LLAMBA: SCALING DISTILLED RECURRENT MODELS FOR EFFICIENT LANGUAGE PROCESSING

Anonymous authors
Paper under double-blind review

ABSTRACT

We present the Llamba model series, a family of highly efficient recurrent language models distilled from the Llama-3.x family into the Mamba architecture. The series includes Llamba-1B, Llamba-4B, and Llamba-8B, delivering fast inference throughput while maintaining competitive benchmark performance. Beyond its computational advantages, Llamba showcases the effectiveness of the MOHAWK (Bick et al., 2024) distillation framework, achieving high-quality performance while being distilled with less than 0.1% of the data typically used for models of similar size. We also provide an optimized implementation of the Llamba models for deployment on resource-constrained devices, such as smartphones and edge platforms, providing a practical and memory-efficient alternative to traditional Transformer architectures. Overall, these models set new standards for speed, memory efficiency, and accessibility of language models.

1 INTRODUCTION

This technical paper introduces the **Llamba model family**, a suite of advanced language models that includes Llamba-1B, Llamba-3B, and Llamba-8B. These models are designed for high inference throughput, further emphasizing the growing efficiency gap between subquadratic architectures and Transformers. Building on the Mamba-2 layer (Dao & Gu, 2024), Llamba models retain the overall Llama architecture, combining recurrent layers with multilayer perceptrons (MLPs). This design enables them to achieve state-of-the-art performance across various benchmarks while maintaining computational efficiency.

Importantly, Llamba achieves its strong 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 the **MOHAWK distillation framework** (Bick et al., 2024) to transfer the knowledge from strong pretrained Transformer-based LLMs (in particular, Llama-3) 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 distillation methods, and in particular, MOHAWK, in transferring knowledge from larger models while significantly reducing both data and computational demands. As a result, Llamba presents a scalable and cost-effective solution for high-quality language modeling.

Extending the benefits of their efficient design, **we provide on-device implementations of the Llamba models**, 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.

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.

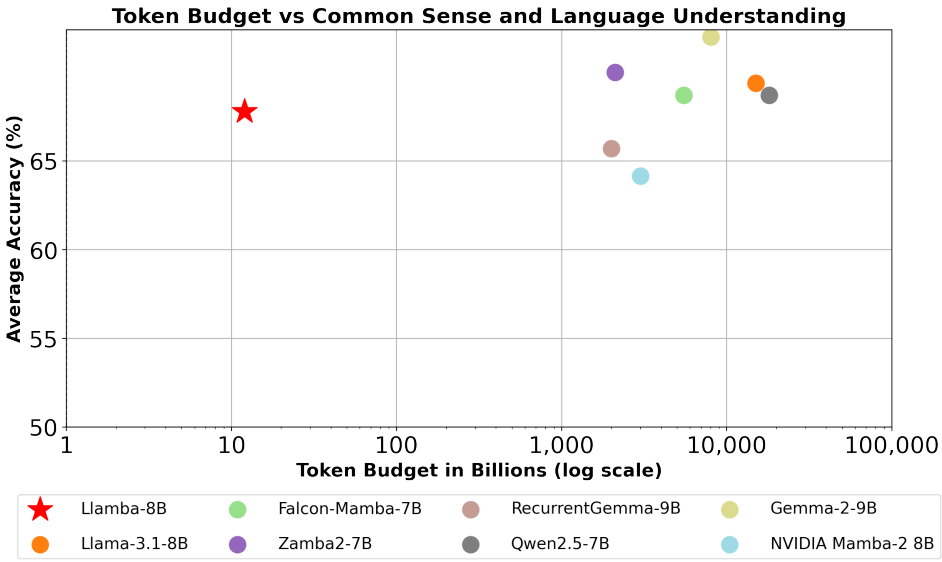

Figure 1: Average accuracy is measured over multiple benchmarks, including ARC Challenge, ARC Easy, PIQA, Winogrande, HELLASWAG, OpenBookQA, and MMLU, providing a comprehensive assessment of a model's Common Sense and Language Understanding.

- **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 Model Architecture

Unlike the Mamba and Mamba2 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 3). 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-4B, 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-4B, and 4096 for Llamba-8B. In addition, Llamba differs from the original Mamba-2 architecture (Dao & Gu, 2024) in the following ways (see Figure 7):

- **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), 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 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 3.1).
- **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 5.2). Furthermore, training becomes simpler and more efficient by eliminating normalization-related all-reduce operations.

