The Mamba in the Llama: Distilling and Accelerating Hybrid Models

Anonymous Authors[1]

Abstract

Recent research suggests that state-space models (SSMs) like Mamba can be competitive with Transformer models for language modeling with advantageous deployment characteristics. Given the focus and expertise on training large-scale Transformer models, we consider the challenge of converting these pretrained models into SSMs for deployment. We demonstrate that it is feasible to distill large Transformers into SSMs by reusing the linear projection weights from attention layers with academic GPU resources. The resulting hybrid model, which incorporates a quarter of the attention layers, achieves performance comparable to the original Transformer. Moreover, we introduce a hardware-aware speculative decoding algorithm that accelerates the inference speed of state-space models. Overall we show how, with limited computation resources, we can distill a large Transformer into a hybrid SSM and decode it efficiently.

1. Introduction

While Transformers (Vaswani et al., 2017) have been an essential architecture in deep learning and have driven the success of large language models such as GPT (Brown et al., 2020), Llama (Touvron et al., 2023), and Mistral (Jiang et al., 2023), they are prohibitively slow for very long sequences due to their quadratic complexity with respect to sequence length and large Key-Value cache requirement. Recent linear RNN models (Mamba (Gu & Dao, 2023), GLA (Yang et al., 2023b), RetNet (Sun et al., 2023), Griffin (De et al., 2024)) have been shown to beat Transformers at small to medium scale. While linear RNN models (Mamba (Gu & Dao, 2023)) show fast inference (5× higher throughput) than Transformers, larger Transformers still significantly outperform linear RNN models on downstream tasks. On the other hand, the training times of these linear RNN models are similar to those of Transformers, and scaling up these models requires substantial computational resources.

The dominance of Transformers for large language model training motivates us to investigate whether a large Transformer model can be distilled into a primarily state space model (SSM) using affordable resources. This SSM model can then be used for efficient inference without requiring training from scratch. The inference benefits of SSMs can unlock new applications currently bottlenecked by the large KV cache of Transformers, such as reasoning over multiple long documents (Guo et al., 2021; Shaham et al., 2022; Peng et al., 2023) and files in large codebases (Roziere et al., 2023; Li et al., 2023a)). Emerging workflows with agents (Yao et al., 2022; Yang et al., 2024) require large-batch inference to explore more trajectories and long-context to model complex environments. The challenge is that training large SSMs from scratch still requires expensive compute, heavy training infrastructure, and lots of data. To distill a pretrained Transformer to an SSM, one would need to make good use of the pretrained weights to initialize the SSMs, as random initialization would require extensive re-training. The technical challenges are two-fold: how to map pretrained Transformer weights to SSMs weights for the best initialization, and how to adapt Transformer inference techniques such as speculative decoding to SSMs where there is no longer any KV cache.

We summarize our contributions in the following:

- We show that by reusing weights from attention layers, it is possible to distill a large transformer into a large hybrid-SSM by using 8 A100 80 GB GPUs within three days while preserving much of its generation quality. To mimic Transformer better, we propose a modified Mamba architecture that can be directly initialized from the attention block of a

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Under review by the Workshop on Efficient Systems for Foundation Models (ES-FoMO) at ICML 2024. Do not distribute.

pretrained model.

- We propose a multistage distillation approach that combines progressive distillation, supervised fine-tuning (Kim & Rush, 2016), and directed preference optimization (Rafailov et al., 2024), which shows better perplexity and downstream evaluation compared with vanilla distillation.
- We develop a targeted speculative sampling algorithm and kernel, and show that speculative decoding can be effectively applied to this hybrid architecture.

Our experiments distill a large-scale open chat LLM, Zephyr-7B (Tunstall et al., 2023) to a hybrid Mamba model, using only 3B tokens of training. Results show that the distilled approach matches the teacher model in standard Chat benchmarks (Zheng et al., 2023; Li et al., 2023b). We also show that it performs on par or better than Mamba 7B models (Mercat et al., 2024; Gu & Dao, 2023) trained from scratch with 1.2T tokens in multiple tasks (e.g., MMLU (Hendrycks et al., 2021), TruthfulQA (Lin et al., 2022)) in LLM evaluation benchmark (Gao et al., 2023).

2. Transferring Transformers to State-Space Models

2.1. Attention and Linear RNNs

We begin by reviewing multihead attention to clarify the shapes of intermediate objects. Notationally, we use explicit subscripts for the sequence position instead of matrices, to better highlight similarities between the two models.

Attention is computed in parallel for multiple differently parameterized heads $h \in \{1 \dots H\}$. Each head takes sequence $\boldsymbol{o}$ with hidden size D as an argument and computes,

$$\mathbf{Q}_t = \mathbf{W}^Q \boldsymbol{o}_t, \quad \mathbf{K}_t = \mathbf{W}_t^K \boldsymbol{o}_t, \ \mathbf{V}_t = \mathbf{W}^V \boldsymbol{o}_t \ \text{ for all } t,$$

$$\alpha_1 \dots \alpha_T = \text{softmax}\left(\frac{[m_{1,t}\mathbf{Q}_t^\top \mathbf{K}_1 \dots m_{T,t}\mathbf{Q}_t^\top \mathbf{K}_T]}{\sqrt{D}}\right) \quad \boldsymbol{y}_t = \sum_s m_{s,t}\alpha_s \mathbf{V}_s$$

$$\text{where } \boldsymbol{o}_t \in \mathbb{R}^{D\times 1}, \ \mathbf{W} \in \mathbb{R}^{N\times D} \ \mathbf{Q}_t, \mathbf{K}_t, \mathbf{V}_t \in \mathbb{R}^{N\times 1} \ m_{s,t} = \mathbf{1}(s \le t)$$

Recent work has argued that linear RNNs can be serious competitors to attention in large language models. Several different linear RNN formulations have been proposed with similar formulations. In this work, we focus on a system with selective parameters from Gu & Dao (2023) of the following form, described again for a single head $h \in \{1 \dots H\}$:

$$\boldsymbol{h}_t = \mathbf{A}_t \boldsymbol{h}_{t-1} + \mathbf{B}_t \boldsymbol{x}_t, \qquad \boldsymbol{y}_t = \mathbf{C}_t \boldsymbol{h}_t \tag{1}$$

For now, we leave the shapes of the parameters $\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t$ abstract. Linear RNNs have several computational advantages over attention. During training, all $\boldsymbol{y}_t$ values can be computed more efficiently than attention using an associative scan algorithm. During inference, each next $\boldsymbol{y}_t$ can be computed serially without requiring a cache.

