MiniMax-M1: Scaling Test-Time Compute Efficiently with Lightning Attention

MiniMax[1]

We introduce MiniMax-M1, the world's first open-weight, large-scale hybrid-attention reasoning model. MiniMax-M1 is powered by a hybrid Mixture-of-Experts (MoE) architecture combined with a lightning attention mechanism. The model is developed based on our previous MiniMax-Text-01 model (MiniMax et al., 2025), which contains a total of 456 billion parameters with 45.9 billion parameters activated per token. The M1 model natively supports a context length of 1 million tokens, 8x the context size of DeepSeek R1. Furthermore, the lightning attention mechanism in MiniMax-M1 enables efficient scaling of test-time compute – For example, compared to DeepSeek R1, M1 consumes 25% of the FLOPs at a generation length of 100K tokens. These properties make M1 particularly suitable for complex tasks that require processing long inputs and thinking extensively. MiniMax-M1 is trained using large-scale reinforcement learning (RL) on diverse problems ranging from traditional mathematical reasoning to sandbox-based, real-world software engineering environments. In addition to the inherent efficiency advantage of lightning attention for RL training, we propose CISPO, a novel RL algorithm to further enhance RL efficiency. CISPO clips importance sampling weights rather than token updates, outperforming other competitive RL variants. Combining hybrid-attention and CISPO enables MiniMax-M1's full RL training on 512 H800 GPUs to complete in only three weeks, with a rental cost of just $534,700. We release two versions of MiniMax-M1 models with 40K and 80K thinking budgets respectively, where the 40K model represents an intermediate phase of the 80K training. Experiments on standard benchmarks show that our models are comparable or superior to strong open-weight models such as the original DeepSeek-R1 and Qwen3-235B, with particular strengths in complex software engineering, tool utilization, and long-context tasks. Through efficient scaling of test-time compute, MiniMax-M1 serves as a strong foundation for next-generation language model agents to reason and tackle real-world challenges. We publicly release MiniMax-M1 at https://github.com/MiniMax-AI/MiniMax-M1.

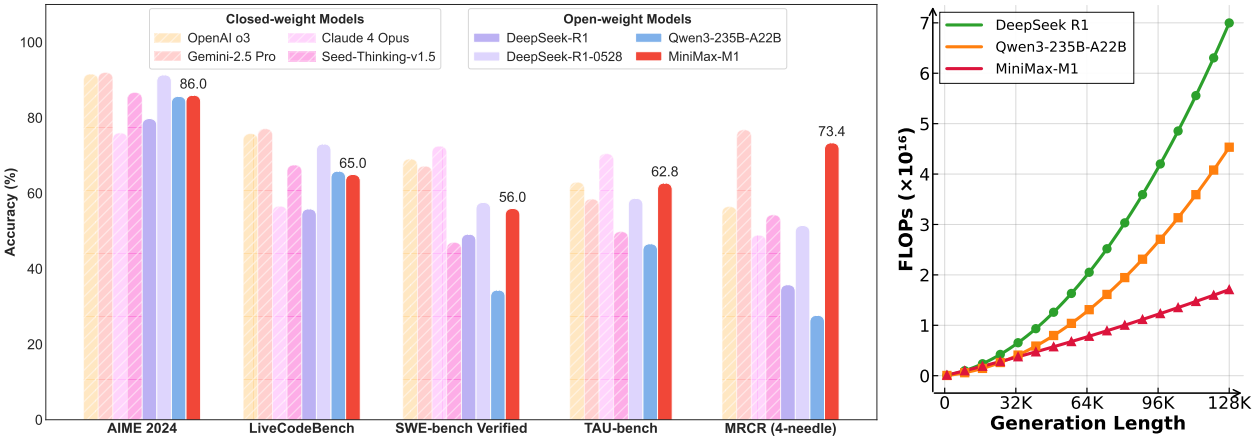

Figure 1 | **Left**: Benchmark performance comparison of leading commercial and open-weight models across competition-level mathematics, coding, software engineering, agentic tool use, and long-context understanding tasks. We use the MiniMax-M1-80k model here for MiniMax-M1. **Right**: Theoretical inference FLOPs scaling with generation length (# tokens).

[1]Please send correspondence to model@minimax.io.

1. Introduction

Large reasoning models (LRMs), such as OpenAI o1 (OpenAI, 2024a) and DeepSeek-R1 (DeepSeek-AI et al., 2025), have demonstrated remarkable success by extending the length of reasoning through large-scale reinforcement learning (RL). In recent months, both the open-source community and commercial organizations have followed this trend, achieving significant advances on complex tasks such as Olympiad mathematics competitions and competitive programming (Anthropic, 2025; Google DeepMind, 2025; Hu et al., 2025; Kimi Team, 2025; Seed et al., 2025; Yu et al., 2025; Zeng et al., 2025). The success of LRMs has been primarily attributed to a new scaling dimension of test-time compute—As more FLOPs are dedicated to extended reasoning processes during generation, model performance shows consistent improvement, particularly for complex real-world applications (Jimenez et al., 2024; OpenAI, 2025).

However, continuously extending the reasoning process is challenging within the traditional transformer architecture (Vaswani et al., 2017), due to the inherent quadratic computational complexity of the softmax attention mechanism. While previous works have proposed various techniques to mitigate this issue—such as sparse attention (Beltagy et al., 2020; Lu et al., 2025; Yuan et al., 2025; Zaheer et al., 2020), linear attention (Arora et al., 2024; Choromanski et al., 2021; Du et al., 2025; He et al., 2024; Katharopoulos et al., 2020; Peng et al., 2024b, 2021; Qin et al., 2021, 2022a,b, 2024a,c; Shen et al., 2024; Sun et al., 2025, 2023; Zhang et al., 2024), linear attention with delta decay (Peng et al., 2025; Yang et al., 2024a,b), state space models (Dao and Gu, 2024; Glorioso et al., 2024; Gu and Dao, 2024; Gu et al., 2020, 2022, 2023; Gupta et al., 2022; Jamba Team, 2024; Ren et al., 2024), and linear RNNs (Behrouz et al., 2024; Chou et al., 2024; Chung and Ç, 2014; Hochreiter and Schmidhuber, 1997; Martin and Cundy, 2018; Peng et al., 2023, 2024a; Qin et al., 2023, 2024d; Siems et al., 2025; Sun et al., 2024; von Oswald et al., 2025)—these approaches have not been fully validated in large-scale reasoning models, and nearly all competitive LRMs to date still rely on traditional attention designs. An exception is the Hunyuan-T1 model (Tencent AI Lab, 2025) that employs the Mamba architecture (Dao and Gu, 2024; Gu and Dao, 2024). However, this model is not open-sourced and few details are disclosed. In this work, we aim to build and open-source a large reasoning model that can efficiently scale up test-time compute and compete with the state-of-the-art reasoning models.

We introduce MiniMax-M1, a reasoning model with a hybrid Mixture-of-Experts (MoE) architecture and Lightning Attention (Qin et al., 2024b), an I/O-aware implementation of a linear attention variant (Qin et al., 2022a). MiniMax-M1 is developed based on our previous MiniMax-Text-01 (MiniMax et al., 2025) model, and comprises 456 billion parameters in total, with 45.9 billion activations and 32 experts. In our attention design, a transformer block with softmax attention follows every seven transnormer blocks (Qin et al., 2022a) with lightning attention. This design theoretically enables efficient scaling of reasoning lengths to hundreds of thousands of tokens, as illustrated in Figure 1 (Right). For example, compared to DeepSeek R1, M1 consumes less than 50% of the FLOPs at a generation length of 64K tokens, and approximately 25% of the FLOPs at a length of 100K tokens. This substantial reduction in computational cost makes M1 significantly more efficient during both inference and large-scale RL training. Furthermore, owing to its lightning attention mechanism and in line with MiniMax-Text-01, our M1 model natively supports a context length of up to 1 million tokens – eight times the context size of DeepSeek R1 and an order of magnitude greater than all open-weight LRMs available to date. These features make M1 particularly well-suited for addressing complex, real-world tasks that require processing long inputs and generating extended thinking. A comparison of the maximum input and output lengths of M1 and other leading models is demonstrated in Table 1.

