LLAMBA: SCALING DISTILLED RECURRENT MODELS FOR EFFICIENT LANGUAGE PROCESSING

Aviv Bick[*]
Carnegie Mellon University
abick@cs.cmu.edu

Tobias Katsch, Nimit Sohoni & Arjun Desai
Cartesia.ai

Albert Gu
Carnegie Mellon University & Cartesia.ai

ABSTRACT

We introduce Llamba, a family of efficient recurrent language models distilled from Llama-3.x into the Mamba architecture. The series includes Llamba-1B, Llamba-3B, and Llamba-8B, which achieve higher inference throughput and handle significantly larger batch sizes than Transformer-based models, while maintaining comparable benchmark performance. Furthermore, Llamba demonstrates the effectiveness of cross-architecture distillation using MOHAWK (Bick et al., 2024), achieving these results with less than 0.1% of the training data typically used for models of similar size. To take full advantage of their efficiency, we provide an optimized implementation of Llamba for resource-constrained devices such as smartphones and edge platforms, offering a practical and memory-efficient alternative to Transformers. Overall, Llamba improves the tradeoff between speed, memory efficiency, and performance, making high-quality language models more accessible.

1 INTRODUCTION

Transformer-based LLMs dominate language modeling, but their quadratic attention mechanism makes them computationally expensive and difficult to scale efficiently. This technical paper introduces the **Llamba model family**, a suite of SSM-based language models—including Llamba-1B, Llamba-3B, and Llamba-8B—that address these efficiency challenges. Retaining the overall structure of Llama models, Llamba models are distilled from Llama-3, replacing self-attention with Mamba-2 layers to achieve high inference throughput while maintaining strong performance across benchmarks.

Llamba achieves its performance with drastically reduced training data requirements through *architecture distillation*. Unlike traditional large language models (LLMs) that rely on massive datasets spanning trillions of tokens, Llamba achieves comparable results with significantly fewer resources by using MOHAWK (Bick et al., 2024) to transfer the knowledge from strong pretrained Transformer-based LLMs to a Mamba-based architecture. For example, *Llamba-3.1-8B was distilled using just 12 billion tokens—less than 0.1% of the training data required for Llama-3.1-8B*. This remarkable data efficiency highlights the effectiveness of architecture distillation methods in transferring knowledge from pretrained models while significantly reducing both data and computational demands. As a result, Llamba presents a scalable and cost-effective solution for high-performance language modeling.

Extending the benefits of their efficient design, **we provide on-device implementations of the Llamba models** [1], optimized for deployment on private devices such as smartphones and edge computing platforms with limited memory and computational resources. These implementations highlight the advantages of linear models, such as the Llamba family, by delivering high-quality language modeling on devices where traditional Transformer architectures are often impractical due to their heavy memory and compute demands.

[*] Work was done while at Cartesia.ai

[1] https://github.com/cartesia-ai/edge

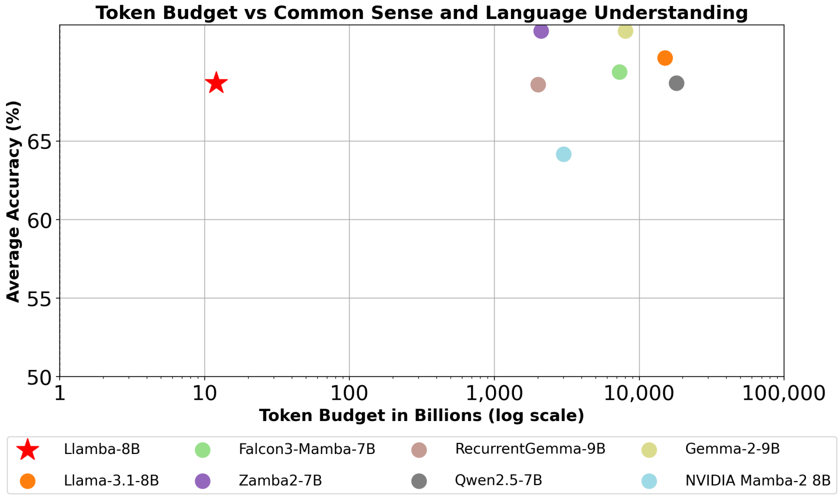


Figure 1: Average accuracy is measured over multiple benchmarks, including ARC Challenge, ARC Easy, PIQA, Winogrande, HELLASWAG, OpenBookQA, and MMLU, providing a comprehensive assessment of a model's Common Sense and Language Understanding.

Overall, the Llamba family introduces several key contributions:

- **Distillation efficiency:** Using the MOHAWK framework, Llamba achieves state-of-the-art performance with less than 0.1% of the training data required by comparable models. This represents a significant advancement in data and compute efficiency for LLMs.
- **On-device deployment:** We provide quantized Llamba models, along with an MLX implementation for edge devices like iPhones and MacBooks, which demonstrate near-constant memory usage, making them ideal for resource-constrained environments.
- **Benchmark performance:** Llamba-1B, Llamba-3B, and Llamba-8B perform on par with traditional models across a wide range of benchmarks, setting a new standard for efficiency and performance in recurrent architectures.

