Effective Distillation to Hybrid xLSTM Architectures

Lukas Hauzenberger[1,2,*], Niklas Schmidinger[1,2,*], Thomas Schmied[1,2], Anamaria-Roberta Hartl[2], David Stap[1], Pieter-Jan Hoedt[2], Maximilian Beck[1,2], Sebastian Böck[1], Günter Klambauer[1,2], Sepp Hochreiter[1,2]

[1]NXAI [2]Johannes Kepler University Linz *Equal contribution.

arXiv:2603.15590v2 [cs.LG] 6 Jul 2026

Abstract

There have been numerous attempts to distill quadratic attention-based large language models (LLMs) into sub-quadratic linearized architectures. However, despite extensive research, such distilled models often fail to match the performance of their teacher LLMs on various downstream tasks. We set out the goal of *lossless distillation*, which we define in terms of tolerance-corrected *Win-and-Tie rates* between student and teacher on sets of tasks. To this end, we introduce an effective distillation pipeline for xLSTM-based students. We propose an additional merging stage, where individually linearized experts are combined into a single model. We show the effectiveness of this pipeline by distilling base and instruction-tuned models from the Llama, Qwen, and Olmo families. In many settings, our xLSTM-based students recover most of the teacher's performance, and even exceed it on some downstream tasks. Our contributions are an important step towards more energy-efficient and cost-effective replacements for transformer-based LLMs.

1 Introduction

Current LLMs require enormous computational resources because their attention mechanisms (Vaswani et al., 2017; Touvron et al., 2023; Team, 2023; OpenAI, 2025) scale quadratically with context length. As a result, these models are energy-intensive and costly to deploy. To address these limitations, many works distill (Schmidhuber, 1991; Hinton et al., 2015) LLMs into linearized, attention-free, or more generally sub-quadratic architectures (Wang et al., 2024; Bick et al., 2024; Wang et al., 2025; Zhang et al., 2025b; Lan et al., 2025; Goldstein et al., 2025). The efficient inference of sub-quadratic distilled LLMs makes them favorable drop-in replacements, if they match their teachers across diverse tasks.

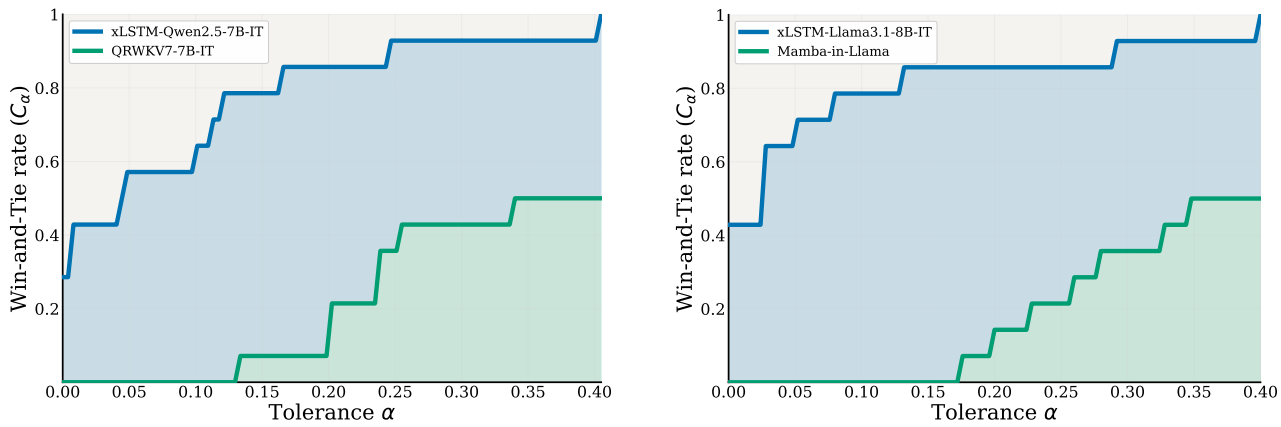


Figure 1: Win-and-Tie rate (C_α) curves of our distilled xLSTM-Qwen2.5-7B-IT**(left)** and xLSTM-Llama3.1-8B-IT**(right)** in comparison against the best sub-quadratic baseline across generation benchmarks spanning math, code, STEM, and chat domains. Higher is better.

Recent post-training linearization has coalesced around a handful of sub-quadratic sequence mixer designs to *substitute* full softmax attention layers and a small set of recurring distillation techniques. LoLCATs (Zhang et al., 2025b) and Liger (Lan et al., 2025) implement intra-layer hybrids that couple linear attention variants with sliding window attention (SWA), RADLADS (Goldstein et al., 2025) adapt RWKV-6 (Peng et al., 2024) and RWKV-7 (Peng et al., 2025) for the distillation setting, and Llamba (Bick et al., 2025) converts layers to Mamba-2 state-space mixers (Dao and Gu, 2024). For linearization, supervision typically involves hidden-state and logit alignment on small subsets of general web-text mixtures or instruction datasets. In contrast, token budgets for conventional LLM pre-training range from tens of billions to trillions of tokens, rendering linearization orders of magnitude more token-efficient than training from scratch. Therefore, linearization is an attractive fine-tuning regime for both exploring novel linear attention designs and lowering the deployment cost of Transformer-based models. However, existing linearization attempts have not yet achieved effective distillation. While linearized models often match the teacher on language understanding or knowledge benchmarks, they fall short on harder generative evaluations that probe the student's mathematical reasoning or code synthesis abilities (see Figures 3b and 4). These outcomes highlight limitations of existing distillation procedures, architectures, and evaluation protocols (see Appendix B.2 for an overview of prior work).

xLSTM as a powerful linear alternative for LLMs. Recently, modern recurrent architectures, such as xLSTM (Beck et al., 2024), Gated Delta Networks (Yang et al., 2024a), and Mamba (Gu and Dao, 2024), have emerged as competitive linear-complexity alternatives to Transformers in language (Beck et al., 2025b), computer vision (Alkin et al., 2025; Pöppel et al., 2025), biological modeling (Schmidinger et al., 2025), decision-making (Schmied et al., 2025), and time series (Auer et al., 2025). Concurrently, specialized kernels enable efficient chunkwise-parallel training for linear recurrent neural networks (RNNs) and xLSTM, substantially improving throughput on high-end accelerators (Beck et al., 2025a). Recent scaling-law analyses further indicate that xLSTM maintains competitive advantages as training and inference contexts grow, positioning it as a strong foundation for efficient long-context models (Beck et al., 2026). We hybridize xLSTM with sparse attention by combining an mLSTM with a synchronous SWA path and sink tokens using learned gates. Conceptually, this is related to recent attention hybrids that blend quadratic key-value (KV) memory with linear fast-weight memory (Irie et al., 2025).

Contributions. To rigorously assess whether linearized students can serve as drop-in replacements, we formalize a reliability criterion via the *Win-and-Tie rate* C_α, which measures how broadly the student recovers teacher-level performance across benchmarks. Using this criterion, we show that prior linearization approaches often preserve language understanding but fall short on harder, free-form generation tasks. To close this gap, we introduce a linearization pipeline that replaces quadratic softmax attention with an efficient mLSTM–SWA hybrid. In our linearization pipeline we introduce a merging stage, where domain-specialized students are distilled independently and consolidated afterwards. In this sense, we demonstrate that linearization can be made modular: linearized models can be consolidated through simple weight-space merging (Wortsman et al., 2022). The resulting merge of distilled xLSTM students closes the performance gap on free-form generation tasks and consistently dominates existing linearization methods across tolerance levels on C_α.

2 Background

Softmax attention and Transformers. The impressive capabilities of Transformer-based LLMs are largely attributed to the effectiveness of the underlying softmax-attention mechanism (Vaswani et al., 2017), which enables fine-grained modeling of long-range dependencies. At each time step t, an attention layer receives an input $\boldsymbol{x}_t \in \mathbb{R}^d$ and projects it to a query $\boldsymbol{q}_t$, key $\boldsymbol{k}_t$, and value $\boldsymbol{v}_t$ via learned linear maps $\boldsymbol{W}_q, \boldsymbol{W}_k \in \mathbb{R}^{d\times d_{qk}}$ and $\boldsymbol{W}_v \in \mathbb{R}^{d\times d_v}$:

$$\boldsymbol{q}_t = \boldsymbol{x}_t\boldsymbol{W}_q \qquad \boldsymbol{k}_t = \boldsymbol{x}_t\boldsymbol{W}_k \qquad \boldsymbol{v}_t = \boldsymbol{x}_t\boldsymbol{W}_v \tag{1}$$

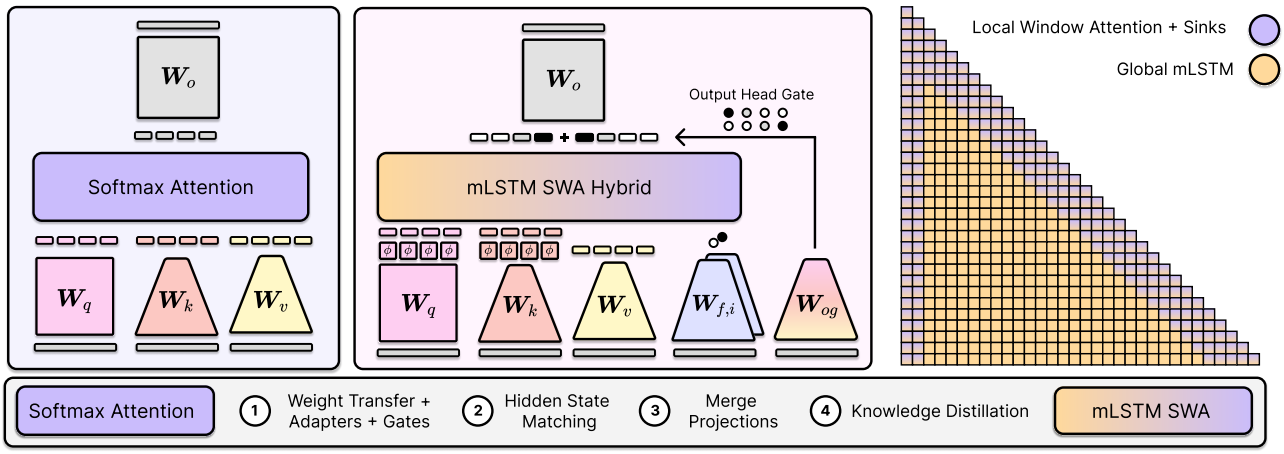


Figure 2: Illustration of our hybrid method consisting of mLSTM, sliding-window attention, and sink tokens. Our approach comprises 4 primary steps: **(1)** transfer the original teacher weights to the student and introduce adapters and gates, **(2)** hidden-state matching, **(3)** subsequent merging of query and key projections, and **(4)** knowledge distillation.

so that $\boldsymbol{q}_t, \boldsymbol{k}_t \in \mathbb{R}^{d_{qk}}$ and $\boldsymbol{v}_t \in \mathbb{R}^{d_v}$. To avoid recomputation, KV caches are maintained whose sizes grow with time, $\boldsymbol{K}_t \in \mathbb{R}^{t \times d_{qk}}$ and $\boldsymbol{V}_t \in \mathbb{R}^{t \times d_v}$, updated by concatenation (denoted as $[\,]$) along the time dimension:

$$\boldsymbol{K}_t = [\,\boldsymbol{K}_{t-1} \;\; \boldsymbol{k}_t\,] \qquad\qquad \boldsymbol{V}_t = [\,\boldsymbol{V}_{t-1} \;\; \boldsymbol{v}_t\,] \tag{2}$$

The output is then read from memory using *scaled softmax attention*:

$$\boldsymbol{h}_t = \text{softmax}\left(\frac{1}{\sqrt{d_{qk}}}\boldsymbol{q}_t \boldsymbol{K}_t^\top\right)\boldsymbol{V}_t \tag{3}$$

For a given query $\boldsymbol{q}_t$ at t, the dot product between the query and all stored keys $\boldsymbol{K}_t$ up to t is computed. Subsequently, softmax is applied over time steps, and the resulting per-position attention scores are used to compute a weighted average of the stored values $\boldsymbol{V}_t$. During training and context encoding, each query in a sequence of length T is compared with every key, incurring $O(T^2)$ time. During autoregressive inference, KV pairs are appended to the cache. At step t, the attention readout time and the cache size are $O(t)$. Although linear in t, the cache footprint scales with network depth and heads, and the cache must be read at every step, implying $O(t)$ memory-bandwidth cost. For long contexts, this becomes a dominant system bottleneck on modern accelerators, constraining batch size and throughput and increasing latency.

Sparse and Sliding-window attention. To mitigate the training and inference costs of full softmax attention, many LLMs adopt sparse attention patterns in which each head attends only to a subset of past positions (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020; Yuan et al., 2025). A widely used special case is sliding window attention (SWA), which restricts each query to attend to a fixed-length band of its immediate token history. SWA evicts keys and values outside the last W steps:

$$\boldsymbol{K}_t^W = [\,\boldsymbol{K}_{t-1}^{W-1} \;\; \boldsymbol{k}_t\,] \qquad\qquad \boldsymbol{V}_t^W = [\,\boldsymbol{V}_{t-1}^{W-1} \;\; \boldsymbol{v}_t\,] \tag{4}$$

where $\boldsymbol{K}_t^W = \boldsymbol{K}_{\max(1,t-W+1):t} \in \mathbb{R}^{\min(t,W) \times d_{qk}}$ and $\boldsymbol{V}_t^W = \boldsymbol{V}_{\max(1,t-W+1):t} \in \mathbb{R}^{\min(t,W) \times d_v}$. The maximum cache length of SWA therefore never exceeds W, while the core attention computation remains unchanged (see Equation 3). For sequences of length T, training and prefill of SWA can be implemented in linear $O(TW)$ time instead of $O(T^2)$ for full softmax attention. Consequently, during autoregressive decoding, both the computational and memory complexities of SWA are independent of the global sequence length. In Appendix C.1 we discuss the effective receptive field of SWA.

Linear attention replaces the exponential kernel of softmax attention $\kappa_{\exp}(\boldsymbol{q}, \boldsymbol{k}) = \exp(\boldsymbol{q}^\top \boldsymbol{k}/\sqrt{d_{qk}})$ with a finite-dimensional feature map $\phi : \mathbb{R}^{d_{qk}} \to \mathbb{R}^{d_{qk}}$ such that $\kappa_\phi(\boldsymbol{q}, \boldsymbol{k}) = \phi(\boldsymbol{q})^\top \phi(\boldsymbol{k})$ (Katharopoulos et al., 2020). This factorization enables two efficient implementations of causal attention: a chunkwise-parallel form for training and context encoding and a strictly recurrent form for stepwise decoding (see e.g. Yang et al., 2024b). Switching between these views enables prefill and training in linear time and constant-memory generation. In the recurrent view, we maintain a per-head KV state $\boldsymbol{S}_t \in \mathbb{R}^{d_{qk} \times d_v}$ that accumulates prefix statistics via rank-1 outer-product updates, together with an optional normalizer $\boldsymbol{z}_t \in \mathbb{R}^{d_{qk}}$:

$$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \phi(\boldsymbol{k}_t) \otimes \boldsymbol{v}_t \tag{5}$$

$$\boldsymbol{z}_t = \boldsymbol{z}_{t-1} + \phi(\boldsymbol{k}_t) \tag{6}$$

$\otimes$ denotes the outer product. Given a query $\boldsymbol{q}_t$, we perform a normalized read from the current state:

$$\boldsymbol{h}_t = \frac{\phi(\boldsymbol{q}_t)\boldsymbol{S}_t}{\phi(\boldsymbol{q}_t)\boldsymbol{z}_t} \tag{7}$$

Here, $\boldsymbol{q}_t, \boldsymbol{k}_t \in \mathbb{R}^{d_{qk}}$ and $\boldsymbol{v}_t \in \mathbb{R}^{d_v}$.

mLSTM. Inspired by the LSTM cell (Hochreiter and Schmidhuber, 1997), the mLSTM (Beck et al., 2024) augments the linear attention update with three data dependent gates that control distinct aspects of the update: $\boldsymbol{w}_i \in \mathbb{R}^{d \times 1}$, $\boldsymbol{w}_f \in \mathbb{R}^{d \times 1}$, $\boldsymbol{W}_{og} \in \mathbb{R}^{d \times d_v}$ where the input gate activations $i_t = \exp(\boldsymbol{x}_t \boldsymbol{w}_i)$ set the strength of the new KV write, the forget gate activations $f_t = \sigma(\boldsymbol{x}_t \boldsymbol{w}_f)$ decay the accumulated state, and the output gate activations $\boldsymbol{o}_t = \sigma(\boldsymbol{x}_t \boldsymbol{W}_{og})$ modulate the readout:

$$\boldsymbol{S}_t = f_t\, \boldsymbol{S}_{t-1} + i_t\, \phi(\boldsymbol{k}_t) \otimes \boldsymbol{v}_t \tag{8}$$

$$\boldsymbol{z}_t = f_t\, \boldsymbol{z}_{t-1} + i_t\, \phi(\boldsymbol{k}_t) \tag{9}$$

Numerical stabilization for the exponential input gate is omitted for simplicity. A query then performs a normalized read, and the output gate modulates the retrieved value:

$$\hat{\boldsymbol{h}}_t = \boldsymbol{o}_t \odot \frac{\phi(\boldsymbol{q}_t)\boldsymbol{S}_t}{\phi(\boldsymbol{q}_t)\boldsymbol{z}_t} \tag{10}$$

3 xLSTM distillation pipeline

In this work, we propose a distillation pipeline for creating efficient LLMs, substituting full softmax attention with a sub-quadratic attention proxy. The core of our method involves replacing the standard self-attention mechanism in a pre-trained LLMs with a hybrid attention block that combines SWA with mLSTM[1] (Beck et al., 2024) via data-dependent gating.