To develop our M1 model, we first continue pretraining MiniMax-Text-01 on 7.5T tokens from a carefully curated, reasoning-intensive corpus. Subsequently, we perform supervised fine-tuning (SFT)

Table 1 | The maximum supported input length and output length (# tokens) of different reasoning models. For Claude-4 we refer to the Claude-4-Opus model. "DS-R1" represents the latest `DeepSeek-R1-0528` model.

	o3	Gemini 2.5 Pro	Claude 4	DS-R1	Qwen3-235B	MiniMax-M1-80k
Max Input	200K	1M	200K	128K	128K	1M
Max Output	100K	64K	32K	64K	32K	80K

to inject certain chain-of-thought (CoT) (Wei et al., 2022) patterns, establishing a strong foundation for reinforcement learning, the core stage of M1 development. Notably, our RL scaling with M1 is made efficient through innovations from two key perspectives: (1) We propose a novel RL algorithm, CISPO, which abandons the trust region constraint and instead clips the importance sampling weights to stabilize training. This approach always leverages all tokens for gradient computations, achieving enhanced efficiency compared to GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025) empirically – For example, on a controlled study based on Qwen2.5-32B models (Qwen et al., 2025), CISPO achieves a 2x speedup compared to DAPO; (2) Although the hybrid-attention design in M1 naturally allows for efficient RL scaling, unique challenges arise when scaling RL with this architecture. For instance, we find a precision mismatch between the training and inference kernels of our architecture, which prevents reward growth during RL training. We develop targeted solutions to address these challenges and successfully scale up RL with this hybrid architecture. In the end, our efficient RL framework enables us to complete a full RL run of MiniMax-M1 within 3 weeks using 512 H800 GPUs—equivalent to a rental cost of approximately $0.53M USD.

In addition to methodological innovations, we curate a diverse set of problems and environments for RL training. Our data encompasses both verifiable and non-verifiable problems. For verifiable problems that are typically considered critical for reasoning learning, we not only include mathematical reasoning and competitive programming problems as commonly used in related works, but also leverage our previous data synthesis framework SynLogic (Liu et al., 2025a) to generate diverse logical reasoning problems spanning 41 distinct tasks. Furthermore, we construct sandboxes for complex software engineering (SE) environments derived from SWE-bench (Jimenez et al., 2024), and conduct RL on real-world SE problems with execution-based rewards to improve M1's performance in challenging SE scenarios. Our unverifiable problems span a broad range of domains such as question answering and creative writing, where we use generative reward models to provide the feedback.

We train two versions of MiniMax-M1 models with 40K and 80K tokens of maximum generation length respectively, which leads to two models MiniMax-M1-40k and MiniMax-M1-80k. MiniMax-M1-80k outperforms MiniMax-M1-40k on complex mathematical and coding tasks, further demonstrating the benefits of scaling test-time compute. As shown in Figure 1 (Left), MiniMax-M1 surpasses previous leading open-weight models such as the original DeepSeek-R1 and Qwen-235B overall, with particular advantages in complex software engineering, tool-using, and long-context tasks. Compared to the latest DeepSeek-R1-0528 model, MiniMax-M1 lags in mathematical and coding competitions but achieves comparable or superior performance in more realistic tool-using and long-context scenarios. Notably, MiniMax-M1 outperforms Gemini 2.5 Pro on the agentic tool use benchmark TAU-Bench (Yao et al., 2025), and surpasses OpenAI o3 and Claude 4 Opus on long-context understanding benchmarks. With efficient test-time scaling, we contend that MiniMax-M1 establishes a strong foundation for next-generation language model agents to address real-world challenges.

To facilitate collaboration and advancement in the field, we have made our models publicly available at GitHub and Hugging Face. They are now supported by both the `vLLM` and `Transformers` frameworks, with detailed deployment guides available at vLLM and Transformers respectively. This

enables easy integration of MiniMax-M1 into modern inference pipelines. We also provide commercial standard API at minimax.io.

2. Preparation for Scalable RL: Continual Pretraining and SFT

In this work, we focus on scaling up reinforcement learning to enhance reasoning capabilities of Minimax-Text-01. To facilitate scalable RL training, we first carry out continual pretraining of our base model to strengthen its intrinsic reasoning abilities. Subsequently, we perform a cold-start supervised fine-tuning (SFT) stage to inject specific reasoning patterns to the model, thereby providing a stronger foundation for the subsequent RL phase.

2.1. Continual Pre-Training: Foundation for RL Scaling

To enhance the reasoning and long context capabilities of the foundation model while ensuring diversity, we continue training the MiniMax-Text-01 model with additional 7.5T tokens with optimized data quality and mixture.

Training Data. We refine our pretraining Web and PDF parsing mechanisms and enhance our heuristic cleaning rules to ensure a high recall rate for mathematical and code-related data. We prioritize the extraction of natural Question-Answer (QA) pairs from a diverse range of sources, including webpages, forums, and textbooks, while strictly avoiding the use of synthetic data. Additionally, we conduct semantic deduplication on the QA data to maintain its diversity and uniqueness. Furthermore, we increase the proportion of STEM (Science, Technology, Engineering, and Mathematics), code, book, and reasoning-related data to 70%. This significantly enhances the foundation model's ability to handle complex tasks without compromising its other general capabilities.

Training Recipe. We decrease the coefficient of the MoE auxiliary loss and adjust the parallel training strategy to support a larger training micro batch size, which mitigates the detrimental effects of the auxiliary loss on overall model performance. Based on MiniMax-Text-01, we continue training with a constant learning rate of 8e-5 for 2.5T tokens, followed by a decay schedule over 5T tokens down to 8e-6.

Long Context Extension. For a hybrid-lightning architecture model with higher convergence complexity, we have observed that excessively aggressive extensions of the training length can lead to a sudden gradient explosion that may occur during the training process. This makes the optimization process extremely challenging. We attribute this to the parameter optimization of the earlier layers not keeping up with the changes in the later layers – For lightning attention, the earlier and later layers have different decay rates, which makes the earlier layers focus more on local information. We alleviate this issue by adapting a smoother extension of context length across four stages, starting from a 32K context window length and ultimately extending the training context to 1M tokens.

2.2. Supervised Fine-Tuning: Focused Alignment for Efficient RL

After continual pretraining, we conduct Supervised Fine-Tuning (SFT) to instill desired behaviors like reflection-based Chain-of-Thought (CoT) reasoning using high-quality examples, creating a strong starting point for more efficient and stable RL in the next stage. Specifically, we curate data samples with long CoT responses. These data samples cover diverse domains such as math, coding, STEM, writing, QA, and multi-turn chat. Math and coding samples account for around 60% of all the data.

3. Efficient RL Scaling: Algorithms and Lightning Attention

As shown in Figure 1 (Right), the M1 architecture demonstrates a clear efficiency advantage during inference. This naturally facilitates efficient RL scaling where increasingly longer responses are generated. However, as pioneers in scaling up RL with this hybrid architecture, we encounter unique challenges during the process, and the RL procedure can become unstable or even fail due to various issues. To address these difficulties, we develop targeted solutions that enable us to successfully scale up RL training for M1. In addition, we propose a new RL algorithm that achieves greater RL efficiency compared to existing methods. These dual contributions yield an efficient and scalable RL framework for training M1, where the complete training cycle requires 3 weeks on 512 H800 GPUs—equivalent to a rental cost of approximately $0.53M USD. In this section, we first provide general context on RL and present our novel RL algorithm, and then describe the specific challenges we face with the hybrid architecture, along with the solutions we devise to overcome them.

