M
Llama→MetaLA	477M
Llama→DeltaNet	448M
Llama→VanillaLA	448M
Llama→Rebased	448M
Llama→Hedgehog	448M

Table 7: Model list with corresponding parameter count

D Experiment 1: Convergence Behaviour

Figure 3 provides an overview of loss trajectories across training stages for each model under all three stage configurations.

Architecture	Mixing Matrix $\boldsymbol{P}$	Decay / Mask Term	Reference
Linear Attention	$\boldsymbol{P} = (\phi(\boldsymbol{Q})\phi(\boldsymbol{K})^\top) \odot \boldsymbol{CM}$		
+ Vanilla	$\phi(x) = elu(x) + 1$	-	
+ Rebased	$\phi(x) = (\gamma \cdot norm(x) + \beta)^2$	-	
+ Hedgehog	$\phi(x) = \exp(Wx + b)$	-	
GLA	$\boldsymbol{P} = ((\boldsymbol{Q} \odot \boldsymbol{B})(\frac{\boldsymbol{K}}{\boldsymbol{B}})^\top) \odot \boldsymbol{CM}$	$\boldsymbol{B} = \prod_{j=i+1}^{t} \alpha_j^\top 1$	Yang et al. (2024), Section 4.1
mLSTM	$\boldsymbol{P} = \boldsymbol{QK}^\top \odot (\boldsymbol{F} \odot exp(\tilde{\boldsymbol{I}}))$	$\boldsymbol{F}_{i,j} = \begin{cases} 0, & \text{if } i < j \\ 1, & \text{if } i = j \\ \prod \sigma(\tilde{f}_k), & \text{if } i > j \end{cases}$	Beck et al. (2024), Appendix A.3
RetentionNet	$\boldsymbol{P} = \boldsymbol{QK}^\top \odot \boldsymbol{D}$	$\boldsymbol{D}_{i,j} = \begin{cases} 0, & \text{if } i < j \\ \gamma^{i-j} & \text{if } i \geq j \end{cases}$	Sun et al. (2023), Section 2.1 Eq. 5
DeltaNet	$\boldsymbol{P} = (\boldsymbol{QK}^\top \odot \boldsymbol{CM}) \odot \boldsymbol{T}$	$T = (\boldsymbol{I} + tril(diag(\beta)\boldsymbol{KK}^\top, -1))^{-1} \cdot diag(\beta)$	Yang et al. (2025), Section 3.2

Table 6: Overview of attention matrix approximations for different sequence mixer backbones.