These advancements position Llamba as a versatile and scalable solution for efficient language modeling, bridging the gap between performance, resource efficiency, and accessibility.

2 Related Work

Language Models. Transformer-based models, such as those in the Llama (Touvron et al., 2023), and Qwen (Yang et al., 2024) series, have shown strong performance across various language modeling tasks. These models underwent extensive pretraining on large-scale corpora and incorporate techniques like instruction tuning and curated datasets to enhance generalization in few-shot and zero-shot settings on various tasks.

While Transformers remain dominant, recent work has explored alternatives to their purely quadratic attention mechanisms to improve efficiency while maintaining strong performance. Structured state space models (SSMs) (Gu & Dao, 2023; Dao & Gu, 2024) have emerged as a promising direction, offering a distinct approach to sequence modeling. At large scales, Falcon-Mamba (Zuo et al., 2024), a fully SSM-based model stacking Mamba-1 layers, has matched and even outperformed Transformers on key tasks. Falcon3-Mamba extends this by pretraining for an additional 1.5 trillion tokens, incorporating high-quality post-training data, and expanding the context length from 8K to 32K tokens, further enhancing its capabilities. However, despite these advances, SSM-based models still underperform Transformers on algorithmic tasks (Jelassi et al., 2024; Wen et al., 2024; Waleffe et al., 2024).

To balance these trade-offs, hybrid models combining attention and SSMs have gained interest for leveraging the strengths of both architectures. Examples include RecurrentGemma (Botev et al., 2024), which integrates gated linear recurrences with local attention, and Zyphra's Zamba (Glorioso et al., 2024b), which combines Mamba-1 layers with a shared attention mechanism spanning the network. Zamba-2 (Glorioso et al., 2024a) builds on this by replacing Mamba-1 blocks with Mamba-2 for improved efficiency, increasing shared attention layers from one to two for enhanced global context modeling, and applying Low-Rank Adaptation (LoRA) (Hu et al., 2021) to shared MLP blocks for parameter-efficient depth adjustments. Other hybrid architectures (Lieber et al., 2024; Waleffe et al., 2024) further underscore the interest in models that balance expressiveness and efficiency.

Distillation. Various methods have been proposed to distill large Transformer-based models into more efficient architectures while maintaining performance. SUPRA (Mercat et al., 2024) propose a procedure to linearize softmax attention into a form of linear attention by copying the weight matrices and fine-tuning. LoLCATs (Zhang et al., 2024) introduces a linearization approach that replaces softmax attention with linear attention through attention transfer, followed by low-rank adaptation, enabling the creation of large-scale linearized models with improved efficiency. (Wang et al., 2025) leverages the State-Space Duality (SSD) in Dao & Gu (2024) to transfer the linear projection weights from the attention layers into Mamba-based models. Their experiments include hybrid models with an increasing proportion of interleaved attention layers, demonstrating that a balanced combination of state-space models (SSMs) and attention preserves performance while improving efficiency. MOHAWK (Bick et al., 2024) distills Transformers into SSMs through progressive alignment, allowing subquadratic models to leverage Transformer training resources effectively. These approaches demonstrate the viability of distilling computationally expensive Transformers into efficient models while retaining competitive performance.

3 Model Architecture

Unlike the Mamba and Mamba-2 architectures, which were designed for training from scratch, *Llamba is directly motivated by architectural distillation*. In particular, the Mohawk distillation framework involves aligning sub-networks of the model at various levels of granularity (Section 4). This constraints Llamba to retain the overall architecture of the teacher model, ideally modifying only the attention matrix mixer by replacing it with a subquadratic alternative.

The Llamba models—Llamba-1B, Llamba-3B, and Llamba-8B—comprise 16, 28, and 32 residual Mamba-2 blocks, respectively, followed by feed-forward layers. These models share the tokenizer and vocabulary of Llama-3.1, with hidden dimensions of 2048 for Llamba-1B, 3072 for Llamba-3B, and 4096 for Llamba-8B. In addition, Llamba differs from the original Mamba-2 architecture (Dao & Gu, 2024) in the following ways (see Figure 2b):

- **Alternating MLP blocks**: Llamba interleaves Llama's Gated MLP components between each Mamba-2 mixing layer, unlike Mamba-1 and Mamba-2, which consist solely of SSM blocks.
- **Multi-head structure**: Llama-3.x models use grouped-query attention (GQA) (Ainslie et al., 2023; Shazeer, 2019), which employs 32 query heads and 8 key-value heads to boost inference speed and reduce the size of the decoding cache. However, Mamba's recurrent layers don't rely on a cache, so these optimizations aren't needed. Instead, *Llamba blocks feature a Multi-Head variant* of Mamba-2 with 32 heads and dimensions of 64, 96, or 128, along with a state size of 64. While this design differs from Mamba-2's "multi-value attention" (MVA) architecture, it still keeps inference costs low.
- **Non-linearities**: We remove the normalization before the output projection and the activation after convolution, as these are non-linear operations that do not exist in the attention block and hurts alignment (See Section 4.1).
- **Discretization**: Llamba uses *Discrete-Mamba-2*, a variant that projects the matrix $\mathbf{A}$ directly from the input, eliminating the discretization parameters Δ to better match the inherently discrete attention mechanisms.