3.1. Efficient RL Scaling with CISPO

Background. For questions q from a dataset $\mathcal{D}$, we denote π as the policy model parameterized by θ, and o as the response generated by the policy. PPO (Schulman et al., 2017) adopts the following objective to optimize the policy to maximize the expected return, and a clipping operation is applied to stabilize training:

$$\mathcal{J}_{\text{PPO}}(\theta) = \mathbb{E}_{q\sim\mathcal{D}, o_i\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \min\left(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}\big(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_{i,t} \right) - \beta D_{KL}(\pi_\theta||\pi_{\text{ref}}) \right], \tag{1}$$

where $r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}$ is the importance sampling (IS) weight, which is used to correct the distribution during off-policy updates, because we use $\pi_{\theta_{\text{old}}}$ to collect trajectories to update the policy via multiple steps in a minibatch manner. While PPO requires a separate value model to compute the advantage $\hat{A}_{i,t}$, GRPO (Shao et al., 2024) eliminates the value model and defines the advantage as the output reward relative to other responses in the group:

$$\hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_j\}_{j=1}^G)}{\text{std}(\{R_j\}_{j=1}^G)}, \tag{2}$$

where R_i is the reward of the response, and G responses $\{o_i\}_{i=1}^G$ are sampled for each question. The reward is either from rule-based verifiers such as in mathematical problem solving, or from a reward model.

Issues of Token Clipping. In our initial experiments with the hybrid architecture under the zero-RL setting, we observed that the GRPO algorithm adversely affected training performance and failed to effectively promote the emergence of long CoT reasoning behaviors. Through a series of controlled ablation studies, we ultimately identified the undesirable clipping operation in the original PPO/GRPO loss as the primary factor contributing to degraded learning performance. Specifically, we found that tokens associated with reflective behaviors (e.g., `However`, `Recheck`, `Wait`, `Aha`), which often serve as "forks" in reasoning paths, were typically rare and assigned low probabilities by our base model. During policy updates, these tokens were likely to exhibit high $r_{i,t}$ values. As a result, these tokens were clipped out after the first on-policy update, preventing them from contributing to subsequent off-policy gradient updates. This issue was particularly pronounced in our hybrid-architecture model and further hindered the scalability of reinforcement learning. These low-probability tokens, however,

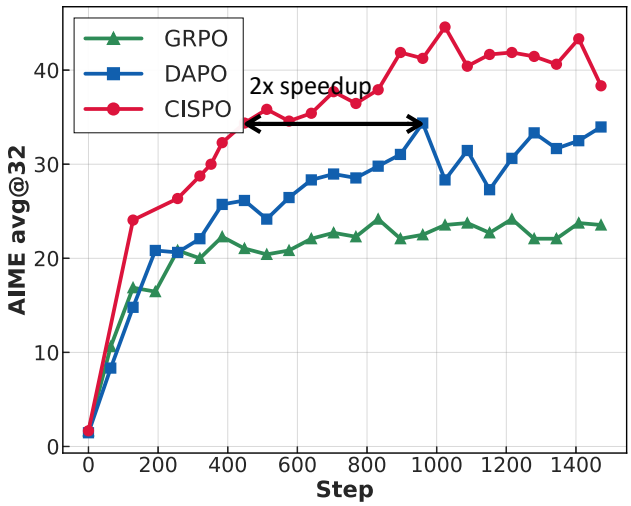

Figure 2 | Comparison of GRPO, DAPO, and our proposed CISPO on AIME 2024, based on Qwen2.5-32B-base. CISPO outperforms both GRPO and DAPO in terms of performance at the same number of training steps, and achieves comparable performance to DAPO using 50% of the training steps.

are often crucial for stabilizing entropy (Cui et al., 2025) and facilitating scalable RL (Wang et al., 2025). Although DAPO attempts to mitigate this issue by increasing the upper clipping bound (Yu et al., 2025), we found this approach to be less effective in our setup, which involved 16 rounds of off-policy updates per generation batch.

The CISPO Algorithm. In response, we propose a new algorithm that explicitly avoids dropping tokens, even those associated with large updates, while inherently maintaining entropy within a reasonable range to ensure stable exploration. First, recall that the vanilla REINFORCE objective with corrected distribution for offline updates is:

$$\mathcal{J}_{\text{REINFORCE}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},o_i\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \text{sg}(r_{i,t}(\theta))\hat{A}_{i,t} \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}) \right], \tag{3}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operation. Rather than clipping the token updates as in PPO/GRPO, we instead clip the importance sampling weight in Eq. 3 to stabilize training. We term our approach CISPO (**C**lipped **IS**-weight **P**olicy **O**ptimization). Adopting the group relative advantage from GRPO and the token-level loss (Liu et al., 2025b; Yu et al., 2025), CISPO optimizes the following objective:

$$\mathcal{J}_{\text{CISPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{\sum_{i=1}^{G}|o_i|} \sum_{i=1}^{G}\sum_{t=1}^{|o_i|} \text{sg}(\hat{r}_{i,t}(\theta))\hat{A}_{i,t} \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}) \right], \tag{4}$$

where $\hat{r}_{i,t}(\theta)$ is the clipped IS weight:

$$\hat{r}_{i,t}(\theta) = \text{clip}\left(r_{i,t}(\theta), 1-\epsilon^{IS}_{low}, 1+\epsilon^{IS}_{high}\right). \tag{5}$$

We note that without weight clipping, $\mathcal{J}_{\text{CISPO}}$ reduces to the standard policy gradient objective. In our experiments, we did not impose a lower bound on the IS weight by setting ϵ^{IS}_{low} to a large value;

instead, we only tuned ϵ^{IS}_{high}. Although the gradient of Eq. 4 is slightly biased due to weight clipping, this approach preserves gradient contributions from all tokens, especially in long responses. CISPO proves effective in our experiments, helping reduce variance and stabilizing RL training. In addition, we utilize the dynamic sampling and length penalty techniques from Yu et al. (2025). There is no KL penalty term in CISPO similar to other recent works (Hu et al., 2025; Yu et al., 2025).

A General Formulation. While we adopt CISPO in our experiments, here we further present a unified formulation by introducing a token-wise mask into the CISPO objective. This allows for hyperparameter tuning to control whether, and under what conditions, gradients from specific tokens should be dropped:

$$\mathcal{J}_{\text{unify}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{\sum_{i=1}^{G}|o_i|} \sum_{i=1}^{G}\sum_{t=1}^{|o_i|} \text{sg}(\hat{r}_{i,t}(\theta))\hat{A}_{i,t} \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}) M_{i,t} \right]. \tag{6}$$

The mask $M_{i,t}$ is equivalent to the mask implicitly defined in the PPO trust region:

$$M_{i,t} = \begin{cases} 0 & \text{if } \hat{A}_{i,t} > 0 \text{ and } r_{i,t}(\theta) > 1+\epsilon_{\text{high}}, \\ 0 & \text{if } \hat{A}_{i,t} < 0 \text{ and } r_{i,t}(\theta) < 1-\epsilon_{\text{low}}, \\ 1 & \text{otherwise.} \end{cases} \tag{7}$$

This unified loss formulation can flexibly represent different clipping strategies under a common framework.

Empirical Validation of CISPO. To validate the effectiveness of CISPO, we empirically compare it with DAPO and GRPO in a zero-RL training setting. Specifically, we apply different RL algorithms to train the Qwen2.5-32B-base model on the mathematical reasoning dataset from Yu et al. (2025), and report performance on the AIME 2024 benchmark. As shown in Figure 2, CISPO significantly outperforms both DAPO and GRPO with the same number of training steps. Notably, CISPO demonstrates superior training efficiency compared to other approaches; for example, it matches DAPO's performance with only 50% of the training steps.

