M1: Towards Scalable Test-Time Compute with Mamba Reasoning Models

Junxiong Wang[1], Wen-Ding Li[2], Daniele Paliotta[3*]
Daniel Ritter[2], Alexander M. Rush[2], Tri Dao[1,4]

[1]TogetherAI, [2]Cornell University, [3]University of Geneva, [4]Princeton University

Abstract

Effective reasoning is crucial to solving complex mathematical problems. Recent large language models (LLMs) have boosted performance by scaling test-time computation through long chain-of-thought reasoning. However, transformer-based models are inherently limited in extending context length due to their quadratic computational complexity and linear memory requirements. In this paper, we introduce a novel hybrid linear RNN reasoning model, **M1**, built on the Mamba architecture, which allows memory-efficient inference. Our approach leverages a distillation process from existing reasoning models and is further enhanced through RL training. Experimental results on the AIME and MATH benchmarks show that **M1** not only outperforms previous linear RNN models but also matches the performance of state-of-the-art Deepseek R1 distilled reasoning models at a similar scale. We also compare our generation speed with a highly performant general purpose inference engine, vLLM, and observe more than a 3x speedup compared to a same size transformer. With throughput speedup, we are able to achieve higher accuracy compared to DeepSeek R1 distilled transformer reasoning models under a fixed generation time budget using self-consistency voting. Overall, we introduce a hybrid Mamba reasoning model and provide a more effective approach to scaling test-time generation using self-consistency or long chain of thought reasoning. Code and pre-trained checkpoints are open-sourced at github.com/jxiw/M1.

1 Introduction

Robust and effective reasoning is the cornerstone for successfully performing tasks in domains such as mathematics and programming. Additionally, performance on reasoning tasks can often be boosted by generating longer sequences and/or generating many sequences in parallel (Snell et al., 2024). However, current transformer-based large language models (LLMs) face significant challenges when tasked with processing long sequences with large batch sizes. These models are constrained by a quadratic increase in computational complexity as the sequence length grows, coupled with a linear escalation in memory requirements. This combination makes it increasingly difficult for models to scale efficiently when handling large inputs.

Although linear hybrid RNN models (Gu & Dao, 2024; Dao & Gu, 2024; Beck et al., 2024; Yang et al., 2024; Peng et al., 2023) have shown great potential as an alternative to transformer-based on general language models, their effectiveness on reasoning tasks remains unclear. Since modern reasoning models typically generate long chains of thought for challenging math questions, it is uncertain whether the performance of hybrid linear RNNs diminishes in such scenarios.

*Work done when interned at TogetherAI

In this paper, we propose **M1** and show that it is possible to derive strong hybrid reasoning models by efficiently transferring reasoning capabilities from a large transformer model. Our training process involves distilling knowledge, incorporating math and reasoning abilities through supervised fine-tuning (SFT), and finally, boosting performance using reinforcement learning (RL) training. In total, the training process requires fewer than 50 billion tokens. In contrast, DeepSeek-R1-Distill-Qwen-1.5B is finetuned from Qwen2.5 MATH 1.5B which is trained using over 1 trillion MATH tokens on top of Qwen2.5.

We demonstrate that our hybrid models achieve a 3x speedup compared to transformers of the same size when served using a highly performant general purpose inference engine, vLLM, at large batch sizes. This gain is mainly due to large batches and long sequences, decoding being generally memory-bound. Lower memory usage of hybrid models can transform this advantage into a speed gain. The decoding speedup is approximately linear with the volume of model's memory access (Yuan et al., 2025).

Notably, this speedup can be converted to a gain in reasoning accuracy. Studies (Snell et al., 2024; Li, 2025; Chen et al., 2025) show that techniques such as self-consistency (Wang et al., 2023) and verification (Cobbe et al., 2021) at test time can significantly boost model reasoning performance. Under these conditions, a high-throughput model can further enhance its performance by generating more samples.

The paper is organized as follows. Section 2 covers related work, Section 3 introduces our pipeline for distilling a hybrid reasoning model, and Section 4.1 presents our results evaluating **M1** on math benchmarks. Sections 4.2 and 4.3 evaluate the performance gains of **M1** in terms of both inference speed and scaling test-time compute. Section 5 provides some additional analysis of the impact of different generation lengths when training on RL, and of the impact of the different steps of the distillation pipeline we propose on performance.

Overall, we show that **M1** performs on par with DeepSeek-R1-Distill-Qwen-1.5B, achieving scores of 82 on MATH500 (Hendrycks et al., 2021), 23 on AIME25 (MAA, 2025), 28 on AIME24 (MAA, 2024), and 47 on OlympiadBench (He et al., 2024), while offering 3x faster inference throughput, even compared to the highly optimized vLLM (Kwon et al., 2023) implementation for Transformer models.