3.1 Architecture & student initialization

We use a pretrained causal Transformer-based LLM as the *teacher* model, similar to prior work (Zhang et al., 2025b). The *student* adopts the same high-level architecture design as the teacher, while replacing every multi-head attention block with a hybrid of SWA and mLSTM. This allows us to recycle the parameters of the original embedding and attention layers and the multi-layer perceptron (MLP) blocks. The fundamental motivation for our hybrid approach is to combine the strengths of two distinct and efficient sequence-mixing paradigms: the local context capturing ability of SWA and the linear complexity of mLSTM. Both components operate in parallel, and their outputs are dynamically fused using a learned, data-dependent gate.

[1]We use the $\text{xLSTM}_{[1:0]}$ configuration, which employs xLSTM blocks with mLSTM cells only.

mLSTM adaptations. Recent instantiations of gated linear operators replace the classical normalizer state with normalization layers such as LayerNorm (Sun et al., 2023; Yang et al., 2024b; Beck et al., 2025a). In the linearization setting, we observe that adding normalization immediately before the output projections degrades student-teacher alignment. Similar observations have been made in Bick et al. (2025). For this reason, we opt for the original normalizer design (cf. Equation 10) without normalization layers.

Instead of one output gate per channel, as in the original mLSTM, we use per-head scalar output gates to keep the parameter count closer to that of the teacher model. Furthermore, we found that using a concatenation of the head inputs over the feature dimension $\left[\, \boldsymbol{q}_t \;\; \boldsymbol{k}_t \;\; \boldsymbol{v}_t \,\right]$ instead of the input activations $\boldsymbol{x}_t$ at time t provides a better input signal for the output gate projections $\boldsymbol{W}_{og}$. Due to the strictly linear nature of the combination of $\boldsymbol{W}_q$, $\boldsymbol{W}_k$, $\boldsymbol{W}_v$ projections and $\boldsymbol{W}_{og}$, this can be merged to a single linear projection with input $\boldsymbol{x}_t$ at any stage. We augment the query and key inputs to the mLSTM with head-wise feature maps, applying softmax over the feature dimension as the activation function (Zhang et al., 2024).

Attention hybridization & data-dependent output mixing. We combine mLSTM and sparse attention into a single unified attention block, similar to Zhang et al. (2025b) and Dong et al. (2025), rather than alternating both operators at every layer. We opt for a sparse attention pattern using SWA over the most recent token history and four initial tokens per sequence to preserve attention sinks, similar to Xiao et al. (2024). The combination of SWA and sink tokens enables both efficient KV cache compression and a good initial approximation of full softmax attention. For a discussion on attention sinks, we refer to Appendix C.2. In Section 4.3, we demonstrate that all three components are critical for strong performance. Moreover, in Appendix B.1, we contextualize our architectural design relative to contemporary hybrid linear-attention architectures.

For a given input batch, we compute query, key, and value activations and apply rotary position embeddings (RoPEs) (Su et al., 2024). The output of the *local* SWA + sink branch is computed using a sparse attention kernel (Dong et al., 2024). For the *global* mLSTM branch, we transform queries and keys with our head-wise feature maps ϕ and pass them together with input and forget gate activations to the mLSTM cell. Finally, the output gate produces a sigmoid-bounded scalar per head that modulates the global mLSTM against the local SWA + sink outputs, similar to Yuan et al. (2025) and Irie et al. (2025):

$$\hat{\boldsymbol{h}}_t = o_t \,\mathrm{mLSTM}(\boldsymbol{q}_t) + (1 - o_t)\,\mathrm{SWA}(\boldsymbol{q}_t) = o_t \frac{\phi(\boldsymbol{q}_t)\boldsymbol{S}_t}{\phi(\boldsymbol{q}_t)\boldsymbol{z}_t} + (1 - o_t)\,\mathrm{sm}\left(\frac{\boldsymbol{q}_t \boldsymbol{K}_t^{W\top}}{\sqrt{d_{qk}}}\right)\boldsymbol{V}_t^W, \tag{11}$$

where sm is used as a short form for softmax. This simple yet effective combination of mLSTM and SWA yields a harmonic interplay between modeling short and long-term dependencies.

3.2 Linearization fine-tuning

Linearization stage I: layer-wise hidden-state alignment. Following prior linearization work (see Appendix B.2), we first align the per-layer representations of the student to the attention outputs of the teacher using a mean-squared error (MSE) objective. For each layer ℓ and time step t, let $\boldsymbol{h}_t^{(\ell)} = \mathrm{SoftmaxAttention}(\boldsymbol{q}_t^{(\ell)}, \boldsymbol{K}_t^{(\ell)}, \boldsymbol{V}_t^{(\ell)})$ denote the teacher's attention output and let $\hat{\boldsymbol{h}}_t^{(\ell)}$ denote the corresponding student hidden state as defined in Equation (11). The layer-wise objective is:

$$\min_{\boldsymbol{\theta}_\ell} \, \| \, \boldsymbol{h}_t^{(\ell)} - \hat{\boldsymbol{h}}_t^{(\ell)} \, \|_2^2, \tag{12}$$

where $\boldsymbol{\theta}_\ell$ denotes the newly introduced parameters, i.e., the parameters of the head-wise feature maps and gate projections. The embedding and MLP weights from the teacher are frozen in this stage. The full batch loss is then computed as the sum of Equation (12) over layers and time.

Linearization stage II: sparse knowledge distillation. Following the hidden-state alignment stage, we unfreeze all student parameters $\boldsymbol{\theta}$ and fine-tune end-to-end. This stage can be viewed as cross-architecture knowledge distillation from a Transformer teacher into a recurrent/hybrid student, conceptually related to prior work on transferring inductive biases between heterogeneous model families such as Transformers and LSTMs (Abnar et al., 2020). The objective for this stage interpolates between next-token prediction and matching the teacher distribution via the Kullback-Leibler divergence (KL):

$$\min_{\boldsymbol{\theta}}\Big\{-\sum_{t=1}^{T}\gamma\,\log p_{\boldsymbol{\theta}}\left(y_t \mid \mathbf{x}_{1:t}\right)+\beta\ \mathrm{KL}\left[p_T^{(k)}\left(\cdot \mid \mathbf{x}_{1:t}\right) \,\|\, p_{\boldsymbol{\theta}}^{(k)}\left(\cdot \mid \mathbf{x}_{1:t}\right)\right]\Big\}, \tag{13}$$

where $p_T(\cdot \mid \cdot)$ and $p_{\boldsymbol{\theta}}(\cdot \mid \cdot)$ denote the teacher and student distributions, respectively. The superscript (k) denotes the distribution over the top-k tokens, giving rise to a sparse KL. For our experiments, we set $k = 256$ (cf. Team, 2025a). The sparse KL in Equation (13) makes it possible to precompute and store teacher targets over the full distillation dataset. As a result, the teacher does not need to be accessed directly during stage II. This is especially advantageous for long-context distillation, where querying an online teacher can become prohibitively costly. Scaling this regime efficiently will be an important focus of future work.

Optional stage III: expert merging. Stages I–II can be applied either in a multi-task setting (one generalist student) or in a *decentralized* setting where K domain experts (e.g., math, code, STEM, etc.) are trained in parallel, all starting from the same initialized seed weights $\boldsymbol{\theta}^{(0)}$. This branch-train-merge workflow mirrors a broader trend in post-training pipelines that target specific capabilities and later consolidate them into a single deployable model (DeepSeek-AI, 2025; Cohere, 2025; Team, 2026). Concretely, after distilling linear experts $\{\boldsymbol{\theta}^{(i)}\}_{i=1}^{K}$, we form a single student via simple linear weight merging (Wortsman et al., 2022):

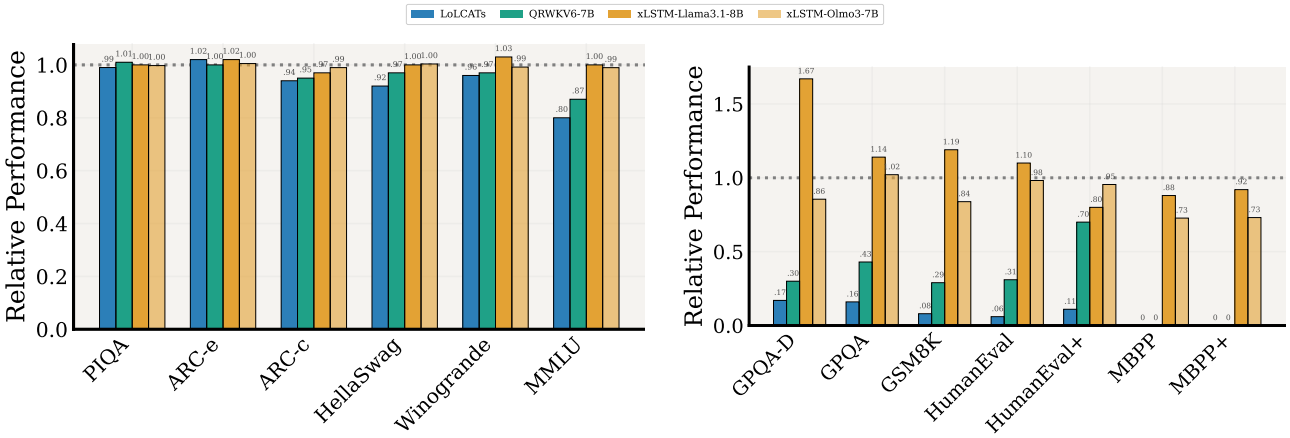
$$\boldsymbol{\theta}_{\text{merge}} = \sum_{i=1}^{K}\lambda_i\,\boldsymbol{\theta}^{(i)}, \qquad \lambda_i \geq 0,\ \sum_{i=1}^{K}\lambda_i = 1, \tag{14}$$

with uniform weights by default and optional validation-tuned λ_i when emphasizing particular capabilities. In our setting, this enables *capability patching*: researchers can independently improve a specific domain expert and update the final hybrid student by re-merging, without retraining the full model end-to-end. Moreover, the expert-centric setup is particularly well-suited for applying domain-specific fine-tuning or on-policy distillation to each expert before merging, i.e., learning from self-generated trajectories with teacher feedback, which we leave for future work (Agarwal et al., 2024). For a brief overview of decentralized post-training pipelines and model merging, see Section B.3.

4 Experiments

In this section, we apply our linearization protocol to both base models and instruction-tuned models from the Llama, Qwen, and Olmo families. We conduct downstream evaluations of the resulting hybrid models on established benchmarks across two important domains: **(1)** language understanding & knowledge tasks, and **(2)** language generation & reasoning tasks. Across benchmarks, we compare our distilled xLSTM students both against its teacher model and state-of-the-art linearization alternatives, including LoLCATs (Zhang et al., 2025b), RADLADS (Goldstein et al., 2025), and Mamba-in-Llama (Wang et al., 2024). We leverage `lm-eval` (Sutawika et al., 2025) for conducting our evaluations (see Appendix E.3 for details). For mathematical evaluations, we use the `Math-Verify` evaluation system.

Metrics for effective distillation: teacher-recovery rate and tolerance-corrected win-and-tie rate. Similar to Goldstein et al. (2025), we report the respective *teacher-recovery rate* as a primary per-benchmark metric, defined as the ratio between student and teacher performance. A recovery rate > 1 indicates that the student exceeds its teacher on the respective benchmark. We refer to Appendix Section E.3 for absolute scores. However, when comparing distilled models across a diverse suite of benchmarks, recovery rates alone do not quantify whether a student is a *reliable*



(a) Language Understanding

(b) Language Generation

Figure 3: **Downstream evaluations** for **(a)** language understanding and **(b)** language generation tasks. We report the recovery rate relative to teacher scores for our mLSTM-based student and established baselines with comparable parameter counts. The dotted line at 1.0 indicates parity with the Transformer teacher. Our model matches the teacher's performance across language understanding tasks, while exceeding the teacher on four of the considered generation tasks.

drop-in replacement. In particular, simple aggregates of recovery (e.g., mean/median recovery) can obscure substantial regressions on a subset of tasks, and ratio-based summaries can be uninformative when the teacher scores are small, yielding misleadingly large or noisy relative changes. We therefore complement recovery rates with a *tolerance-corrected win-and-tie* metric that summarizes *task-level win-rate* across benchmarks. Following our definition of (approximately) lossless distillation (Appendix Section A), we compute the win-and-tie rate C_α, i.e., the fraction of benchmarks on which the student matches or exceeds teacher performance within a tolerance α. This metric captures parity coverage across heterogeneous evaluations and distinguishes truly lossless distillation from partial recovery. For compact model comparison, we report α^*: the minimum tolerance α such that $C_\alpha \geq 0.5$. Lower α^* indicates a better student, and thus a better distillation process, since less tolerance is required to match the teacher on half of the benchmarks.

4.1 Base Model Evaluation: Validating the Hybrid Architecture

To assess the generality of our linearization pipeline for base models, we distill both Llama3.1-8B and Olmo3-7B. Olmo's fully open pre-training corpus provides a unique opportunity to evaluate whether matching the teacher on the original data distribution improves distillation compared to using alternative public datasets.

Experimental setup. For both models, we conduct stage I hidden-state alignment over 655M tokens with a sequence length of 4K using a standard linear-warmup to peak learning rate of 10^{-2} and cosine decay to 10^{-5}. For Llama we leverage the Dolmino dataset[2], and for Olmo we use the Dolmino 3 midtraining mix[3] released as part of Olmo2 OLMo (2025) and Olmo3 Olmo (2025) and maintain the originally proposed mixing weights.

For stage II, we further distill our aligned Llama checkpoints on an additional 5 billion tokens from the same data mixes and context size as in phase I. For Olmo, we extend the token budget to 20 billion tokens to align the budget with the protocol used for instruction-tuned models (cf. Section 4.2). For both models, we train using $\gamma = 0.9$ and $\beta = 0.1$ for cross-entropy (CE) and KL losses, respectively, and rewarm to a constant learning rate of 10^{-5}. Moreover, we provide additional experiment details, including a description of training settings and hyperparameters in Appendix D.

[2] https://huggingface.co/datasets/allenai/dolmino-mix-1124
[3] https://huggingface.co/datasets/allenai/dolma3_dolmino_mix-100B-1025

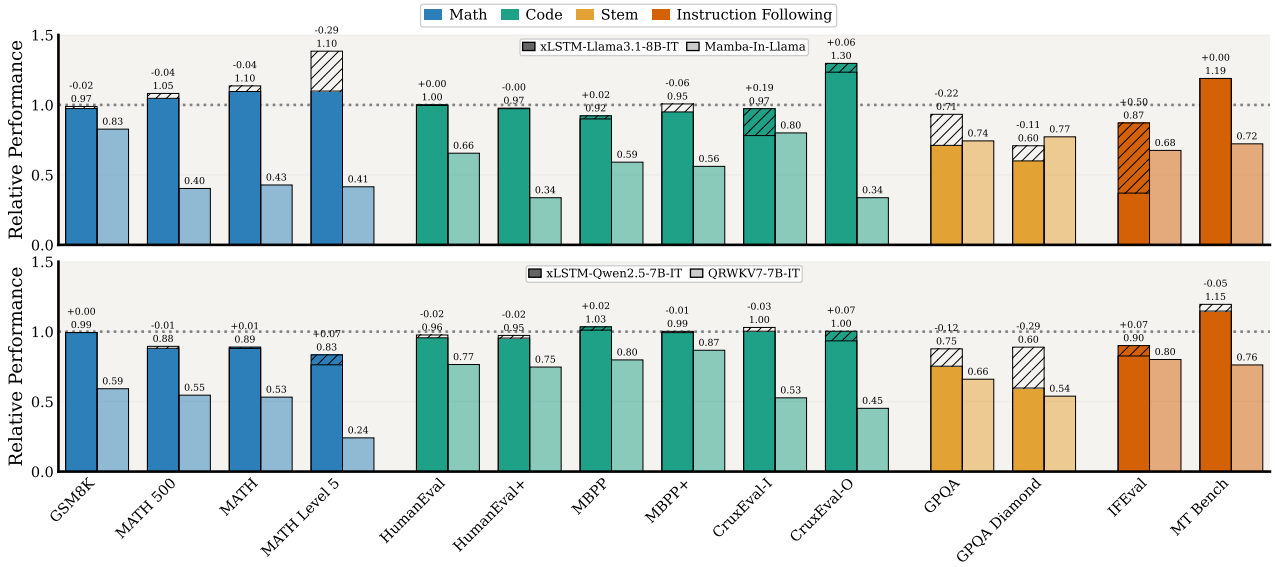


Figure 4: **Teacher-recovery rates for instruction-tuned xLSTM students and the effect of expert merging. Top:** xLSTM-Llama3.1-8B-IT distilled from Llama3.1-8B-IT vs. Mamba-in-Llama; **Bottom:** xLSTM-Qwen2.5-7B-IT distilled from Qwen2.5-7B-IT vs. QRWKV7-7B-IT. For each benchmark (x-axis; grouped by domain color), we report *relative performance* as the student/teacher score ratio (y-axis); the dotted line at 1.0 indicates parity with the Transformer teacher. For our method (left bar in each pair), the *merged* student is shown. For a given task, the striped area on top of the bar indicates gains (colored) or losses (empty) compared to our linearized *domain expert* before merging. For the baselines (right bar in each pair), the light bar shows the recovery rate.