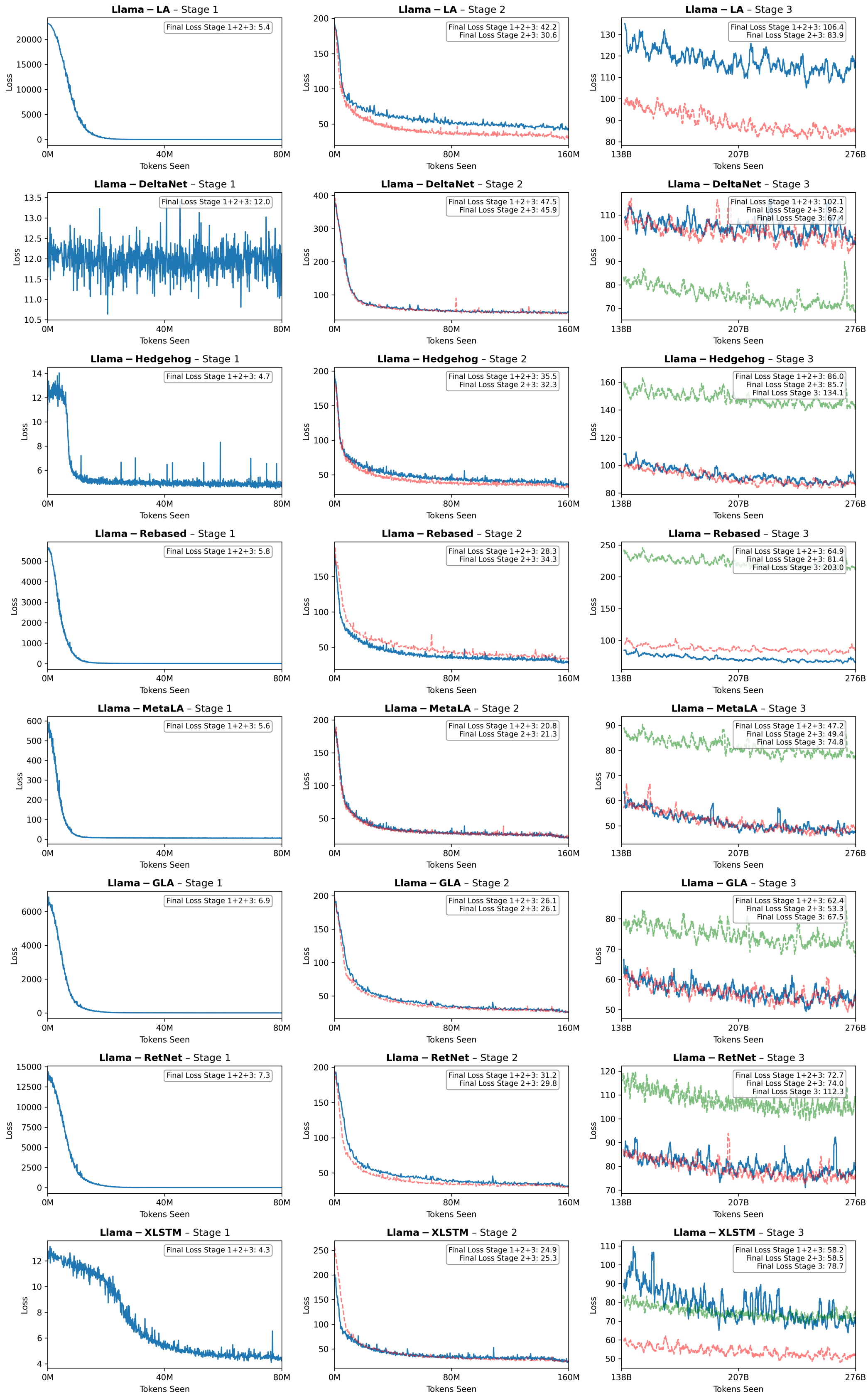

Figure 3: Loss plots for all runs conducted in Experiment 1. Green line plots indicate only Stage 3 training, while red and blue indicate Stage 2+3 and 1+2+3 Stage respectively.

E Experiment 3: Full Results for Explicit vs. Implicit Attention Approximation

For completeness, we include the full results of Experiment 3.

F Experiment 4: Full Results for the Longe Context experiments

For completeness, we include the full results of Experiment 4.

G Ablation: SmolLM-xLSTM Collection

As an outlook, we trained xLSTM student models, based on the SmolLM collection. We used the same training setup as described in Section 4. For the 1.7B model equivalent we also trained a version with a lower learning rate to adjust for size. Results are shown in Table 10.

H Ablation: Efficiency Comparison.

Figure 4 shows token generation speed and memory usage across models. Transformer models like Llama incur higher costs due to softmax attention and growing key-value caches. In contrast, linear attention and recurrent models (e.g., xLSTM, GLA) maintain constant or subquadratic memory and achieve faster, linear-time inference through efficient state updates.