2 Related Work

2.1 Reasoning models

Recent models like Deepseek-R1 (DeepSeek-AI et al., 2025) have shown the potential of RL training to improve performance on verifiable reasoning tasks, such as math problem solving and programming. Additional work has proposed methods for inducing this reasoning behavior via supervised fine-tuning, either on curated data (Muennighoff et al., 2025) or on generated pairs of traces (Yang et al., 2025). Other approaches also combine search procedures such as MCTS with language models (Qi et al., 2024) or alter standard RL training schemes to control the length of generated outputs (Aggarwal & Welleck, 2025). After training, these models solve complex tasks by generating long chains of thought, which often include subtasks of the overall problem, multiple attempted solutions, and backtracking over prior attempts (Gandhi et al., 2025). Since the performance of these models, both during training and inference, relies on generating lengthy chains of thought, more efficient architectures can enable larger scale training and less costly generation.

2.2 Enhancing Reasoning via Scaled Inference Compute

Increasing the computational budget during inference has become a promising approach to boost LLM performance. Methods like Chain of Thought (CoT) and its derivatives have achieved notable gains on reasoning benchmarks by breaking down complex tasks into intermediate steps (Wei et al., 2023; Yao et al., 2023). Although decomposing tasks improves reasoning, it also lengthens generation sequences and raises computational costs. Some recent studies even indicate that this extra computation might itself enhance model capabilities (Pfau et al., 2024). In addition, adaptive compute allocation during inference

has been explored. For example, Goyal et al. (2024) incorporated pause tokens into the vocabulary, allowing models to distribute compute more efficiently and improve both reasoning and overall task performance. LightTransfer (Zhang et al., 2024c) introduces a lightweight method that detects lazy layers and replaces their full attention with streaming attention—slashing KV-cache overhead and boosting throughput.

Another strategy involves generating several outputs and selecting the best one. Researchers have developed various sampling algorithms to diversify and enhance the quality of generated responses, thereby increasing the chances of retrieving the most accurate answer (Wang et al., 2023; Renze & Guven, 2024; Zhang et al., 2023). Moreover, outcome and process reward models (ORMs and PRMs) have been introduced to evaluate responses and steer intermediate generation steps (Lightman et al., 2023; Zhang et al., 2024a; Luo et al., 2024; Uesato et al., 2022).

Recent investigations reveal that, under fixed compute budgets, smaller LLMs augmented with inference-time compute techniques (such as majority voting or PRM-guided search) can outperform larger models (Snell et al., 2024; Wu et al., 2024; Beeching et al., 2024). However, these results are mainly confined to Transformer-based architectures, leaving open questions about whether similar scaling laws hold for subquadratic architectures, which offer faster inference but might compromise on expressiveness.

2.3 Alternatives to Transformer Architectures

Even though most reasoning models are based on the Transformer architecture (Grattafiori et al., 2024; Qwen et al., 2025), alternatives have been proposed to alleviate their high computational cost. Models built on top of RNNs (Beck et al., 2024; Peng et al., 2023), state space models (SSMs) (Gu et al., 2022; Gu & Dao, 2024), and linear attention mechanisms (Katharopoulos et al., 2020; Yang et al., 2024) demonstrate superior inference and memory efficiency, particularly for long-context tasks and large-batch generation. The Mamba series (Mamba-1 and Mamba-2) notably introduced selective state spaces to enable linear-time sequence modeling with strong performance (Gu & Dao, 2024; Dao & Gu, 2024). In addition, hybrid architectures that combine a few self-attention layers with subquadratic layers (e.g., Mamba) have emerged, showing advantages over both pure Transformer and pure subquadratic designs (Lieber et al., 2024; Ren et al., 2024). Such architectures are particularly suited to meet the high compute demands of inference-time scaling, and our work investigates their scaling properties.

2.4 Knowledge Distillation Strategies

Knowledge distillation has proven to be an effective means of transferring capabilities from large teacher models to smaller, more efficient student models (Hinton et al., 2015). In LLMs, this process compresses a larger pre-trained model into a more compact version while preserving core knowledge and functionality (Gu et al., 2024; Xu et al., 2024). Although larger models tend to exhibit superior reasoning abilities due to scaling properties (Xu et al., 2025; Wei et al., 2022), distillation techniques have enabled smaller models to achieve competitive reasoning performance (DeepSeek-AI et al., 2025; Labs, 2025). While most efforts have focused on intra-architecture distillation (e.g., Transformer-to-Transformer), recent studies have ventured into cross-architecture distillation. For instance, pretrained Transformers have been distilled into architectures such as RNNs (Kasai et al., 2021; Mercat et al., 2024), linear attention models (Zhang et al., 2024b; Zhang et al.), convolutional networks (Ralambomihanta et al., 2024), and SSMs (Bick et al., 2024; Wang et al., 2024b; Paliotta et al., 2025). Whether the robust reasoning abilities of Deepseek R1 (DeepSeek-AI et al., 2025) distilled models can be effectively transferred across different architectures remains an open question.