Results. First, we evaluate our xLSTM-based students on six established multiple-choice (MC) and log-likelihood tasks, such as MMLU (Hendrycks et al., 2021), that test for general language understanding and knowledge. Among publicly available baselines, LoLCATs is distilled from the same Llama3.1-8B teacher, enabling a direct recovery-rate comparison, while QRWKV6-7B is distilled from a Qwen-family teacher (Goldstein et al., 2025). We observe that our distilled students achieve full (xLSTM-Llama3.1-8B) or near-full (xLSTM-Olmo3-7B) teacher parity, while LoLCATs and QRWKV6-7B exhibit a significant performance gap. We report the respective teacher-recovery rates in Figure 3a. Additionally, absolute scores are reported in Table 5. Next, we evaluate our distilled models on a broad battery of commonly used language generation and reasoning tasks that span important domains such as mathematics and coding (Cobbe et al., 2021; Austin et al., 2021). Unlike language understanding tasks, these benchmarks test the model's ability to produce consistent and relevant answers. In Figure 3b, we report the recovery rate of our xLSTM-based students and established baselines (see Table 6 for the raw scores). We discover that prior methods exhibit significant performance gaps compared to the teacher, with LoLCATs and QRWKV6-7B both yielding $\alpha^\star = 1.0$. In contrast, our hybrid models achieve strong relative scores across most tasks, achieving $\alpha^\star = 0.0$ for Llama3.1-8B and $\alpha^\star = 0.01$ for Olmo3-7B.

4.2 Instruction-Tuned Model Evaluation: Validating Decentralized Linearization

Experimental setup. Next, we apply our linearization pipeline to *post-trained* models, focusing on Llama3.1-8B-IT and Qwen2.5-7B-IT. To ensure coverage of the capabilities of interest, we use our decentralized linearization scheme: starting from the teacher weights, we train four linearized specialists targeting **math**, **STEM**, **code**, and **instruction-following/chat**. Each expert is trained for ∼5B tokens on a domain-specific mixture constructed from Nemotron Nano-2/3 and Olmo-3 data (NVIDIA, 2025b,c; Olmo, 2025; full mixtures and sampling weights in Table 2). As for base models, we use the Stage II objective in Eq. (13) with $\gamma = 0.9$ and $\beta = 0.1$, and train with a constant learning rate

of 7×10^{-7} for Qwen and 10^{-5} for Llama. After training, we consolidate the specialists into a single deployable student via linear weight merging (Eq. (14)). We choose merge weights via lightweight sweeps on small sets of downstream evaluations and simple heuristics such as downweighting experts that underperform across domains.

Results. Beyond language understanding benchmarks, we evaluate both the individual experts and the merged student on an expanded suite of generative benchmarks spanning our target domains (math, STEM reasoning, code, and instruction-following). Figure 4 reports teacher-recovery rates for each benchmark and highlights the effect of merging by comparing the merged checkpoint with its constituent specialists.

We compare against instruction-tuned linearization baselines with aligned teachers. For Llama3.1-8B-IT, we use Mamba-in-Llama, a Mamba2 attention hybrid that retains 50% of the original softmax attention layers. For Qwen2.5-7B-IT, we compare to QRWKV7-7B-IT (Goldstein et al., 2025). We find that our decentralized distillation pipeline yields strong linearized students that match their teachers on language-understanding benchmarks and code-generation evaluations, while recovering most of the teacher performance on mathematical reasoning tasks. We additionally assess instruction-following quality on MT-bench (Zheng et al., 2023) using LLM-as-a-judge, where GPT-5.1 grades generated responses. Across both the Llama and Qwen families, our students receive higher preference scores than their respective teachers (see Appendix E Figure 9 for detailed results). The largest remaining gap appears in STEM reasoning, where the merged student underperforms the dedicated STEM expert. Overall our merged students exhibit strong performance with xLSTM-Llama3.1-8B-IT and xLSTM-Qwen2.5-7B-IT achieving $\alpha^\star = 0.02$ and $\alpha^\star = 0.05$ respectively. We therefore conclude that expert training and subsequent model merging is a promising strategy to distill capabilities of interest in parallel and unify linearized models.

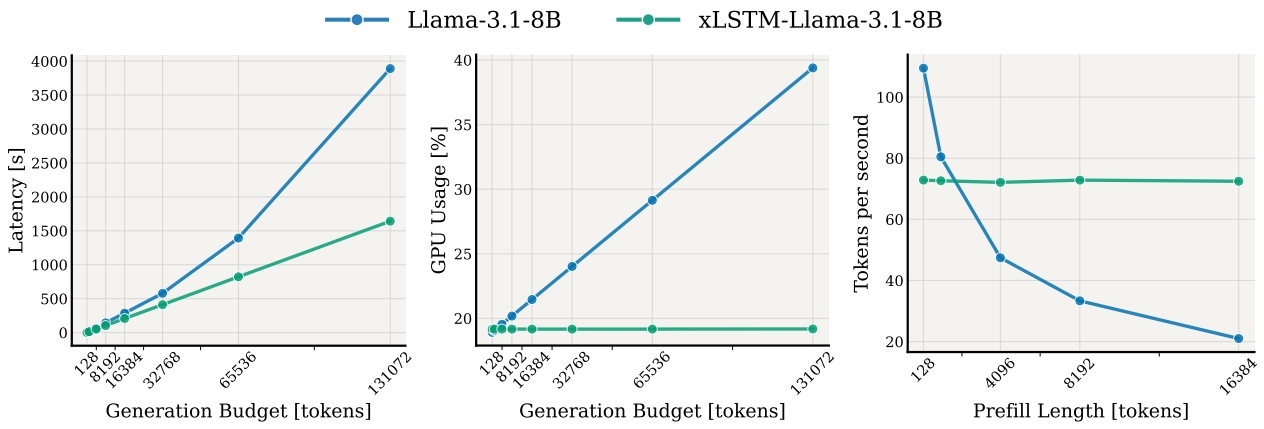
Model merging discussion Across both model families, we observe *positive transfer* when unifying independently trained experts into a single hybrid student. Most notably, merging substantially improves instruction-following for Llama, where we recover a large fraction of the gap on IFEval compared to the instruction expert alone. At the same time, merging is not uniformly beneficial. Both students exhibit the most pronounced degradations on STEM-oriented evaluations (e.g., GPQA and GPQA-Diamond), indicating interference between domain updates. In contrast, math and code capabilities are largely robust to merging, and for Qwen2.5-7B-IT we also observe comparatively minor changes in instruction-following. Overall, these results validate that simple weight-space merging can be effective even for *fully linearized* architectures, opening a practical path toward consolidating independently developed efficient students.

4.3 Ablations

Setup. For the following ablations we distill linear student models from Llama3.1-8B with a token budget of 2.5 billion tokens.

Effect of mLSTM, SWA & Sinks. Our hybrid approach utilizes a combination of mLSTM, SWA with a fixed size of 512 tokens and 4 sink tokens. In Figure 15, we empirically validate the individual contributions of these components. In addition, we compare against a pure linear attention baseline to understand the contribution of mLSTM. Pure mLSTM exhibits a considerably lower loss than linear attention, which highlights the effectiveness of its gating mechanism. The combination of mLSTM and SWA results in a striking improvement, pointing towards a harmonic relationship between modeling short and long-term dependencies via SWA and mLSTM, respectively. We report additional analyses on the importance of sink tokens and SWA in Appendix F.1. Moreover, we analyze the mixture weights produced by the data-dependent output gate and find that both contribute considerably to the final outputs (see Appendix E.2).

Effect of distillation objective. The distillation objective is another critical component of every distillation recipe. Therefore, we conduct an ablation in which we vary the mixture weights γ and β for the CE and KL losses used in stage II, respectively. We observe that $\gamma = 0.9$ and $\beta = 0.1$ result



(a) Latency, $B = 1$ (b) GPU RAM %, $B = 1$ (c) Throughput, $B = 8$

Figure 5: Inference comparison for the **generation stage** between the Transformer-based teacher and our xLSTM-based student. In **(a)**, we show generation latency at different generation budgets ($B = 1$). In **(b)**, we report the memory consumption in % of GPU memory during the generation ($B = 1$). In **(c)**, we show the generation throughput when generating 100 tokens with varying prefill lengths and $B = 8$.

in a low cross-entropy loss, while preventing the student from drifting too far away from its teacher (see Appendix F.2). In contrast, training purely via KL resulted in worse performance compared to a mixed objective due to overconstraining the linear student to the teacher. This trend persists even when distilling on the teacher's original training data, indicating that the gains are not merely an artifact of *up-training* the student on higher-quality data.

PEFT vs. FFT. Prior linearization recipes use parameter-efficient fine-tuning (PEFT) via low-rank adaptation (LoRA, Hu et al., 2022) to recover performance lost during conversion (Zhang et al., 2025b; Nguyen et al., 2025). While LoRA is cheaper, it is also less expressive than full fine-tuning (FFT). We find that FFT results in considerably improved performance and therefore adopt it by default in our distillation recipe (see Appendix F.3).

4.4 Inference Comparison

A key motivation for distilling Transformer teachers into recurrent xLSTM students is improved serving efficiency. Following prior work (De et al., 2024; Beck et al., 2025b), we report inference results separately for *prefill* (prompt encoding) and *generation* (autoregressive decoding) (Pope et al., 2023). Our inference tests are characterized by three core hyperparameters, the batch size B, the context length C, and the generation budget G. Both teacher and student are implemented in `transformers` (Wolf et al., 2020) and optimized with `torch.compile`, FlashAttention (Dao, 2024), and fused mLSTM kernels (Beck et al., 2025a); the student uses a static cache that stores mLSTM states plus sliding-window KV (details in Appendix E.4).

Prefill. Figure 10 reports throughput and time to first token (TTFT) on a single H100 80GB for the largest feasible batch-size/context-length pairs. Our hybrid student is consistently faster, with $\sim 2\times$ higher throughput at B=1, C=65K and an overall $\sim 2\times$ reduction in TTFT.

Generation. Figure 5 compares decoding latency, memory, and throughput. Without prefill (to isolate decoding complexity), the student roughly halves latency and GPU memory at G=131K, while maintaining constant memory over time. With prefill and B=8, the student achieves up to $\sim 4\times$ higher generation throughput as context length grows; the teacher runs out of memory (OOM) at larger batches (Appendix E.4).

Overall, we demonstrate strong efficiency benefits of our xLSTM-based student at inference time, in terms of latency, throughput, and memory consumption. We note that other linearized methods, such as LoLCATs (Zhang et al., 2025b), exhibit similar inference advantages (but with a larger student-teacher gap on downstream tasks) and therefore omit an additional comparison.

Serving systems perspective. Beyond the experimental measurements reported here, production deployment of hybrid models requires integrating linear-complexity sequence-mixing layers into serving stacks such as vLLM or SGLang, which are built around paged KV-memory abstractions such as PagedAttention (Kwon et al., 2023). As hybrid and otherwise heterogeneous architectures become more common, including recent releases such as Qwen3.5 (Qwen Team, 2026), Nemotron 3 (NVIDIA, 2025c), and Olmo Hybrid (Merrill et al., 2026), the system problem of efficiently scheduling, caching, and allocating memory across mixed layer types is becoming increasingly important. Recent work such as Jenga makes this challenge explicit for heterogeneous LLM serving and points to the need for serving runtimes that can handle non-uniform memory and access patterns efficiently (Zhang et al., 2025a). We view this as essential systems work and as an important direction for research.

5 Conclusion

We presented a linearization pipeline that replaces quadratic softmax attention with an efficient mLSTM–SWA hybrid. To assess whether distilled students are reliable drop-in replacements, we formalized lossless distillation through the *Win-and-Tie rate* C_α: the fraction of benchmarks on which the student matches or exceeds teacher performance (optionally within tolerance α), and its critical tolerance α^*, the minimum α such that $C_\alpha \geq 0.5$, i.e., the tolerance needed to match the teacher on at least half of benchmarks. Across base and instruction-tuned teachers from the Llama, Qwen, and Olmo families, our xLSTM students attain substantially higher C_α and Pareto-dominate prior linearization methods across tolerances, reflecting more effective distillation across benchmarks rather than gains on a small subset. As a result, our hybrid student models are prime candidates for a drop-in replacement of full-attention transformers when inference efficiency matters. We further showed that *distilling domain experts* and consolidating them via simple weight-space merging improves C_α, indicating that weight merging remains effective after full linearization and enabling modular capability development and targeted updates.

Limitations and future work. The remaining deficits are most visible on synthetic long-context evaluations (e.g., Needle-in-a-Haystack) and on select reasoning benchmarks, where interference between independently trained experts can reduce recovery after merging. A key next step is to further probe which expert domains are most beneficial to distill in isolation and how to consolidate them more reliably. For long-context behavior, we plan to explore stronger attention hybrids and memory designs. Finally, we aim to scale this recipe to larger teachers, including Sparse-Mixture-of-Experts models, and to study on-policy distillation or RL-based expert refinement prior to merging.

Acknowledgments

This work was supported by European Union's Horizon Europe research and innovation programme under grant agreement number 101214398 (ELLIOT) and the Austrian Science Fund (FWF) 10.55776/COE12. The ELLIS Unit Linz, the LIT AI Lab, the Institute for Machine Learning, are supported by the Federal State Upper Austria. We acknowledge EuroHPC Joint Undertaking for awarding us access to Leonardo at CINECA, Italy, Deucalion at MACC, Portugal, and Discoverer at SofiaTech, Bulgaria.

References

Samira Abnar, Mostafa Dehghani, and Willem Zuidema. Transferring inductive biases through knowledge distillation. *arXiv preprint arXiv:2006.00555*, 2020.

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 12. OpenReview, 2024.

Benedikt Alkin, Maximilian Beck, Korbinian Pöppel, Sepp Hochreiter, and Johannes Brandstetter. Vision-LSTM: xLSTM as Generic Vision Backbone. In *International Conference on Learning Representations*, volume 13, Singapore, Singapore, 2025. OpenReview.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Re. Simple linear attention language models balance the recall-throughput tradeoff. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 1763–1840, Vienna, Austria, 2024. PMLR. ISSN: 2640-3498.

Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. TiRex: Zero-Shot Forecasting Across Long and Short Horizons with Enhanced In-Context Learning. In *Advances in Neural Information Processing Systems*, volume 38, San Diego, CA, USA, 2025. Curran Associates, Inc.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program Synthesis with Large Language Models, 2021. arXiv:2108.07732 [cs].

Federico Barbero, Alvaro Arroyo, Xiangming Gu, Christos Perivolaropoulos, Petar Veličković, Razvan Pascanu, and Michael M. Bronstein. Why do LLMs attend to the first token? In *Conference on Language Modeling*, volume 2, Montréal, QC, Canada, 2025. OpenReview.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended Long Short-Term Memory. In *Advances in Neural Information Processing Systems*, volume 37, pages 107547–107603, Vancouver, BC, Canada, 2024. Curran Associates, Inc. doi: 10.52202/079017-3417.

Maximilian Beck, Korbinian Pöppel, Phillip Lippe, and Sepp Hochreiter. Tiled Flash Linear Attention: More Efficient Linear RNN and xLSTM Kernels. In *Advances in Neural Information Processing Systems*, volume 38, San Diego, CA, USA, 2025a. Curran Associates, Inc.

Maximilian Beck, Korbinian Pöppel, Phillip Lippe, Richard Kurle, Patrick M. Blies, Günter Klambauer, Sebastian Böck, and Sepp Hochreiter. xLSTM 7B: A Recurrent LLM for Fast and Efficient Inference. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pages 3335–3357, Vancouver, BC, Canada, 2025b. PMLR.

Maximilian Beck, Kajetan Schweighofer, Sebastian Böck, Sebastian Lehner, and Sepp Hochreiter. xLSTM scaling laws: Competitive performance with linear time-complexity. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=bpbU549sSg`.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer, 2020. arXiv:2004.05150 [cs].

Aviv Bick, Kevin Y. Li, Eric P. Xing, J. Z. Kolter, and Albert Gu. Transformers to SSMs: Distilling Quadratic Knowledge to Subquadratic Models. In *Advances in Neural Information Processing Systems*, volume 37, pages 31788–31812, Vancouver, BC, Canada, 2024. Curran Associates, Inc. doi: 10.52202/079017-0999.

Aviv Bick, Tobias Katsch, Nimit Sharad Sohoni, Arjun D. Desai, and Albert Gu. Llamba: Scaling Distilled Recurrent Models for Efficient Language Processing. In *ICLR Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models*, volume 1, Singapore, Singapore, 2025. OpenReview.

Ricardo Buitrago and Albert Gu. Understanding and Improving Length Generalization in Recurrent Models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pages 5856–5874, Vancouver, BC, Canada, 2025. PMLR. ISSN: 2640-3498.

Loïc Cabannes, Maximilian Beck, Gergely Szilvasy, Matthijs Douze, Maria Lomeli, Jade Copet, Pierre-Emmanuel Mazaré, Gabriel Synnaeve, and Hervé Jégou. Short window attention enables long-term memorization, 2025. arXiv:2509.24552 [cs].

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating Long Sequences with Sparse Transformers, 2019. arXiv:1904.10509 [cs].

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems, 2021. arXiv:2110.14168 [cs].

Team Cohere. Command A: An Enterprise-Ready Large Language Model, 2025. arXiv:2504.00698 [cs].