3 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.

3.1 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.

3.2 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 3, 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).

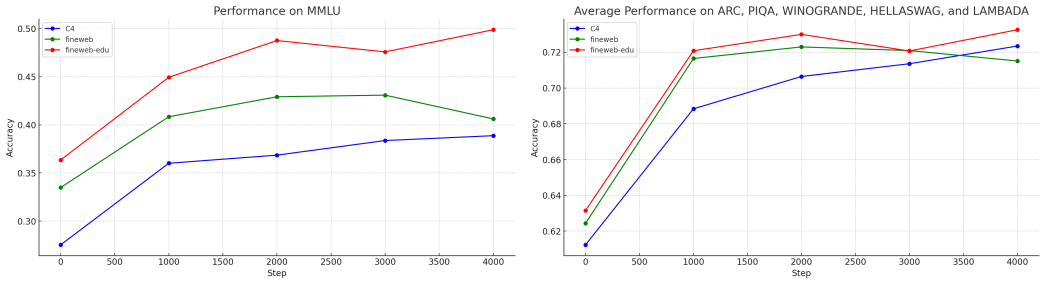

Figure 2: An evaluation of Llamba-8B's knowledge distillation step (MOHAWK's stage 3) across three datasets: C4, fineweb, and fineweb-edu. Each model underwent hidden-state alignment (MOHAWK's stage 2) on its respective dataset using 4 billion tokens and subsequently underwent testing with knowledge 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.

3.3 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 3 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 2, 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.

4 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.

5 EVALUATIONS

5.1 PERFORMANCE

Table 1 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., 2024)) 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).

5.2 THROUGHPUT

Llamba achieves up to 10× higher throughput than other recurrent models (see Figure 3). This performance gain is attributed to simplified architecture design with fewer temporal mixing layers than in Zuo et al. (2024); Botev et al. (2024), each of which incurs high kernel launch overhead. While CUDA Graph recording can reduce this gap, it imposes constraints such as static execution graphs and manual graph capturing. In contrast, *Llamba maintains high throughput without these restrictions*, making it a more practical choice for dynamic execution patterns and varying input sizes.

We have evaluated the throughput of different models (see Figure 3) 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)` for consistent performance baselines. In addition to Llamba's standard compiled version, we also provide a CUDA-graph recorded variant (see Figure 4), which will be released with the model. The primary distinction between the recorded and unrecorded CUDA-graph versions lies in the overhead of kernel launches, which is particularly impactful in generation scenarios.

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.

6 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

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

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need, 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.

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.

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.

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.

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.

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.

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 Comparison of Downstream Performance

We evaluate a range of language models across multiple benchmarks, measuring their performance in both zero-shot and few-shot settings. ARC-Challenge and ARC-Easy (Clark et al., 2018), and PIQA (Bisk et al., 2019) were evaluated with 0 and 25 shots, Winogrande (Sakaguchi et al., 2019) with 0 and 5 shots, HellaSwag (Zellers et al., 2019) and OpenBookQA Mihaylov et al. (2018) with 0 and 10 shots, and Lambada (Paperno et al., 2016) and MMLU (Hendrycks et al., 2021) with 0 and 5 shots. The results, reported in Tables 1 and 2, use normalized logits for ARC-Challenge, ARC-Easy, PIQA, HellaSwag, and OpenBookQA. These benchmarks cover a range of reasoning, commonsense, and language comprehension tasks, providing insight into the models' ability to process different types of contextual dependencies.

The evaluation includes both quadratic and sub-quadratic architectures. Specifically, we evaluate the Transformer-based Llama-3.2-1B, Llama-3.2-3B, and Llama-3.1-8B (Touvron et al., 2023), Phi1.5-1.4B and Phi3-3.8B (Gunasekar et al., 2023), as well as Qwen2.5-3B (Yang et al., 2024). Falcon-Mamba-7B and Falcon3-Mamba-7B (Zuo et al., 2024), RecurrentGemma-2B and RecurrentGemma-9B (Botev et al., 2024), and Mamba2-2.8B (Dao & Gu, 2024) are included as models integrating recurrent components. Additionally, we evaluate Zamba2-7B (Glorioso et al., 2024), a hybrid that incorporates both attention and SSMs. To provide a more comprehensive comparison, we also assess Llamba-1B, Llamba-3B, and Llamba-8B, which apply recurrent architectures across different scales. These models vary in their training strategies, data sources, and architectural choices, allowing for a broad comparison of language model capabilities across different methodological approaches.