Despite the superficially different form, there is a natural relationship between linear RNNs and attention. Linearizing the attention formula by removing the softmax yields:

$$\boldsymbol{y}_t = \sum_s m_{s,t}\alpha_s \mathbf{V}_s = \frac{1}{\sqrt{D}}\mathbf{Q}_t \sum_s (m_{s,t}\mathbf{K}_s \mathbf{V}_s) = \frac{1}{\sqrt{D}}\mathbf{Q}_t \sum_s m_{s,t}\mathbf{K}_s \mathbf{W}^V \boldsymbol{o}_s.$$

This implies that there exists a linear RNN form of linear attention, specifically:

$$\boldsymbol{h}_t = \boldsymbol{h}_{t-1} + \mathbf{K}_t \mathbf{W}^V \boldsymbol{o}_t \quad \boldsymbol{y}_t = \frac{1}{\sqrt{D}}\mathbf{Q}_t \boldsymbol{h}_t$$

If the two models have the same heads H and the head size N, we can set $\mathbf{B}_t = \mathbf{W}^K \boldsymbol{x}_t$, $\mathbf{C}_t = \mathbf{W}^Q \boldsymbol{x}_t$, $\boldsymbol{x}_t = \mathbf{W}^V \boldsymbol{x}_t$. This relationship motivates moving between attention and linear RNN representations.

2.2. Mamba

Unfortunately linearizing attention leads to a degraded representation of the original model, as the softmax nonlinearity is critical. Previous work has developed kernel methods to improve this approximation (Schlag et al., 2021; Irie et al., 2021; Zhang et al., 2024). These approaches increase the size of the hidden state representation to $\boldsymbol{h}$ to better match the modeling capacity of softmax.

Algorithm 1 Transformer to Mamba

Shapes: B - Batch, L - Length, D - Hiddens,
H - Heads, N - D / H
Input: $\boldsymbol{o}$: (B, L, D)
Output: output: (B, L, D) = 0
for each head $\mathbf{W}^k, \mathbf{W}^q, \mathbf{W}^v, \mathbf{W}^o : (N, D)$
expanding grouped KVs **do**
Head Parameter: $\mathbf{A} : (N)$
$\boldsymbol{x} : (B, L, N) \leftarrow \mathbf{W}^V \boldsymbol{o}$
$\mathbf{B} : (B, L, N) \leftarrow \mathbf{W}^K \boldsymbol{o}$
$\mathbf{C} : (B, L, N) \leftarrow \mathbf{W}^Q \boldsymbol{o}$
$\Delta : (B, L, N) \leftarrow \text{MLP}(\boldsymbol{x})$
$\mathbf{A}, \mathbf{B} : (B, L, N, N) \leftarrow \text{S6}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \Delta)$
$\boldsymbol{y} \leftarrow \textsc{Scan}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \boldsymbol{x})$
output $\leftarrow$ output $+ \mathbf{W}^{O\top} \boldsymbol{y}$
return output

In this work, we use the parameterization from Mamba Gu & Dao (2023) to increase the hidden state size, while initializing from the attention representation. Mamba uses a continuous time state-space model (SSM) to parameterize a linear RNN at run time, described by the

differential equation: $\boldsymbol{h}'(t) = \mathbf{A}\boldsymbol{h}(t) + \mathbf{B}(t)\boldsymbol{x}(t), \boldsymbol{y}(t) = \mathbf{C}(t)\boldsymbol{h}(t)$ where $\mathbf{A}$ is a diagonal matrix. We overload the continuous time and linear RNN names for simplicity. To apply it to a discrete-time problem like language modeling, we need to produce a sequence of sampling intervals Δ_t that map our samples to discrete time. Given these sampling intervals, and samples of $\mathbf{A}, \mathbf{B}, \mathbf{C}, \boldsymbol{c}$, Mamba uses a zero-order hold to map from the continuous time representation onto a linear RNN, $\mathbf{A}_{1\ldots T}, \mathbf{B}_{1\ldots T}, \mathbf{C}_{1\ldots T} = S6(\mathbf{A}, \mathbf{B}, \mathbf{C}, \Delta)$ For our purposes, we can treat this as a black box for producing a more expressive hidden state representation that can a minimal reproduce linear attention. We note that in addition to mapping from continuous-time to discrete-time, Mamba also uses this step to increase the hidden size of the representation through a hardware-aware factorization. For each head and element in the batch, S6 takes in $\mathbf{B}_t, \mathbf{C}_t \in \mathbb{R}^{N\times 1}$ and $\Delta_t \in \mathbb{R}^{N'}$, but outputs $\mathbf{B}_t, \mathbf{C}_t \in \mathbb{R}^{N'\times N\times 1}$. This allows the model to use different sampling intervals for the initial hidden state and effectively increases the hidden size by a factor of D over the naive linear attention.

3. Speculative Decoding for Mamba

Language model generation is inherently bottlenecked by the serial dependency of autoregressive models. Systems cannot utilize all available compute, as they need to wait for the generation of previous tokens to proceed (Spector & Re, 2023; Leviathan et al., 2023; Chen et al., 2023a; Xia et al., 2023; Cai et al., 2024). Speculative decoding has emerged as a method for breaking this bottleneck by spending extra compute to speculate on future generations. The method uses two models, a draft model, θ_D, and a verification model, θ_V. The fast draft model produces potential future completions $\arg\max_{y_1\ldots y_N} p(y_1, \ldots, y_N; \theta_D)$ and the larger verification model checks that these are top outputs at each time step for θ_V. The longer a chain before a verification failure the faster the output.

