Fast Quiet-STaR: Thinking Without Thought Tokens

Wei Huang[1]*, Yizhe Xiong[2,3]*, Xin Ye[4]*, Zhijie Deng[5],
Hui Chen[2,3]†, Zijia Lin[2]†, Guiguang Ding[2,3]

[1]School of Computer Science, Beijing University of Posts and Telecommunications,
[2]Tsinghua University,
[3]Beijing National Research Center for Information Science and Technology (BNRist),
[4]Kuaishou Technology, [5]Shanghai Jiao Tong University, Shanghai, China
Correspondence†: jichenhui2012@gamil.com, linzijia07@tsinghua.org.cn

Abstract

Large Language Models (LLMs) have achieved impressive performance across a range of natural language processing tasks. However, recent advances demonstrate that further gains—particularly in complex reasoning tasks—require more than merely scaling up model sizes or training data. One promising direction is to enable models to "think" during the reasoning process. Recently, Quiet-STaR significantly improves reasoning by generating token-level thought traces, but incurs substantial inference overhead. In this work, we propose Fast Quiet-STaR, a more efficient reasoning framework that preserves the benefits of token-level reasoning while reducing computational cost. Our method introduces a curriculum-learning-based training strategy that gradually reduces the number of thought tokens, enabling the model to internalize more abstract and concise reasoning processes. We further extend this approach to the standard Next Token Prediction (NTP) setting through reinforcement learning-based fine-tuning, resulting in Fast Quiet-STaR NTP, which eliminates the need for explicit thought token generation during inference. Experiments on four benchmark datasets with Mistral 7B and Qwen2.5 7B demonstrate that Fast Quiet-STaR consistently outperforms Quiet-STaR in terms of average accuracy under the same inference time budget. Notably, Fast Quiet-STaR NTP achieves an average accuracy improvement of 9% on Mistral 7B and 5.7% on Qwen2.5 7B, while maintaining the same inference latency. Our code will be available at https://github.com/huangwei200012/Fast-Quiet-STaR.

1 Introduction

Artificial intelligence has made remarkable progress in recent years (Xiong et al., 2023), particularly with large language models (LLMs)

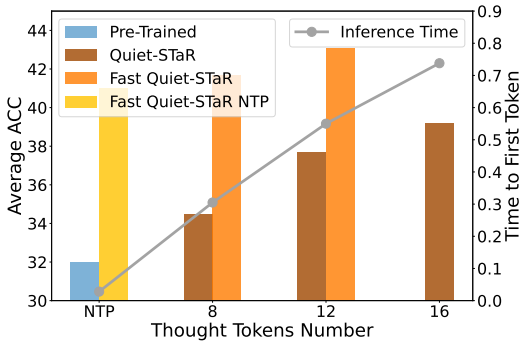

Figure 1: Performance comparison between Fast Quiet-STaR, Quiet-STaR and the pre-trained model (NTP). "Inference Time" represents the Time-to-First-Token (TTFT) of each model variant. Note that with the same number of thought tokens, Fast Quiet-STaR shares the same inference time with Quiet-STaR, but enjoys a significant performance boost. Additionally, Fast Quiet-STaR can be extended to the NTP setting, improving model performance without additional inference time overhead.

(Achiam et al., 2023; Grattafiori et al., 2024a) through pre-training models with billions of parameters on massive datasets. However, merely scaling up model size or increasing the amount of training data is insufficient for enabling strong performance on tasks that require complex reasoning or long-term planning. To further enhance model capabilities, one promising direction is to enable models to engage in autonomous "thinking" before producing final answers. Recently, a growing body of research has explored this paradigm to strengthen the reasoning abilities of LLMs. Notably, models such as OpenAI o1 (OpenAI et al., 2024), DeepSeek-R1 (Guo et al., 2025), QwQ (Zheng et al., 2024), and Kimi-1.5 (Team et al., 2025) have demonstrated impressive performance across a variety of challenging tasks, such as mathematical competition problems (Hendrycks et al., 2021; Cobbe et al., 2021).

Recently, Quiet-STaR (Quiet Self-Taught Rea-

*These authors contributed equally to this work.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 18771–18781

soner) (Zelikman et al., 2024) has been proposed as a novel reasoning paradigm that shifts the thinking process from the problem level to a finer, token-level granularity. In Quiet-STaR, before predicting the next token, the model first generates a sequence of intermediate thought trace (represented as <|*start_of_thought*|>,*thought_token1, thought_token2, ...* <|*end_of_thought*|>), based on which the model predicts the next token. Compared to other approaches, Quiet-STaR can significantly enhance the model's reasoning ability through a lightweight unsupervised training process. For instance, it achieves a 10% performance gain on CommonsenseQA (Talmor et al., 2018) for the Mistral 7B (Jiang et al., 2023) by continue pre-training with only **0.2M** tokens, demonstrating remarkable improvements through efficient training.