Tri Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *International Conference on Learning Representations*, volume 12, Vienna, Austria, 2024. OpenReview.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 10041–10071, Vienna, Austria, 2024. PMLR. 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. Griffin: Mixing Gated Linear Recurrences with Local Attention for Efficient Language Models, 2024. arXiv: 2402.19427 [cs.LG].

DeepSeek-AI. Deepseek-v3.2: Pushing the frontier of open large language models, 2025. arXiv:2512.02556 [cs].

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex Attention: A Programming Model for Generating Optimized Attention Kernels, 2024. arXiv:2412.05496 [cs].

Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan Celine Lin, Jan Kautz, and Pavlo Molchanov. Hymba: A Hybrid-head Architecture for Small Language Models. In *International Conference on Learning Representations*, volume 13, Singapore, Singapore, 2025. OpenReview.

Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. Zamba: A Compact 7B SSM Hybrid Model, 2024. arXiv:2405.16712 [cs].

Daniel Goldstein, Eric Alcaide, Janna Lu, and Eugene Cheah. RADLADS: Rapid Attention Distillation to Linear Attention Decoders at Scale, 2025. arXiv:2505.03005 [cs].

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. In *Conference on Language Modeling*, volume 1, Philadelphia, PA, USA, 2024. OpenReview.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, Wanjia Zhao, John Yang, Shreyas Pimpalgaonkar, Kartik Sharma, Charlie Cheng-Jie Ji, Yichuan Deng, Sarah Pratt, Vivek Ramanujan, Jon Saad-Falcon, Jeffrey Li, Achal Dave, Alon Albalak, Kushal Arora, Blake Wulfe, Chinmay Hegde, Greg Durrett, Sewoong Oh, Mohit Bansal, Saadia Gabriel, Aditya Grover, Kai-Wei Chang, Vaishaal Shankar, Aaron Gokaslan, Mike A. Merrill, Tatsunori Hashimoto, Yejin Choi, Jenia Jitsev, Reinhard Heckel, Maheswaran Sathiamoorthy, Alexandros G. Dimakis, and Ludwig Schmidt. OpenThoughts: Data Recipes for Reasoning Models, 2025. arXiv:2506.04178 [cs].

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *International Conference on*

Learning Representations, volume 9, virtual, 2021. OpenReview.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network, 2015. arXiv:1503.02531 [stat].

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780, 1997. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco.1997.9.8.1735.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the Real Context Size of Your Long-Context Language Models? In *Conference on Language Modeling*, volume 1, Philadelphia, PA, USA, 2024. OpenReview.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, volume 10, virtual, 2022. OpenReview.

Kazuki Irie, Morris Yau, and Samuel J. Gershman. Blending Complementary Memory Systems in Hybrid Quadratic-Linear Transformers. In *Advances in Neural Information Processing Systems*, volume 38, San Diego, CA, USA, 2025. Curran Associates, Inc.

Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A. Smith. Finetuning Pretrained Transformers into RNNs. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10630–10643, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.830.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 5156–5165, virtual, 2020. PMLR. arXiv: 2006.16236.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626, 2023.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing Frontiers in Open Language Model Post-Training, 2024. arXiv:2411.15124 [cs].

Disen Lan, Weigao Sun, Jiaxi Hu, Jusen Du, and Yu Cheng. Liger: Linearizing Large Language Models to Gated Recurrent Structures. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pages 32452–32466, Vancouver, BC, Canada, 2025. PMLR.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang W. Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. DataComp-LM: In search of the next generation of training sets for language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 14200–14282, Vancouver, BC, Canada, 2024. Curran Associates, Inc. doi: 10.52202/079017-0455.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A. Smith, and Luke

Zettlemoyer. Branch-train-merge: Embarrassingly parallel training of expert language models, 2022. arXiv:2208.03306 [cs].

Yixing Li, Ruobing Xie, Zhen Yang, Xingwu Sun, Shuaipeng Li, Weidong Han, Zhanhui Kang, Yu Cheng, Chengzhong Xu, Di Wang, and Jie Jiang. TransMamba: A Sequence-Level Hybrid Transformer-Mamba Language Model, 2026. arXiv:2503.24067 [cs].

Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the Effective Receptive Field in Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 29, pages 4898–4906, Barcelona, Spain, 2016. Curran Associates, Inc.

Jean Mercat, Igor Vasiljevic, Sedrick Scott Keh, Kushal Arora, Achal Dave, Adrien Gaidon, and Thomas Kollar. Linearizing Large Language Models. In *Conference on Language Modeling*, volume 1, Philadelphia, PA, USA, 2024. OpenReview.

William Merrill, Yanhong Li, Tyler Romero, Anej Svete, Caia Costello, Pradeep Dasigi, Dirk Groeneveld, David Heineman, Bailey Kuehl, Nathan Lambert, Chuan Li, Kyle Lo, Saumya Malik, DJ Matusz, Benjamin Minixhofer, Jacob Morrison, Luca Soldaini, Finbarr Timbers, Pete Walsh, Noah A. Smith, Hannaneh Hajishirzi, and Ashish Sabharwal. Olmo Hybrid: From theory to practice. `https://allenai.org/papers/olmo-hybrid`, March 2026. Allen Institute for AI.

Evan Miller. Attention Is Off By One, 2023. URL `https://www.evanmiller.org/attention-is-off-by-one.html`.

MiniMax. MiniMax-01: Scaling Foundation Models with Lightning Attention, 2025. arXiv:2501.08313 [cs].

Chien Van Nguyen, Ruiyi Zhang, Hanieh Deilamsalehy, Puneet Mathur, Viet Dac Lai, Haoliang Wang, Jayakumar Subramanian, Ryan A. Rossi, Trung Bui, Nikos Vlassis, Franck Dernoncourt, and Thien Huu Nguyen. Lizard: An Efficient Linearization Framework for Large Language Models, 2025. arXiv:2507.09025 [cs].

NVIDIA. Nemotron-H: A Family of Accurate and Efficient Hybrid Mamba-Transformer Models, 2025a. arXiv:2504.03624 [cs].

NVIDIA. NVIDIA Nemotron Nano 2: An Accurate and Efficient Hybrid Mamba-Transformer Reasoning Model, 2025b. arXiv:2508.14444 [cs].

NVIDIA. Nemotron 3 Nano: Open, Efficient Mixture-of-Experts Hybrid Mamba-Transformer Model for Agentic Reasoning, 2025c. arXiv:2512.20848 [cs].

Team OLMo. 2 OLMo 2 Furious, 2025. arXiv:2501.00656 [cs].

Team Olmo. Olmo 3, 2025. arXiv:2512.13961 [cs].

OpenAI. gpt-oss-120b & gpt-oss-20b Model Card, 2025. arXiv:2508.10925 [cs].

Guilherme Penedo, Hynek Kydlíček, Loubna B. Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale. In *Advances in Neural Information Processing Systems*, volume 37, pages 30811–30849, San Diego, CA, USA, 2024. Curran Associates, Inc. doi: 10.52202/079017-0970.

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, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the Transformer Era. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.936.

Bo Peng, Daniel Goldstein, Quentin Gregory Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Teddy Ferdinan, Kranthi Kiran Gv, Haowen Hou, Satyapriya Krishna, Ronald McClelland Jr, Niklas Muennighoff, Fares Obeid, Atsushi Saito, Guangyu Song, Haoqin Tu, Ruichong

Zhang, Bingchen Zhao, Qihang Zhao, Jian Zhu, and Rui-Jie Zhu. Eagle and Finch: RWKV with Matrix-Valued States and Dynamic Recurrence. In *Conference on Language Modeling*, volume 1, Philadelphia, PA, USA, 2024. OpenReview.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, Nathan Wilce, Johan S. Wind, Tianyi Wu, Daniel Wuttke, and Christian Zhou-Zheng. RWKV-7 "Goose" with Expressive Dynamic State Evolution, 2025. arXiv:2503.14456 [cs].

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Re. Hyena Hierarchy: Towards Larger Convolutional Language Models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 28043–28078, Honolulu, HI, USA, 2023. PMLR. ISSN: 2640-3498.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently Scaling Transformer Inference. In *Proceedings of Machine Learning and Systems*, volume 5, pages 606–624, Miami, FL, USA, 2023. MLSys.

Korbinian Pöppel, Richard Freinschlag, Thomas Schmied, Wei Lin, and Sepp Hochreiter. pLSTM: parallelizable Linear Source Transition Mark networks. In *Advances in Neural Information Processing Systems*, volume 38, San Diego, CA, USA, 2025. Curran Associates, Inc.

Zhen Qin, Weigao Sun, Dong Li, Xuyang Shen, Weixuan Sun, and Yiran Zhong. Various Lengths, Constant Speed: Efficient Language Modeling with Lightning Attention. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 41517–41535, Vienna, Austria, 2024. PMLR. ISSN: 2640-3498.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. ISSN 1533-7928.

Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple Hybrid State Space Models for Efficient Unlimited Context Language Modeling. In *International Conference on Learning Representations*, volume 13, Singapore, Singapore, 2025. OpenReview.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear Transformers Are Secretly Fast Weight Programmers. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 9355–9366, virtual, 2021. PMLR. ISSN: 2640-3498.

Jürgen Schmidhuber. *Neural sequence chunkers*. Inst. für Informatik, 1991.

Niklas Schmidinger, Lisa Schneckenreiter, Philipp Seidl, Johannes Schimunek, Pieter-Jan Hoedt, Johannes Brandstetter, Andreas Mayr, Sohvi Luukkonen, Sepp Hochreiter, and Günter Klambauer. Bio-xLSTM: Generative modeling, representation and in-context learning of biological and chemical sequences. In *International Conference on Learning Representations*, volume 13, Singapore, Singapore, 2025.

Thomas Schmied, Thomas Adler, Vihang Prakash Patil, Maximilian Beck, Korbinian Pöppel, Johannes Brandstetter, Günter Klambauer, Razvan Pascanu, and Sepp Hochreiter. A Large Recurrent Action Model: xLSTM enables Fast Inference for Robotics Tasks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pages 53343–53387, Vancouver, BC, Canada, 2025. PMLR.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's Razor: Why Online Reinforcement Learning Forgets Less, 2025. arXiv:2509.04259 [cs].

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive Network: A Successor to Transformer for Large Language Models, 2023. arXiv:2307.08621.

Lintang Sutawika, Hailey Schoelkopf, Leo Gao, Baber Abbasi, Stella Biderman, Jonathan Tow, ben fattori, Charles Lovering, farzanehnakhaee70, Jason Phang, Anish Thite, Fazz, Aflah, Niklas Muennighoff, Thomas Wang, sdtblck, nopperl, gakada, tttyuntian, researcher2, Julen Etxaniz, Chris, Hanwool Albert Lee, Leonid Sinev, Zdeněk Kasner, Khalid, KonradSzafer, Jeffrey Hsu, Anjor Kanekar, and Pawan Sasanka Ammanamanchi. EleutherAI/lm-evaluation-harness: v0.4.8, 2025.

Gemma Team. Gemma 3 Technical Report, 2025a. arXiv:2503.19786 [cs].

Jamba Team. Jamba: Hybrid Transformer-Mamba Language Models. In *International Conference on Learning Representations*, volume 13, Singapore, Singapore, 2025b. OpenReview.

Llama Team. The Llama 3 Herd of Models, 2024. arXiv:2407.21783 [cs].

Qwen Team. Qwen Technical Report, 2023. arXiv:2309.16609 [cs].

Qwen Team. Qwen3-Next: Towards Ultimate Training & Inference Efficiency, 2025c. URL `https://qwen.ai/blog?id=4074cca80393150c248e508aa62983f9cb7d27cd`.

Xiaomi Team. MiMo-V2-Flash Technical Report, 2026. arXiv:2601.02780 [cs].

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. LLaMA: Open and Efficient Foundation Language Models, 2023. arXiv:2302.13971 [cs].

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, Long Beach, CA, USA, 2017. Curran Associates, Inc. arXiv: 1706.03762.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander M. Rush, and Tri Dao. The Mamba in the Llama: Distilling and Accelerating Hybrid Models. In *Advances in Neural Information Processing Systems*, volume 37, pages 62432–62457, Vancouver, BC, Canada, 2024. Curran Associates, Inc. doi: 10.52202/079017-1996.

Junxiong Wang, Wen-Ding Li, Daniele Paliotta, Daniel Ritter, Alexander M. Rush, and Tri Dao. M1: Towards Scalable Test-Time Compute with Mamba Reasoning Models, 2025. arXiv:2504.10449 [cs].

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 Rush. Transformers: State-of-the-Art Natural Language Processing. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, virtual, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 23965–23998, Baltimore, MD, USA, 2022. PMLR.

Guangxuan Xiao. Why Stacking Sliding Windows Can't See Very Far, 2025. URL `https://guangxuanx.com/blog/stacking-swa`.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient Streaming Language Models with Attention Sinks. In *International Conference on Learning Representations*, volume 12, Vienna, Austria, 2024. OpenReview.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A. Raffel, and Mohit Bansal. TIES-Merging: Resolving Interference When Merging Models. In *Advances in Neural Information Processing*

Systems, volume 36, pages 7093–7115, New Orleans, LA, USA, 2023. Curran Associates, Inc.

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

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated Linear Attention Transformers with Hardware-Efficient Training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 56501–56523, Vienna, Austria, 2024b. PMLR. ISSN: 2640-3498.

Tianyu Yu, Bo Ji, Shouli Wang, Shu Yao, Zefan Wang, Ganqu Cui, Lifan Yuan, Ning Ding, Yuan Yao, Zhiyuan Liu, Maosong Sun, and Tat-Seng Chua. RLPR: Extrapolating RLVR to General Domains without Verifiers, 2025. arXiv:2506.18254 [cs].

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native Sparse Attention: Hardware-Aligned and Natively Trainable Sparse Attention. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23078–23097, Vienna, Austria, 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1126.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for Longer Sequences. In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297, virtual, 2020. Curran Associates, Inc.

Chen Zhang, Kuntai Du, Shu Liu, Woosuk Kwon, Xiangxi Mo, Yufeng Wang, Xiaoxuan Liu, Kaichao You, Zhuohan Li, Mingsheng Long, et al. Jenga: Effective memory management for serving llm with heterogeneity. In *Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles*, pages 446–461, 2025a.

Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Re. The Hedgehog & the Porcupine: Expressive Linear Attentions with Softmax Mimicry. In *International Conference on Learning Representations*, volume 12, Vienna, Austria, 2024. OpenReview.

Michael Zhang, Simran Arora, Rahul Chalamala, Benjamin Frederick Spector, Alan Wu, Krithik Ramesh, Aaryan Singhal, and Christopher Re. LoLCATs: On Low-Rank Linearizing of Large Language Models. In *International Conference on Learning Representations*, volume 13, Singapore, Singapore, 2025b. OpenReview.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623, New Orleans, LA, USA, 2023. Curran Associates, Inc.

Appendix

A Definition of Knowledge Distillation Goals **20**

B Related Work **22**

B.1 Modern Recurrent and Hybrid Architectures. 22

B.2 Linearizing LLMs. 23

B.3 Decentralized model training and weight merging. 23

C Extended Background **24**

C.1 Receptive Field of SWA 24

C.2 Attention Sinks 24

D Experimental & Implementation Details **24**

D.1 Hyperparameters and Data Mixes 24

D.2 Generalist vs. Merged Expert Student 25

E Additional Results **25**

E.1 Sink Analysis 27

E.2 Output Gate Analysis 27

E.3 Downstream Evaluations 29

E.4 Inference Time Analysis 32

F Ablations **35**

F.1 Effects of SWA, Attention Sinks & Gating 35

F.2 Effect of Distillation Objective 35

F.3 PEFT vs. FFT 37

F.4 Effect of Phases I&II 37

A Definition of Knowledge Distillation Goals

Lossless and α-Tolerant Knowledge Distillation

Knowledge distillation (KD) aims to transfer the knowledge of a pre-trained *teacher* model to a strictly more efficient *student* model. KD is *lossless* if the transfer process yields a student that matches or outperforms its teacher across a broad spectrum of downstream tasks under an identical evaluation protocol.

More formally, let $\mathcal{B} = \{b_1, \ldots, b_n\}$ be a set of benchmarks, $A_S(b)$ and $A_T(b)$ be the accuracies of the student and the teacher on benchmark b, respectively. A student is at least as good as the teacher up to a tolerance of $\alpha \in [0, 1]$:

$$A_S(b) \geq (1 - \alpha) A_T(b)$$

.

$$\mathbf{1}_\alpha(b) = \begin{cases} 1 & \text{if } A_S(b) \geq (1-\alpha)\, A_T(b), \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

The *Win-and-Tie rate at a tolerance level of* α of the student relative to the teacher is then

$$C_\alpha = \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \mathbf{1}_\alpha(b). \tag{16}$$

We can now define *lossless distillation* when a student exhibits $C_0 \geq 0.5$, i.e. equal or better accuracy on at half of the benchmarks without any tolerance. *α-tolerant distillation* occurs if a student reaches a Win-and-Tie rate $C_\alpha \geq 0.5$ at tolerance level α.

The quality of a distillation process can be assessed by

$$\alpha^\star = \inf\{\alpha \mid C_\alpha \geq 0.5\} \tag{17}$$

Smaller $\alpha^\star$ indicates a more conservative and higher-quality distillation, as less tolerance is required for the student to match the teacher on at least half of the benchmarks.