Transformer models are particularly amenable to speculation, as they are slow at generation due to sequential attention, but fast at verification due to their ability to check multiple tokens in parallel. Linear RNN models like Mamba have significantly different performance characteristics that make them less amenable to speculative decoding. Their sequential decoding mode using recurrent style sampling is significantly faster than Transformers. Their parallel mode, used at training, is more efficient than Transformers but is optimized for extremely long sequences. In addition, the optimization for efficient parallel scans explicitly avoids instantiating the intermediate state representation. These properties make it difficult to use for speculation as chains are relatively short, and it is unknown when a conflict will occur.

Algorithm 2 MultiStep Speculative Decoding

function VERIFY($\boldsymbol{x}_{1:k}, j, \boldsymbol{h}_i$) ▷ $\boldsymbol{x}_{1:k}$ are draft, j is last verified, $\boldsymbol{h}_i$ is a cached state with $i \le j$
  $\boldsymbol{y}_{j:k}, \boldsymbol{h}_j, \boldsymbol{h}_k \leftarrow$ MULTISTEP($\boldsymbol{h}_i, \boldsymbol{x}_{1:k}, i, j, k$)
  $k' \leftarrow$ FIRSTCONFLICT($\boldsymbol{y}_{j:k}, \boldsymbol{x}_{j:k}$)
  return $k', \boldsymbol{h}_k$ if $k' = k$ else $\boldsymbol{h}_j$
function SPECULATE(K) ▷ draft K tokens per step
  cache $\leftarrow \boldsymbol{h}_0$
  $j \leftarrow 0$
  while $\boldsymbol{x}_j$ is not end **do**
    k $\leftarrow$ j + K
    $\boldsymbol{x}_{1:k} \leftarrow$ DRAFT($\boldsymbol{x}_{1:j}, K$)
    j, cache $\leftarrow$ VERIFY($\boldsymbol{x}_{1:k}, j$, cache)

We therefore design a new algorithm for Mamba speculative decoding using hardware-aware multi-step generation. The algorithm is based on a new generation kernel for mamba that computes: $\boldsymbol{y}_{j:k}, \boldsymbol{h}_j, \boldsymbol{h}_k \leftarrow$ MULTISTEP($\boldsymbol{h}_i, \boldsymbol{x}_{1:n}, i, j, k; \mathbf{A}, \mathbf{B}, \mathbf{C}, \Delta$) Where i is the starting hidden state, $i \le j \le k$, and $j \ldots k$ is the range of $\boldsymbol{y}$ outputs needed. We say the function is hardware-aware because it is designed to avoid materializing key terms off of the fast GPU memory. Specifically it avoids instantiating most $\boldsymbol{h}_{1:n}$ as well as the discrete-time linear RNN parameters.

Utilizing this function we can run Algorithm 2 for verification. The algorithm maintains only one linear RNN hidden state for verification and advances it lazily based on the success of the multistep kernel.

Additionally, since our distilled models contain transformer layers, we extend speculative decoding to Transformer/Mamba hybrid architectures. In this setting, the Mamba layers perform verification according to Algorithm 2, while the transformer layers simply perform parallel verification.

4. Experimental Setup

Target models We perform all experiments using the chat model Zephyr-7B (Tunstall et al., 2023) as our target model, which is a fine-tuned Mistral 7B (Jiang et al., 2023) model. We use this model because of its strong performance in both chat and academic benchmarks, and because the supervised fine-tuning and preference alignment protocols used are well documented [1]

[1] https://github.com/huggingface/alignment-handbook

and models are highly reproducible. For the distilled model we use a hybrid one with 50% Attention layers and one with 25% Attention layers.

Training We reiterate that distillation does not require any language modeling pretraining data, but instead uses the post-training process to adapt the new model. We use a three-stage process. In the first stage, we use UltraChat (Ding et al., 2023) and UltraFeedback (Cui et al., 2023) as seed prompts and use the teacher model Zephyr (Tunstall et al., 2023) to generate pseudo-labels. The student model is trained in one epoch using the loss $\mathcal{L}$ in Eq 2 with $\alpha = 1$ and $\beta = 0.1$. Models are trained using AdamW optimizer with $\beta = (0.9, 0.98)$ with a batch size 64. We used a linear learning rate warm-up (for the first 500 steps) followed by cosine annealing. In the second stage, we use supervised finetuning with our model on the UltraChat (Ding et al., 2023) and OpenHermes 2.5 (Teknium, 2023) datasets using dSFT in one epoch, with the same configuration as Zephyr (Tunstall et al., 2023). In the final stage, we do distilled alignment with our model using dDPO on the UltraFeedback (Cui et al., 2023) dataset in one epoch (1.9k steps in total) and evaluate models for every 1k steps and pick the best. We only freeze Gated MLP in the first stage, while in the second and final stage all parameters are trained. The total distillation process takes three days in 8x80G A100.

Hybrid Speculative decoding We perform speculative decoding using the distilled hybrid models. We run experiments using both Hybrid Mamba 50% and Hybrid Mamba 25% as main models. For the draft models, we train 2 and 4-layer transformer draft models on the OpenHermes2.5 dataset (Teknium, 2023), for approximately 3 full epochs, following the "shrink and fine-tune" approach from (Shleifer & Rush, 2020). Specifically, we initialize the draft layers using layers from the Zephyr-7B model (we take layers at indices $[0, 31]$ for the 2-layer model and $[0, 10, 20, 31]$ for the 4-layer model), and the embeddings and language model head also from the Zephyr-7B model (Tunstall et al., 2023). We perform loss masking on the prompt, thus only considering next token prediction loss (cross-entropy) on the chat continuations from the training set. Speculative decoding experiments are run on a single NVIDIA RTX 3090 on data from OpenHermes2.5.