Notably, these changes not only facilitate the distillation process but also improve training efficiency. Alternating with MLPs **reduces the number of temporal mixing layers**, enabling Llamba to achieve faster computation than other models of comparable size (see Section 6.2). Furthermore, training becomes simpler and more efficient by eliminating normalization-related all-reduce operations.

	STAGE 1	STAGE 2	STAGE 3	OVERALL TOKENS
LLAMBA-1B	300M	2.7B	5B	8B
LLAMBA-3B	500M	4B	5.5B	10B
LLAMBA-8B	500M	5B	6.5B	12B

Table 1: Token allocations during the distillation process for different Llamba models and MOHAWK stages (Matrix Orientation, Hidden-State Alignment, and Knowledge distillation).

4 DISTILLATION

The Llamba models were distilled using MOHAWK (Bick et al., 2024) from Meta's Llama-3.x family (Touvron et al., 2023). Specifically, Llamba-3.1-1B was distilled from Llama-3.2-1B-Instruct, Llamba-3B from Llama-3.2-3B-Instruct, and Llamba-8B from Llama-3.1-8B-Instruct.

4.1 MOHAWK

Following the MOHAWK framework (Bick et al., 2024), Llamba models were initialized by setting the convolution layer of the Mamba block to an identity kernel (nullifying its effect) and configuring the multiplicative skip connection to directly pass the input unchanged, effectively initializing the block as an identity function. Subsequently, three key steps were implemented: *Matrix Orientation*, *Hidden-State Alignment*, *Weight Transfer with Knowledge Distillation*.

Matrix Orientation. This phase is used to align the student and teacher matrix mixers. Specifically, we minimize the distance between the materialized Llamba matrix mixer and Llama's self-attention matrix. Notably, Llama uses an MQA architecture, which results in 32 attention heads with shared weights. Since Llamba's 32 heads are not tied (it uses a Multi-Head architecture), it learns independent weights, unlike the dependent matrices of its teacher.

Hidden-State Alignment. For Hidden-State Alignment, each Mamba-2 block of the Llamba model was aligned independently using the L2 distance, guided by the output of the preceding layer.

Weight Transfer & Knowledge Distillation. We begin this stage by transferring the MLP weights, normalization layers, input embedding, and output head from the Llama-3.x models to each Llamba model. Unlike previous works (Wang et al., 2024; Bick et al., 2024), we did not freeze the MLP components and optimized them using the same learning rate of the mixing components. During Knowledge Distillation, each Llamba model was aligned with the respective teacher model using the cross-entropy loss of their logits. After this phase's loss saturation, all models were further distilled from Llama-3.1-70B-Instruct for their remaining tokens.

4.2 TRAINING DETAILS

The Llamba models were trained using mixed precision and Fully Sharded Data Parallel (FSDP) on a single node with 8 H100 GPUs, with activation checkpointing enabled. Training used the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a weight decay of 0.1. The maximum learning rates were 1×10^{-4} for the first two MOHAWK stages across all models, 5×10^{-5} for the third stage of Llamba-1B and Llamba-3B, and 1×10^{-5} for the third stage of Llamba-8B. Batch sizes were set to 64 in the first MOHAWK stage and 128 in the second and third stages. We used the Warm-Stable-Decay (WSD) scheduler (Hu et al., 2024), with a minimum learning rate of 1×10^{-8} and warm-up and decay phases each spanning 10% of total training steps.

During the distillation process Table 1, a total of 12 billion tokens were processed for Llamba-8B, 10 billion tokens for Llamba-3B and Llamba-1B used only 8 billion tokens, highlighting its significantly smaller allocation of training data used for distillation compared to training without any teacher supervision (see Figure 1).

4.3 Data

Data quality is critical for accurately modeling temporal interactions in the MOHAWK distillation setting. MOHAWK transfers only the MLP weights that affect the hidden dimensions, excluding the sequence mixer weights related to the time dimension. This omission limits the ability to capture time-dependent information directly. For the distillation process, two datasets were used. The first, fineweb-edu-4.0, is derived from fineweb-edu (Penedo et al., 2024), itself a subset of the broader fineweb dataset. This refined subset includes only highly educational web pages, filtered using a 4.0 classifier score threshold - higher than the 3.0 threshold used for fineweb-edu. Since distillation requires relatively few tokens, this focused approach was practical and effective.

The *Matrix Orientation* and *Hidden-State Alignment* processes were conducted using the fineweb-edu-4.0 dataset with packed sequences of length 2048 (see Table 1 for more details). In contrast, *Knowledge Distillation* was initially performed using fineweb-edu-4.0, and subsequently, the OpenHermes-2.5 dataset was employed for an additional 4 epochs, processing 200 million tokens per epoch with sequences of length 4096. The combination of these datasets played a pivotal role in enhancing the MMLU score.

Our results demonstrate that this dataset selection significantly improves the performance of MMLU (Hendrycks et al., 2021). As shown in Figure 3, while the C4 (Raffel et al., 2023) and fineweb datasets achieve similar scores across most benchmarks, fineweb-edu drives a marked improvement in MMLU. Beyond this, our approach highlights an important takeaway: *we achieve strong results using only established open-source datasets, in contrast to many alternative models that rely on highly curated proprietary datasets*. This demonstrates the feasibility of leveraging openly available resources for high-quality model performance.