3 The M1 Reasoning Model

In this section, we present a multi-stage process for building our hybrid linear RNN reasoning model, M1. The approach has three stages: distillation, SFT, and RL. We begin by

Algorithm 1 Initializing MAMBAINLLAMA

Shapes: B - Batch, L - Length, D - embed size, $N = D/\text{Attention\_heads}$,$N'$ - expand
Input: $\boldsymbol{o}_t$: (B, D)
Output: output: (B, D)
New Params: MLP, $\mathbf{A}$
for each head $\mathbf{W}^K, \mathbf{W}^Q, \mathbf{W}^V, \mathbf{W}^o : (N, D)$ after expanding to same dimension **do**
  Head Parameter: $\mathbf{A} : (N, N')$
  for all positions t:
  $\boldsymbol{x}_t : (B, N) \leftarrow \mathbf{W}^V \boldsymbol{o}_t$
  $\mathbf{B}_t : (B, N) \leftarrow \mathbf{W}^K \boldsymbol{o}_t$
  $\mathbf{C}_t : (B, N) \leftarrow \mathbf{W}^Q \boldsymbol{o}_t$
  $\Delta_t : (B, N') \leftarrow \text{MLP}(\boldsymbol{x}_t)$
  $\overline{\mathbf{A}}_{1:T}, \overline{\mathbf{B}}_{1:T}, \overline{\mathbf{C}}_{1:T} : (B, N, N') \leftarrow \text{DISC}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \Delta)$
  $\boldsymbol{y} \leftarrow \text{LINEARRNN}(\overline{\mathbf{A}}, \overline{\mathbf{B}}, \overline{\mathbf{C}}, \boldsymbol{x})$
  output $\leftarrow$ output $+ \mathbf{W}^{O\top} \boldsymbol{y}$
end for
return output

distilling a Transformer model into a Mamba architecture, adapting the method of Wang et al. (2024a), which initializes the hybrid model's weights from a transformer model. We then perform math-specific supervised fine-tuning (SFT) on general mathematical datasets to enhance the model's mathematical performance, first without yet incorporating datasets generated by reasoning-focused models, and then with reasoning data leveraging multiple large-scale datasets generated by the R1 model series. Finally, we apply R1's GRPO method to further enhance the model's math reasoning capability.

Stage 1: Distillation. The first step in building our **M1** model is distilling a pretrained transformer model into a Mamba model. We adapt the distillation approach introduced by Wang et al. (2024a).

The MAMBAINLLAMA framework (Wang et al., 2024a) proposes distilling hybrid Transformer-Mamba models by reusing weights from attention layers. In this distillation procedure, outlined in Algorithm 1, linear projections for **Q**, **K**, **V**, and **O** are initialized from the corresponding projections for **C**, **B**, **X**, and **O**, respectively. The newly introduced parameters in the Mamba layers are the sampling rate Δ and the dynamic parameter **A**, which control the resulting Mamba module via a discretization function. Specifically, the sampling rate $\Delta \in \mathbb{R}^{N'}$ discretizes $\mathbf{B}_t, \mathbf{C}_t \in \mathbb{R}^{N \times 1}$, yielding $\overline{\mathbf{B}}_t, \overline{\mathbf{C}}_t \in \mathbb{R}^{N' \times N \times 1}$, as detailed in Algorithm 1. Different from Wang et al. (2024a), we introduce two additional linear layers to project from `head_dim * kv_head` to `head_dim * n_head`. This is because GQA (Ainslie et al., 2023) is used in the transformer model to reduce the KV cache. As Mamba does not utilize a KV cache, this expansion can increase the expressiveness of **B** and **X**.

We directly reuse the MLP layers; however, unlike the original approach, we replace the attention layers with Mamba layers in a single step. Subsequently, we fine-tune the entire model to expedite the training process. The distillation step involves minimizing the token-level KL divergence, aligning the entire probability distribution of the student model, $p(\cdot;\theta)$, with the teacher model, $p(\cdot;\theta_T)$, for every candidate token at position t. We use the reverse KL divergence, $D_{\text{KL}}(p(\cdot;\theta) \parallel p(\cdot;\theta_T))$, as our loss function rather than the forward KL divergence. We choose the reverse KL divergence due to its mode-seeking properties, which results in improved empirical performance.

We reimplement the distillation and SFT framework using the Axolotl [1]training framework. We apply the model chat template, mask the user prompt, and compute the loss only over the tokens generated in the assistant's output. To speed up training, we use data packing to merge different sequences into a single one until we reach the maximum sequence length which is set to 8192. We find that data packing achieves significantly better results compared

[1] https://github.com/axolotl-ai-cloud/axolotl

to the non-packing version in distillation for the same training steps. We use the AdamW optimizer with learning rate 1×10^{-5} with cosine decay, $\beta = (0.9, 0.95)$ and a weight decay of 0.1.

Stage 2: SFT Following the distillation procedure, we finetune the model on a large set of math problems, OpenMathInstruct-2 (Toshniwal et al., 2024). As in the distillation stage, we apply the chat template to the prompts, mask the user prompt, and compute the loss only over the tokens generated in the assistant's output. We train for two epochs using the same optimizer as distillation.

After the initial fine-tuning stage, we finetune on an additional set of math problems and solutions generated by reasoning models. We collect a mixed reasoning dataset, including OpenR1-Math-220k [2], OpenThoughts-114k-math[3], and ServiceNow-AI-R1-Distill[4], Magpie-Reasoning-250K[5] for a total of 10B reasoning tokens. The first two datasets were generated from R1, while the last two was generated from the R1 distilled Qwen 32B model and R1 distilled Llama 70B model. We extended the training length to 24,576 because we found that it covers 99% of the data items. We train the model for five epochs using the same optimizer as before but changing the peak learning rate to 6×10^{-6}.