3.2. Efficient RL Scaling with Lightning Attention – Challenges and Recipes

As shown in Figure 1 (Right), we emphasize that our hybrid attention inherently enables more efficient RL scaling compared to traditional attention designs, since rollout computation and latency are often the primary bottlenecks in RL training. However, as pioneers in conducting large-scale RL experiments with this novel architecture, we encountered unique challenges and developed targeted solutions, as we describe below.

Computational Precision Mismatch in Generation and Training. RL training is highly sensitive to computational precision. During our RL training, we observed a significant discrepancy in the probabilities of rolled-out tokens between training-mode and inference-mode, as shown in Figure 3 (Left). This discrepancy arose from a precision mismatch between the training and inference kernels. The issue was detrimental and prevented reward growth in our experiments. Interestingly, this issue did not appear in smaller, dense models with softmax attention. Through layer-by-layer analysis, we identified high-magnitude activations in the LM head at the output layer as the primary source of error. To address this, we increased the precision of the LM output head to FP32, thereby realigning the two theoretically identical probabilities, as demonstrated in Figure 3 (Right). This adjustment improved the correlation between training and inference probabilities from approximately 0.9x to

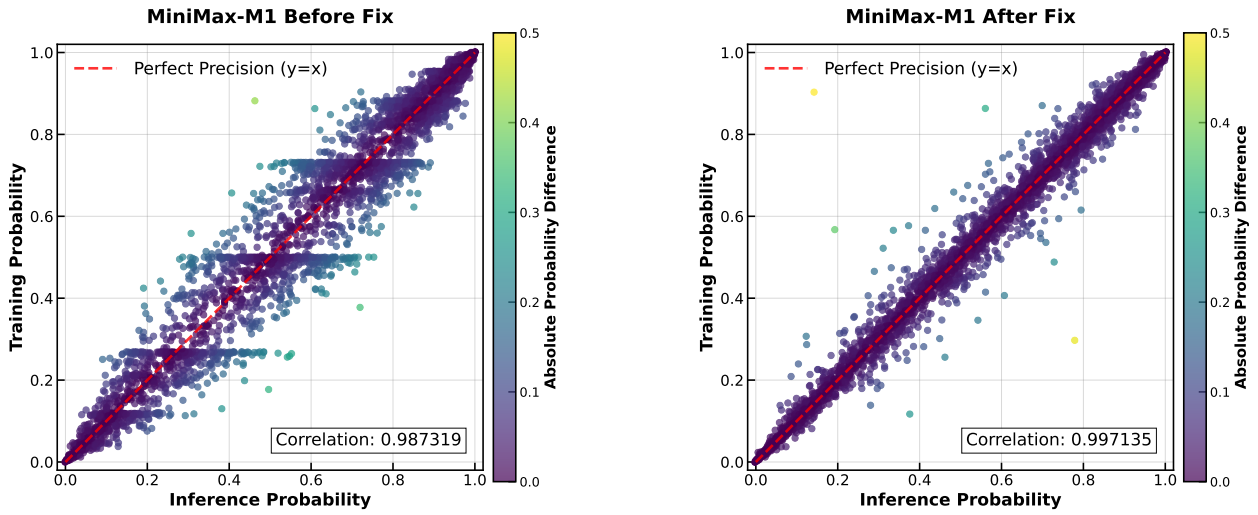

Figure 3 | Probability of tokens in training-mode code vs. probability of tokens in inference-mode code. Each point in the figures represents an individual token. The Pearson correlation coefficient is indicated in the figures. Theoretically, the two probabilities should be identical, and all the tokens should be exactly on the diagonal line. **Left:** Correlation of the M1 model before our fix; **Right:** Correlation of the M1 model after applying our fix of using FP32 precision for the LM output head.

0.99x. Notably, this correlation metric remained stable throughout training, enabling successful reward increase.

Optimizer Hyperparameter Sensitivity. We employ the AdamW (Loshchilov and Hutter, 2019) optimizer, and inappropriate configurations of β_1, β_2, and ϵ can lead to non-convergence during training. (Molybog et al., 2023). For instance, using the default configuration from VeRL (Sheng et al., 2024), where betas = (0.9, 0.999) and eps = 1e-8, can result in such issues. We have observed that the gradient magnitudes in MiniMax-M1 training span a wide range, from 1e-18 to 1e-5, with the majority of the gradients being smaller than 1e-14. Furthermore, the correlation between the gradients of adjacent iterations is weak. Based on this, we set $\beta_1 = 0.9$, $\beta_2 = 0.95$, and eps=1e-15.

Early Truncation via Repetition Detection. During RL training, we found that complex prompts could induce pathologically long and repetitive responses, whose large gradients threatened model stability. Our goal was to preemptively terminate these generation loops rather than penalize the already repetitive text. As simple string-matching is ineffective against varied repetition patterns, we developed a heuristic based on token probabilities. We observed that once a model enters a repetitive cycle, the probability for each token soars. Consequently, we implemented an early truncation rule: generation is halted if 3,000 consecutive tokens each have a probability above 0.99. This method successfully prevents model instability and improves generation throughput by eliminating these pathological, long-tail cases.

4. Scaling Reinforcement Learning with Diverse Data

In this section, we describe the data and reward we adopted for our RL stage. We incorporate a diverse set of environments in our RL training pipeline, including tasks that can be verified by rules and general tasks that need to be verified through reward models. All these environments are integrated into the RL stage using a carefully designed curriculum.

4.1. Reasoning-Intensive Tasks with Rule-based Verification

Below, we introduce our data that can be verified by deterministic rules. For all the following tasks, we employ rule-based final correctness as the correctness reward, complemented by a format reward.

Mathematical Reasoning. Our initial mathematical dataset comprises hundreds of thousands of high-quality, competition-level problems, meticulously curated and organized from public sources and official mathematics competitions. These problems span a wide range of difficulty levels, each paired with a standard reference solution. Our data cleaning pipeline begins with the removal of incomplete samples and those exhibiting formatting or typographical errors. We subsequently apply embedding-based deduplication across the RL data sources and enforce a strict separation from the SFT dataset to avoid any overlap, as leakage from the SFT phase into the RL stage hinders exploration and undermines training effectiveness. Additionally, we employ both n-gram and embedding-based methods to eliminate potential contamination from commonly used mathematical benchmark test sets, thereby ensuring the integrity and fairness of our evaluations. We filter out samples containing multiple sub-problems, proof-based questions, and binary questions (e.g., true/false) that are susceptible to random guessing. Multiple-choice questions are reformulated into open-ended formats to better align with our reinforcement learning framework. Next, we employ our internal model to extract the final answers from the reference solution, retaining only those samples whose extracted answers can be correctly parsed by our rule-based answer checker. Finally, we use a strong reasoning model to compute the pass@10 for each question and retain only those samples with a pass rate strictly between 0 and 0.9, resulting in a curated dataset of nearly 50K high-quality mathematical samples for our RL training.

Logical Reasoning. For logical reasoning data, we carefully select 41 logical reasoning tasks requiring non-trivial reasoning ability such as cipher and Sudoku, then we implement a data synthesis framework to synthesize all the data. Concretely, we utilize our SynLogic framework (Liu et al., 2025a) to implement the data synthesis pipeline featuring task-specific data generators and rule-based task-specific verifiers, enabling automatic logical data generation. We meticulously configure the difficulty parameters during generation, ensuring the appropriate learning challenge of the generated data. Specifically, to prevent inclusion of overly difficult instances, we establish an upper difficulty bound based on the solvability limits of current strong reasoning models, requiring their pass@10 rates greater than zero. Similarly, we set a lower difficulty bound using the lowest difficulty parameters for which the MiniMax-Text-01 model achieves pass rates between 0 and 0.5. This approach ensures the data maintains a balance between difficulty and learnability. In addition, as the model capabilities improve during training, we increase the difficulty of the data in the later stages. Using this framework, we synthesize approximately 53K logical reasoning samples for RL training.