Win-and-Tie rate curves. The definition above allows us to investigate the Win-and-Tie rate C_α in dependency of different values for α for each model. With increasing tolerance α the student is more often considered as equal or better than the teacher, which means increasing C_α. If the student matches or outperforms the teacher on at least half of the benchmarks, the models can be considered as at least equally performant and thus as a *successful distillation process*. Win-and-tie rate curves show at which tolerance values, successful distillation would be reached.

Model comparison and Pareto front. These curves also allow for model comparison: the higher the curve the better. The **Pareto-front** is at the top curve: at a given tolerance value, the best pick is the method that provides the best Win-and-Tie rate.

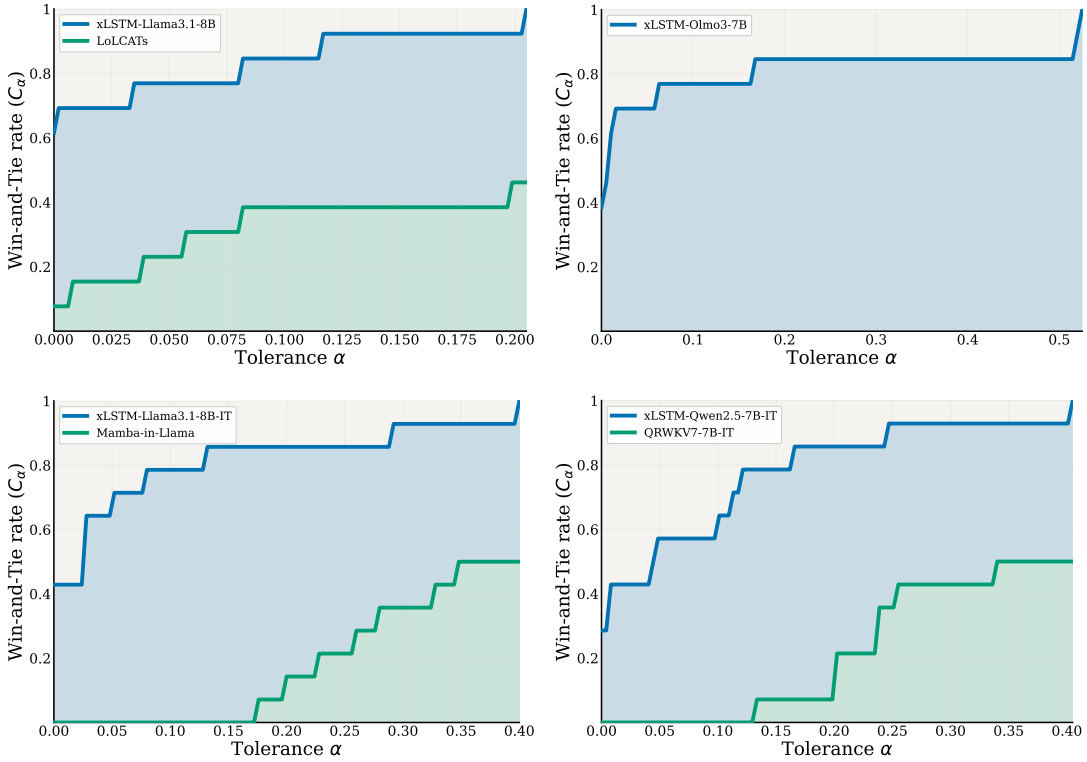


Figure 6: Win-and-Tie rate (C_α) curves of our xLSTM-distilled Models (xLSTM-Llama3.1-8B, xLSTM-Olmo3-7B, xLSTM-Llama3.1-8B-IT, xLSTM-Qwen2.5-7B-IT) and their respective best performing subquadratic baselines across generations benchmarks spanning math, code, STEM, and chat domains (for Olmo3-7B there are no available Baselines at the time of writing). On the y-axis, we show the Win-and-Tie rate C_α between the student and teacher for a given tolerance α. Distillation can be considered successful if the student matches or outperforms the teacher in 50% of the benchmarks. Thus, C_α values above 0.5 can be considered successful distillation attempts.

B Related Work

B.1 Modern Recurrent and Hybrid Architectures.

Linear attention and SSM alternatives to softmax attention. A broad line of work targets sub-quadratic sequence operators with linear-time training and constant-memory decoding. Beyond state space models (SSMs) such as Mamba (Dao and Gu, 2024), this space includes data-dependent convolutions such as Hyena (Poli et al., 2023) and linear-attention families augmented with expressive gating mechanisms such as GLA, DeltaNet, Gated-DeltaNet, RWKV, and xLSTM (Yang et al., 2024b; Schlag et al., 2021; Peng et al., 2023; Beck et al., 2024). Among recurrent gated variants, xLSTM offers two operators. *sLSTM* uses a scalar state with exponential input gating. *mLSTM* maintains a matrix-valued state with head-wise scalar gates and is fully parallelizable, with a recurrent formulation for decoding. These operators provide stable long-horizon memory, efficient rank-one fast-weight updates, and constant-size states at inference.

Hybrid models: inter-layer vs. intra-layer. A growing literature blends quadratic attention with linear attention or SSM primitives. These works can be categorized into inter-layer and intra-layer hybrids.

Inter-layer hybrids interleave attention and linear sequence-mixing blocks. Notable early works include Jamba (Team, 2025b) and Zamba (Glorioso et al., 2024), which alternate between Mamba and global softmax attention layers. Samba (Ren et al., 2025) replaces the global attention layers in earlier designs with sliding-window attention, yielding a fully linear architecture that combines global memory with precise short-range recall. SWAX (Cabannes et al., 2025) alternates sliding-window attention and mLSTM and applies stochastic window resizing to strengthen long-context capability. Recent production-scale efforts extend this pattern. Nemotron-H (NVIDIA, 2025a) and the Nemotron-Nano-2 (NVIDIA, 2025b) replace most attention layers with Mamba. Qwen-Next (Team, 2025c) interleaves Gated DeltaNet layers and gated softmax attention layers. MiniMax-01 (MiniMax, 2025) combines Lightning Attention (Qin et al., 2024), a gated linear-attention variant, with global softmax attention layers. Recently, GPT-OSS combines global attention with sliding window attention (OpenAI, 2025).

Intra-layer hybrids fuse attention and linear or SSM branches within a block and mix outputs through addition or learned gates. Head-wise designs allocate some heads to attention and others to SSMs. Sequence-wise designs split tokens by absolute or relative position. Recent synchronous designs allow linear mixers and attention layers to process the full input sequence at the same time. Representative head-wise models include Hymba (Dong et al., 2025), which assigns half of the heads to parallel attention and the remainder to Mamba. Sequence-wise hybrids include TransMamba (Li et al., 2026), which can switch between attention and SSM mechanisms at different transition points for different layers. As an early influential SWA plus linear hybrid, BASED (Arora et al., 2024) utilized linear attention to compress tokens outside the sliding window into a compact global state.

Our method fits the intra-layer category. Every layer combines sliding-window attention with mLSTM. Unlike Hymba, we do not separate sliding-window and linear branches across heads. Each head models both local and global dependencies with both branches. We chose this design for two reasons. In distillation, the teacher's attention heads are already pre-trained, and assigning local or global roles would require analysis. The design also grants each head more expressive power. Unlike BASED, we do not restrict the linear branch to tokens that leave the sliding window. Both branches model the full input sequence and can exploit their complementarity. This design has recently also been shown to be effective for pre-training at smaller scales (Irie et al., 2025). We fuse the outputs by repurposing the mLSTM output gate, which yields a data-dependent combination of the two attention streams. In distillation, we also find that modeling sink tokens is important. Similar interventions appear in work on KV-cache compression (Xiao et al., 2024) and in hybrids such as Hymba. Unlike Hymba, which introduces learned meta tokens, we simply include the first four sink tokens in the sliding window. Recent work also seeks to mitigate the emergence of attention sinks through pre-training interventions (OpenAI, 2025; Miller, 2023). For those models, there is no need to handle sinks explicitly.

B.2 Linearizing LLMs.

To reduce the prohibitive inference cost of Transformers on long sequences, recent work studies *linearization*, a post-training procedure that replaces some or all softmax self-attention layers in a pre-trained model with linear-time sequence mixers such as gated RNNs or state-space models. Notable examples include T2R (Kasai et al., 2021), SUPRA (Mercat et al., 2024), LoLCATs (Zhang et al., 2025b), Mamba-in-Llama (Wang et al., 2024), RADLADS (Goldstein et al., 2025), MOHAWK (Bick et al., 2024) and Llamba (Bick et al., 2025). Most methods copy compatible weights from the teacher into the student, which yields far greater data efficiency than training from scratch. During the initial alignment stage, LoLCATs, MOHAWK, and Llamba match hidden states with an MSE loss, and MOHAWK and Llamba also match attention maps, which improves alignment but requires materializing the attention matrix with $O(n^2)$ memory. We follow the recipe of copying weights and then calibrating a small set of new gating and head-projection parameters with MSE.

Adaptation strategies divide into low-rank updates and full fine-tuning. LoLCATs and Lizard utilize low-rank adapters to reduce training cost. Mamba-in-Llama, Llamba, MOHAWK, and RADLADS perform full-model updates, which are more compute-intensive but reduce approximation error between student and teacher. Supervision choices also differ. LoLCATs and Lizard optimize next-token cross-entropy, while Mamba-in-Llama, Llamba, MOHAWK, and RADLADS add logit alignment with a KL loss term. We adopt a mixed objective that combines cross-entropy loss with a KL penalty, and we align only k sampled logits so that teacher outputs can be precomputed once, similar to Team (2025a).

Architecturally, some systems hybridize linear sequence operators with variants of softmax attention, and others are fully softmax-free. LoLCATs and Lizard propose intra-layer hybrids that mix sliding-window attention with a linear path, with Lizard adding gated linear attention and global meta tokens. Mamba-in-Llama and MOHAWK study inter-layer hybrids in which only some layers are linearized (Bick et al., 2024). RADLADS and Llamba convert all attention to linear-time mixers. We follow the intra-layer route, pairing sliding-window attention with an mLSTM path and adding sink tokens, then gating the two paths for data-dependent mixing.

Training budgets and datasets vary widely, from about 20M to 12B tokens. RADLADS relies on DCLM (Li et al., 2024) and adds OpenThoughts (Guha et al., 2025) for hybrid reasoning models, MOHAWK uses C4 (Raffel et al., 2020), and Llamba reports gains from FineWeb-Edu (Penedo et al., 2024). Lizard and LoLCATs distill on small instruction datasets, for example, Alpaca. Our schedule uses about 650M tokens for hidden-state matching and 1.3B tokens for end-to-end fine-tuning on a Dolmino-derived mid-training mixture.

B.3 Decentralized model training and weight merging.

A growing set of post-training pipelines decentralize (often within a research organization) by training multiple capability- or domain-specialized variants and then consolidating them into a single deployable model. Early on, Branch-Train-Merge (BTM) showed that independently trained domain experts can be ensembled or collapsed back into a single model via parameter averaging (Li et al., 2022). Recent large-scale systems follow related patterns: DeepSeek-V3.2 unifies expert behaviors by sampling from specialists and then performing SFT on the resulting traces to consolidate capabilities (DeepSeek-AI, 2025), while Command-A reports merging specialists in a way closely aligned with our proposed workflow (Cohere, 2025). Beyond direct weight fusion, MiMo Flash V2 uses multi-teacher on-policy distillation as a mechanism to combine models through teacher feedback on self-generated trajectories (Team, 2026). When consolidation is performed in weight space, simple averaging can already be surprisingly effective (e.g., weight soups) (Wortsman et al., 2022), but more robust methods such as TIES explicitly mitigate parameter interference by resolving sign conflicts and trimming small-magnitude updates (Yadav et al., 2023).

C Extended Background

C.1 Receptive Field of SWA

Although a depth-L stack of SWA layers has a nominal receptive field of LW, in practice the effective receptive field grows much more slowly and is biased toward recent tokens. Empirical measurements, as well as signal-propagation arguments, suggest sublinear growth in L (Xiao, 2025). This mirrors classic results for deep convolutional networks, where the effective receptive field is Gaussian-like and occupies only a small fraction of the theoretical context (Luo et al., 2016).

C.2 Attention Sinks

Transformers often place large, persistent attention mass on initial tokens (e.g., <BOS>). Barbero et al. (2025) argue that sinks are a useful stabilizer that prevents over-mixing through depth and preserves token identity, explaining why *sink patterns* emerge broadly even when those tokens are semantically irrelevant. Sink behavior has been previously exploited for effective KV cache compression. StreamingLLM (Xiao et al., 2024) preserves a small sink prefix ($1-4$ tokens) plus a sliding window for recent context and evicts the remaining tokens in the cache. This compression partially recovers full-attention quality, improves length-generalization, and yields substantial decoding speedups. A complementary perspective is that row-wise softmax in attention forces every head to allocate its entire probability mass across the sequence, encouraging spurious focus on sinks. Minimal fixes have been proposed, replacing $\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$ with $\text{softmax}(x)_i = \frac{e^{x_i}}{e^b + \sum_j e^{x_j}}$, where b is either set to 0 or a learned bias (Miller, 2023; OpenAI, 2025). This is effectively equivalent to adding a no-op *null* key and value.

D Experimental & Implementation Details

All experiments were run on 8 H100 GPUs using PyTorch fully sharded data parallel (FSDP). We configured our training with a global batch size of 64 (using gradient accumulation), mixed precision (bfloat16 for operations, float32 for gradient all-gather), and gradient clipping at a threshold of 1.0 for full finetuning. To maximize GPU utilization, input sequences were packed to fill the maximum context length. We found that preserving the attention mask across these packed sequences, rather than truncating it, improved performance for our hybrid architecture. This finding is consistent with prior work (Buitrago and Gu, 2025).

D.1 Hyperparameters and Data Mixes

Hyperparameters. Table 1 summarizes the hyperparameters used throughout our distillation pipeline. We instantiate the student with an xLSTM [0:1] configuration (mLSTM-only), using 32 mLSTM heads with head dimension 128 and rotary position embeddings, and train at a context length of 4096. Optimization is performed with AdamW (weight decay 0.1) at a global batch size of 64. We enable sequence packing to maximize context utilization while preserving attention masks across packed sequences, which improves throughput without truncating examples. In **Phase I**, we perform layer-wise hidden-state alignment with an MSE objective (cosine learning-rate schedule with peak LR 10^{-2}), keeping the optimization focused on the newly introduced mixer/gating parameters. In **Phase II**, we switch to end-to-end knowledge distillation with a warmup followed by a low constant learning rate (10^{-5} for LLaMa, 7×10^{-7} for Qwen). The Phase II stage interpolates between next-token cross-entropy and KL distillation, using weights $\gamma = 0.9$ (CE) and $\beta = 0.1$ (KL).

Data Mix. Table 2 outlines the domain-specific data mixtures used to train the individual experts(*math*, *code*, *stem*, *chat/instruction-following*), as well a the multi-task mixture used for a *generalist*-xlstm student. Across both LLama and Qwen variants, the data mixes are mostly con-

Table 1: Hyperparameters.

Setting	**Value**
mLSTM/sLSTM configuration	`[1:0]`
mLSTM head dimension	128
Number of mLSTM heads	32
Position embeddings	true
Context size	4096
Total token budget	20B (5B per expert)
Weight decay	0.1
Optimizer	AdamW
Batch size	64
Sequence packing	true
Learning rate (Phase I: MSE matching)	Cosine schedule, max LR 1e−2
Learning rate (Phase II: Knowledge Distillation)	Warmup + LR (1e−5 Llama, 7e−7 Qwen)
CE loss weight	0.9
KL loss weight	0.1
Linear merge weights (Llama)	$\texttt{math} \cdot 0.35 + \texttt{code} \cdot 0.35 + \texttt{stem} \cdot 0.20 + \texttt{chat} \cdot 0.10$
Linear merge weights (Qwen)	$\texttt{math} \cdot 0.20 + \texttt{code} \cdot 0.40 + \texttt{stem} \cdot 0.15 + \texttt{chat} \cdot 0.25$

structed from *Nemotron Nano Pre- and Post-training* datasets. Moreover, for some of the experts (e.g., code), we enhance the datamix with appealing domain-targeted data, such as *OpenCodeInstruct* and *Dolci-Think-SFT*. The stem expert differs the most between the two model families; the Llama stem expert was trained solely on the *Nemotron STEM Post-Training* dataset, whereas for Qwen we enhanced our datamix with more specialized *STEM/MCQ* traces, to minimize the gap to the teacher.

D.2 Generalist vs. Merged Expert Student

To assess the effectiveness of decentralized expert training, we compare a *generalist* student trained on the multi-task mixture against a *merged expert* student obtained by training four domain specialists and combining them via linear merging. In the expert setting, each specialist is trained on its corresponding domain mixture (Table 2) for ∼5B tokens, for a total of 20B tokens, and then merged using the fixed weights in Table 1. In the generalist setting, a single student is trained for the same 20B token budget on the multi-task mixture, keeping architecture and optimization matched.

Table 7 shows that the *generalist* student trained on the multi-task mixture lags behind the *merged-expert* student, with the largest gaps appearing on specialized evaluations that directly probe individual capabilities. xLSTM-Llama3.1-8B-IT consistently outperforms xLSTM-Llama3.1-8B-IT-Generalist on math and reasoning (MATH500 0.54 vs. 0.37, MATH 0.55 vs. 0.37), code generation (HumanEval 0.63 vs. 0.50, HumanEval+ 0.56 vs. 0.47), and instruction-following (IFEval 0.69 vs. 0.49, MT-Bench 6.05 vs. 5.45). xLSTM-Qwen2.5-7B-IT similarly remains ahead of xLSTM-Qwen2.5-7B-IT-Generalist across the same specialized benchmarks (e.g., MATH 0.66 vs. 0.62, MATH Level 5 0.42 vs. 0.36, GPQA-D 0.26 vs. 0.19, MT-Bench 5.96 vs. 5.55).