Stage 3: Reasoning RL. To further enhance performance, we integrate Mamba with a RL pipeline for further training.[6] We use GRPO as the loss function. Differing from (Shao et al., 2024), we remove the KL penalty term as empirically we find it destabilizes training. Additionally, we include an entropy bonus to encourage a more diverse policy. The resulting formula is,

$$L_{\text{GRPO}}(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta_{\text{old}}}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} \hat{A}(s,a) \right] + \eta \, H(\pi_\theta) \tag{1}$$

where $\hat{A}(s,a)$ is the estimate of the advantage from multiple rollouts. We use a batch size of 128 and a PPO batch size of 64, which also determines the number of PPO iterations, $\mu = 2$. We set the number of generations for each sequence to 8 and the maximum generation length to 32k. For optimization, we use the Adam optimizer with a learning rate of 1×10^{-6}. We train for 50 steps, and pick the best checkpoint with the highest critic reward. We append the simple prompt *"Let's think step by step and output the final answer within \boxed{}"* to the end of each question in both training and evaluation.

4 Experiments

Model. We adopt the Llama3.2-3B-Instruct models as distillation target models. For Mamba layers, we set the SSM state size to 16. Consequently, the number of SSM groups after expansion is 3072/16 = 192 for the 3B model. We use 6 interleaved attention layers among 28 total layers.

Evaluation Dataset. Following common practice in evaluating reasoning models, we use a similar set of math benchmarks, including competition-level problems: MATH500 (Hendrycks et al., 2021), AIME25 (MAA, 2025), AIME24 (MAA, 2024), AMC23 (MAA, 2023), and OlympiadBench (He et al., 2024).

[2] https://huggingface.co/datasets/open-r1/OpenR1-Math-220k

[3] https://huggingface.co/datasets/open-thoughts/OpenThoughts-114k

[4] https://huggingface.co/datasets/ServiceNow-AI/R1-Distill-SFT

[5] https://huggingface.co/datasets/Magpie-Align/Magpie-Reasoning-V2-250K-CoT-Deepseek-R1-Llama-70B

[6] We add it into the popular VeRL (Sheng et al., 2024) framework. In doing so, we addressed and resolved the CUDA graph incompatibility issues that previously arose during training with PyTorch's FSDP module. As a result, the updated framework now efficiently supports Mamba generation with CUDA graph enabled, making it 5x faster than with CUDA Graph disabled

Evaluation Metrics. Our model's performance is assessed using two key metrics: coverage and accuracy. In fields such as coding and formal proofs, where answers can be automatically verified, coverage translates directly to enhanced performance and is widely utilized (Chen et al., 2021; Brown et al., 2024). Coverage is often measured using the pass@k metric, with k indicating the number of samples per problem (Chen et al., 2021; Brown et al., 2024). This metric estimates the likelihood that at least one correct solution exists among the k samples. To minimize variance when calculating coverage, we employ the unbiased estimation formula from Chen et al. (2021). Specifically, we generate $N \geq k$ total samples per task. The probability that a correct solution exists among a pool of k generated samples can then be determined given the total number of correct solutions C_i for each task.

$$\text{pass@k} = \frac{1}{\text{\# of problems}} \sum_{i=1}^{\text{\# of problems}} \left(1 - \frac{\binom{N-C_i}{k}}{\binom{N}{k}}\right)$$

We implement this formula using a numerically stable approach as recommended by Chen et al. (2021).

When using additional compute, we employ multiple aggregation strategies. The most straightforward method is majority voting, also known as self-consistency decoding (Wang et al., 2023), which takes the majority response among k samples as the predicted answer, and uses that to compute the accuracy.

4.1 Reasoning Evaluation

Model	AIME25	AIME24	MATH500	AMC23	OlympiadBench
Qwen2.5-Math-7B-Instruct	-	13.3	79.8	50.6	40.7
rStar-Math-7B (Guan et al., 2025)	-	26.7	78.4	47.5	47.1
Eurus-2-7B-PRIME (Cui et al., 2025)	-	26.7	79.2	57.8	42.1
Qwen2.5-7B-SimpleRL (Zeng et al., 2025)	-	26.7	82.4	62.5	43.3
DeepSeek-R1-Qwen-1.5B	23.0	28.8	**82.8**	**62.9**	43.3
M1-3B	**23.5**	**28.9**	82.1	62.8	**47.3**

Table 1: Evaluation results for **M1**-3B, DeepSeek-R1-Distill-Qwen-1.5B and other MATH models on MATH benchmarks

Model	AIME25		AIME24		MATH500		AMC23		OlympiadBench	
	Pass@1	Maj@32	Pass@1	Maj@32	Pass@1	Maj@32	Pass@1	Maj@32	Pass@1	Maj@32
DeepSeek-R1-Qwen-1.5B	23.0	**35.0**	28.8	49.2	**82.8**	91.0	**62.9**	54.2	43.3	80.3
M1-3B	**23.5**	34.6	**29.0**	**50.5**	82.1	**91.8**	62.8	**55.0**	**47.3**	80.1

Table 2: Maj@32 results comparing **M1**-3B with DeepSeek-R1-Distill-Qwen-1.5B.