Although Quiet-STaR significantly enhances the model's reasoning capabilities, it substantially increases inference overhead due to the requirement of generating a thought trace for every token. As shown in Figure 1, even when using only 8 thought tokens, the average Time-To-First-Token (TTFT) of Quiet-STaR remains over **10 times higher** than that of conventional next token prediction (NTP) models. Despite the high inference costs, these thought tokens cannot be directly reduced or eliminated as they are the main contributor to performance improvements. For example, as shown in Figure 1, halving the number of thought tokens from 16 to 8 leads to a 4.7% accuracy drop. Such dilemma on efficiency severely undermines the practical value of Quiet-STaR.

Prior work has shown that LLMs are capable of skipping reasoning steps by omitting non-essential steps without sacrificing overall reasoning performance (Liu et al., 2024). Inspired by this, we believe that within the Quiet-STaR reasoning paradigm, **the model can maintain its strong reasoning abilities obtained from long thought-trace training by compressing the number of thought tokens and keeping only a more abstract thought trace**. To improve the efficiency of the Quiet-STaR reasoning paradigm with minimal performance degradation, we propose **Fast Quiet-STaR**. We employ a multi-stage training strategy that progresses from easy to hard. That is, we gradually guide the model from generating a detailed thought trace using more thought tokens to generating a concise thought trace using fewer thought tokens. To further accelerate the Quiet-STaR inference paradigm to NLP-level efficiency, we employ a reinforcement learning-based fine-tuning strategy for Fast Quiet-STaR model under the NTP setting. The resulting Fast Quiet-STaR NTP model preserves the original thinking abilities of Fast Quiet-STaR while eliminating reliance on generating explicit thought trace during inference.

We evaluate our method on two open-source models, Mistral 7B (Jiang et al., 2023) and Qwen2.5 7B (Qwen et al., 2025) across four public datasets. Extensive experiments show that, given the same number of thought tokens(same inference time), Fast Quiet-STaR achieves substantial performance gains over Quiet-STaR. Furthermore, under equivalent inference time, Fast Quiet-STaR NTP improves the average accuracy by 9% on Mistral 7B and 5.7% on Qwen2.5 7B compared to the original pre-trained models.

We summarize our contribution as follows:

- We propose Fast Quiet-STaR, a novel training paradigm that compresses token-level thought traces to significantly reducing inference overhead while preserving the strong reasoning abilities imparted by the Quiet-STaR framework.
- We introduce a curriculum learning-based multi-stage training strategy that progressively guides the model to learn a more concise thought trace, enabling it to internalize efficient reasoning patterns and express them compactly without performance degradation. We further accelerate Fast Quiet-STaR to the standard NTP-level setting via reinforcement learning-based fine-tuning, enabling implicit reasoning without explicit thought token generation.
- Extensive experiments show that Fast Quiet-STaR achieves comparable or even better performance than standard Quiet-STaR while reducing thought tokens. Fast Quiet-STaR NTP significantly outperforms the pre-trained model without increasing the inference time.

2 Related Works

2.1 LLM Reasoning

In recent years, enhancing the reasoning capabilities of large language models has become a major research focus (Rajani et al., 2019; Zhang et al., 2025; Pan et al., 2025). The Chain-of-Thought

(CoT) (Kojima et al., 2022) prompting technique explicitly guides models to generate intermediate reasoning steps. Tree of Thoughts (ToT) (Yao et al., 2023) explores multiple reasoning paths through a tree-structured search. The CPO (Zhang et al., 2024) method combines ToT with Direct Preference Optimization (DPO) (Rafailov et al., 2023), using reasoning paths generated by ToT as paired training data to directly optimize the model's CoT abilities. Self-Consistency (Wang et al., 2022) samples multiple reasoning paths for the same problem and selects the final answer through a voting mechanism. Methods based on Monte Carlo Tree Search (MCTS) (Qi et al., 2024) introduce classical planning algorithms into the reasoning process. Coconut (Hao et al., 2024) explores the potential of unconstrained reasoning in latent spaces, highlighting the structural thinking capabilities of LLMs.

In the latest research, reinforcement learning (RL) (Schulman et al., 2017) has emerged as a new paradigm for enhancing LLM reasoning. OpenAI's o1 (OpenAI et al., 2024) achieves significant improvements in reasoning performance. Similarly, models such as DeepSeek-R1 (Guo et al., 2025), Kimi 1.5 (Team et al., 2025), and QWQ (Zheng et al., 2024) incorporate reinforcement learning (Shao et al., 2024) into pretrained models, exhibiting strong reasoning abilities.