Furthermore, we emphasize that architecture distillation (e.g. the MOHAWK framework) and data curation are orthogonal and synergistic, and we hypothesize that our distillation results could be improved further by incorporating even higher-quality datasets.

5 On-Device Implementation

The advantages of sub-quadratic language models are particularly impactful in compute- and memory-constrained environments, making them ideal for on-device applications. To support efficient inference, we implemented optimized Mamba-2 kernels, including state-space model and Conv1D layers, using Apple's Metal framework. These kernels are specifically designed for Apple Silicon, leveraging its GPU parallelism and unified memory architecture for efficient execution.

Our implementation integrates seamlessly with MLX (Hannun et al., 2023), a machine learning framework optimized for Apple Silicon. MLX enables dynamic graph construction and efficient tensor operations while utilizing unified memory to minimize data transfer overhead. Additionally, we support 4-bit quantization to further reduce memory usage, enabling models to run effectively on devices with limited resources.

These optimizations allow our models to maintain high throughput and low memory consumption, even in long-context scenarios, making them highly suitable for real-time, on-device applications. The implementation is available in the released repository `https://github.com/cartesia-ai/edge`.

6 Evaluations

6.1 Performance

Comparison Against Pretrained Models. Table 2 presents a comparative analysis of downstream evaluation results across different models and tasks. The evaluation includes recent advanced models such as hybrids of subquadratic and attention layers (e.g., Zamba2-7B (Glorioso et al., 2024a)) and purely subquadratic models (e.g., RecurrentGemma-9B (Botev et al., 2024), Falcon-Mamba-7B (Zuo et al., 2024)). Additionally, we include Llama-3.2-1B, Llama-3.2-3B, and Llama-3.1-8B to benchmark performance against state-of-the-art non-hybrid Transformer models.

We evaluate the models' performance in both zero-shot and few-shot settings across a range of standard datasets: ARC (Clark et al., 2018), PIQA (Bisk et al., 2019), Winogrande (WG) (Sakaguchi et al., 2019), HellaSwag (HS) (Zellers et al., 2019), Lambada OpenAI (LO) (Paperno et al., 2016), MMLU (Hendrycks et al., 2021), and OpenBookQA (OBQA) (Mihaylov et al., 2018). All evaluations were conducted using `bfloat16` precision and the *lm-eval-harness v0.4.4* Python library (Gao et al., 2024).

Comparison Against Distilled Models. Table 4 compares Llamba-8B with other distilled models of similar size. We specifically focus on MMLU, which is known to be difficult for recurrent models (Waleffe et al., 2024), and has the biggest gap for distilled models from prior work. Llamba significantly improves MMLU relative to the teacher model.

We found that MMLU performance takes much longer to improve compared to other benchmarks in our end-to-end distillation. Llamba reached teacher performance on other tasks in a very small number of tokens, while MMLU took longer to improve.

We also note that some of the baselines are actually hybrid models, which have a 1:1 ratio of attention to recurrent layers. We note that even sliding window attention has a strong effect because the MMLU context size is very small. Although Llamba still has a gap to the teacher model, we consider this result a large step forward for the performance of distilled recurrent models.

6.2 Throughput

Llamba achieves higher throughput than its Llama-3.1-8B teacher (see Figure 4), even when Llama-3.1-8B generates four times fewer tokens. This performance gain stems from Llamba's recurrent Mamba-2 layers, whose state size remains constant regardless of sequence length. Additionally, Llamba incorporates MLPs with fewer temporal mixing layers than Dao & Gu (2024), enabling it to: (1) scale to batches twice as large as a pure Mamba-2 model, as MLPs are stateless in time, and (2) reduce kernel launch overhead when CUDA graph optimization is not applied, since Mamba layers typically require more kernel preparation time.

We have evaluated the throughput of Llama-3.1-8B and Llamba-8B models on a single NVIDIA H100 80GB HBM3. To ensure a fair comparison, all models were tested under a reasonable level of optimization, using `torch.compile(model, fullgraph=True)` and CUDA graph for consistent performance baselines.

Furthermore, on-device evaluation results highlight Llamba's exceptional performance in decoding scenarios with constrained hardware. Specifically, on Apple Silicon M3 Pro (36GB) using MLX, Llamba maintains consistent high throughput and low memory consumption at 4-bit quantization (see Figure 5). In contrast, the inference performance of Llama-3.1-8B deteriorates linearly with increasing context size, emphasizing the superior efficiency of Llamba in handling long sequences.

7 Conclusion

The Llamba model family represents a significant step forward in creating efficient and scalable recurrent language models. It achieves high performance with less than 0.1% of the data typically required for similar models while maintaining strong performance across various benchmarks.

We see great promise in distilling pre-trained transformers into subquadratic architectures. Future directions include improving the quality and diversity of datasets used in distillation, optimizing the handling of long contexts, and expanding Llamba's deployment to real-time, low-power applications such as IoT devices and wearable technology. Refining the distillation process further could unlock new capabilities and broaden the applications of this model family, solidifying its impact on efficient language modeling.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints, 2023. URL https://arxiv.org/abs/2305.13245.