We evaluate our models using a temperature setting of 0.7 and a sequence length of 32k with evaluation tools in VeRL. We use 32k because it has become the standard for evaluating performance on reasoning models (DeepSeek-AI et al., 2025; Luo et al., 2025). We report the pass@1 metric averaged over 64 runs; for majority voting, we repeat the metric calculation 100 times.

We report the accuracy of **M1**-3B and DeepSeek-R1-Distill-Qwen-1.5B in Table 1 and 2. We use the baseline DeepSeek-R1-Distill-Qwen-1.5B since a 3B R1 reasoning model is still not available. Although **M1**-3B has more parameters than DeepSeek-R1-Distill-Qwen-1.5B, its speed is still comparable even with shorter contexts, so we believe this is a fair comparison. Our model's performance is competitive with state-of-the-art open reasoning models in the same model size range and outperforms larger nonreasoning math transformer models. Our model performs slightly worse on AIME24 compared to the DeepSeek-R1-Distill-Qwen-1.5B model. Notably, DeepSeek-R1-Distill-Qwen-1.5B is built on top of the Qwen2.5 MATH models, which were finetuned with over 1T MATH tokens on top of the Qwen2.5 models, significantly more training data than what **M1**-3B used in total.

4.2 Speed Evaluation

We benchmark inference time with our model against a transformer model (Llama-3.2.-3B (Grattafiori et al., 2024)) of the same size. We use vLLM (version 0.6.3), which is the version used in VeRL for efficient rollouts. We also compare against DeepSeek-R1-Distill-Qwen-1.5B (DeepSeek-AI et al., 2025), a reasoning transformer model that is half the size of **M1**. This model has the same number of layers as the 3B parameter transformer, but the hidden dimension is half the size.

According to Luo et al. (2025), the average generation length of reasoning models on MATH questions is 4k to 5k. We therefore fix a decoding length of 4096 (and prompt length of 256) and benchmark our model across a range of batch sizes. We vary the batch size from 8 to 512, measuring the inference latency across different models.

We perform our benchmarking on a single NVIDIA H100 GPU with greedy decoding. To ensure that every model generates up to the set maximum number of tokens, we use `ignore_eos=True`. Before recording results, we warm up the system with two runs. The final performance metrics are then averaged over three subsequent runs. The inference speeds of the models across batch sizes are shown in Figure 1. **M1** achieves a 3× speedup over similarly-sized transformers when using a batch size of 512 and a decoding length of 4096, demonstrating its effectiveness in large-batch generation settings.

The maximum length of generated sequences is also an important factor in RL training, as longer sequences allow the model to use more compute during learning by generating longer chains-of-thought, shown in Figure 5. To benchmark our model in this setting, we fix the batch size to 128, and vary the generation length. We compare against the same two models as in the batch size varying case, and the results are shown in Figure 2. As the generated sequence length increases, **M1** achieves increasing speedups relative to the baseline models, and consistently generates at least 2x faster than Llama-3.2-3B (2.64x faster for the longest sequence length).

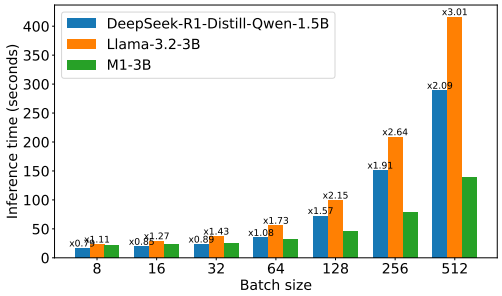

Figure 1: Inference latency when using prompt length 256 and decoding length 4096.

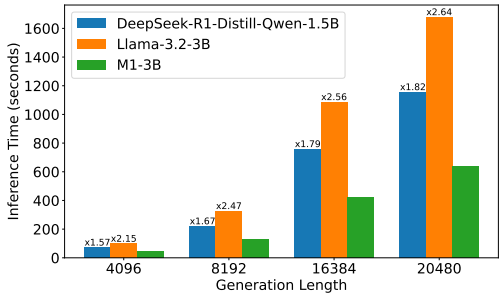

Figure 2: Inference latency when using batch size 128.

It is well-known that LLM inference comprises a prefilling (compute-bound) and a decoding (memory-bound) stage. For math reasoning models, it is common to assume that decoding takes much longer than prefilling, since prefilling only uses a short MATH question, while decoding generates long answers. Under these settings, the process is memory-bound. Given that Mamba is highly memory-efficient and we only use a SSM state size of 16, these memory advantages translate into improved speed.

4.3 Test-Time Scaling

Given a fixed time budget, **M1** can generate more sequences or longer sequences compared to a transformer model, which can hopefully boost its performance. We evaluate the effect of test-time compute scaling on model performance. We scale both the number of samples generated as well as the length of generated samples, to see if **M1** benefits from additional compute along these axes. We aim to investigate whether the speed benefit from section 4.2 can translate into an accuracy gain.

Scaling with majority vote.