Unlike most approaches that prompt models to "think before answering" on a per-question basis, Quiet-STaR (Zelikman et al., 2024) shifts the reasoning process to a finer-grained, token-level paradigm. By encouraging deep reasoning at every token generation step, Quiet-STaR further enhances reasoning quality. However, as it requires long-range reasoning at every token, it incurs substantial inference latency, which limits its applicability in real-world scenarios.

2.2 Curriculum Learning

Curriculum Learning is a training strategy that organizes the learning process by first presenting simpler examples and gradually introducing more complex ones. In recent years, curriculum learning has been widely adopted in the training of Large Language Models (LLMs) (Xu et al., 2020; Naïr et al., 2024). LDCAL (Li et al., 2024) leverages LLMs themselves to assess the difficulty of training instances, guiding the model to learn in an easy-to-hard sequence. TAPIR (Yue et al., 2024) constructs a task-aware curriculum scheduling framework that dynamically adjusts the task distribution and progressively increases task complexity. Moreover, curriculum learning has also been employed to improve LLMs' understanding of long contexts by gradually increasing the context window size during training (Grattafiori et al., 2024b). Kimi 1.5 (Team et al., 2025) integrates a curriculum learning strategy during the reinforcement learning stage, allowing the model to start with simpler question before transitioning to more complex ones.

Unlike existing studies that apply curriculum learning at the data or task scheduling level, our approach integrates curriculum learning into the token-level reasoning process. By combining this strategy with the Quiet-STaR inference paradigm, our Fast Quiet-STaR better learns reasoning behaviors under limited Thought Tokens.

3 Methodology

The training procedure of Fast Quiet-STaR is illustrated in Figure 2. Building upon Quiet-STaR, we propose a progressive, multi-stage training framework inspired by the principles of curriculum learning. This approach facilitates a gradual transition from easy to hard reasoning paradigms. In particular, during the final stage of training, we incorporate reinforcement learning to transition the reasoning paradigm of Quiet-STaR to the the standard NTP paradigm.

3.1 Quiet-STaR

Quiet-STaR (Zelikman et al., 2024) is a method for enabling language models to autonomously learn to generate internal rationales—referred to as "thoughts"—in order to improve their ability to predict future tokens.The training process consisting of three distinct phases—Think, Talk, and Learn.

3.1.1 Think Process

Given a token sequence $X = \{x_0, x_1, ..., x_t\}$, Quiet-STaR **n-m** (n and m represent the number of thought/ahead tokens, respectively) generates a corresponding thought of length **n-1**, i.e., $T_i = (t_{i1}, t_{i2}, ..., t_{i(n-1)})$, after each token x_i. Each thought is enclosed by learned meta-tokens <|*start_of_thought*|> and <|*end_of_thought*|>, which serve to activate and terminate the generation of thought, respectively. This process is executed in parallel using a custom attention mask, ensuring that each generated thought attends only to the corresponding prefix of the input sequence and the previously generated tokens within the same thought, denoted as:

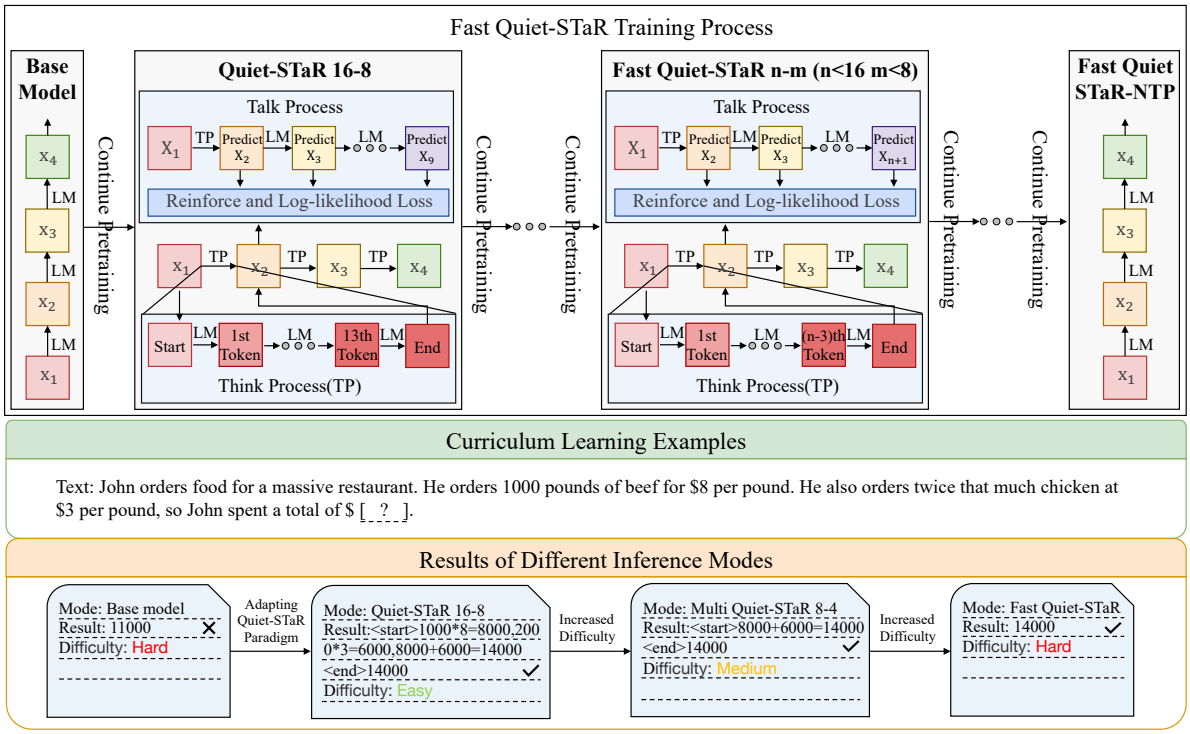

Figure 2: Fast Quiet-STaR training pipeline and Curriculum Learning Examples.

$P(t_{ij}|x_1, ..., x_{i-1}, \text{<|}start\_of\_thought\text{|>}, ..., t_{i(j-1)})$. Note that m will be introduced in the Learn Process section.

3.1.2 Talk Process

Quiet-STaR introduces a learnable interpolation mechanism. By employing a shallow MLP head to compute an interpolation weight w, conditioned on the hidden states of both the <|*end_of_thought*|> token and the original input tokens. This weight modulates the influence of post-thought logits on the final prediction. The resulting mixed log-probability is defined in Equation 1.

$$\log p_i^{\text{talk}} = w_i \log p_i^{\text{base}} + (1 - w_i) \log p_i^{\text{thought}} \tag{1}$$

Among them, p_i^{base} represents the logits before thought, and p_i^{thought} represents the logits after thought, w.r.t the token x_i.

3.1.3 Learn Process

Quiet-STaR leverages the REINFORCE algorithm (Phan et al., 2023) to optimize thought selection based on utility. It maximizes the log-likelihood of the next **m** (the number of ahead tokens) ground-truth tokens $X_{j+1:j+m+1}$ given prior context and a candidate rationale T_j. To reduce variance, multiple rationale continuations are sampled per token. The reward r_j for each T_j is defined as the difference between its log-likelihood $\log p_{j:j+m}^{\text{talk}}$ and the mean log-likelihood across all sampled rationales (see Eq. 2). Quiet-STaR incorporates this reward

$$r_j = \log p_{j:j+m}^{\text{talk}}(X_{j+1:j+m+1}) - \log \bar{p}_{j:j+m}^{\text{talk}}(X_{j+1:j+m+1}) \tag{2}$$

into a REINFORCE loss to update the model parameters θ, encouraging thoughts that exceed the average, as shown in Equation 3. Additionally, Quiet-STaR includes a log-likelihood loss term, denoted as L_i^{NLL}, to ensure that the model not only learns to optimize the talking-head but also continues to receive next-token prediction signals for the base language model head.

$$\nabla_\theta \mathcal{L}_j^{\text{REINFORCE}} = -r_j \cdot \nabla_\theta \log p_\theta(T_j \mid [X_{:j}; \text{<|}start\_of\_thought\text{|>}]) \tag{3}$$

3.2 Fast Quiet-STaR

3.2.1 Fast Quiet-STaR

Compared with the mainstream NTP reasoning paradigm, Quiet-STaR introduces a new reasoning mechanism of "think first, talk later" for each token. Since it allows thinking, this mechanism effectively reduces the difficulty of predicting the next token, making the model perform better. Within the Quiet-STaR framework, a key hyperparameter, n, denotes

the number of thought tokens, which has a significant impact on model performance. As illustrated in Figure 1, the model exhibits stronger reasoning abilities when more thought tokens are provided, while its performance degrades noticeably as the number of thought tokens decreases.