Aviv Bick, Kevin Y. Li, Eric P. Xing, J. Zico Kolter, and Albert Gu. Transformers to ssms: Distilling quadratic knowledge to subquadratic models, 2024. URL https://arxiv.org/abs/2408.10189.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Aleksandar Botev, Soham De, Samuel L Smith, Anushan Fernando, George-Cristian Muraru, Ruba Haroun, Leonard Berrada, Razvan Pascanu, Pier Giuseppe Sessa, Robert Dadashi, Léonard Hussenot, Johan Ferret, Sertan Girgin, Olivier Bachem, Alek Andreev, Kathleen Kenealy, Thomas Mesnard, Cassidy Hardin, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Armand Joulin, Noah Fiedel, Evan Senter, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, David Budden, Arnaud Doucet, Sharad Vikram, Adam Paszke, Trevor Gale, Sebastian Borgeaud, Charlie Chen, Andy Brock, Antonia Paterson, Jenny Brennan, Meg Risdal, Raj Gundluru, Nesh Devanathan, Paul Mooney, Nilay Chauhan, Phil Culliton, Luiz Gustavo Martins, Elisa Bandy, David Huntsperger, Glenn Cameron, Arthur Zucker, Tris Warkentin, Ludovic Peran, Minh Giang, Zoubin Ghahramani, Clément Farabet, Koray Kavukcuoglu, Demis Hassabis, Raia Hadsell, Yee Whye Teh, and Nando de Frietas. Recurrentgemma: Moving past transformers for efficient open language models, 2024. URL https://arxiv.org/abs/2404.07839.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

Paolo Glorioso, Quentin Anthony, Yury Tokpanov, Anna Golubeva, Vasudev Shyam, James Whittington, Jonathan Pilault, and Beren Millidge. The zamba2 suite: Technical report, 2024a. URL https://arxiv.org/abs/2411.15242.

Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. Zamba: A compact 7b ssm hybrid model, 2024b. URL https://arxiv.org/abs/2405.16712.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2023.

Awni Hannun, Jagrit Digani, Angelos Katharopoulos, and Ronan Collobert. MLX: Efficient and flexible machine learning on apple silicon. https://github.com/ml-explore, 2023. Version 0.0.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

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, 2021. URL https://arxiv.org/abs/2106.09685.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. Minicpm: Unveiling the potential of small language models with scalable training strategies, 2024. URL https://arxiv.org/abs/2404.06395.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying, 2024. URL https://arxiv.org/abs/2402.01032.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A hybrid transformer-mamba language model, 2024. URL https://arxiv.org/abs/2403.19887.

Jean Mercat, Igor Vasiljevic, Sedrick Keh, Kushal Arora, Achal Dave, Adrien Gaidon, and Thomas Kollar. Linearizing large language models, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering, 2018. URL https://arxiv.org/abs/1809.02789.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context, 2016. URL https://arxiv.org/abs/1606.06031.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben 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, 2024. URL https://arxiv.org/abs/2406.17557.

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, 2023. URL https://arxiv.org/abs/1910.10683.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Noam Shazeer. Fast transformer decoding: One write-head is all you need, 2019. URL https://arxiv.org/abs/1911.02150.

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.

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, Garvit Kulshreshtha, Vartika Singh, Jared Casper, Jan Kautz, Mohammad Shoeybi, and Bryan Catanzaro. An empirical study of mamba-based language models, 2024. URL https://arxiv.org/abs/2406.07887.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander M. Rush, and Tri Dao. The mamba in the llama: Distilling and accelerating hybrid models, 2024. URL https://arxiv.org/abs/2408.15237.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander M. Rush, and Tri Dao. The mamba in the llama: Distilling and accelerating hybrid models, 2025. URL https://arxiv.org/abs/2408.15237.

Kaiyue Wen, Xingyu Dang, and Kaifeng Lyu. Rnns are not transformers (yet): The key bottleneck on in-context retrieval, 2024. URL https://arxiv.org/abs/2402.18510.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024. URL `https://arxiv.org/abs/2407.10671`.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL `https://arxiv.org/abs/1905.07830`.

Michael Zhang, Simran Arora, Rahul Chalamala, Alan Wu, Benjamin Spector, Aaryan Singhal, Krithik Ramesh, and Christopher Ré. Lolcats: On low-rank linearizing of large language models, 2024. URL `https://arxiv.org/abs/2410.10254`.

Jingwei Zuo, Maksim Velikanov, Dhia Eddine Rhaiem, Ilyas Chahed, Younes Belkada, Guillaume Kunsch, and Hakim Hacid. Falcon mamba: The first competitive attention-free 7b language model, 2024. URL `https://arxiv.org/abs/2410.05355`.

A Appendix

A.1 Benchmark Accuracy Across Reasoning Tasks (Part I)

Table 2 compares zero-shot and few-shot accuracy on ARC, PIQA, and Winogrande. The first panel highlights that recurrent Llamba models already match or surpass their Transformer teachers despite being trained on two orders of magnitude fewer tokens. The second and third panels show the same trend at larger model sizes, with hybrid systems (e.g. Zamba2-7B) edging ahead only when both attention and SSMs are combined. Overall, the table sets the stage for later efficiency analyses by showing that strong accuracy does not have to come from quadratic attention.