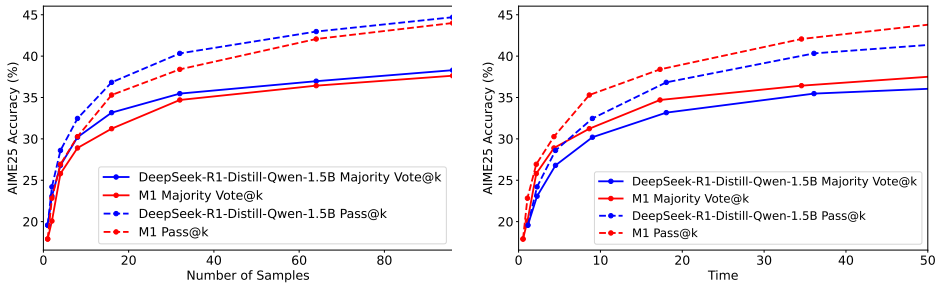

Figure 3: Number of samples vs. AIME25 accuracy (left) and generation time (seconds) vs. AIME25 accuracy (right). Both graphs include pass@1 and majority voting accuracies for **M1** and DeepSeek-R1-Distill-Qwen-1.5B.

The left side of Figure 3 shows the effect of scaling the number of generated samples (while fixing the maximum decoding length) on AIME25 accuracy. Both the baseline model and **M1** see increasing accuracy as the number of samples increases, with **M1** nearly matching the baseline performance for larger sample sizes. The efficient generation of **M1** also means that generating large number of samples at test-time is faster than for the baseline transformer model.

We quantify this efficiency in the right side of Figure 3, which compares the number of seconds spent generating samples against the resulting accuracy. To compute the time values on the x-axis, we find an optimal throughput value (in tokens per second) for each model by increasing batch sizes until throughput decreases. The optimal values were 7263 T/s for DeepSeek-R1-Distill-Qwen-1.5B, and 15169 T/s for **M1**. We then assume that each generated sample is maximum length (8K), and compute the seconds required for one sample from one model as 8K divided by the throughput. We then convert the left graph of Figure 3 into the right graph, by multiplying the number of samples for each datapoint by the seconds required per sample for each model. As an example, **M1** requires roughly a half second (8K/15K) per sample, so the accuracy value for **M1** at 32 samples on the left graph appears at approximately 16 seconds on the right graph.

Scaling with longer sequences

Figure 4 shows the effect of scaling the maximum length of the generated answer, while fixing the number of generated samples to one. For both the baseline and **M1**, increasing the maximum sequence length leads to increased accuracy, as shown in the left graph in Figure 4. After converting from generation length to the seconds required to generate (done in the same way as Figure 3, but dividing the generation length by throughput), we can see the accuracy gain per time spent generating on the right side of Figure 4. In this case, **M1** actually gets a higher accuracy for the same amount of time spent generating at 4 of the 5 evaluated sequence lengths, showing the benefits of efficient generation for test-time compute scaling.

5 Analysis

Increasing Training Length in RL boosts model performance

With more efficient models, we can increase the length of sequences used in RL training, resulting in improved performance. Empirically, we see this in Figure 5, which shows an increase in accuracy on AIME25 as we scale up the length of sequences generated when training with GRPO. Training with sequences of maximum length 4096 results in accuracy below 10%, while allowing sequences up to length 24K boosts the accuracy up to 23%.

MATH Accuracy at each training stage

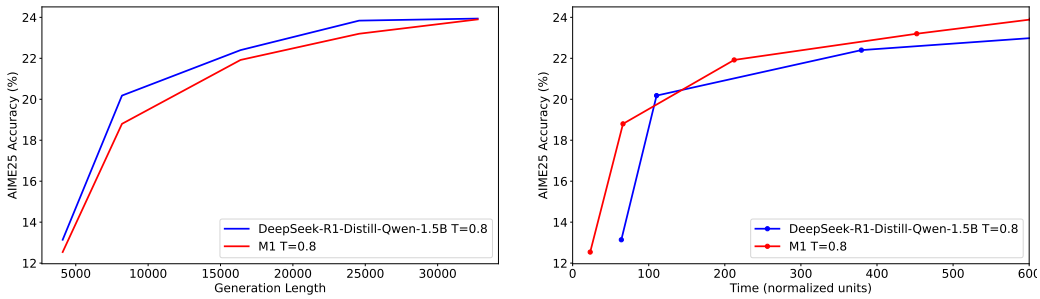

Figure 4: Generation length vs. AIME25 accuracy (left) and generation time (seconds) vs. AIME25 accuracy (right). Sampling for both models is done using a temperature of 0.8.

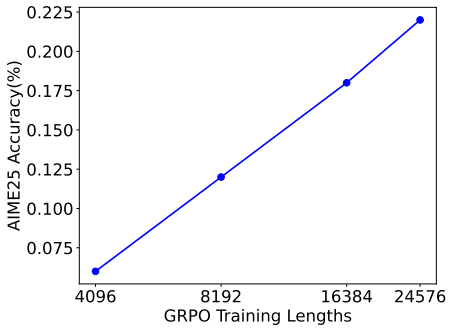

Figure 5: Pass@1 vs. maximum sequence length in GRPO training

	MATH500	AIME24
Distill	38	0
Distill + SFT(MATH)	45	0
Distill + SFT(MATH) + SFT(Reason)	74	22
Distill + SFT(MATH) + SFT(Reason) + RL	82	28

Table 3: **M1** Accuracy after each training stage on MATH500 and AIME24.