We attribute this behavior to the multi-task distillation. Although the total token budget for distillation is the same, the generalist allocates fewer updates to the specialized traces that matter for our benchmarks (math, code, and instruction-format data), whereas each expert is trained on a single-domain data distribution. Linear merging then retains much of this specialization in the merged checkpoint, in line with decentralized post-training recipes (Cohere, 2025).

E Additional Results

In this section, we provide additional details on experiments and provide additional results to complement the main text.

Table 2: Data mixes for expert and multi-task linearization.

Model	Dataset	Split	Mixing Weight (%)
xLSTM-Llama Math Expert			
	nvidia/Nemotron-Pretraining-SFT-v1	math	50
	nvidia/Llama-Nemotron-Post-Training-Dataset	math	25
	nvidia/Nemotron-Post-Training-Dataset-v2	math	25
xLSTM-Llama Code Expert			
	nvidia/Llama-Nemotron-Post-Training-Dataset	code	21
	nvidia/Nemotron-Post-Training-Dataset-v2	code	1
	nvidia/Nemotron-Pretraining-SFT-v1	code	64
	nvidia/OpenCodeInstruct		11
	allenai/Dolci-Think-SFT-Python		3
xLSTM-Llama STEM Expert			
	nvidia/Nemotron-Post-Training-Dataset-v1	stem	100
xLSTM-Llama STEM Expert FT			
	nvidia/Nemotron-Post-Training-Dataset-v2	stem	100
xLSTM-Llama Instruction Following Expert			
	nvidia/Nemotron-Pretraining-SFT-v1	general	90
	nvidia/Nemotron-Post-Training-Dataset-v1	chat	10
xLSTM-Llama Instruction Following Expert FT			
	nvidia/Nemotron-Post-Training-Dataset-v2	chat	100
xLSTM-Qwen Math Expert			
	nvidia/Nemotron-Pretraining-SFT-v1	math	76
	nvidia/Llama-Nemotron-Post-Training-Dataset	math	20
	nvidia/Nemotron-Post-Training-Dataset-v2	math	4
xLSTM-Qwen Code Expert			
	nvidia/Llama-Nemotron-Post-Training-Dataset	code	20
	nvidia/Nemotron-Post-Training-Dataset-v2	code	1.2
	nvidia/Nemotron-Pretraining-SFT-v1	code	60
	nvidia/OpenCodeInstruct		12
	allenai/Dolci-Think-SFT-Python		6.8
xLSTM-Qwen STEM Expert			
	nvidia/Nemotron-Pretraining-Specialized-v1		90.6
	nvidia/Nemotron-Science-v1	MCQ	3.4
	nvidia/Nemotron-Post-Training-Dataset-v2	stem	6
xLSTM-Qwen Instruction Following Expert			
	nvidia/Nemotron-Pretraining-SFT-v1	general	88
	nvidia/Nemotron-Post-Training-Dataset-v2	chat	8
	nvidia/Nemotron-Instruction-Following-Chat-v1	chat_if	4
Multi-task			
	nvidia/Nemotron-Pretraining-SFT-v1	math	19
	nvidia/Llama-Nemotron-Post-Training-Dataset	math	5
	nvidia/Nemotron-Post-Training-Dataset-v2	math	1
	nvidia/Nemotron-Pretraining-Specialized-v1	stem	22.66
	nvidia/Nemotron-Science-v1	MCQ	0.84
	nvidia/Nemotron-Post-Training-Dataset-v2	stem	1.5
	nvidia/Nemotron-Pretraining-SFT-v1	general	22
	nvidia/Nemotron-Post-Training-Dataset-v2	chat	2
	nvidia/Nemotron-Instruction-Following-Chat-v1	chat_if	1
	nvidia/Llama-Nemotron-Post-Training-Dataset	code	5
	nvidia/Nemotron-Post-Training-Dataset-v2	code	0.3
	nvidia/Nemotron-Pretraining-SFT-v1	code	15
	nvidia/OpenCodeInstruct		3
	allenai/Dolci-Think-SFT-Python		1.7

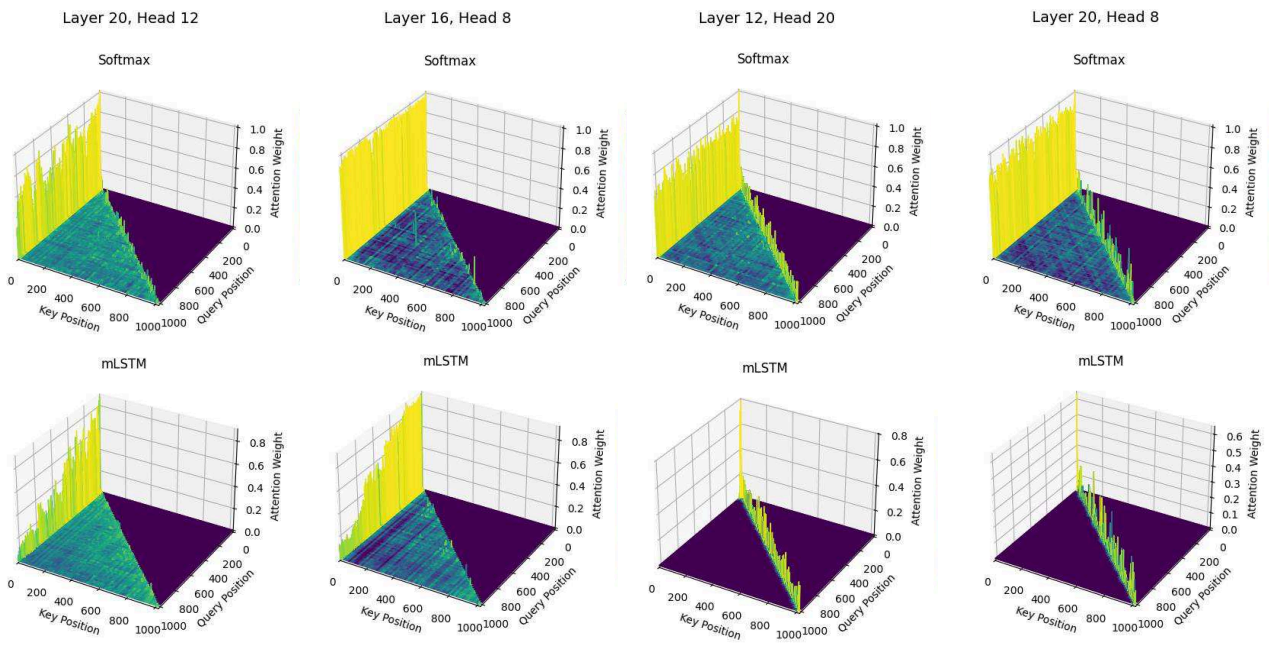


Figure 7: Illustration of **attention sinks** in the Llama 3.1 8B teacher model.

E.1 Sink Analysis

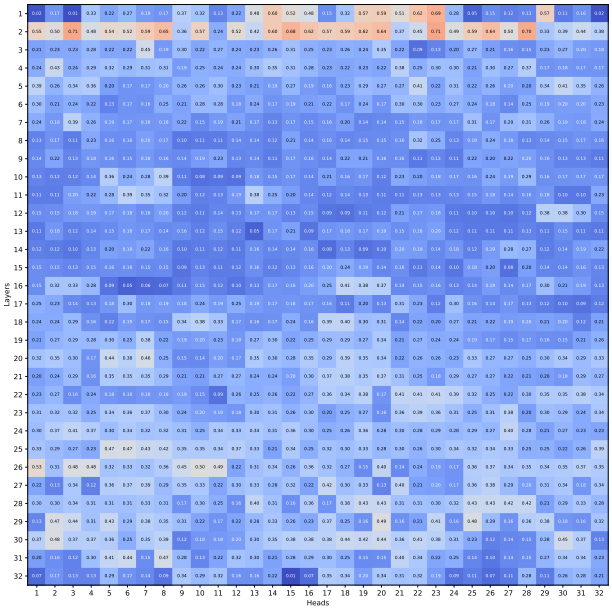
As described in Section 2, Transformers often place large, persistent attention mass on the first on a small set of initial tokens. Barbero et al. (2025) argue that sinks are a useful stabilizer that prevents over-mixing through depth and preserves token identity. Consequently, many established LLMs exhibit sink patterns if they are not equipped with counter-measurements to prevent them from emerging. To illustrate the sink patterns in our Llama 3.1 8B teacher model, in Figure 7 we plot attention maps for two layers and two heads in a 3-dimensional grid. Indeed, we find that the Transformer teacher puts the majority of its attention mass on the first token. In contrast, when analyzing the distilled mLSTM-only checkpoints, we find that they struggle to represent the sink patterns present in the pre-trained teacher model. We observe that this forgetting effect worsens as the input sequence length grows. We suspect this is attributable to the decaying effect of the mLSTM's forget gate.

Therefore, modeling sink behavior is critical for strong performance. In our experiments, we empirically confirmed that modeling sink patterns in combination with the sliding window results in a considerably lower loss and stronger downstream performance, as illustrated in our components ablation in Figure 15.

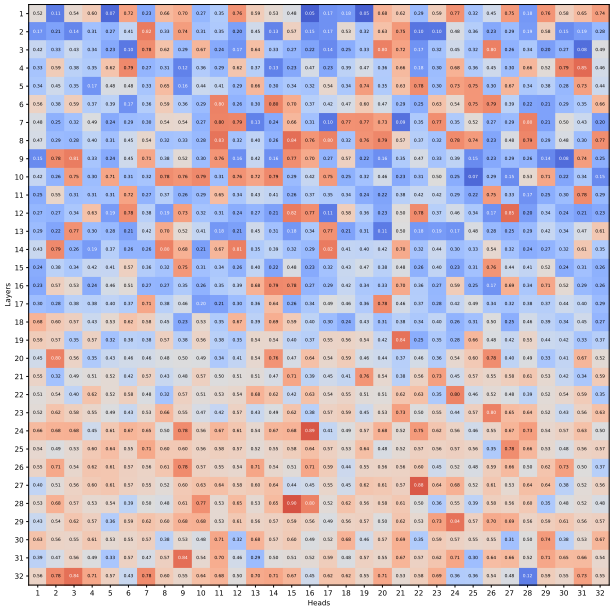
E.2 Output Gate Analysis

As described in Section 3 combine the individual outputs of the mLSTM and SWA components via a data-dependent output gate. To better understand the relative contributions of each component to the final prediction of our student models, we analyze the activations of the output gate. To this end, we forward 128 sequences (sequence length 4096) through our model and record the average activation values for each layer and attention head, resulting in a layer $\times$ head gating matrix.

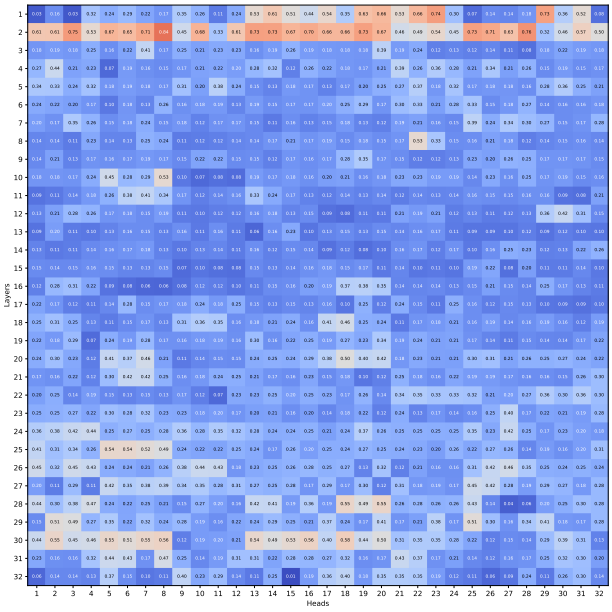
In Figure 8, we observe that both components contribute significantly to the final output prediction across all layers. Noticeably, mLSTM dominates in the first two layers, suggesting that the global contextual information carried by mLSTM blocks is integrated early on in the layer stack. The middle layers ($3 - 16$) are predominantly influenced by SWA, while the final layers ($17 - 32$) exhibit a more balanced contribution from both components, with neither clearly dominating.



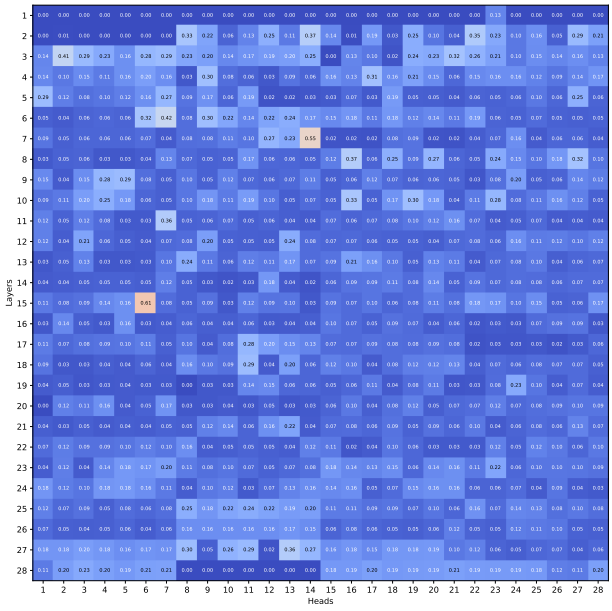
(a) xLSTM-Llama3.1-8B



(b) xLSTM-Olmo3-7B



(c) xLSTM-Llama3.1-8B-IT



(d) xLSTM-Qwen2.5-7B-IT

Figure 8: Illustration of the data-dependent **output gate** output matrix. After training, we analyze how the mLSTM and SWA contributions are mixed across layers and heads. The plots above show the median output gate's sigmoid activation across layers and heads over 128 randomly drawn data samples with a context size of 4096. For a given example, 0 (blue) indicates that only sliding window is used, whereas an activation of 1 (red) means only mLSTM is used. We observe balanced to high mLSTM activations across model families.

Table 3: Downstream **evaluation configurations** for language understanding, generation, and quality benchmarks for all student and teacher `base` models.

Task	**# of shots**	**Generation Budget**
Language Understanding		
PIQA	0	–
ARC-e	0	–
ARC-c	25	100
HellaSwag	10	–
Winogrande	5	–
MMLU	5	10
Language Generation		
GPQA Diamond CoT	0	2048
GPQA Main CoT	0	2048
GSM8K	8	1024
HumanEval	0	1024
HumanEval Plus	0	1024
MBPP	3	256
MBPP Plus	3	1024
Language Quality		
MT-Bench	–	1024

E.3 Downstream Evaluations

After training, we evaluate our distilled student models on downstream tasks, which we group into 4 categories:

- **Language Understanding**: log-likelihood and commonly used multiple-choice benchmarks
- **Language Generation & Reasoning**: mathematics, coding and other established reasoning benchmarks
- **Language Generation Quality via MT-Bench**: across 6 tasks contained in MT-Bench (Zheng et al., 2023)
- **Needle in a Haystack**: we evaluate long context retrieval via Needle in a Haystack tasks.

We conduct all our evaluations using `lm-eval` released by Sutawika et al. (2025). To ensure a consistent and fair comparison, we maintain the same number of few-shot examples, context lengths, and generation budgets across all teacher and student models. Where available, we adopt the same evaluation settings as used by Lambert et al. (2024) and Touvron et al. (2023). We provide all evaluation configurations for `base` and `instruct` models in Tables 3 and 4, respectively.

Language Understanding & Knowledge. To complement the relative teacher scores reported in Figure 3, we report the raw performance scores across all language understanding tasks for all student and teacher models in Table 5. We find that our distilled student models match and in some tasks even slightly exceed their respective teacher performances. In contrast, other linearization recipes fall short of their respective teachers, exhibiting a significant performance gap.

Language Generation & Reasoning. Similarly, we report the raw performance scores across all language generation and reasoning tasks for all student and teacher models in Table 6. Again, we observe that our distilled students almost match their respective teacher performances, while alternative distillation recipes fall short.

MT-bench. Figure 9 shows MT-Bench (Zheng et al., 2023) performance as evaluated by GPT5.1 broken down by category. Both xLSTM-Qwen2.5-7B-ITand xLSTM-Llama3.1-8B-IToutperform their teacher models on all 7 MT-bench categories.

Table 4: Downstream **evaluation configurations** for language understanding, generation, and quality benchmarks for all student and teacher `instruct` models.

Task	# of shots	Gen Budget	Chat	Multiturn ICL
Language Understanding				
PIQA	0	–	✓	✗
ARC-e	0	–	✓	✗
ARC-c	25	100	✓	✓
HellaSwag	10	–	✓	✓
Winogrande	5	–	✓	✓
MMLU	5	10	✓	✓
Language Generation				
GPQA Diamond CoT	0	2048	✓	✗
GPQA Main CoT	0	2048	✓	✗
GSM8K CoT	5	3072	✓	✓
MATH 500	0	3072	✓	✓
MATH	0	3072	✓	✓
Math Level 5	4	1024	✓	✓
HumanEval (64) Instruct	0	3072	✓	✗
HumanEval+ (64) Instruct	0	3072	✓	✗
MBPP	3	1024	✓	✓
MBPP+	3	1024	✓	✓
Cruxeval-O	0	3072	✓	✓
Cruxeval-I	2	3072	✓	✓
Language Quality				
MT-Bench	–	1024	✓	✓

Table 5: Raw scores for **downstream evaluations** on **language understanding** tasks. Our models perform comparably to their Llama3.1-8B, Olmo3-7B, Llama3.1-8B-IT, and Qwen2.5-7B-IT teacher models across language understanding tasks.