Table 2: Comparison of downstream performance (accuracy %) across various models in zero-shot and few-shot settings. For ARC-Challenge, ARC-Easy, and PIQA, we have used normalized logits' results. Along with accuracy, each model is annotated with the number of training or distillation tokens (in trillions) and its architecture—Recurrent (R), Transformer (T), or Hybrid (H). For models with a sliding window, the window size is also specified. We use an instruct-tuned version whenever one is available; however, we exclude this label for brevity.

Model	Arch.	Tokens (T)	ARC Challenge		ARC Easy		PIQA		Winogrande	
			0-shot	25-shot	0-shot	25-shot	0-shot	10-shot	0-shot	5-shot
Llama-3.2-1B (Teacher)	T	≤ 9	38.1	42.0	68.5	71.8	74.4	75.4	59.7	62.0
Llamba-1B	R	0.008	37.2	41.8	69.5	71.2	74.0	74.3	60.6	58.1
Mamba-1.4B	R	0.3	32.9	36.0	60.9	66.6	73.7	74.4	60.6	60.1
RecurrentGemma-2B	H ($w = 2048$)	≤ 2	35.6	48.0	51.2	73.3	67.2	75.8	55.7	64.1
Qwen2.5-3B	T	≤ 18	48.1	60.8	72.9	85.1	78.3	79.8	69.8	71.3
Llama-3.2-3B (Teacher)	T	9	45.6	52.1	74.3	79.8	75.8	77.7	67.6	68.8
Llamba-3B	R	0.01	48.5	53.0	79.0	81.1	78.6	79.5	70.4	72.4
Mamba2-2.8B	R	0.3	35.9	39.5	64.3	71.8	75.6	76.4	63.4	64.6
Qwen2.5-7B	T	18	55.1	67.0	81.3	89.5	80.3	82.4	71.1	75.1
Llama-3.1-8B (Teacher)	T	15	55.1	60.0	81.7	85.8	81.1	82.4	73.9	77.3
Llamba-8B	R	0.012	54.6	60.0	82.5	85.8	80.9	81.5	73.3	76.9
Falcon3-Mamba-7B	R	7.3	53.2	65.9	72.5	86.7	79.7	82.3	69.1	72.1
Zamba2-7B	H	2.1	56.1	68.3	80.6	88.7	81.1	81.3	76.9	80.1
RecurrentGemma-9B	H ($w = 2048$)	2	57.1	60.2	78.9	84.5	80.6	81.7	73.7	75.6

A.2 Benchmark Accuracy Across Knowledge-Heavy Tasks (Part II)

Table 3 focuses on HellaSwag, Lambada, MMLU, and OpenBookQA. Transformer baselines hold a small edge on Lambada, which rewards long-range context, but Llamba remains highly competitive while keeping its recurrent footprint. The gap narrows further on MMLU when shot count increases, suggesting that the recurrent architecture can absorb in-context examples effectively. Hybrid models again shine on HellaSwag, hinting that a small attention window still helps narrative completion.

Table 3: Comparison of downstream performance (accuracy %) across various models in zero-shot and few-shot settings. For HellaSwag and OpenBookQA, we have used normalized logits' results. Along with accuracy, each model is annotated with the number of training or distillation tokens (in trillions) and its architecture—Recurrent (R), Transformer (T), or Hybrid (H). For models with a sliding window, the window size is also specified. We use an instruct-tuned version whenever one is available; however, we exclude this label for brevity.

Model	Arch.	Tokens (T)	HellaSwag		Lambada		MMLU		OpenBookQA	
			0-shot	10-shot	0-shot	10-shot	0-shot	5-shot	0-shot	10-shot
Llama-3.2-1B (Teacher)	T	≤ 9	60.8	59.4	60.1	53.1	46.0	45.5	34.6	37.6
Llamba-1B	R	0.008	61.2	60.2	48.4	39.0	38.0	31.3	37.0	38.0
Mamba-1.4B	R	0.3	59.1	59.6	64.4	57.0	24.7	24.8	36.8	37.4
RecurrentGemma-2B	H ($w = 2048$)	≤ 2	60.3	69.4	52.5	53.0	40.2	42.1	30.4	43.2
Qwen2.5-3B	T	≤ 18	74.9	75.2	65.8	58.1	65.5	66.4	41.8	46.2
Llama-3.2-3B (Teacher)	T	9	70.4	73.2	65.9	61.9	60.4	59.8	35.8	39.6
Llamba-3B	R	0.01	73.8	74.3	65.8	60.0	52.7	50.3	42.8	42.8
Mamba2-2.8B	R	0.3	66.2	66.6	68.1	61.2	25.7	25.1	40.4	42.0
Qwen2.5-7B	T	18	80.5	81.3	69.5	62.7	71.7	74.4	48.6	52.0
Llama-3.1-8B (Teacher)	T	15	79.3	80.0	73.0	65.6	68.0	68.4	43.0	48.2
Llamba-8B	R	0.012	77.6	78.7	69.4	65.0	61.0	60.0	43.4	45.8
Falcon3-Mamba-7B	R	7.3	79.8	81.6	67.5	63.6	65.0	66.0	48.0	50.2
Zamba2-7B	H	2.1	81.5	83.5	74.6	68.6	64.7	67.2	45.2	52.4
RecurrentGemma-9B	H ($w = 2048$)	2	80.1	80.9	54.1	69.6	55.1	56.5	46.0	49.2