Metrics We evaluate the model on chat and academic task benchmarks. For chat, we use MT-Bench (Zheng et al., 2023), a multi-turn benchmark including 160 questions in eight different knowledge areas. Each model response is rated by GPT-4 on a scale from 1 to 10 and the final score is determined by averaging the scores from the two turns, and AlpacaEval (Li et al., 2023b) v2, a single-turn benchmark in which a model needs to generate responses to 805 questions on different topics, focused on helpfulness. It evaluates the win rate against GPT-4, scored by GPT-4 Turbo. For academic benchmarks, we use a subset of tasks in the LLM Eval Benchmark (Gao et al., 2023) which is consistent with teacher model Zephyr (Tunstall et al., 2023). Note that our aim is not to replicate full general-purpose LLM ability, so we do not target matching perplexity but focus on transferring task ability.

5. Results

Distillation Our primary goal is to produce a model competitive with Zephyr on chat-based benchmarks. We evaluate our models using single-turn and multi-turn chat benchmarks. These benchmarks assess the model's ability to follow instructions and respond to challenging prompts across a wide variety of domains.

Model	Size	MT-Bench (score)	AlpacaEval (win %)
Xwin-LM v0.1	7B	6.19	-
Mistral-Instruct v0.1	7B	6.84	-
Zephyr	7B	**7.34**	$10.99_{0.96}$
Hyb Mamba (50% att)	7B	6.69	$\mathbf{12.60}_{1.01}$
Hyb Mamba (25% att)	7B	6.10	$9.32_{0.87}$
Falcon-Instruct	40B	5.17	3.3
Llama2-Chat	7B	6.26	5.0
Llama2-Chat	13B	6.65	7.7
Llama2-Chat	70B	6.86	13.9
GPT-3.5-turbo	-	7.94	14.10
GPT-4	-	**8.99**	50.00

Table 1: Chat benchmark results for open-access and proprietary models on MT-Bench and AlpacaEval. MT-Bench scores model responses using GPT-4. AlpacaEval version two measures the win-loss rate between baseline models and GPT-4 scored by GPT-4 Turbo.

Hybrid Speculative Decoding Table 2 shows results for hybrid speculative decoding with a lookahead size of 4. For both the 50% and 25% distilled models, we achieve speedups of over 1.8x compared to the non-speculative baseline. We also show that the 4-layer draft model we trained achieves a higher acceptance rate, but it adds some additional overhead due to the increased draft model size.

Draft Model	Target Model	# Gen. Tokens	Throughput (tok/s)	Speedup
2 layers	Hybrid 1/2	2.48	115	1.8x
	Hybrid 1/4	2.64	120	1.88x
4 layers	Hybrid 1/2	3	116	1.81x
	Hybrid 1/4	3	115	1.8x

Table 2: Performance metrics for different draft and target model configurations for $K = 4$ on data from OpenHermes2.5. *# Gen* is the average number of generated tokens per speculative decoding step and includes an additional token from the last Verifier logits.

References

Arora, S., Eyuboglu, S., Timalsina, A., Johnson, I., Poli, M., Zou, J., Rudra, A., and Ré, C. Zoology: Measuring and improving recall in efficient language models. *arXiv preprint arXiv:2312.04927*, 2023.

Arora, S., Eyuboglu, S., Zhang, M., Timalsina, A., Alberti, S., Zinsley, D., Zou, J., Rudra, A., and Ré, C. Simple linear attention language models balance the recall-throughput tradeoff. *arXiv preprint arXiv:2402.18668*, 2024.

Bhendawade, N., Belousova, I., Fu, Q., Mason, H., Rastegari, M., and Najibi, M. Speculative streaming: Fast llm inference without auxiliary models. *arXiv preprint arXiv:2402.11131*, 2024.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Cai, T., Li, Y., Geng, Z., Peng, H., Lee, J. D., Chen, D., and Dao, T. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L., and Jumper, J. Accelerating Large Language Model Decoding with Speculative Sampling, 2023a.

Chen, Z., Yang, X., Lin, J., Sun, C., Huang, J., and Chang, K. C.-C. Cascade speculative drafting for even faster llm inference. *arXiv preprint arXiv:2312.11462*, 2023b.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Cui, G., Yuan, L., Ding, N., Yao, G., Zhu, W., Ni, Y., Xie, G., Liu, Z., and Sun, M. Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377*, 2023.

De, S., Smith, S. L., Fernando, A., Botev, A., Cristian-Muraru, G., Gu, A., Haroun, R., Berrada, L., Chen, Y., Srinivasan, S., et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

Ding, N., Chen, Y., Xu, B., Qin, Y., Hu, S., Liu, Z., Sun, M., and Zhou, B. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3029–3051, 2023.

Fu, D., Arora, S., Grogan, J., Johnson, I., Eyuboglu, E. S., Thomas, A., Spector, B., Poli, M., Rudra, A., and Ré, C. Monarch mixer: A simple sub-quadratic gemm-based architecture. *Advances in Neural Information Processing Systems*, 36, 2024a.

Fu, D. Y., Dao, T., Saab, K. K., Thomas, A. W., Rudra, A., and Re, C. Hungry hungry hippos: Towards language modeling with state space models. In *The Eleventh International Conference on Learning Representations*, 2022.

Fu, Y., Bailis, P., Stoica, I., and Zhang, H. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*, 2024b.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

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

Gu, A., Goel, K., and Ré, C. Efficiently Modeling Long Sequences with Structured State Spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Gu, A., Goel, K., Gupta, A., and Ré, C. On the Parameterization and Initialization of Diagonal State Space Models. *Advances in Neural Information Processing Systems*, 35:35971–35983, 2022.

Guo, M., Ainslie, J., Uthus, D., Ontanon, S., Ni, J., Sung, Y.-H., and Yang, Y. Longt5: Efficient text-to-text transformer for long sequences. *arXiv preprint arXiv:2112.07916*, 2021.

Gupta, A., Gu, A., and Berant, J. Diagonal State Spaces are as Effective as Structured State Spaces. *Advances in Neural Information Processing Systems*, 35:22982–22994, 2022.