Model	PIQA↑	ARC-e↑	ARC-c↑	HellaSwag↑	Winogrande↑	MMLU↑	Avg.↑
T: Qwen2.5-7B(Yu et al., 2025)	78.7	80.3	**63.7**	**80.2**	**76.4**	**75.3**	**75.8**
S: QRWKV6-7B(Goldstein et al., 2025)	**79.4**	**80.6**	60.2	77.9	73.9	65.4	72.9
T: Llama3.1-8B(Team, 2024)	**80.1**	81.6	**57.8**	81.9	76.9	65.9	74.0
S: LoLCATs(Zhang et al., 2025b)	79.5	**83.2**	54.6	75.3	74.0	52.9	69.9
S: xLSTM-Llama3.1-8B	80.0	82.9	55.9	**82.3**	**79.0**	**66.1**	**74.4**
T: Olmo3-7BOlmo (2025)	**78.1**	80.7	**58.4**	56.7	**73.4**	**66.4**	**69.0**
S: xLSTM-Olmo3-7B	77.9	**81.1**	57.8	**56.9**	72.8	65.7	68.7
T: Qwen2.5-7B-IT(Yu et al., 2025)	74.5	68.6	59.4	74.5	64.2	**74.5**	69.3
S: QRWKV7-7B-IT(Goldstein et al., 2025)	76.9	74.9	**62.0**	**78.1**	69.9	68.2	**71.7**
S: xLSTM-Qwen2.5-7B-IT	**79.4**	**80.2**	60.8	60.0	74.7	73.7	71.4
S: xLSTM-Qwen2.5-7B-IT-Generalist	79.1	73.7	54.1	58.9	**75.4**	66.6	68.7
T: Llama3-8B-IT(Team, 2024)	77.3	**76.9**	**66.2**	**72.4**	**73.9**	**68.6**	**72.6**
S: Mamba-in-Llama(Wang et al., 2024)	**82.8**	74.8	62.1	64.1	59.2	56.8	66.6
T: Llama3.1-8B-IT(Team, 2024)	**79.9**	**81.8**	56.7	**59.3**	**78.0**	68.9	70.1
S: xLSTM-Llama3.1-8B-IT	79.8	81.2	56.0	58.7	76.9	68.0	**71.4**
S: xLSTM-Llama3.1-8B-IT-Generalist	78.8	71.1	**58.4**	57.9	73.2	**72.8**	67.9

Table 6: Raw scores for **downstream evaluations** on **language generation tasks** tasks. Our models perform comparably or slightly exceed their respective Llama3.1-8Band Olmo3-7Bteacher models, while significantly outperforming distilled models with alternative linearization recipes.

Model	GPQA-D (0)↑	GPQA (0)↑	GSM8K (8)↑	HumanEval (0)↑	HumanEval+ (0)↑	MBPP (3)↑	MBPP+ (3)↑	Avg.↑
T: Qwen2.5-7B(Yu et al., 2025)	**32.3**	**30.8**	**80.8**	**65.9**	**47.6**	**62.6**	**80.4**	**57.2**
S: QRWKV6-7B(Goldstein et al., 2025)	9.60	13.2	23.8	20.7	33.5	0.0	27.0	14.4
T: Llama3.1-8B(Team, 2024)	10.6	13.8	48.4	35.4	**29.9**	**48.4**	**61.9**	35.5
S: LoLCATs(Zhang et al., 2025b)	1.77	2.23	3.87	2.13	3.35	0.0	0.0	1.9
S: xLSTM-Llama3.1-8B	**17.7**	**15.8**	**57.8**	**39.0**	23.8	42.8	56.9	**36.3**
T: Olmo3-7BOlmo (2025)	**20.7**	19.0	**67.5**	**32.9**	**28.7**	**50.6**	**71.7**	**41.6**
S: xLSTM-Olmo3-7B	17.7	**19.4**	56.6	32.3	27.4	36.8	52.4	34.7

Table 7: Raw scores for **downstream evaluations** on **language generation tasks** tasks. Our models perform comparably to their respective Llama3.1-8B-ITand Qwen2.5-7B-IT teacher models, while significantly outperforming distilled models with alternative linearization recipes.

Model	GSM8K (8)↑	MATH500 (0)↑	MATH (0)↑	MATH Level 5 (0)↑	HumanEval (0)↑	HumanEval+ (0)↑	MBPP (3)↑	MBPP+ (3)↑	CruxEval-O (0)↑	CruxEval-I (0)↑	GPQA-D (0)↑	GPQA (0)↑	IfEval (0)↑	MT-Bench (0)↑	Avg.↑
T: Qwen2.5-7B-IT(Yu et al., 2025)	**0.90**	**0.74**	**0.74**	**0.49**	**0.81**	**0.74**	0.61	**0.79**	0.41	0.41	**0.34**	**0.36**	**0.74**	5.20	0.95
S: QRWKV7-7B-IT(Goldstein et al., 2025)	0.53	0.40	0.39	0.12	0.62	0.55	0.49	0.69	0.18	0.22	0.22	0.19	0.59	3.96	0.65
S: xLSTM-Qwen2.5-7B-IT	**0.90**	0.65	0.66	0.42	0.78	0.71	**0.63**	**0.79**	**0.42**	**0.42**	0.26	0.22	0.67	**5.96**	**0.96**
S: xLSTM-Qwen2.5-7B-IT-Generalist	0.89	0.63	0.62	0.36	0.76	0.68	0.62	0.78	0.37	0.36	0.19	0.14	0.62	5.55	0.90
T: Llama3.1-8B-IT(Team, 2024)	**0.85**	0.51	0.50	0.19	**0.63**	**0.57**	**0.59**	0.69	0.26	**0.28**	**0.30**	**0.32**	**0.78**	5.08	0.83
S: xLSTM-Llama3.1-8B-IT	0.83	**0.54**	**0.55**	**0.22**	**0.63**	0.56	0.54	0.66	**0.34**	0.27	0.21	0.20	0.69	**6.05**	**0.88**
S: xLSTM-Llama3.1-8B-IT-Generalist	0.82	0.37	0.37	**0.22**	0.50	0.47	0.54	**0.71**	0.29	0.25	0.16	0.14	0.49	5.45	0.77
T: Llama3-8B-IT(Team, 2024)	**0.82**	**0.29**	**0.29**	**0.09**	**0.57**	**0.54**	**0.56**	**0.74**	**0.26**	**0.20**	**0.34**	**0.29**	**0.77**	**5.50**	**0.80**
S: Mamba-in-Llama(Wang et al., 2024)	0.68	0.12	0.12	0.04	0.38	0.34	0.33	0.41	0.09	0.16	0.25	0.22	0.52	3.97	0.54

* 50% attention layers

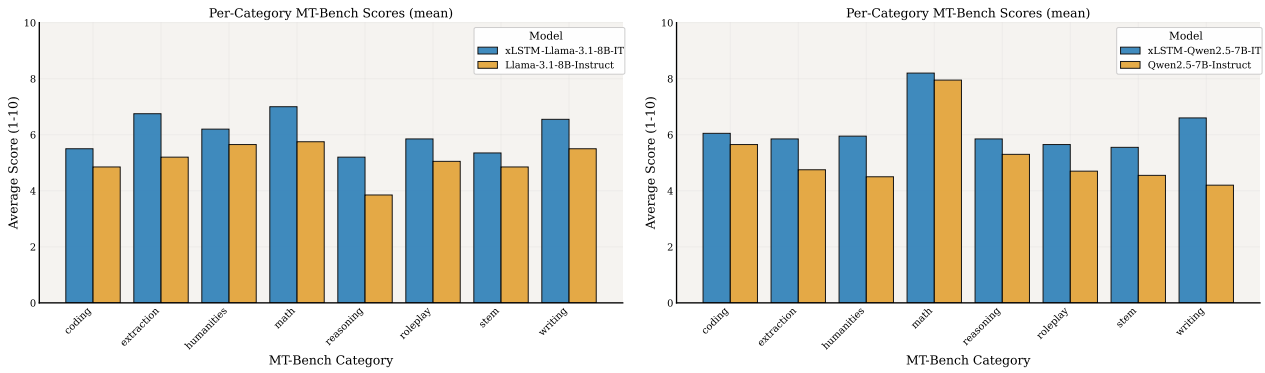


Figure 9: MT-Bench performance per category as judged by GPT-5.1.

Table 8: Comparison of Llama3.1-8B-IT and Qwen2.5-7B-IT against our Students on Needle-in-a-Haystack tasks.

Model	NIAH Single				NIAH Multi			
	1024	**4096**	**8192**	**16384**	**1024**	**4096**	**8192**	**16384**
Llama3.1-8B-IT	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
xLSTM-Llama3.1-8B-IT	0.686	0.148	0.078	0.024	0.664	0.130	0.088	0.034
Qwen2.5-7B-IT	0.998	0.998	1.000	1.000	1.000	1.000	1.000	0.998
xLSTM-Qwen2.5-7B-IT	0.656	0.140	0.078	0.024	0.640	0.116	0.088	0.034

Needle in a Haystack. In Appendix Table 8, we report Needle-in-a-Haystack (NIAH; Hsieh et al. (2024)) accuracy for both the single-needle and multi-needle variants across four context lengths (1k, 4k, 8k, and 16k tokens). Overall, the instruction-tuned baselines (Llama3.1-8B-IT and Qwen2.5-7B-IT) maintain near-perfect recall across all lengths, whereas our distilled Students degrade with increasing context, with the largest drop occurring at 4k and compounding further at 8k and 16k. We note that we also observe substantial long-context degradation when continually fine-tuning the Transformer baseline without linearization, and because our Student was not exposed to long-context distillation data during adaptation, it remains unclear whether the sharp decline reflects an inherent memory limitation of the fixed-size state or could be partially mitigated with improved long-context instruction tuning; we leave this question to future work.

E.4 Inference Time Analysis

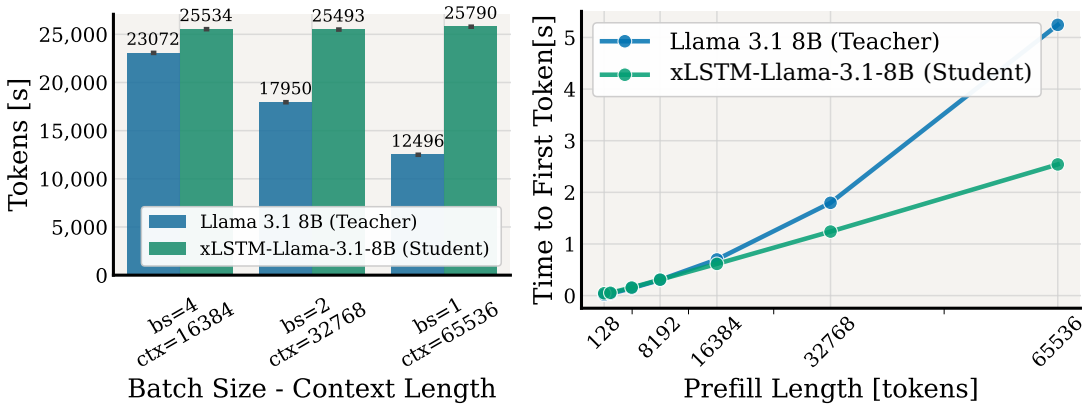
In this section, we provide additional details on our inference time tests.

Setup. We run all our inference time tests on a single H100 GPU with 80GB of memory. Our implementations for both student and teacher are based on the `transformers` library Wolf et al. (2020) and their respective classes for Llama 3 (Touvron et al., 2023). For our hybrid student model, we replace the self-attention mechanism of the teacher model with our hybrid mechanism of mLSTM and SWA. To accelerate runtimes, we leverage `torch.compile` using a static KV-cache. For the Transformer-based teacher, we leverage FlashAttention-2 (Dao, 2024). Similarly, for our hybrid student, we make use of the Triton kernels released by Beck et al. (2025a) for the mLSTM part and FlashAttention for the sliding window part. To enable compilation of our hybrid student via `torch.compile`, we utilize a custom static cache implementation that retains both the mLSTM states and the relevant keys/values of sink tokens and the sliding window.

Prefill vs. generation. We separate our inference time tests into two stages: prefilling and generation. While the prefilling stage encodes the input prompt by the user and populates the KV cache, the generation or decoding stage autoregressively samples tokens until sequence termination, starting from the pre-filled KV cache (Pope et al., 2023). Our inference tests are characterized by three core hyperparameters, the batch size B, the context length C, and the generation budget G. Consequently, if one would only perform prefilling without generating any tokens, then $G = 0$. Similarly, if we only perform generation without any prefill sequence, then $C = 0$ These two scenarios are reflected in Figure 10a and Figure 11a, respectively. For every combination of B, C, and G, we always first conduct three warmup runs, which include the compilation of our model. Afterwards, we record runtimes, memory consumption, and throughput across five runs and average metrics across them.

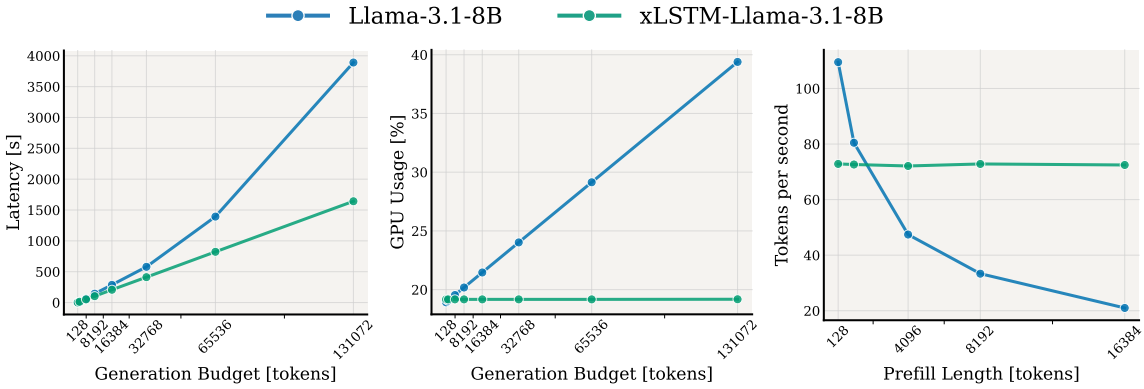
Generation latency & memory consumption. To complement the results that we presented in Section 4.4, we report additional metrics for generation latency and memory consumption across varying batch sizes $B \in [1, 4, 8]$ and sequence lengths $C \in [128, 1024, 4096, 16384, 32768, 65536, 131072]$ in Figures 12 and 13, respectively. The purpose of this experiment is to better understand how the inference time advantages behave with increasing batch sizes and sequence lengths.

In Figure 12, we make two important observations. First, our hybrid student only exhibits a slight



(a) Prefill Throughput (b) Prefill Latency, $B = 1$

Figure 10: Inference comparison for the **prefilling stage** between the Transformer-based teacher and our mLSTM-based student. In **(a)**, we report prefill throughput for varying context lengths and batch sizes. In **(b)**, we show the prefilling latency for varying prefill lengths and $B = 1$.



(a) Latency, $B = 1$ (b) GPU RAM %, $B = 1$ (c) Throughput, $B = 8$

Figure 11: Inference comparison for the **generation stage** between the Transformer-based teacher and our xLSTM-based student. In **(a)**, we show generation latency at different generation budgets ($B = 1$). In **(b)**, we report the memory consumption in % of GPU memory during the generation ($B = 1$). In **(c)**, we show the generation throughput when generating 100 tokens with varying prefill lengths and $B = 8$.

increase in total inference latency when increasing the batch size from 1 to 8. This is because its computational complexity does not grow with the sequence length, due to the recurrent inference mode of mLSTM and the fixed sliding window of 256 tokens. Consequently, for the largest batch size we compare ($B = 8$), the computation remains memory-bound. Second, we observe that the computational demand of the Transformer-based teacher grows faster with increasing sequence length, due to the quadratic complexity of self-attention. For example, when increasing the batch size from 1 to 4, the runtime at $C = 65K$ of our teacher model increases two-fold. Similarly, the required memory grows quickly as the batch size increases and the KV cache gets larger, causing the model to OOM, as indicated by missing dots for the teacher. The differences in RAM consumption are further highlighted in Figure 13. The RAM consumption of the Transformer-based teacher grows quickly with increasing sequence length and batch size. In contrast, our hybrid student remains constant along the sequence length and only exhibits a slight increase with larger batch sizes due to the constant memory complexity.