A.3 Comparison with Alternative Distillation Pipelines

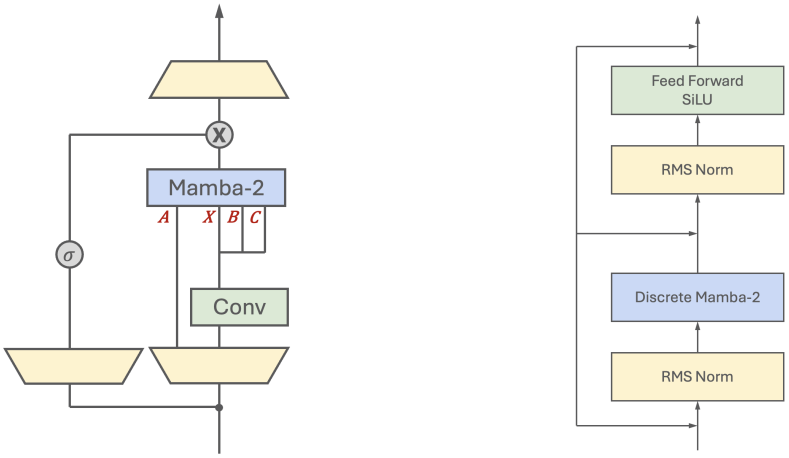
Table 4 places Llamba-8B beside recent distillation efforts such as SUPRA, Mamba2-Llama, and LoLCATs. When normalized by teacher performance, Llamba-8B achieves the highest relative MMLU score (80.6 %), reinforcing that its block-wise recurrent distillation strategy is data-efficient. The result also underscores that token budget alone does not dictate quality; architectural fit between student and teacher matters.

Model	Arch.	Tokens (B)	Teacher	MMLU (5-shot)	Relative Score (%)
SUPRA	R	100	Mistral 7B	34.2	24.6
Mamba2-Llama 3	R	20	Llama 3 8B	43.2	43.8
Mamba2-Llama 3	H	20	Llama 3 8B	56.7	76.2
LoLCATs	H	0.04	Mistral 7B	51.4	70.5
LoLCATs	H	0.04	Llama 3 8B	52.8	66.8
Llamba-8B	R	12	Llama 3.1 8B	**60.0**	**80.6**

Table 4: MMLU (5-shot) performance of various models, specifying their architectures, training tokens, and teacher models. The number of training or distillation tokens is given in billions (B), and model architectures are categorized as Recurrent (R) or Hybrid (H). The Relative Score represents the model's MMLU accuracy as a percentage of its teacher's performance, with a baseline of 25 (random guessing).

A.4 Architectural Building Blocks

Figure 2 juxtaposes a Discrete Mamba-2 block with the full Llamba layer stack. Panel (a) strips Mamba-2 of extra normalizations to keep the signal path short, while panel (b) shows how Llamba alternates this mixer with a standard feed-forward sub-layer. The schematic clarifies why Llamba can reuse pretrained Llama weights: only the attention mixer is swapped for its discrete recurrent counterpart, leaving the rest of the block intact.



(a) The Discrete Mamba-2 block Bick et al. (2024) modifies the original Mamba-2 architecture by removing both post-convolution activation and pre-output projection normalization. Additionally, the Discrete Mamba-2 sequence mixer eliminates the Δ discretization parameter and directly projects the $\mathbf{A}$ matrix from the input.

(b) Llamba models—Llamba-1B, Llamba-3B, and Llamba-8B—are based on the architecture of their Llama teacher models. Each block comprises two sub-blocks with residual connections: (1) RMS Normalization followed by a Discrete Mamba-2 layer. (2) RMS Normalization followed by a feed-forward layer.

Figure 2: Comparison of the Discrete Mamba-2 block and the Llamba architecture.

A.5 Effect of Pre-Training Corpus on Distillation Quality

Figure 3 evaluates Llamba-8B after hidden-state alignment on three corpora. C4 and fineweb give nearly identical downstream curves, but fineweb-edu delivers a noticeable MMLU lift, likely because its educational content covers many of the exam-style topics found in the benchmark. The plot suggests that modest corpus curation can translate directly into higher reasoning scores without extra compute.