He, Z., Zhong, Z., Cai, T., Lee, J. D., and He, D. Rest: Retrieval-based speculative decoding. *arXiv preprint arXiv:2311.08252*, 2023.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Irie, K., Schlag, I., Csordás, R., and Schmidhuber, J. Going beyond linear transformers with recurrent fast weight programmers. *Advances in neural information processing systems*, 34:7703–7717, 2021.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Kim, Y. and Rush, A. M. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327, 2016.

Leviathan, Y., Kalman, M., and Matias, Y. Fast Inference from Transformers via Speculative Decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Li, R., Allal, L. B., Zi, Y., Muennighoff, N., Kocetkov, D., Mou, C., Marone, M., Akiki, C., Li, J., Chim, J., et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023a.

Li, X., Zhang, T., Dubois, Y., Taori, R., Gulrajani, I., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 2023b.

Lieber, O., Lenz, B., Bata, H., Cohen, G., Osin, J., Dalmedigos, I., Safahi, E., Meirom, S., Belinkov, Y., Shalev-Shwartz, S., et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

Lin, S., Hilton, J., and Evans, O. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, 2022.

Liu, X., Hu, L., Bailis, P., Stoica, I., Deng, Z., Cheung, A., and Zhang, H. Online speculative decoding. *arXiv preprint arXiv:2310.07177*, 2023.

Massaroli, S., Poli, M., Fu, D., Kumbong, H., Parnichkun, R., Romero, D., Timalsina, A., McIntyre, Q., Chen, B., Rudra, A., et al. Laughing hyena distillery: Extracting compact recurrences from convolutions. *Advances in Neural Information Processing Systems*, 36, 2024.

Mehta, H., Gupta, A., Cutkosky, A., and Neyshabur, B. Long Range Language Modeling via Gated State Spaces. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=5MkYIYCbva.

Mercat, J., Vasiljevic, I., Keh, S., Arora, K., Dave, A., Gaidon, A., and Kollar, T. Linearizing large language models. *arXiv preprint arXiv:2405.06640*, 2024.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Cappelli, A., Alobeidli, H., Pannier, B., Almazrouei, E., and Launay, J. The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023. URL https://arxiv.org/abs/2306.01116.

Peng, B., Quesnelle, J., Fan, H., and Shippole, E. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.

Poli, M., Massaroli, S., Nguyen, E., Fu, D. Y., Dao, T., Baccus, S., Bengio, Y., Ermon, S., and Ré, C. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pp. 28043–28078. PMLR, 2023.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Ralambomihanta, T. R., Mohammadzadeh, S., Islam, M. S. N., Jabbour, W., and Liang, L. Scavenging hyena: Distilling transformers into long convolution models. *arXiv preprint arXiv:2401.17574*, 2024.

Roziere, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Remez, T., Rapin, J., et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Schlag, I., Irie, K., and Schmidhuber, J. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pp. 9355–9366. PMLR, 2021.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shaham, U., Segal, E., Ivgi, M., Efrat, A., Yoran, O., Haviv, A., Gupta, A., Xiong, W., Geva, M., Berant, J., et al. Scrolls: Standardized comparison over long language sequences. *arXiv preprint arXiv:2201.03533*, 2022.

Shleifer, S. and Rush, A. M. Pre-trained summarization distillation. *CoRR*, abs/2010.13002, 2020. URL https://arxiv.org/abs/2010.13002.

Spector, B. and Re, C. Accelerating llm inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623*, 2023.

Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., and Wei, F. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena: A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2020.

Teknium. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants, 2023. URL https://huggingface.co/datasets/teknium/OpenHermes-2.5.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Tunstall, L., Beeching, E., Lambert, N., Rajani, N., Rasul, K., Belkada, Y., Huang, S., von Werra, L., Fourrier, C., Habib, N., et al. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, J., Yan, J. N., Gu, A., and Rush, A. M. Pretraining without attention. *arXiv preprint arXiv:2212.10544*, 2022.

Xia, H., Ge, T., Wang, P., Chen, S.-Q., Wei, F., and Sui, Z. Speculative Decoding: Exploiting Speculative Execution for Accelerating Seq2seq Generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3909–3925, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.257. URL https://aclanthology.org/2023.findings-emnlp.257.

Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K., and Press, O. Swe-agent: Agent computer interfaces enable software engineering language models, 2024.

Yang, N., Ge, T., Wang, L., Jiao, B., Jiang, D., Yang, L., Majumder, R., and Wei, F. Inference with reference: Lossless acceleration of large language models. *arXiv preprint arXiv:2304.04487*, 2023a.

Yang, S., Wang, B., Shen, Y., Panda, R., and Kim, Y. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023b.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Zhang, M., Bhatia, K., Kumbong, H., and Ré, C. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. *arXiv preprint arXiv:2402.04347*, 2024.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

A. Full Initialization and Hybrid Stepwise Training

To initialize an SSM layer from a modern attention layer we use the procedure shown in Algorithm 1 and Figure 1. This algorithm needs to handle the conversion from attention to the SSM parameterization and then from the SSM form to linear attention. In addition, it requires processing additional components like grouped query attention that shares keys and values across heads. We note that this model differs from the commonly-used Mamba architecture, which combines MLP and SSM layers into one and is single head. Our version replaces attention heads directly with SSM layers. We also keep the MLP layers as is and do not train them.

This initialization allows us to replace any attention block with a Mamba block. In practice, we experiment with hybrid models where we keep every n attention layers. Empirically we found that replacing layers in a stepwise manner was the most effective strategy, i.e. we first keep every 2 layers, distill, and then every 4, and continue distillation.