Table 1: Comparison of downstream performance (accuracy %) across various models on in zero-shot and few-shot settings. For ARC-Challenge, ARC-Easy and PIQA we have used normalized logits' results.

Model	ARC Challenge		ARC Easy		PIQA		Winogrande	
	0-shot	25-shots	0-shot	25-shots	0-shot	10-shots	0-shot	5-shots
1–2B Parameters								
Llama-3.2-1B	38.1	42.0	68.5	71.8	74.4	75.4	59.7	62.0
Llamba-1B	37.2	41.8	69.5	71.2	74.0	74.3	60.6	58.1
Phi1.5-1.4B	48.0	53.8	73.3	80.3	76.0	77.7	73.3	72.4
RecurrentGemma-2B	35.6	48.0	51.2	73.3	67.2	75.8	55.7	64.1
Mamba2-2.8B	35.9	39.5	64.3	71.8	75.6	76.4	63.4	64.6
3–6B Parameters								
Llama-3.2-3B	45.6	52.1	74.3	79.8	75.8	77.7	67.6	68.8
Llamba-3B	48.5	53.0	79.0	81.1	78.6	79.5	70.4	72.4
Qwen2.5-3B	48.1	60.8	72.9	85.1	78.3	79.8	69.8	71.3
Phi3-3.8B	59.9	63.1	82.9	86.2	79.9	81.7	72.7	74.4
7B+ Parameters								
Zamba2-7B	56.1	68.3	80.6	88.7	81.1	81.3	76.9	80.1
Falcon3-Mamba-7B	53.2	65.9	72.5	86.7	79.7	82.3	69.1	72.1
Falcon-Mamba-7B	57.6	62.8	82.5	86.1	82.3	83.4	72.6	76.1
Llama-3.1-8B	55.1	60.0	81.7	85.8	81.1	82.4	73.9	77.3
Llamba-8B	54.6	60.0	82.5	85.8	80.9	81.5	73.3	76.9
RecurrentGemma-9B	57.1	60.2	78.9	84.5	80.6	81.7	73.7	75.6

Table 2: 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.

MODEL	HELLASWAG		LAMBADA		MMLU		OPENBOOKQA	
	0-shot	10-shots	0-shot	10-shots	0-shot	5-shots	0-shot	10-shots
1–2B Parameters								
LLAMA-3.2-1B	60.8	59.4	60.1	53.1	46.0	45.5	34.6	37.6
LLAMBA-1B	61.2	60.2	48.4	39.0	38.0	31.3	37.0	38.0
PHI1.5-1.4B	62.5	64.1	52.6	69.6	55.1	56.5	47.6	47.6
RECURRENTGEMMA-2B	60.3	69.4	52.5	53.0	40.2	42.1	30.4	43.2
MAMBA2-2.8B	66.2	66.6	68.1	61.2	25.7	25.1	40.4	42.0
3–6B Parameters								
LLAMA-3.2-3B	70.4	73.2	65.9	61.9	60.4	59.8	35.8	39.6
LLAMBA-3B	73.8	74.3	65.8	60.0	52.7	50.3	42.8	42.8
QWEN2.5-3B	74.9	75.2	65.8	58.1	65.5	66.4	41.8	46.2
PHI3-3.8B	77.7	79.5	62.0	62.0	69.0	69.3	46.0	53.2
7B+ Parameters								
ZAMBA2-7B	81.5	83.5	74.6	68.6	64.7	67.2	45.2	52.4
FALCON3-MAMBA-7B	79.8	81.6	67.5	63.6	65.0	66.0	48.0	50.2
FALCON-MAMBA-7B	78.2	79.1	69.2	65.0	60.5	60.5	47.0	47.2
LLAMA-3.1-8B	79.3	80.0	73.0	65.6	68.0	68.4	43.0	48.2
LLAMBA-8B	77.6	78.7	69.4	65.0	61.0	60.0	43.4	45.8
RECURRENTGEMMA-9B	80.1	80.9	54.1	69.6	55.1	56.5	46.0	49.2

B SUPPLEMENTARY FIGURES

Figure 3 presents the throughput of different models as a function of batch size, illustrating how token processing scales with increasing batch sizes. The experiment uses `torch.compile(model, fullgraph=True)` for compilation. The batch size varies between 1 and 128, with an initial prompt length of 128 and a generated sequence of 2048 tokens. The results highlight that Llamba-8B achieves the highest throughput, particularly at larger batch sizes, demonstrating efficient scaling behavior.