Competitive Programming. For the competitive programming problems, we collect publicly available problems from online judge platforms and popular coding websites. For problems lacking test cases, we develop an LLM-based workflow and use the MiniMax-Text-01 model to generate comprehensive test suites. Similar to our approach with mathematical reasoning datasets, we filter problems based on quality and difficulty using pass rates from model sampling, retaining moderately challenging and high-quality algorithmic problems. Through this process, we generate 30K competitive programming data samples for RL training.

Software Engineering. For the software engineering domain, inspired by SWE-bench (Jimenez et al., 2024), we construct verifiable reinforcement learning environments by leveraging real-world data from public GitHub repositories. Our dataset primarily comprises issues and pull requests (PRs) that encapsulate common software development challenges, including bug localization, code repair, and test case synthesis. To facilitate effective reinforcement learning, we develop a sophisticated containerized sandbox environment that simulates a realistic software development workflow. This

environment enables the actual execution of code, providing direct and verifiable feedback on the correctness and efficacy of an agent's proposed interventions. The pass/fail status of pre-defined or newly generated test cases serves as the primary reward signal for our RL framework. A successful execution that passes all relevant test cases yields a positive reward, while compilation errors, runtime failures, or test case regressions result in a zero or negative reward, thus providing a clear signal for policy optimization. Through this process, we curate several thousand high-quality data samples. Each sample includes a problem description (e.g., bug report from an issue), the initial faulty code, and a set of associated test cases. This setup allows our RL agent to learn to accurately pinpoint bugs, propose correct code fixes, and even synthesize new, effective test cases, with performance directly verifiable through the execution within our sandboxed environment.

4.2. General Domain Tasks with Model-based Feedbacks

In this section, we further extend the RL scope to a wider array of general domain tasks. As these tasks cannot be easily verified by rules, we utilize reward models to provide the feedback.

4.2.1. Data and Reward Models

Our general RL dataset consists of a total of 25K complex samples. These can be broadly categorized into two types: samples with ground-truth answers that are verifiable but difficult to validate using rules, and samples without ground-truth answers.

Tasks with Ground Truth. This category primarily includes STEM and other factual problems where answers are objective but may have multiple valid expressions. Such diversity often renders rule-based answer checkers inaccurate. Our data cleaning process is similar to that used in mathematical reasoning, while we use our Generative Reward Model (GenRM) as a verifier, instead of relying on rule-based checkers. To evaluate consistency between ground-truth answers and model responses, we adopt a five-grade reward scale to evaluate the two components. First, we construct a human-annotated reward model benchmark, which covers a range of objective tasks across diverse knowledge and task domains, especially the pairs of model response–ground truth that rule-based checkers fail to judge accurately. Second, we evaluate the GenRM's effectiveness by comparing the Best-of-N (BoN) responses selected by GenRM against the pass@N metrics across several benchmarks. GenRM performance is assessed using its accuracy on the human-annotated benchmark and the performance gap between BoN and pass@N. These metrics guide experiments to optimize both the data distribution and the prompt design used during the GenRM training.

Tasks without Ground Truth. This category encompasses a wider range of tasks, including instruction-following, creative writing, etc. Prompts are sampled from a large pool based on our internal tagging system, ensuring a balanced training distribution across fine-grained domains. Even though these queries are typically open-ended and do not have a ground-truth answer, we seek to pair a reference answer for each query, which serves as a reference for reward model judgment. To this end, we first generate responses by various internal and external models, and then these reference answers will undergo our internal quality evaluation. During RL training, we adopt a pairwise comparison framework to evaluate model responses. Each comparison yields a score of -1, 0, or 1, indicating whether the model's output is worse than, similar to, or better than a reference answer. For instruction-following tasks with constraints particularly, we utilize both the rule-based reward to assess whether the response satisfies the constraint, and model-based reward to evaluate response's quality. As with the ground-truth setting, we first build a human-annotated benchmark, incorporating multiple blind preference judgments from reliable annotators. We then refine our scoring criteria and preference prompt to optimize accuracy as well as potential biases, which would be mentioned in §4.2.2 below.

To minimize the potential biases, training data are also optimized by several methods, such as multiple-blind consistent judgment, position-switched consistent judgment, etc. Once an optimal GenRM is trained, a Swiss Round scoring system is performed across the training dataset to determine the most suitable reference answer for RL training.

4.2.2. Addressing Bias of Generative Reward Models for Long CoT

Effective general RL for complex CoT reasoning tasks is critically dependent on accurate and unbiased reward models. Assessing such CoT responses turns out to be challenging, and we found that GenRMs preferred longer outputs over potentially superior concise alternatives, irrespective of actual reasoning quality. This **length bias** is a significant issue as it may substantially misguide RL policy optimization, incentivizing verbosity without substance and inducing reward hacking. Our initial efforts to improve GenRM fidelity include standard offline strategies: (1) Diversifying training data with a wide range of response lengths, sources, and quality tiers; (2) Incorporating adversarial examples to expose vulnerabilities; and (3) Refining model architectures. However, empirical analysis revealed that purely offline evaluation and preemptive mitigation of length bias in GenRMs frequently failed to prevent length bias during RL training.

Consequently, our core strategy incorporates continuous online monitoring of length bias during RL training. Specific metrics are established to detect whether the RL policy disproportionately extends output lengths to maximize GenRMs rewards without gains in task success or reasoning depth. Upon detecting such detrimental length-seeking behavior, indicative of exploiting GenRMs length bias, immediate GenRMs recalibration is triggered. This iterative adjustment is vital to preempt reward hacking related to output length, ensuring the policy prioritized substantive capability enhancement over superficial text inflation. Complementing this adaptive approach, RL-side techniques including reward shaping, value clipping, and normalization are systematically employed. These mechanisms desensitize reward signals to extreme values from superficial characteristics (e.g., length), thereby directing policy optimization toward substantive quality and correctness of its long CoT reasoning.

4.3. Curriculum of Incorporating Diverse Data

Given that our RL data spans a wide spectrum of categories, a core challenge is training a single policy capable of excelling on both reasoning-intensive tasks and general domain tasks. To address this, our approach entails a carefully managed curriculum and dynamic weighting strategy for reasoning and general-domain tasks during the RL training process with CISPO: we start with only the reasoning-intensive tasks with rule-based reward, and then gradually mix in the general domain tasks. This ensures that the model continues to refine its verifiable skills (e.g., in math and code) while progressively enhancing its performance on a diverse spectrum of general tasks, from complex instruction following to open-ended CoT reasoning. This mixed RL training encourages the model to learn context-dependent application of its reasoning abilities—applying rigorous, step-by-step deduction for verifiable problems and more flexible, adaptive generation for general queries—all within a unified policy framework. It prevents catastrophic forgetting of specialized skills while fostering broader generalization.

5. Extending RL Scaling to Longer Thinking

Our first RL training is performed with an output length limit of 40K tokens. Given that the hybrid architecture of M1 natively supports near-linear scaling for longer sequences, as demonstrated in Figure 1 (Right), we further extend the generation length during RL training to 80K tokens. This results in a new model, which we refer to as MiniMax-M1-80k.

Data. To efficiently train our RL model for an 80K output length, we utilize our previously trained 40K model to guide the data filtering process. First, we evaluate the pass rates on the curated dataset described in §4 and remove samples that are easily solved. We then adjust the data distribution to favor more challenging examples, such as difficult mathematical and coding problems. Additionally, we downsample synthetic reasoning data after observing that it destabilizes long-context RL training. Specifically, outputs generated from this data type often become repetitive and homogenous, and continued exposure to these patterns proves detrimental to the model's overall performance.

Length Scaling Strategy. To gradually increase the output length, we employ a staged window expansion RL strategy. We begin with an output length of 40K and incrementally expand it to 48K, 56K, 64K, 72K, and ultimately 80K. This staged approach ensures training stability at each step. The transition to a subsequent length is determined by a set of empirical indicators. These include the convergence of perplexity on the generated sequences and whether the 99th percentile of the output lengths is approaching the current context window limit. These signals offer valuable insights into the model's readiness for scaling, which allows us to maintain robust training throughout the process.