To identify which components of our training pipeline have the greatest impact on performance, we also evaluate intermediate versions of the model on MATH500 (Hendrycks et al., 2021) and AIME24 (MAA, 2024). The results of these evaluations are presented in Table 3. Each step of the training pipeline provides a boost to performance, with particularly large gains from fine-tuning on solutions from reasoning models (+29% on MATH500 and +17% on AIME24).

Direct Distillation from Reasoning Models We also attempted to distill from Deepseek-R1-Qwen-1.5B instead of Llama-3.2-3B. In this case, we did not SFT on OpenMathInstruct, and instead only SFT on the 10B reasoning data that we collected after distillation. We found that the distilled model's performance was poor (38% and 3.3% pass@1 accuracy on MATH500 and AIME24, resspectively). Our hypothesis for why this occurs is that 10B tokens is insufficient to effectively transfer reasoning skills from the transformer to Mamba. Although curating a high-quality reasoning dataset demands significant time and effort, we begin by leveraging the standard MATH distillation dataset from OpenMathInstruct (Toshniwal et al., 2024) to first distill a strong MATH model. We then transform this MATH model into a reasoning model via SFT on the dedicated reasoning dataset. This approach achieves strong performance with a much smaller number of reasoning tokens.

6 Conclusion

In this paper, we introduced M1, a hybrid reasoning model built on the Mamba architecture, designed to address the scalability challenges of the Transformer models. We demonstrated effective techniques for distillation and finetuning to develop M1, which achieves mathematical reasoning performance comparable to state-of-the-art reasoning models of similar size. Notably, M1 delivers over 3x faster inference than similar-sized Transformer models, even when using the heavily optimized vLLM inference engine, particularly at large batch sizes. This improved efficiency can make the resource-intensive inference-time strategies, such as self-consistency, more practical. Our findings establish M1 as a strong alternative to Transformer-based architectures, paving the way for more efficient and high-performing reasoning models.

References

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.04697.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, 2023.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory, 2024. URL https://arxiv.org/abs/2405.04517.

Edward Beeching, Lewis Tunstall, and Sasha Rush. Scaling test-time compute with open models, 2024. URL https://huggingface.co/spaces/HuggingFaceH4/blogpost-scaling-test-time-compute.

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.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024. URL https://arxiv.org/abs/2407.21787.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and et. al. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wangxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality, 2024. URL https://arxiv.org/abs/2405.21060.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and et. al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars, 2025. URL https://arxiv.org/abs/2503.01307.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens, 2024. URL https://arxiv.org/abs/2310.02226.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and et. al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL https://arxiv.org/abs/2312.00752.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022. URL https://arxiv.org/abs/2111.00396.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models, 2024. URL https://arxiv.org/abs/2306.08543.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A. Smith. Finetuning pretrained transformers into rnns, 2021. URL https://arxiv.org/abs/2103.13076.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020. URL https://arxiv.org/abs/2006.16236.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Bespoke Labs. Bespoke-stratos: The unreasonable effectiveness of reasoning distillation. www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation, 2025. Accessed: 2025-01-22.

Xinzhe Li. A survey on llm test-time compute via search: Tasks, llm profiling, search algorithms, and relevant frameworks, 2025. URL https://arxiv.org/abs/2501.10069.

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.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision, 2024. URL https://arxiv.org/abs/2406.06592.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2, 2025. Notion Blog.

MAA. American invitational mathematics examination 2023, 2023. URL https://artofproblemsolving.com/wiki/index.php/American_Invitational_Mathematics_Examination?srsltid=AfmBOoqiDCiaGTLQrsRTKsZui8RFnjOZqM4qIqY3yGB3sBaqOaxwf_Xt.

MAA. American invitational mathematics examination 2024, 2024. URL https://artofproblemsolving.com/wiki/index.php/American_Invitational_Mathematics_Examination?srsltid=AfmBOoqiDCiaGTLQrsRTKsZui8RFnjOZqM4qIqY3yGB3sBaqOaxwf_Xt.

MAA. American invitational mathematics examination 2025, 2025. URL https://artofproblemsolving.com/wiki/index.php/American_Invitational_Mathematics_Examination?srsltid=AfmBOoqiDCiaGTLQrsRTKsZui8RFnjOZqM4qIqY3yGB3sBaqOaxwf_Xt.

Jean Mercat, Igor Vasiljevic, Sedrick Keh, Kushal Arora, Achal Dave, Adrien Gaidon, and Thomas Kollar. Linearizing large language models, 2024. URL https://arxiv.org/abs/2405.06640.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

Daniele Paliotta, Junxiong Wang, Matteo Pagliardini, Kevin Y Li, Aviv Bick, J Zico Kolter, Albert Gu, François Fleuret, and Tri Dao. Thinking slow, fast: Scaling inference compute with distilled reasoners. *arXiv preprint arXiv:2502.20339*, 2025.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Jiaju Lin, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, and et. al. Rwkv: Reinventing rnns for the transformer era, 2023. URL https://arxiv.org/abs/2305.13048.

Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models, 2024. URL https://arxiv.org/abs/2404.15758.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. Mutual reasoning makes smaller llms stronger problem-solvers, 2024. URL https://arxiv.org/abs/2408.06195.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and et. al. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Tokiniaina Raharison Ralambomihanta, Shahrad Mohammadzadeh, Mohammad Sami Nur Islam, Wassim Jabbour, and Laurence Liang. Scavenging hyena: Distilling transformers into long convolution models, 2024. URL https://arxiv.org/abs/2401.17574.

Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling, 2024. URL https://arxiv.org/abs/2406.07522.

Matthew Renze and Erhan Guven. The effect of sampling temperature on problem solving in large language models, 2024. URL https://arxiv.org/abs/2402.05201.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data, 2024. URL https://arxiv.org/abs/2410.01560.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback, 2022. URL https://arxiv.org/abs/2211.14275.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander Rush, and Tri Dao. The mamba in the llama: Distilling and accelerating hybrid models. *Advances in Neural Information Processing Systems*, 37:62432–62457, 2024a.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander M. Rush, and Tri Dao. The mamba in the llama: Distilling and accelerating hybrid models. *arXiv preprint arXiv:2408.15237*, 2024b.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022. URL https://arxiv.org/abs/2206.07682.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models, 2024. URL https://arxiv.org/abs/2408.00724.

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Chenyang Shao, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Yu Li, Jie Feng, Chen Gao, and Yong Li. Towards large reasoning models: A survey of reinforced reasoning with large language models, 2025. URL https://arxiv.org/abs/2501.09686.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models, 2024. URL https://arxiv.org/abs/2402.13116.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training, 2024. URL https://arxiv.org/abs/2312.06635.

Wang Yang, Hongye Jin, Jingfeng Yang, Vipin Chaudhary, and Xiaotian Han. Thinking preference optimization, 2025. URL https://arxiv.org/abs/2502.13173.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL https://arxiv.org/abs/2305.10601.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025.

Weihao Zeng, Yuzhen Huang, Wei Liu, Keqing He, Qian Liu, Zejun Ma, and Junxian He. 7b model and 8k examples: Emerging reasoning with reinforcement learning is both effective and efficient. https://hkust-nlp.notion.site/simplerl-reason, 2025. Notion Blog.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search, 2024a. URL https://arxiv.org/abs/2406.03816.

Michael Zhang, Simran Arora, Rahul Chalamala, Benjamin Frederick Spector, Alan Wu, Krithik Ramesh, Aaryan Singhal, and Christopher Re. Lolcats: On low-rank linearizing of large language models. In *The Thirteenth International Conference on Learning Representations*.

Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Ré. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry, 2024b. URL https://arxiv.org/abs/2402.04347.

Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B. Tenenbaum, and Chuang Gan. Planning with large language models for code generation, 2023. URL https://arxiv.org/abs/2303.05510.

Xuan Zhang, Fengzhuo Zhang, Cunxiao Du, Chao Du, Tianyu Pang, Wei Gao, and Min Lin. Lighttransfer: Your long-context llm is secretly a hybrid model with effortless adaptation. *arXiv preprint arXiv:2410.13846*, 2024c.

A Limitations and Future Work

Speedup. Our current hybrid model is only 3× faster than a Transformer of the same size when serving inference with vLLM. Recently, NVIDIA introduced a new hybrid Mamba kernel[7], which could further boost the speed of hybrid models. Additionally, our attention implementation in hybrid models does not yet leverage the optimizations available in vLLM. Integrating **M1** into vLLM could further boost performance by taking advantage of these attention speedups.

Why do we not distill Qwen2.5 1.5B MATH model. We considered using the Qwen2.5 1.5B MATH Instruct model as the distillation target in the first stage. However, we found that the cross entropy loss of the Qwen 1.5B MATH model on the OpenMATH Instruct dateset (Toshniwal et al., 2024) exceeded 1.8, which is much higher than that of the Llama models (0.5). This suggests that, to mimic the Qwen2.5 model, we need a dataset generated from a large Qwen2.5 series model rather than this one generated from the Llama models. Dataset curation from Qwen Math models goes beyond the scope of this work.

[7] https://github.com/NVIDIA/Megatron-LM/commit/b957578e76a921209ef873cbbd389114a4042542

Improvement on RL training speed Recently, DeepSeek R1 (DeepSeek-AI et al., 2025) showed that reinforcement learning (RL) is a key component in improving model reasoning performance during post-training. Since then, recent research has predominantly relied on reinforcement learning (RL) as a training paradigm for reasoning models. However, training with RL requires the efficient generation of long sequences. For example, in VeRL (Sheng et al., 2024), the typical training batch size ranges from a few thousand to several thousand. DeepscaleR (Luo et al., 2025) also shows a significant accuracy boost when training RL with longer sequences, as it tends to enhance model performance by providing more steps for thorough reasoning. However, this shift towards reinforcement learning has resulted in the generation process becoming a significant bottleneck in reasoning model training, taking more than three times as long as the actor's weight update (forward + backward) according to the time profiling done for DeepscaleR (Luo et al., 2025). This need for efficient generation in RL presents a significant challenge for transformer models, namely due to the heavy computational burden imposed by large key-value caches during generation, especially for large batch sizes. Given their generation speed advantages, linear RNN models may be better suited for scaling RL training.