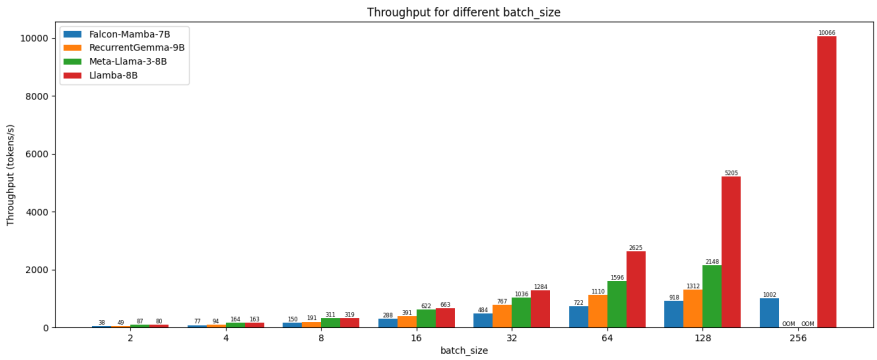

Figure 3: Tokens processed for various batch sizes across different models. All models were compiled with `torch.compile(model, fullgraph=True)`. Batch sizes vary between 1 and 128 with initial prompt of length 128 and a generated sequence of 2048 tokens. The chart demonstrates the scaling behavior of each model, showing that Llamba-8B achieves the highest throughput, particularly at larger batch sizes. This speedup comes from a streamlined architecture with fewer temporal mixing layers, reducing kernel launch overhead. While CUDA Graph recording helps bridging this gap, it requires static execution graphs and manual capturing. Llamba sustains high throughput without these constraints, making it more practical for dynamic execution and variable input sizes.

Figure 4 compares the throughput of Llamba-8B with and without CUDA-graph compilation on an NVIDIA H100 80GB HBM3. The CUDA-graph compiled version exhibits improved efficiency by reducing kernel launch overhead, leading to a significant performance boost in generation-heavy scenarios.

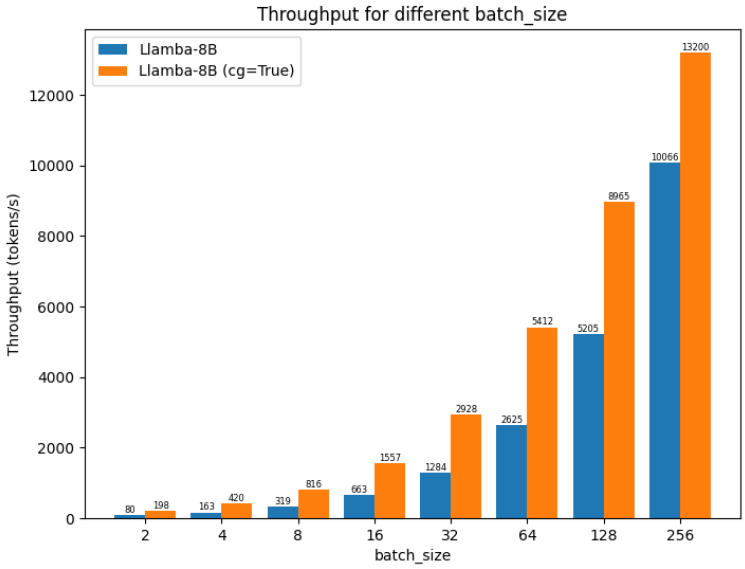

Figure 4: Throughput comparison of Llamba-8B with and without CUDA-graph compilation on a single NVIDIA H100 80GB HBM3. The CUDA-graph compiled version shows reduced kernel launch overhead, improving performance in generation scenarios.

Figure 5 evaluates the on-device decoding throughput and memory consumption of Llamba-8B compared to Llama-3.1-8B, both quantized to 4-bit, running in MLX on an Apple Silicon M3 Pro with 36GB of memory. The results show that Llamba maintains consistently high throughput and low memory usage, whereas Llama-3.1-8B's inference performance degrades linearly as context size increases.