MODEL	MAT. MIXING	LAMB. acc.	WINOG. acc.	ARC-E acc. norm.	ARC-C acc. norm.	PIQA acc_norm	HELLAS. acc. norm.	AVG.↑
Llama→xLSTM$_{mohawk}$	Explicit	35.71	56.43	60.40	32.51	70.95	50.37	51.06
Llama→xLSTM$_{mohawk}$	Implicit	36.05	55.09	59.85	33.28	70.95	49.87	50.84
Llama→GLA$_{mohawk}$	Explicit	35.05	53.67	60.94	32.42	70.35	50.17	50.43
Llama→GLA$_{mohawk}$	Implicit	35.06	54.62	61.07	33.36	70.51	50.19	50.80
Llama→RetNet$_{mohawk}$	Explicit	31.54	53.83	59.97	32.00	70.35	48.47	49.35
Llama→RetNet$_{mohawk}$	Implicit	32.27	54.62	59.60	32.42	70.67	48.26	49.64
Llama→MetaLA$_{mohawk}$	Explicit	36.39	54.22	61.07	32.68	71.22	50.21	50.95
Llama→MetaLA$_{mohawk}$	Implicit	35.54	54.14	62.08	32.94	71.00	50.31	51.00
Llama→DeltaNet$_{mohawk}$	Explicit	28.38	52.01	56.86	31.83	70.18	45.98	47.54
Llama→DeltaNet$_{mohawk}$	Implicit	26.83	50.36	57.20	30.80	69.80	45.84	46.80
Llama→LA$_{mohawk}$	Explicit	30.66	53.43	56.51	31.06	69.53	46.13	47.88
Llama→LA$_{mohawk}$	Implicit	30.94	53.75	55.68	31.48	70.02	46.33	48.03
Llama→Rebased	Explicit	34.41	52.80	57.83	32.42	69.75	48.60	49.30
Llama→Rebased	Implicit	33.14	53.49	57.37	31.06	70.51	48.13	48.95
Llama→Hedgehog$_{mohawk}$	Explicit	30.72	53.99	56.99	30.38	70.57	46.18	48.13
Llama→Hedgehog$_{mohawk}$	Implicit	30.44	52.17	56.69	32.17	70.62	46.02	48.01

Table 8: Comparison of explicit and implicit alignment of the token mixer backbone. When applying both approaches an additional 80M tokens is allocated from the 3B token budget.

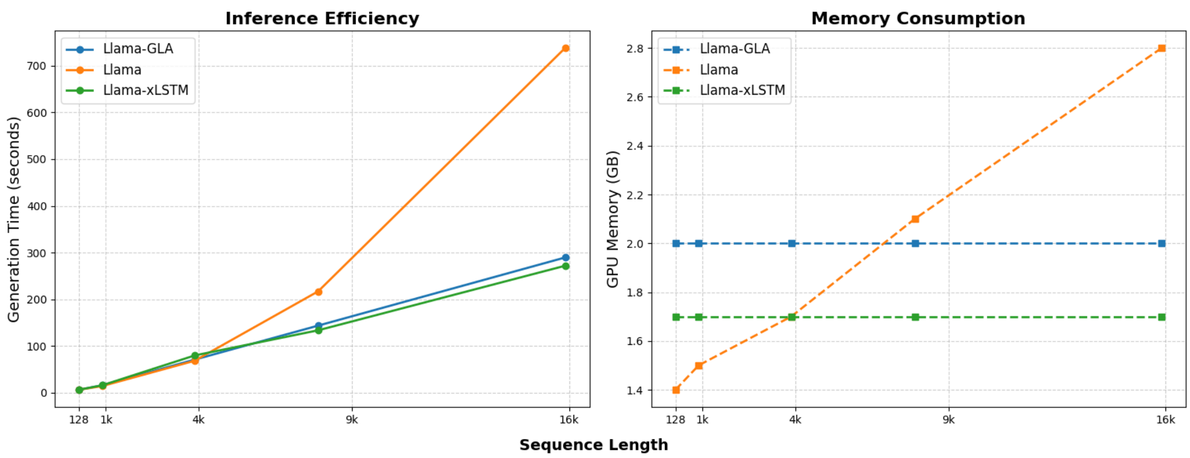

Figure 4: Inference efficiency and memory consumption of linear and softmax attention models, evaluated across single sequences of varying lengths.