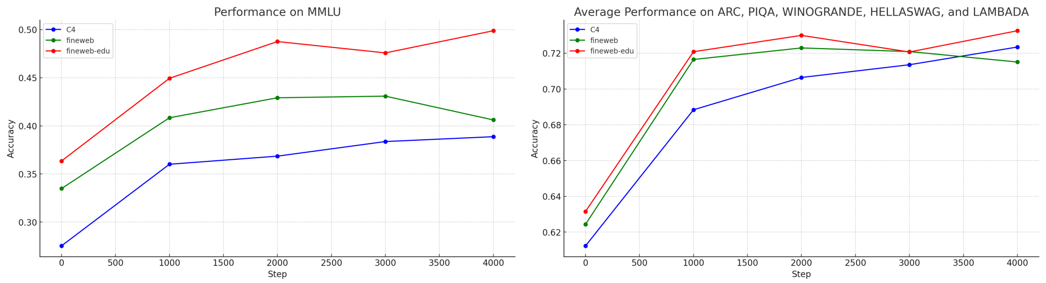


Figure 3: An evaluation of Llamba-8B's knowledge distillation step (MOHAWK's stage 3) across three datasets: C4, fineweb, and fineweb-edu. Each model underwent hidden-state alignment (MOHAWK's stage 2) on its respective dataset using 4 billion tokens and subsequently underwent testing with knowledge distillation on 1 billion tokens. It is observed that although all datasets yield similar outcomes across most benchmarks, MMLU shows notable improvement when utilizing fineweb-edu, unlike with fineweb and C4.

A.6 Throughput as a Function of Batch Size

Figure 4 reports generated tokens per second for Llamba-8B and two Llama-3.1-8B settings. At small batches the curves are similar, yet once the batch exceeds 256, Llamba maintains linear scaling while the Transformer either stalls or runs out of memory. This mirrors the theoretical $O(n)$ recurrent complexity versus $O(n^2)$ attention cost and motivates Llamba for large-scale inference services.

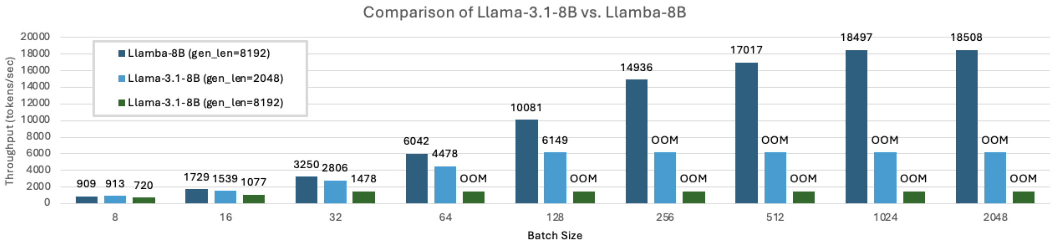


Figure 4: Tokens processed at different batch sizes across various models. All models were compiled using `torch.compile(model, fullgraph=True)` with CUDA graph compilation. We evaluated three settings: (1) Llamba-8B with `gen_len=8192`, (2) Llama-3.1-8B with `gen_len=2048`, and (3) Llama-3.1-8B with `gen_len=8192`. Each was tested with `prompt_len=1` and batch sizes ranging from 8 to 2048. The results show that Llamba-8B achieves the highest throughput, particularly at larger batch sizes, where Transformers either slow down or run out of memory (OOM).

A.7 On-Device Inference Efficiency

Figure 5 measures 4-bit quantized decoding on an M3 Pro laptop GPU. Llamba's memory footprint stays flat across context lengths, letting it hold steady throughput even at 8k tokens. Llama-3.1-8B, by contrast, sees both memory use and latency rise linearly, limiting mobile deployments. The result underscores Llamba's suitability for edge devices where RAM is scarce.

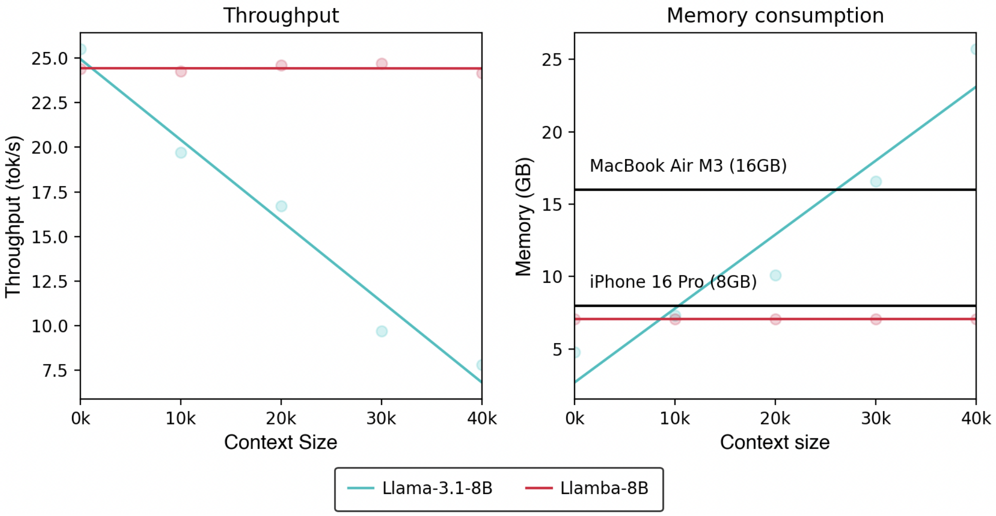


Figure 5: Comparison of on-device decoding throughput and memory consumption between Llamba-8B and Llama-3.1-8B at 4 bit quantization in MLX running on Apple Silicon M3 Pro (36GB). Llamba maintains constant high throughput and low memory consumption while the inference performance of Llama drops linearly with increasing context size.