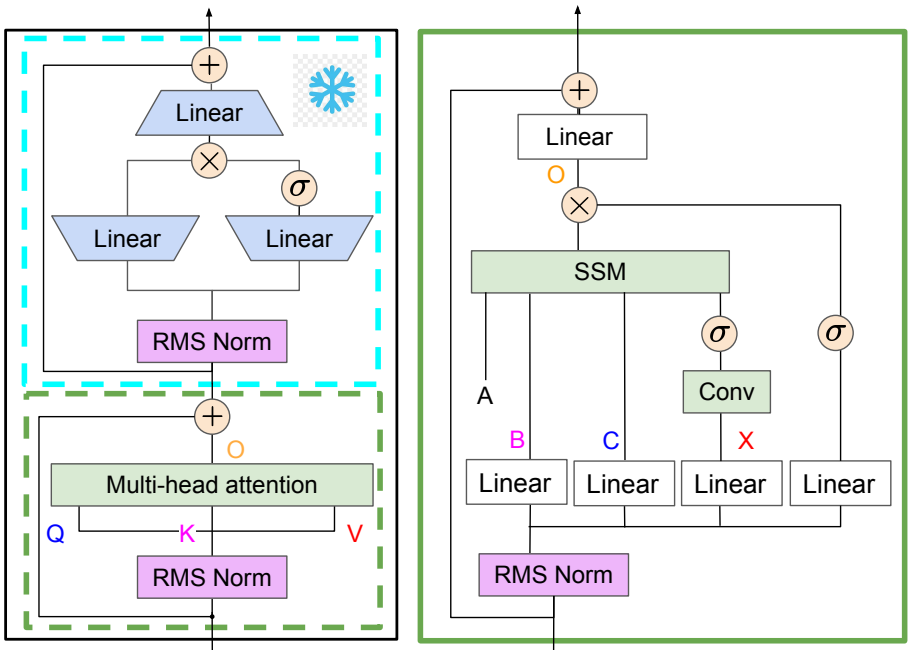


Figure 1: Transferring Transformer to Mamba. Weights in same color are initialize from transformer. We replace only individual Attention heads by SSM layers, and then fine-tune these blocks while freeze the MLP blocks. Shapes are kept mainly the same. New parameters are introduced for the learned $\mathbf{A}$ and Δ parameters.

B. Knowledge Distillation for Aligned LMs

Knowledge distillation (KD) (Hinton et al., 2015) serves as a compression technique aimed at training a smaller network that mimics the behavior of a larger teacher network. After initializing the Mamba model from the original transformer parameters, we aim to distill it to perform on par with the original language model. We assume that most of the knowledge from the transformer is maintained in the MLP layers which were transferred from the original model, and focus on distilling the fine-tuning and alignment steps of the LLM. During this stage, the MLP layers are kept frozen and the Mamba layers are trained as in Figure 1.

Supervised Fine-Tuning We first apply knowledge distillation to redo the supervised fine-tuning (SFT) stage of language model adaptation. During this stage, an LLM is trained to maximize the likelihood of a response y given an input prompt x, i.e. $p(y \mid x)$. In this sense, the task looks similar to conditional generation.

There are two common approaches for distillation in this setting. One method is to use word-level KL-Divergence. In this setting, the full probability distribution of the student model $p(\cdot;\theta)$ is trained to match the full distribution of the teacher model $p(\cdot;\theta_T)$ by minimizing the KL divergence over the entire set of next possible tokens at

position t. The second method is sequence-level knowledge distillation (SeqKD) (Kim & Rush, 2016). SeqKD suggests a simple method for distillation on this style of task, by replacing the ground truth text $y_{1\dots t}$ with the teacher generation output $\hat{y}_{1\dots t}$, also known as pseudo-labels.

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \alpha \ \log \ p(\hat{y}_{t+1} \mid \hat{y}_{1:t}, x, \theta) + \beta \ \mathrm{KL}\left[p(\cdot \mid \hat{y}_{1:t}, x, \theta_T) \mid\mid p(\cdot \mid \hat{y}_{1:t}, x, \theta)\right] \tag{2}$$

Here θ is trainable parameters of the student model and α and β control the weights of sequence and word loss term respectively.

Preference Optimization The second stage of instruction-tuning for LLMs is to align them to a set of user preferences. During this stage, a set of desired preference pairs is used to improve the model's output. The objective is to produce outputs y to prompts x that maximize a reward model r while maintaining close to a reference model. Typically the reference model is chosen to be the model after supervised fine-tuning. For distillation, we can conveniently utilize the original teacher, i.e.

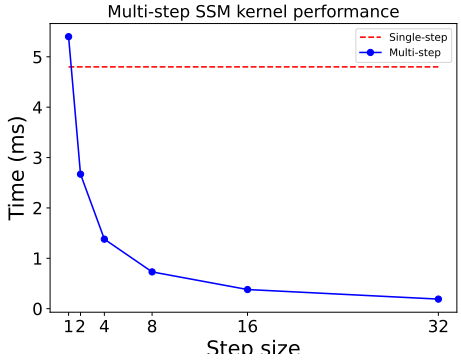
$$\max_{\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim p(y|x;\theta)}\left[r_\phi(x,y)\right] - \beta \mathrm{KL}\left[p(y \mid x;\theta) \mid\mid \pi(y \mid x;\theta_T)\right] \tag{3}$$

This preference model is defined by a reward function $r_\phi(x, y)$ dependent on the method used. Previous research utilizatin AI feedback has primarily focused on employing reinforcement learning methods, such as proximal policy optimization (PPO) (Schulman et al., 2017), to optimize ϕ concerning this reward. Recently, methods using direct preference optimization (DPO) (Rafailov et al., 2024) have been effective at optimizing this objective with direct gradient updates. Specifically, DPO shows that, if we have access to preferred y_w and dispreferred y_l outputs for a given prompt x, we can reformulate this optimization problem as,

$$\pi_\theta = \max_{\theta} \mathop{\mathbb{E}}_{(x,y_w,y_l)\sim\mathcal{D}} \log \sigma \left(\beta \log \frac{p(y_w|x;\theta)}{p(y_w|x;\theta_T)} - \beta \log \frac{p(y_l|x;\theta)}{p(y_l|x;\theta_T)}\right). \tag{4}$$

This optimization can be performed at the sequence level by scoring the preferred and dispreferred outputs of the model with the teacher and student and then backpropagating to the student. As far as we are aware this is the first use of DPO as a distillation objective.