Model	WikiMQA	MultiFieldQA	NarrativeQA	TREC	TriviaQA	Avg.
			512 Context			
SmolLM-360M	34.30	26.71	30.25	14.96	34.11	28.06
Llama→xLSTM	31.90	23.94	26.54	7.67	30.15	24.04
Llama→GLA	34.12	29.26	28.92	5.75	28.26	25.26
Llama→MetaLA	22.59	21.19	19.46	0.00	25.04	17.66
Llama→RetNet	31.17	26.35	26.53	8.25	27.06	23.87
Llama→DeltaNet	26.19	27.38	27.44	5.25	29.79	23.21
Llama→LA	21.30	19.82	19.72	0.00	23.07	16.78
Llama→Bebased	32.61	28.54	24.78	9.00	27.51	24.49
Llama→Hedgehog	31.66	25.45	26.12	2.75	29.67	23.13
			2K Context			
SmolLM-360M	35.63	27.17	30.06	16.08	33.66	28.52
Llama→xLSTM	32.87	26.88	27.04	5.75	28.10	24.13
Llama→GLA	30.39	29.29	26.79	5.67	31.03	24.63
Llama→MetaLA	22.54	22.06	19.10	0.00	24.59	17.66
Llama→RetNet	18.00	17.40	16.03	1.50	18.48	14.28
Llama→DeltaNet	24.98	24.41	20.49	0.50	24.17	18.91
Llama→LA	11.75	11.36	12.99	0.00	16.06	10.43
Llama→Rebased	21.67	20.75	17.96	0.00	20.18	16.11
Llama→Hedgehog	22.28	20.02	21.13	0.00	18.88	16.46
			4K Context			
SmolLM-360M	33.18	24.51	31.70	15.29	36.68	28.27
Llama→xLSTM	31.16	23.40	25.77	5.00	26.96	22.46
Llama→GLA	33.12	23.05	26.83	2.75	30.10	23.17
Llama→MetaLA	22.73	22.71	19.10	0.00	24.73	17.85
Llama→RetNet	18.07	11.21	16.66	1.25	19.12	13.26
Llama→DeltaNet	16.71	18.49	19.55	0.00	23.44	15.64
Llama→LA	13.97	14.92	17.21	0.00	13.60	11.94
Llama→Rebased	17.41	16.63	25.27	0.00	20.48	15.96
Llama→Hedgehog	21.78	16.43	19.40	0.00	18.57	15.24
			8K Context			
SmolLM-360M	17.84	15.44	17.29	0.17	19.06	14.16
Llama→xLSTM	33.71	27.66	24.86	4.25	27.61	23.62
Llama→GLA	30.63	27.55	28.06	3.50	28.87	23.72
Llama→MetaLA	24.26	22.72	19.10	0.00	25.18	18.25
Llama→RetNet	16.70	15.85	17.25	1.50	15.05	13.27
Llama→DeltaNet	17.21	21.43	18.57	0.00	18.87	15.22
Llama→LA	12.90	13.06	10.79	0.00	12.94	9.94
Llama→Rebased	11.98	15.61	24.65	0.50	20.66	14.68
Llama→Hedgehog	20.65	17.19	17.85	0.00	16.31	14.40
			16K Context			
SmolLM-360M	18.12	18.01	20.29	0.00	20.96	15.47
Llama→xLSTM	30.31	28.19	28.25	4.00	28.77	23.90
Llama→GLA	33.10	29.10	28.48	2.00	29.75	24.49
Llama→MetaLA	25.29	20.55	19.31	0.00	25.34	18.10
Llama→RetNet	17.16	15.89	19.90	0.00	18.19	14.23
Llama→DeltaNet	20.62	18.75	20.08	0.00	22.35	16.36
Llama→LA	13.44	11.28	11.26	0.00	14.02	10.00
Llama→Rebased	13.21	14.81	23.64	0.00	16.25	13.58
Llama→Hedgehog	16.00	17.23	13.62	0.00	16.15	12.60

Table 9: Full evaluation results for long-context evaluation on LongBench benchmark.

MODEL	LAMB. acc.	WINOG. acc.	ARC-E acc. norm.	ARC-C acc. norm.	PIQA acc_norm	HELLAS. acc. norm.	AVG.↑	RECOVERY
SmolLM-135M	32.93	52.88	55.85	29.18	68.23	42.68	46.96	-
SmolLM-360M	41.33	56.51	63.72	36.01	71.49	53.37	53.73	-
SmolLM-1.7B	48.38	60.93	73.48	46.42	76.06	65.74	61.83	-
Llama-xLSTM-180M	26.64	50.51	51.81	26.79	67.57	39.90	43.87	93.42%
Llama-xLSTM-400M	35.71	56.43	60.40	32.51	70.95	50.37	51.06	95.03%
Llama-xLSTM-1.8B	47.08	60.38	56.19	29.05	73.56	57.71	53.99	87.32%
Llama-xLSTM-1.8B$_{low-lr}$	39.99	57.46	66.71	38.57	74.43	60.41	56.26	90.99%

Table 10: Linearized xLSTM models based on the SmolLM collection. All models were trained with the same 3 Stage regime like in Experiment 1. For the SmolLM-1.7B equivalent, we also trained a version with a lower LR of $1e-4$ for Stage 3.