Generation throughput. Finally, we report the average generation throughputs for a fixed generation budget of $G = 100$ tokens with varying prefill lengths and $B \in [1, 4, 8]$ in Figure 14. Again, we observe significantly higher throughputs for our hybrid student of up to almost $4\times$ that of the Transformer-

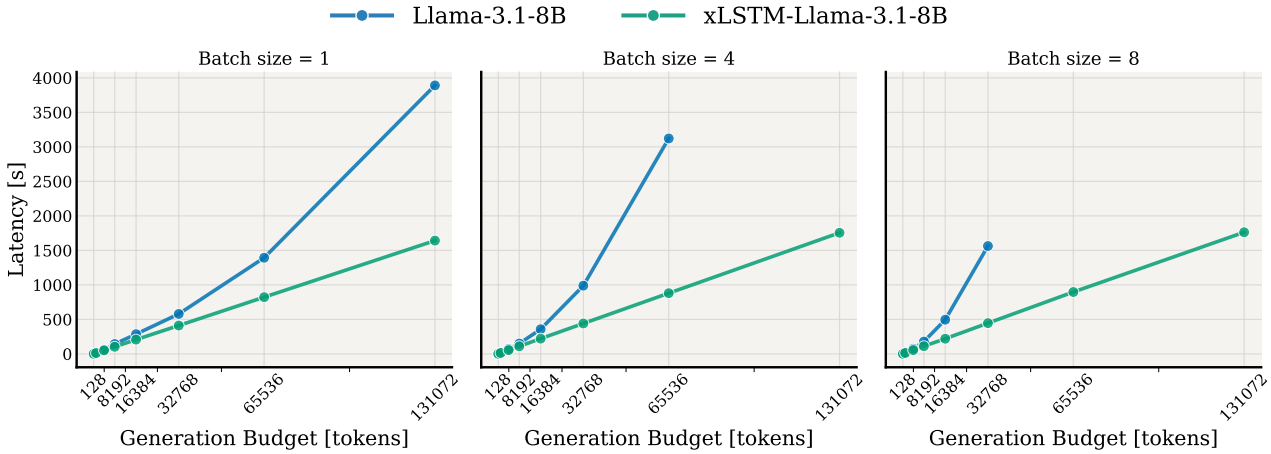


Figure 12: **Latency**. We report the latency for generation with varying token generation budgets and batch sizes. Our mLSTM-based student exhibits lower generation latency than the Transformer-based teacher. This advantage grows with larger generation budgets and batch sizes. Missing dots for the teacher indicate OOM.

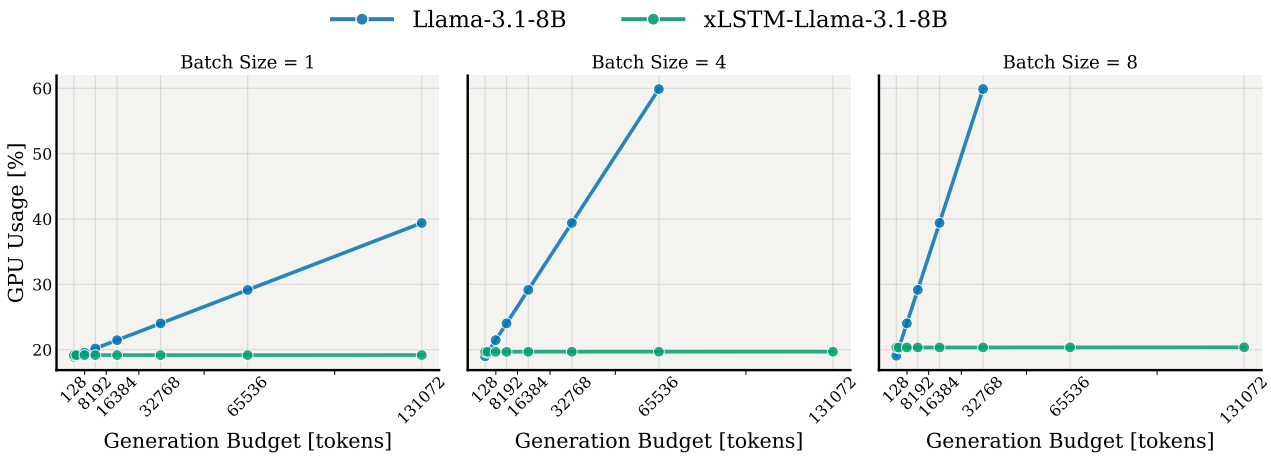


Figure 13: **GPU RAM**. We report the memory consumption in % of GPU memory during the generation for varying batch sizes. Our mLSTM-based student requires significantly less memory compared to the Transformer-based teacher. This advantage grows with larger generation budgets and batch sizes. Missing dots for the teacher indicate OOM.

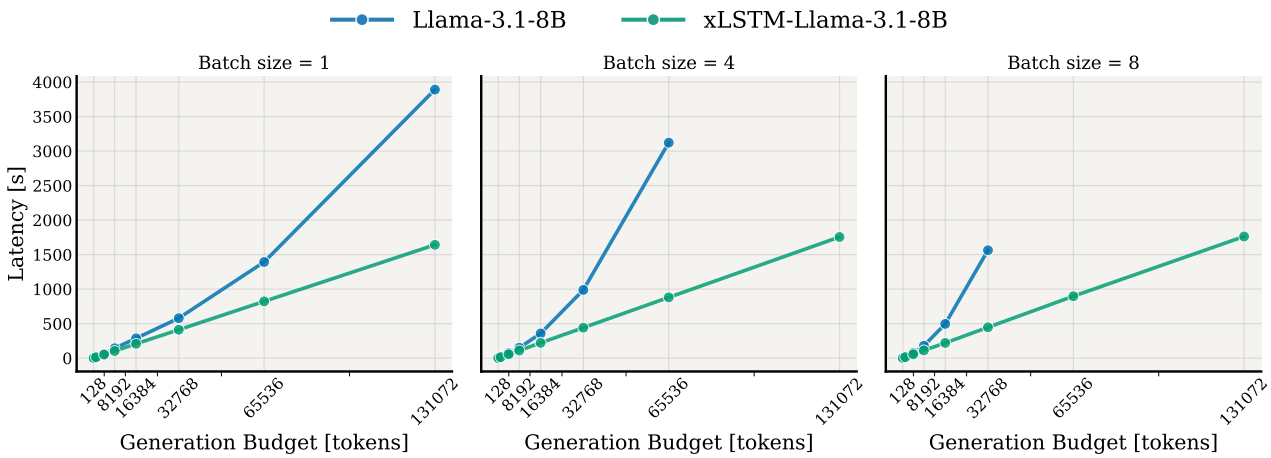


Figure 14: **Throughput**. We report the average throughput for generating 100 tokens with varying prefill lengths and batch sizes. Missing dots for the teacher indicate OOM.

based teacher as the sequence length increases. Note that due to OOMs, we only show metrics for $B = 4$ and $B = 8$ up to $C = 32K$ and $C = 16K$, respectively.

F Ablations

In this section, we empirically analyse a variety of important components of our distillation recipe.

F.1 Effects of SWA, Attention Sinks & Gating

First, we ablate components of our hybrid attention operator under a fixed linearization recipe. We compare four variants: (i) pure linear attention, (ii) mLSTM, (iii) mLSTM + SWA, and (iv) mLSTM + SWA + sink tokens. For SWA, we use a window size $W = 512$ and designate the first four tokens of each sequence as attention sinks. We train with a CE/KL objective weighted by $\gamma = 0.9$ and $\beta = 0.1$. Evaluation loss curves in Figure 15 show that replacing linear attention with gated mLSTM reduces validation CE throughout training, indicating that head-wise gates increase expressivity and better match the teacher. Adding the SWA branch yields a further, uniform CE reduction, consistent with exact short-range recall. Introducing a small prefix of sink tokens provides an additional gain. Overall, the full hybrid (mLSTM + SWA + sinks) converges faster and achieves the lowest CE, i.e., the tightest student–teacher alignment.

F.2 Effect of Distillation Objective

To find the best tradeoff between CE loss and the KL between the student and teacher models, we sweep over CE and KL loss weights γ and β (see Eq. 13). Evaluation losses of the different fine-tuning configurations are shown in Figure 16. As the KL β grows, validation CE rises and the student under-adapts. As $\beta \to 0$, CE is lowest but the student drifts, with KL diverging. Prior work observes that large post-finetuning KL correlates with forgetting of capabilities (Shenfeld et al., 2025). Based on this, we adopt CE γ=0.9 and KL $\beta = 0.1$. This setting achieves CE essentially matching the $\gamma = 1.0, \beta = 0$ configuration while keeping KL dramatically smaller, providing substantial freedom to adapt to the new attention operators without sacrificing teacher alignment. Notably, even a small KL term materially improves alignment. Therefore, we use CE $\gamma = 0.9$ and KL $\beta = 0.1$ as our base setting.

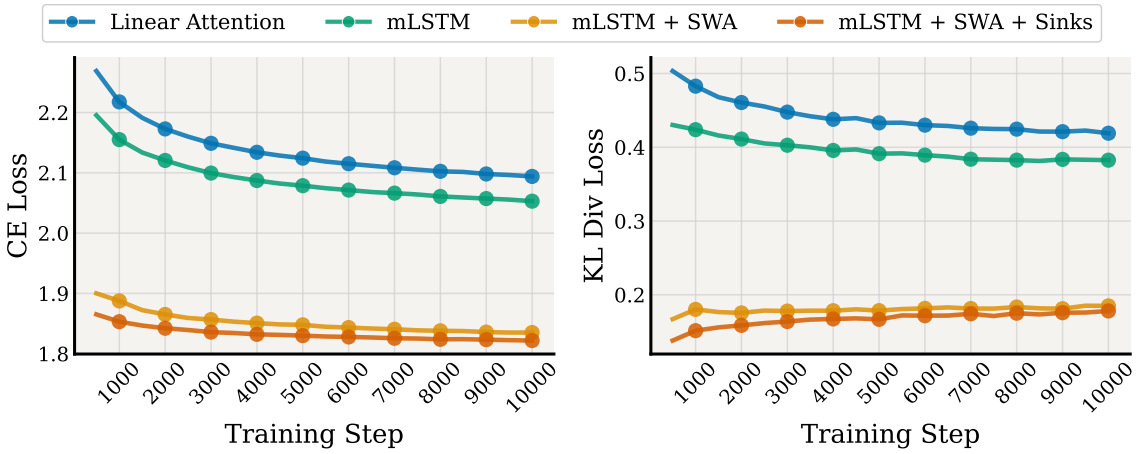


Figure 15: Ablation on the **effects of mLSTM, SWA & sinks** on our hybrid student. We track the cross-entropy loss (left) and KL loss (right) throughout stage II and across individual components. All components contribute considerably to the final performance

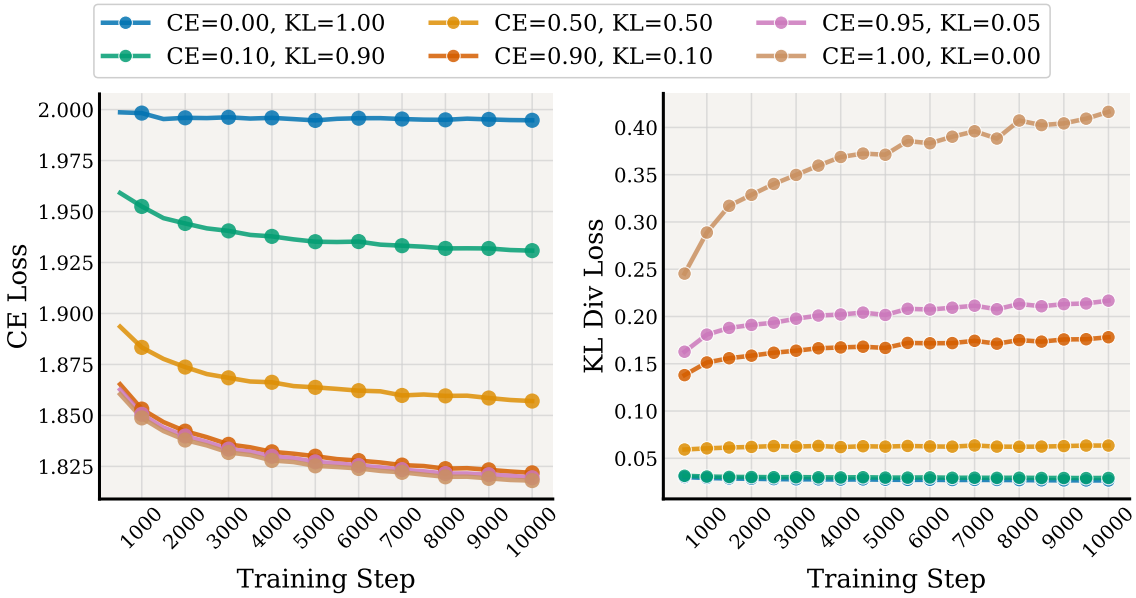


Figure 16: Ablation on the **effects of different loss weightings** on our hybrid student. We track the CE (left) and KL loss (right) throughout stage II. We find that weighting CE loss with 0.9 and KL loss with 0.1 provides a good tradeoff between performance and teacher alignment.

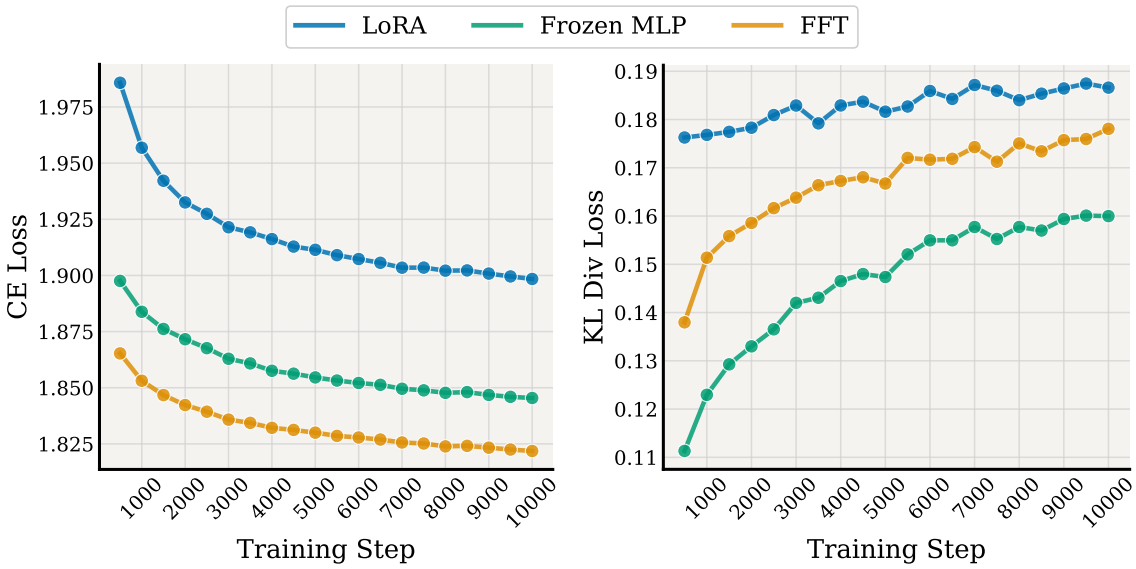


Figure 17: Ablation on the **effects of different training strategies** on our hybrid student. We track the CE loss (left) and KL loss (right) throughout stage II. This analysis reveals that full finetuning achieves a comparable KL while yielding superior downstream performance.

F.3 PEFT vs. FFT

Prior linearization recipes use low-rank adaptation (LoRA, Hu et al., 2022) to recover performance lost during conversion (Zhang et al., 2025b; Nguyen et al., 2025). While low-rank adaptation (LoRA) is attractive for its cost and scalability, it is unclear whether its capacity suffices to close the student-teacher gap. We therefore ablate three strategies spanning the efficiency–expressivity trade-off: (i) LoRA with high ranks ($r = 256$), (ii) updating only the sequence-mixer parameters while freezing all MLP blocks and embeddings, and (iii) FFT. We follow the baseline linearization setup in Section 3 and train with CE weight $\gamma = 0.9$ and KL weight $\beta = 0.1$. Figure 17 reports validation CE and KL. As expected, CE decreases as more parameters are unfrozen: FFT achieves the lowest CE, followed by mixer-only, then LoRA. Surprisingly, this additional flexibility does not increase deviation from the teacher. A small KL penalty is sufficient to keep all three methods comparably close in KL. Consequently, we adopt FFT, which offers the greatest capacity to adapt to the new attention operators while remaining close to the teacher model.

F.4 Effect of Phases I&II

We find that layer-wise hidden-state alignment (Phase I) is necessary but not sufficient to recover most of the teacher's performance within a limited training budget. This finding aligns with previous work by (Zhang et al., 2025b). Consequently, a two-phase approach, hidden-state alignment followed by full finetuning (Phase II), consistently outperforms standard finetuning given the same budget. While Phase I is crucial for aligning the student's intermediate hidden representations with the teacher's, the performance gains from this stage alone quickly plateau after a limited number of steps (Figure 18).

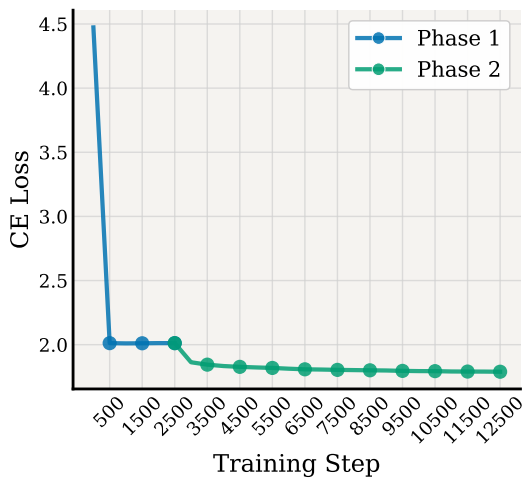


Figure 18. Phase I&II ablation.