C. Speculative Decoding for Mamba



K	# Gen. Tokens	Throughput	Speedup
3	3.01	1411	1.48x
4	3.28	1482	1.56x
5	3.65	1517	1.59x
6	4.00	1531	1.60x

Figure 2: (Left) Performance of the multi-step SSM kernel for generating 32 tokens. (Right) Speedup results for speculative decoding with pure Mamba models (2.8B verifier, 130M draft) on The Pile. K is number of draft tokens produced, # *Gen* includes an additional token from the last Verifier logits.

Results of our hardware-aware multi-step generation are shown in Figure 2(left). We verify the effectiveness of Algorithm 2 we run the speculation using a 2.8B Mamba as the target model and a 130M Mamba as the draft model (model checkpoints from (Gu & Dao, 2023)), using data from The Pile. Results are shown in Figure 2(right).

Additionally, since our distilled models contain transformer layers, we extend speculative decoding to Transformer/Mamba hybrid architectures. In this setting, the Mamba layers perform verification according to Algorithm 2, while the transformer layers simply perform parallel verification.

D. More Experiment Results

Distillation Our primary goal is to produce a model competitive with Zephyr on chat-based benchmarks. We evaluate our models using single-turn and multi-turn chat benchmarks. These benchmarks assess the model's ability to follow instructions and respond to challenging prompts across a wide variety of domains.

Model	Size	Align	MT-Bench (score)	AlpacaEval (LC win %)	AlpacaEval (win %)
Xwin-LM v0.1	7B	dPPO	6.19	-	-
Mistral-Instruct v0.1	7B	-	6.84	-	-
Zephyr	7B	dDPO	**7.34**	$13.20_{0.96}$	$10.99_{0.96}$
Hyb Mamba (50% att)	7B	dDPO	6.69	$\mathbf{14.11}_{1.01}$	$\mathbf{12.60}_{1.01}$
Hyb Mamba (25% att)	7B	dDPO	6.10	$8.92_{0.87}$	$9.32_{0.87}$
Falcon-Instruct	40B	dSFT	5.17	5.6	3.3
Llama2-Chat	7B	RLHF	6.26	5.4	5.0
Llama2-Chat	13B	RLHF	6.65	8.4	7.7
Llama2-Chat	70B	RLHF	6.86	14.7	13.9
GPT-3.5-turbo	-	RLHF	7.94	22.70	14.10
Claude 2	-	RLHF	8.06	28.20	17.20
GPT-4	-	RLHF	**8.99**	50.00	50.00
GPT-4o	-	RLHF	-	$\mathbf{57.46}_{1.47}$	$\mathbf{51.33}_{1.47}$

Table 3: Chat benchmark results for open-access and proprietary models on MT-Bench and AlpacaEval. MT-Bench scores model responses using GPT-4. AlpacaEval version two measures the win-loss rate between baseline models and GPT-4 scored by GPT-4 Turbo.

Table 3 shows the performance of our distilled Mamba model on chat benchmarks compared with large transformer models. The distilled Hyb Mamba (50% att) achieves a similar score in the MT-benchmark as the teacher model, and slightly better than the teacher model on the AlpacaEval benchmark in both LC win rate and overall win rate. The Hyb Mamba (25% att) performance is slightly worse than that of the teacher models in the MT benchmark but still surpasses some large transformers even with more parameters in AlpcaEval.

We also report evaluation on standard academic benchmarks in Table 4. We follow the evaluation of Zephyr by conducting 25 shots in ARC-Challenge (Clark et al., 2018), 10 shots in HellaSwag (Zellers et al., 2019), 5 shots in MMLU (Hendrycks et al., 2021), and zero-shot in TruthfulQA (Lin et al., 2022). For these experiments, we also compare to a pure Mamba-7B model trained from scratch (Mercat et al., 2024) and evaluate with the same number of shots. The distilled models show somewhat degraded performance on these benchmarks compared to Zephyr, but are still competitive. We note that our model, which is trained on only 3 billion tokens, significantly outperforms Mamba 7B, which is trained from scratch with 1.2 trillion tokens, on the Refined Web (Penedo et al., 2023) dataset in some tasks, like MMLU (Hendrycks et al., 2021) and Truthful QA (Lin et al., 2022).

E. Analysis

Does PPL correspond to ability? Table 5 Left compares the PPL of different model variants. We distill using Ultrachat (Ding et al., 2023) in one epoch and compare the perplexity. We find that removing more layers gets significantly worse. We also compare our distillation approach with a previous baseline. This approach distills a Transformer model into a Hyena model (Poli et al., 2023), as proposed in (Ralambomihanta et al., 2024). They use

Model	Size	Align	ARC	Hella Swag	MMLU	Truthful QA
StableLM-Tuned-α	7B	dSFT	31.91	53.59	24.41	40.37
MPT-Chat	7B	dSFT	46.50	75.51	37.62	40.16
Xwin-LM v0.1	7B	dPPO	56.57	79.40	49.98	47.89
Mistral-Instruct v0.1	7B	dSFT	54.52	75.63	55.38	56.28
Zephyr	7B	dDPO	**62.03**	**84.52**	**61.44**	**57.44**
Mamba	7B	-	$52.56_{1.46}$	$80.62_{0.39}$	$33.40_{6.67}$	$29.10_{3.13}$
Hyb Mamba (50% att)	7B	dDPO	$49.15_{1.46}$	$75.07_{0.43}$	$47.98_{10.21}$	$46.67_{5.51}$
Hyb Mamba (25% att)	7B	dDPO	$48.55_{1.46}$	$71.09_{0.45}$	$37.82_{7.31}$	$40.01_{5.36}$
Falcon-Instruct	40B	dSFT	61.60	84.31	55.45	52.52
Llama2-Chat	7B	RLHF	53.07	77.74	45.30	33.29
Llama2-Chat	13B	RLHF	59.39	82.13	54.80	41.74
Llama2-Chat	70B	RLHF	67.32	87.33	69.83	44.92

Table 4: LLM eval benchmark results for open-access models on the Open LLM Leaderboard.