Addressing Training Instability During Scaling. During the scaling process, we encountered a critical issue in the later stages of training at each length window. Specifically, the model exhibited susceptibility to pattern collapse, where the latter portions of generated sequences degraded into incoherent or garbled text. This phenomenon consistently coincided with increased perplexity, indicating compromised generation quality and stability. We identify the root cause: during output length extension, negative samples increase in length substantially faster than positive samples, frequently reaching the context window limit earlier. Consequently, disproportionately large negative gradients accumulate in the latter segments of generation sequences. This imbalance originates from the inherently unequal nature of GRPO's advantage normalization and the token-level loss we adopt. To address this, we implement three key solutions: (1) Detecting repetitive patterns (consecutive high-probability tokens) with early stopping to prevent excessive context window consumption by repetitive responses; (2) Adopting combined sample-level loss and token-level normalization to alleviate negative-positive sample imbalance and mitigate adverse effects; (3) Decreasing both the gradient clipping threshold and ϵ^{IS}_{high} to further stabilize generation.

6. Evaluations

6.1. Core Benchmarks

We conduct a comprehensive evaluation of MiniMax-M1 across several key domains: mathematics, general coding, software engineering, reasoning & knowledge, long context, agentic tool use, factuality, and general assistant ability. We evaluate all tasks using temperature 1.0 and top-p 0.95 sampling.

- **Mathematics:** To evaluate mathematical reasoning capabilities, we utilize several competition level math benchmarks, including MATH-500 (Hendrycks et al., 2021), AIME 2024, AIME 2025. For AIME evaluation, we sample 32 times and compute the average passrate as the final score.
- **General Coding:** We assess general programming proficiency using LiveCodeBench (Jain et al., 2025) and FullStackBench (Liu et al., 2024), which evaluate code generation across diverse programming tasks. For both benchmarks, we report scores as the average passrate of 16 samples.
- **Reasoning & Knowledge:** We assess domain knowledge and reasoning capabilities through GPQA-Diamond (Rein et al., 2024), MMLU-Pro (Wang et al., 2024), and the challenging HLE benchmark (Phan et al., 2025). For GPQA-Diamond, we sample 32 times and report the average passrate. For HLE evaluation, we assess the model without external tools. Additionally, we

Table 2 | **Performance of MiniMax-M1 on core benchmarks.**

Tasks	Leading Close-Weights Models				Open-Weights Models			Our Models	
	OpenAI-o3	**Gemini 2.5 Pro (06-05)**	**Claude 4 Opus**	**Seed-Thinking-v1.5**	**DeepSeek-R1**	**DeepSeek-R1-0528**	**Qwen3-235B-A22B**	**MiniMax-M1-40k**	**MiniMax-M1-80k**
Extended Thinking	*100K*	*64K*	*64K*	*32K*	*32K*	*64K*	*32K*	*40K*	*80K*
Mathematics									
AIME 2024	91.6	92.0	76.0	86.7	79.8	91.4	85.7	83.3	86.0
AIME 2025	88.9	88.0	75.5	74.0	70.0	87.5	81.5	74.6	76.9
MATH-500	98.1	98.8	98.2	96.7	97.3	98.0	96.2	96.0	96.8
General Coding									
LiveCodeBench *(24/8~25/5)*	75.8	77.1	56.6	67.5	55.9	73.1	65.9	62.3	65.0
FullStackBench	69.3	–	70.3	69.9	70.1	69.4	62.9	67.6	68.3
Reasoning & Knowledge									
GPQA Diamond	83.3	86.4	79.6	77.3	71.5	81.0	71.1	69.2	70.0
HLE *(no tools)*	20.3	21.6	10.7	8.2	8.6*	17.7*	7.6*	7.2*	8.4*
ZebraLogic	95.8	91.6	95.1	84.4	78.7	95.1	80.3	80.1	86.8
MMLU-Pro	85.0	86.0	85.0	87.0	84.0	85.0	83.0	80.6	81.1
Software Engineering									
SWE-bench Verified	69.1	67.2	72.5	47.0	49.2	57.6	34.4	55.6	56.0
Long Context									
OpenAI-MRCR *(128k)*	56.5	76.8	48.9	54.3	35.8	51.5	27.7	76.1	73.4
OpenAI-MRCR *(1M)*	–	58.8	–	–	–	–	–	58.6	56.2
LongBench-v2	58.8	65.0	55.6	52.5	58.3	52.1	50.1	61.0	61.5
Agentic Tool Use									
TAU-bench *(airline)*	52.0	50.0	59.6	44.0	–	53.5	34.7	60.0	62.0
TAU-bench *(retail)*	73.9	67.0	81.4	55.7	–	63.9	58.6	67.8	63.5
Factuality									
SimpleQA	49.4	54.0	–	12.9	30.1	27.8	11.0	17.9	18.5
General Assistant									
MultiChallenge	56.5	51.8	45.8	43.0	40.7	45.0	40.0	44.7	44.7

* conducted on the text-only HLE subset.

measure logical reasoning ability using ZebraLogic (Lin et al., 2025).

- **Software Engineering:** We evaluate software engineering capabilities using SWE-bench Verified (Jimenez et al., 2024), which measures the ability to resolve real-world GitHub issues. We report results derived from the Agentless scaffold (Xia et al., 2024). Departing from the original pipeline, our methodology employs a two-stage localization process (without any embedding-based retrieval mechanisms): initial coarse-grained file localization followed by fine-grained localization to specific files and code elements.
- **Long Context:** We evaluate long context understanding using OpenAI-MRCR (OpenAI, 2024b), which tests retrieval and disambiguation of multiple similar items within extended contexts, and LongBench-v2 (Bai et al., 2024), a challenging benchmark with 503 multiple-choice questions

across contexts ranging from 8k to 2M words.

- **Agentic Tool Use:** We assess tool use capabilities through TAU-bench (Yao et al., 2025), which emulates dynamic conversations where agents must utilize API tools while adhering to domain-specific policy guidelines. We evaluate TAU-bench with GPT-4.1 as user model, a general system prompt[2] and without any custom tools. The maximum number of interaction steps is 40.
- **Factuality:** To measure factuality of LLMs, we utilize SimpleQA (Wei et al., 2024), an adversarially-collected benchmark of fact-seeking questions with single, indisputable answers.
- **General Assistant:** We evaluate general assistant capabilities using MultiChallenge (Sirdeshmukh et al., 2025), which assesses LLMs on conducting realistic multi-turn conversations with human users. We report our scores judged by GPT-4o.

Results on Math, Coding, and other General Tasks. Table 2 presents our model's performance compared to state-of-the-art large reasoning models. In mathematical reasoning, the MiniMax-M1 models demonstrate strong performance across multiple benchmarks, achieving results comparable to the close-weight model Seed-Thinking-v1.5 (Seed et al., 2025). Notably, MiniMax-M1-80k achieves 86.0% on AIME 2024, placing it second among open-weight models and trailing only the latest DeepSeek-R1-0528 model. For general coding, MiniMax-M1-80k matches Qwen3-235B on LiveCodeBench while outperforming it on FullStackBench, demonstrating robust capabilities among leading open-weight models. On reasoning & knowledge benchmarks, MiniMax-M1-80k similarly trails DeepSeek-R1-0528 but achieves competitive performance against other top open-weight models. On the factuality benchmark SimpleQA, Minimax-M1 models underperform DeepSeek-R1 while outperforming all other open-weight models and Seed-Thinking-v1.5. On MultiChallenge, both MiniMax models perform comparably to DeepSeek-R1-0528 and Claude 4 Optus, with inferior results only to o3 and Gemini-2.5-Pro.