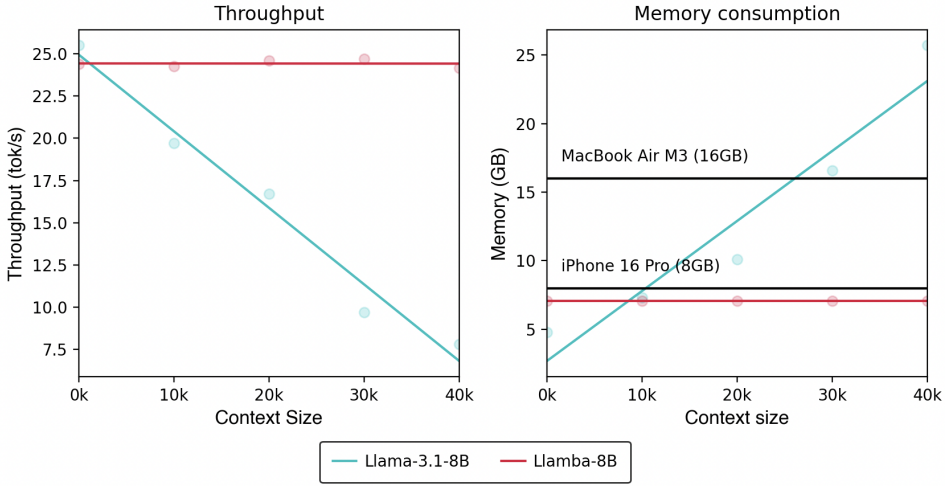

Figure 5: Comparison of on-device decoding throughput and memory consumption between Llamba-8B and Llama-3.1-8B at 4 bit quantization in MLX running on Apple Silicon M3 Pro (36GB). Llamba maintains constant high throughput and low memory consumption while the inference performance of Llama drops linearly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

Figure 6 provides an architectural overview of the Discrete Mamba-2 block, which modifies the original Mamba-2 design by removing post-convolution activation and pre-output projection normalization. Furthermore, the sequence mixer in Discrete Mamba-2 eliminates the Δ discretization parameter, instead directly projecting the $\mathbf{A}$ matrix from the input. In addition, figure 7 illustrates the architecture of Llamba models (Llamba-1B, Llamba-4B, and Llamba-8B), which are derived from their Llama teacher models. Each transformer block consists of two sub-blocks with residual connections: (1) an RMS Normalization layer followed by a Discrete Mamba-2 layer and (2) an RMS Normalization layer followed by a feed-forward layer.

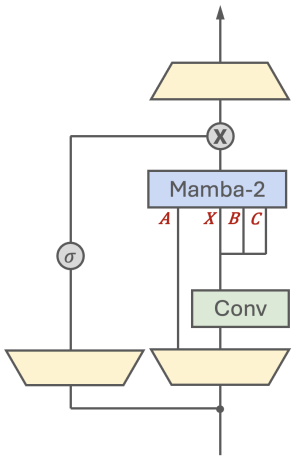

Figure 6: The Discrete Mamba-2 block Bick et al. (2024) modifies the original Mamba-2 architecture by removing both post-convolution activation and pre-output projection normalization. Additionally, the Discrete Mamba-2 sequence mixer eliminates the Δ discretization parameter and directly projects the $\mathbf{A}$ matrix from the input.

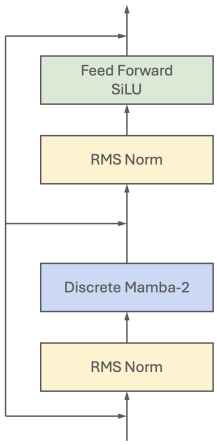

Figure 7: Llamba models—Llamba-1B, Llamba-4B, and Llamba-8B—are based on the architecture of their Llama teacher models. Each block comprises two sub-blocks with residual connections: (1) RMS Normalization followed by a Discrete Mamba-2 layer. (2) RMS Normalization followed by a feed-forward layer.

Table 3 This table outlines the token allocation for different Llamba models across three distinct distillation stages: Matrix Orientation, Hidden-State Alignment, and Knowledge Distillation (MOHAWK). The token budgets illustrate the progressive increase in token usage, reflecting the scaling of model capacity during training.

	Stage 1	Stage 2	Stage 3	Overall
Llamba-1B	300M Tokens	2.7B Tokens	5B Tokens	8B Tokens
Llamba-3B	500M Tokens	4B Tokens	5.5B Tokens	10B Tokens
Llamba-8B	500M Tokens	5B Tokens	6.5B Tokens	12B Tokens

Table 3: Token allocations during the distillation process for different Llamba models and MOHAWK stages (Matrix Orientation, Hidden-State Alignment, and Knowledge distillation).