Model	PPL	Ratio
Teacher: Zephyr (7B)	2.02	1
Hyb Mamba (50% att)	2.09	1.03
Hyb Mamba (25% att)	2.20	1.09
Hyb Mamba (6.25% att)	2.46	1.22
Mamba (0% att)	3.36	1.66
Teacher: Pythia (70M)	51.4	1
Distill Hyena	**121.2**	2.36

Model	Hyb Mamba (50% Att)	Hyb Mamba (25% Att)
Dis	5.55	5.01
Dis+SFT	5.61	4.97
Dis+dDPO	5.42	4.84
Dis+SFT+dDPO	**6.69**	**6.10**

Table 5: (Left) Perplexity comparison between our distillation approach and (Ralambomihanta et al., 2024). (Right) Ablation study of different alignment methods of the Distilled Hybrid Mamba on the MT-benchmark.

a different distillation approach using progressive knowledge transfer, wherein the student model is trained starting from the first layer and progressively extending to subsequent layers. While it is challenging to compare, our distill shows a smaller degradation (1.03 for 50 % attention, 1.09 for 25 % attention, 1.22 for 6.35% attention, and 3.36 for no attention), while the Distill Hyena model is trained in WikiText (Merity et al., 2016) dataset with a much smaller model and shows large perplexity degrade.

Does distilling from preferences help? In Table 5 (Right), we show the impact of different steps in the alignment process of the distillation. We observe that dSFT or dDPO alone does not yield much improvement, while dSFT + dDPO yields the best score.

Ablations We consider several different model ablation studies in Table 6. For these experiments we consider training for 5k steps using the pseudo-label approaches on the Ultrachat (Ding et al., 2023) dataset.

Table 6 (Left) presents the results of distillation with various initializations. According to this table, initializing weights from a transformer is crucial for performance. Without weight initialization from a transformer, perplexity significantly worsens for both pure Mamba models and hybrid models. Also, freezing MLP layers can help the student model focus on learning the interaction of tokens and better mimic attention layers. Table 6 (Right) shows also see smaller benefits from progressive distillation and interleaving the attention layers with Mamba.

Model	Mamba (0% Att) Froz	Mamba (0% Att) -Froz	Hyb Mamba (50% Att) Froz	Hyb Mamba (50% Att) -Froz
Atten-Init	**3.36**	66.7	**2.09**	9.1
-Atten-Init	18.2	20.3	7.4	11.2

Model	Hyb Mamba (25% Att) Step	Hyb Mamba (25% Att) -Step	Hyb Mamba (50% Att) Step	Hyb Mamba (50% Att) -Step
Interleave	**2.20**	2.29	**2.09**	-
-Interleave	2.89	-	2.41	-

Table 6: (Left) Perplexity comparison with different SSM initialization. (Right) Perplexity comparison with different Mamba interleaving layers and stepwise distillation.

F. Related Work

Attention-free models. Attention-free models offer improved computational and memory efficiency, making them increasingly popular for various language processing tasks, including autoregressive language modeling. Models like S4 (Gu et al., 2021) and its subsequent variants (Gupta et al., 2022; Gu et al., 2022) have shown promising results in long-range synthetic tasks (Tay et al., 2020). Gated SSM architectures, such as GSS (Mehta et al., 2023) and BiGS (Wang et al., 2022), incorporate a gating mechanism into SSMs for (bidirectional) language modeling. The recently introduced Mamba model (Gu & Dao, 2023) argues that the static dynamics of these methods fail to incorporate input-specific context selection within the hidden state, which could be crucial for tasks like language modeling. Mamba has been shown to outperform Transformers across different model sizes and scales. Additionally, several other sub-quadratic model architectures (Poli et al., 2023; Yang et al., 2023b; De et al., 2024; Arora et al., 2023; 2024; Fu et al., 2024a) and hybrid architectures (Fu et al., 2022; Lieber et al., 2024) have also been proposed.

Distillation from Transformers. Laughing Hyena (Massaroli et al., 2024) proposes to distill the long convolution into a state space representation, enabling constant time inference in Hyena (Poli et al., 2023). Ralambomihanta et al. (2024) introduces a progressive knowledge approach to distill small transformer models (70M) into Hyena models.

Speculative Decoding. Speculative decoding (Spector & Re, 2023; Leviathan et al., 2023; Chen et al., 2023a; Xia et al., 2023; Cai et al., 2024) has recently emerged as a promising method to accelerate the inference process of large language models, particularly Transformers. This approach utilizes a smaller draft model to speculatively generate candidate tokens, which the larger target model then verifies. Leviathan et al. (2023); Chen et al. (2023a) proposed a rejection sampling scheme to improve inference quality, while Spector & Re (2023) organized candidate tokens into a tree structure to enable more efficient verification. Subsequent work has examined both trained draft models (Bhendawade et al., 2024; Chen et al., 2023b; Liu et al., 2023) and training-free draft models (He et al., 2023; Yang et al., 2023a; Fu et al., 2024b).

G. Discussion: Limitations, Broader Impacts, Conclusion

Limitations We only train our model using chat corpora due to academic budget constraints. Training on general corpora, such as those referenced in (Penedo et al., 2023), may help close the gap between teacher models and is worth exploring further. Additionally, our model is in 7B scale. Further work still needs to be done with models that have more parameters.

Broader Impacts Our models are trained using a collected chat corpus. Recent research has uncovered potential societal biases embedded within many established corpora. While it is beyond the scope of this paper to delve deeply into these biases, we acknowledge the potential risk that our distilled trained models may inherit these biases.

Conclusion We consider the problem of maintaining LLM abilities while increasing decoding speed through a combination of distillation and speculative decoding. We first show that a transformer LLM can be used to effectively initialize a Mamba linear RNN model while maintaining original abilities. We then show that through a combination of distillation on supervised instructions and preferences, we can improve the model's ability with relatively little compute. Finally, we show that the Mamba model can be significantly sped up at inference time through the use of a hardware-aware speculative decoding method. The full model nears LLM chat accuracy,

and is accelerated with speculative decoding. We believe these results show that transformer knowledge can be transferred effectively to other architectures, opening up the potential for customizing the inference profile of LLMs beyond optimizing attention.