Highlights in Complex Scenarios: Software Engineering, Long Context, and Tool use. Benefiting from our execution-based, software engineering environments during RL, MiniMax-M1-40k and MiniMax-M1-80k achieve strong scores of 55.6% and 56.0% on SWE-bench verified respectively. These results are slightly inferior to DeepSeek-R1-0528's 57.6% and significantly surpass other open-weights models. Leveraging its 1M context window, the M1 models significantly outperform all other open-weight models in long-context understanding. They even surpass OpenAI o3 and Claude 4 Opus, ranking second globally and trailing only Gemini 2.5 Pro by a small margin. In agentic tool-use scenarios (TAU-bench), MiniMax-M1-40k surpasses all open-weight models and even Gemini-2.5 Pro. Moreover, MiniMax-M1-80k consistently outperforms MiniMax-M1-40k across most benchmarks, confirming the benefits of scaling test-time compute.

6.2. Effect of RL Scaling

To investigate the effect of RL scaling, we track performance and response length throughout training. Figure 4 presents three representative examples from AIME 2024, AIME 2025, and LiveCodeBench v5, respectively. We observe consistent improvements in both model performance and response length during training. Notably, average response lengths on AIME and LiveCodeBench exceed 20,000 tokens, with AIME 2024 accuracy showing substantial gains from 68% to 80%. Crucially, the strong correlation between accuracy gains and increased response length in these visualizations underscores the importance of extending RL scaling to facilitate more extensive reasoning processes.

[2] "In each round, you need to carefully examine the tools provided to you to determine if any can be used. You must adhere to all of the policies. Pay attention to the details in the terms. Solutions for most situations can be found within these policies."

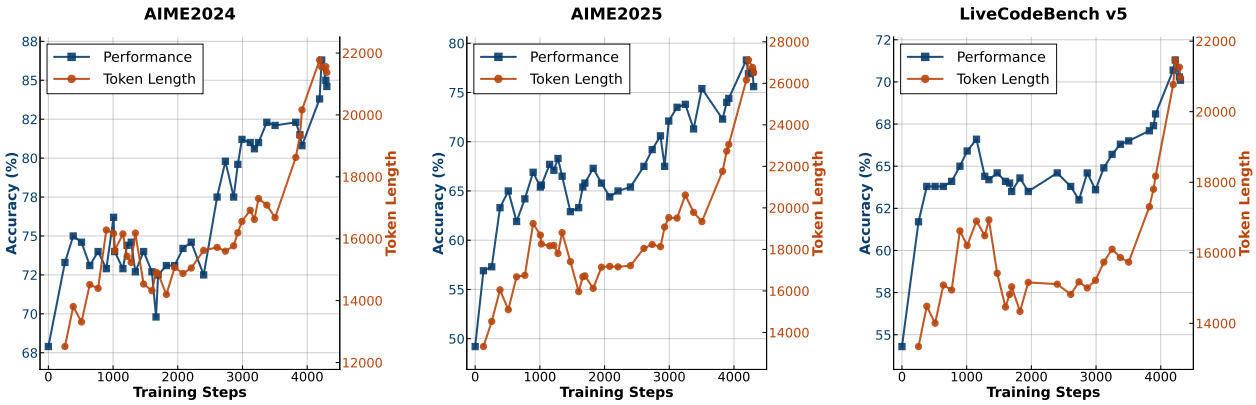

Figure 4 | Accuracy and generation length versus RL training steps for MiniMax-M1.

7. Conclusion and Future work

In this work, we introduce and release MiniMax-M1, the world's first open-weight, large-scale reasoning model featuring a lightning attention mechanism. This efficient attention design enables MiniMax-M1 to natively support inputs of up to 1M tokens and generation lengths of 80K tokens—both significantly exceeding capabilities of other open-weight models. These capabilities render MiniMax-M1 uniquely suited for complex, realistic scenarios requiring long context and extended reasoning, properties empirically validated by its strong performance on software engineering, agentic tool use, and long-context understanding benchmarks. Beyond the inherent efficiency advantages of lightning attention for RL training, this work contributes a novel RL algorithm, CISPO, to accelerate training. Combining architectural advantages with CISPO, we efficiently trained MiniMax-M1, with complete RL training completed in three weeks using 512 H800 GPUs. Across comprehensive evaluations, MiniMax-M1 ranks among the world's best open-weight models alongside DeepSeek-R1 and Qwen3-235B.

Looking forward, as test-time compute continuously scales to power increasingly complex scenarios, we foresee significant potential for such efficient architectures in addressing real-world challenges. These include automating company workflows (Xu et al., 2025) and conducting scientific research (OpenAI, 2025; Si et al., 2024). Real-world applications particularly demand LRMs that function as agents interacting with environments, tools, computers, or other agents—requiring reasoning across dozens to hundreds of turns while integrating long-context information from diverse sources. We envision MiniMax-M1 serving as a strong foundation for such applications with unique advantages, and we are fully dedicated to further evolving MiniMax-M1 toward this goal.

References

Anthropic. Claude 3.7 sonnet and claude code. https://www.anthropic.com/news/claude-3-7-sonnet, 2025. Blog post, February 24, 2025.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Ré. Simple linear attention language models balance the recall-throughput tradeoff. *arXiv preprint arXiv:2402.18668*, 2024.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench. *arXiv preprint arXiv:2412.15204*, 2024.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with Performers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

Yuhong Chou, Man Yao, Kexin Wang, Yuqi Pan, Rui-Jie Zhu, Jibin Wu, Yiran Zhong, Yu Qiao, Bo Xu, and Guoqi Li. Metala: Unified optimal linear approximation to softmax attention map. *Advances in Neural Information Processing Systems*, 37:71034–71067, 2024.

Junyoung Chung and Ç. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, and Yu Cheng. Mom: Linear sequence modeling with mixture-of-memories. *arXiv preprint arXiv:2502.13685*, 2025.

Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. Zamba: A compact 7b SSM. *arXiv preprint arXiv:2405.16712*, 2024.

Google DeepMind. Gemini pro. https://deepmind.google/models/gemini/pro/, 2025. Web page, accessed 2025.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33:1474–1487, 2020.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Albert Gu, Isys Johnson, Aman Timalsina, Atri Rudra, and Christopher Re. How to train your HIPPO: State space models with generalized orthogonal basis projections. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=klK17OQ3KB.

Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal state spaces are as effective as structured state spaces. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/9156b0f6dfa9bbd18c79cc459ef5d61c-Abstract-Conference.html.

Zhihao He, Hang Yu, Zi Gong, Shizhan Liu, Jianguo Li, and Weiyao Lin. Rodimus*: Breaking the accuracy-efficiency trade-off with efficient attentions. *arXiv preprint arXiv:2410.06577*, 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. *arXiv preprint arXiv:2103.03874*, 2021.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025.

Jamba Team. Jamba-1.5: Hybrid T. *arXiv preprint arXiv:2408.12570*, 2024.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

Kimi Team. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. Zebralogic: On the scaling limits of llms for logical reasoning. *arXiv preprint arXiv:2502.01100*, 2025.

Junteng Liu, Yuanxiang Fan, Zhuo Jiang, Han Ding, Yongyi Hu, Chi Zhang, Yiqi Shi, Shitong Weng, Aili Chen, Shiqi Chen, Yunan Huang, Mozhi Zhang, Pengyu Zhao, Junjie Yan, and Junxian He. Synlogic: Synthesizing verifiable reasoning data at scale for learning logical reasoning and beyond. *arXiv preprint arXiv:2505.19641*, 2025a.

Siyao Liu, He Zhu, Jerry Liu, Shulin Xin, Aoyan Li, Rui Long, Li Chen, Jack Yang, Jinxiang Xia, Z. Y. Peng, Shukai Liu, Zhaoxiang Zhang, Ge Zhang, Wenhao Huang, Kai Shen, and Liang Xiang. Fullstack bench: Evaluating llms as full stack coders. *arXiv preprint arXiv:2412.00535*, 2024.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, et al. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025.

Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=HyUNwulC-.

MiniMax, Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, et al. Minimax-01: Scaling foundation models with lightning attention. *arXiv preprint arXiv:2501.08313*, 2025.

Igor Molybog, Peter Albert, Moya Chen, Zachary DeVito, David Esiobu, Naman Goyal, Punit Singh Koura, Sharan Narang, Andrew Poulton, Ruan Silva, Binh Tang, Diana Liskovich, Puxin Xu, Yuchen Zhang, Melanie Kambadur, Stephen Roller, and Susan Zhang. A theory on adam instability in large-scale machine learning. *arXiv preprint arXiv:2304.09871*, 2023.

OpenAI. Introducing openai o1. https://openai.com/o1/, 2024a. Web page, accessed 2024.

OpenAI. Openai mrcr dataset. https://huggingface.co/datasets/openai/mrcr, 2024b. Accessed: 2025-06-15.

OpenAI. Introducing deep research, 2025. URL https://openai.com/index/introducing-deep-research/.

Bo Peng, Eric Alcaide, Quentin Gregory Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Nguyen Chung, Leon Derczynski, et al. Rwkv: Reinventing rnns for the transformer era. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Teddy Ferdinan, Haowen Hou, and Przemysł Kazienko. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence. *arXiv preprint arXiv:2404.05892*, 2024a.

Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Teddy Ferdinan, Haowen Hou, and Przemysł Kazienko. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence. *arXiv preprint arXiv:2404.05892*, 2024b.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, et al. Rwkv-7. *arXiv preprint arXiv:2503.14456*, 2025.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=QtTKTdVrFBB.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Zhen Qin, Xiaodong Han, Weixuan Sun, Dongxu Li, Lingpeng Kong, Nick Barnes, and Yiran Zhong. The devil in linear transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7025–7041, 2022a.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosFormer: Rethinking softmax in attention. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=Bl8CQrx2Up4.

Zhen Qin, Songlin Yang, and Yiran Zhong. Hierarchically gated recurrent neural network for sequence modeling. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 33202–33221, 2023.

Zhen Qin, Yuxin Mao, Xuyang Shen, Dong Li, Jing Zhang, Yuchao Dai, and Yiran Zhong. You only scan once: Efficient multi-dimension sequential modeling with lightnet. *arXiv preprint arXiv:2405.21022*, 2024a.

Zhen Qin, Weigao Sun, Dong Li, Xuyang Shen, Weixuan Sun, and Yiran Zhong. Lightning attention-2: A free lunch for handling unlimited sequence lengths in large language models. *arXiv preprint arXiv:2401.04658*, 2024b.

Zhen Qin, Weigao Sun, Dong Li, Xuyang Shen, Weixuan Sun, and Yiran Zhong. Various lengths, constant speed: Efficient language modeling with lightning attention. In *International conference on machine learning*, pages 41517–41535. PMLR, 2024c.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. HGRN2. *arXiv preprint arXiv:2404.07904*, 2024d.

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, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

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. *arXiv preprint arXiv:2406.07522*, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

ByteDance Seed, Jiaze Chen, Tiantian Fan, Xin Liu, Lingjun Liu, Zhiqi Lin, Mingxuan Wang, Chengyi Wang, Xiangpeng Wei, Wenyuan Xu, et al. Seed1. 5-thinking: Advancing superb reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.13914*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. DeepSeekMath. *arXiv preprint arXiv:2402.03300*, 2024.

Xuyang Shen, Dong Li, Ruitao Leng, Zhen Qin, Weigao Sun, and Yiran Zhong. Scaling laws for linear complexity language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16377–16426, 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.

Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. Can llms generate novel research ideas? a large-scale human study with 100+ nlp researchers. *arXiv preprint arXiv:2409.04109*, 2024.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. Deltaproduct: Improving state-tracking in linear rnns via householder products. *arXiv preprint arXiv:2502.10297*, 2025.

Ved Sirdeshmukh, Kaustubh Deshpande, Johannes Mols, Lifeng Jin, Ed-Yeremai Cardona, Dean Lee, Jeremy Kritz, Willow Primack, Summer Yue, and Chen Xing. Multichallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier llms. *arXiv preprint arXiv:2501.17399*, 2025.

Weigao Sun, Disen Lan, Tong Zhu, Xiaoye Qu, and Yu Cheng. Linear-moe: Linear sequence modeling meets mixture-of-experts. *arXiv preprint arXiv:2503.05447*, 2025.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Tencent AI Lab. Hunyuan-t1: Reasoning efficiency redefined. https://llm.hunyuan.tencent.com/#/Blog/hy-t1/, 2025. Accessed: 2025-06-15.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Johannes von Oswald, Nino Scherrer, Seijin Kobayashi, Luca Versari, Songlin Yang, Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling, Alexander Meulemans, et al. Mesanet: Sequence modeling by locally optimal test-time training. *arXiv preprint arXiv:2506.05233*, 2025.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*, 2024.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying llm-based software engineering agents. *arXiv preprint arXiv:2407.01489*, 2024.

Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. Theagentcompany: Benchmarking llm agents on consequential real world tasks. *arXiv preprint arXiv:2412.14161*, 2025.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2024a.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *arXiv preprint arXiv:2406.06484*, 2024b.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

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.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big Bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025.

Yu Zhang, Songlin Yang, Rui-Jie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, et al. Gated slot attention for efficient linear-time sequence modeling. *Advances in Neural Information Processing Systems*, 37:116870–116898, 2024.

A. Contributors

The contributors to the report are listed in alphabetical order as follows:

Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, Chengjun Xiao, Chengyu Du, Chi Zhang, Chu Qiao, Chunhao Zhang, Chunhui Du, Congchao Guo, Da Chen, Deming Ding, Dianjun Sun, Dong Li, Enwei Jiao, Haigang Zhou, Haimo Zhang, Han Ding, Haohai Sun, Haoyu Feng, Huaiguang Cai, Haichao Zhu, Jian Sun, Jiaqi Zhuang, Jiaren Cai, Jiayuan Song, Jin Zhu, Jingyang Li, Jinhao Tian, Jinli Liu, Junhao Xu, Junjie Yan, Junteng Liu, Junxian He, Kaiyi Feng, Ke Yang, Kecheng Xiao, Le Han, Leyang Wang, Lianfei Yu, Liheng Feng, Lin Li, Lin Zheng, Linge Du, Lingyu Yang, Lunbin Zeng, Minghui Yu, Mingliang Tao, Mingyuan Chi, Mozhi Zhang, Mujie Lin, Nan Hu, Nongyu Di, Peng Gao, Pengfei Li, Pengyu Zhao, Qibing Ren, Qidi Xu, Qile Li, Qin Wang, Rong Tian, Ruitao Leng, Shaoxiang Chen, Shaoyu Chen, Shengmin Shi, Shitong Weng, Shuchang Guan, Shuqi Yu, Sichen Li, Songquan Zhu, Tengfei Li, Tianchi Cai, Tianrun Liang, Weiyu Cheng, Weize Kong, Wenkai Li, Xiancai Chen, Xiangjun Song, Xiao Luo, Xiao Su, Xiaobo Li, Xiaodong Han, Xinzhu Hou, Xuan Lu, Xun Zou, Xuyang Shen, Yan Gong, Yan Ma, Yang Wang, Yiqi Shi, Yiran Zhong, Yonghong Duan, Yongxiang Fu, Yongyi Hu, Yu Gao, Yuanxiang Fan, Yufeng Yang, Yuhao Li, Yulin Hu, Yunan Huang, Yunji Li, Yunzhi Xu, Yuxin Mao, Yuxuan Shi, Yuze Wenren, Zehan Li, Zelin Li, Zhanxu Tian, Zhengmao Zhu, Zhenhua Fan, Zhenzhen Wu, Zhichao Xu, Zhihang Yu, Zhiheng Lyu, Zhuo Jiang, Zibo Gao, Zijia Wu, Zijian Song, Zijun Sun