This phenomenon raises a critical question: **why does a reduced number of thought tokens significantly impair the performance of Quiet-STaR, and how can we minimize the use of thought tokens without compromising model performance?** To investigate this, we analyze the thought traces under varying numbers of thought tokens. As shown in the lower part of Figure 2, reducing the number of thought tokens forces the model to complete the reasoning process within a shorter sequence. This poses greater demands on the model's ability to compress its reasoning steps, presenting a more challenging inference setting. In contrast to learning under easier setting (with more thought tokens), directly training the model on more difficult ones (with fewer thought tokens) proves less effective. This observation aligns with a core insight from curriculum learning: models often struggle to learn effectively when exposed to high-difficulty tasks early in training. Therefore, adopting a curriculum learning strategy, which progressively trains Quiet-STaR from easier settings to harder ones, holds promise for reasoning performance with limited thought tokens.

Based on the above observations, we adopt a curriculum learning strategy to facilitate the acquisition of reasoning paradigms and propose Fast Quiet-STaR approach. This method decomposes the training process into multiple stages, each aligned with a specific level of reasoning difficulty and corresponding modeling objective.

In the initial stage, the model is trained with a larger number of thought tokens. This setting represents a easy reasoning setting. As training progresses, we gradually reduce the number of thought tokens, thereby encouraging the model to engage in more concise and abstract reasoning under increasingly constrained resources. This encourages the model to progressively adapt to more difficult reasoning setting, enhancing both its reasoning efficiency and its generalization capabilities in lower-resource scenarios. Specifically, we begin with 16 thought tokens and 8 ahead tokens (16-8). During training, we gradually reduce it to 12-4 and 8-4.

3.2.2 Fast Quiet-STaR NTP

Although the number of thought tokens has been reduced, inference based on the Quiet-STaR paradigm still requires significantly higher computational resources compared to the NTP approach. To address this issue, we adopt reinforcement learning (Phan et al., 2023) to transition the model's inference paradigm from Fast Quiet-STaR to NTP, called Fast Quiet-STaR NTP. Specifically, we initialize an NTP model using the checkpoint obtained from the last stage of the multi-stage training that includes 8 thought tokens and 4 ahead tokens. The log-likelihood loss **after thinking** of this checkpoint serves as a reference for computing rewards in reinforcement learning. The process of calculating reward is as follows:

$$r_j = \mathcal{L}_{FastQuietSTaR} - \mathcal{L}_{FastQuietSTaR-NTP} \tag{4}$$

where $\mathcal{L}_{FastQuietSTaR}$ represents the negative log-likelihood loss of Fast Quiet-STaR 8-4 at the jth token after a thinking process, and $\mathcal{L}_{FastQuietSTaR-NTP}$ represents the negative log-likelihood loss of Fast Quiet-STaR NTP at the jth token. The final loss function is as follows:

$$\nabla_\theta \mathcal{L}_j^{\text{REINFORCE}} = -r_j \cdot \nabla_\theta \log p_\theta \left(x_j \mid X_{:j}\right) \tag{5}$$

Through reinforcement learning, the model is encouraged to emulate the prediction quality of Fast Quiet-STaR model without explicitly generating intermediate reasoning tokens during inference. Notably, this transition enables Fast Quiet-STaR NTP to effectively internalize the reasoning process, compressing and integrating the previously explicit "thinking" into its latent representations.

4 Experiments

4.1 Experimental Settings

Post-Training Settings. We perform post-training on Mistral 7B (Jiang et al., 2023) and Qwen2.5 7B (Qwen et al., 2025) using the OpenWebMath dataset (Paster et al., 2023) and evaluate its ability to directly predict answers on the CommonsenseQA (Talmor et al., 2018) and GSM8K (Cobbe et al., 2021) benchmarks. Following (Zelikman et al., 2024), we calculate the accuracy rate as: $ACC = \frac{\prod_{i=0}^{l} P(A_i|Q_1,...,Q_k,A_i,...A_{i-1})}{\prod_{i=0}^{l}(\sum_{A_j \in S_{ans}}(P(A_j|Q_1,...,Q_k,A_i,...A_{i-1})))}$, where Q_i represents the question token, A_i represents the answer token, k and l represent their

Table 1: Performance (%) comparison. **Bold** and underline denote the best and second-best performance of models. For each method, we report their time to first token (TTFT, in seconds). Performance Δ represents the difference between Fast Quiet-STaR NTP and Pre-Trained.

Method	Thought Tokens	Ahead Tokens	TTFT (s)	PIQA	SIQA	CommonsenseQA	GSM8K	AVG
Mistral-7B								
Pre-Trained	1	1	**0.028**	45.9	41.6	35.4	4.9	32.0
Quiet-STaR	16	8	0.738	54.7	47.0	45.3	9.8	39.2
Quiet-STaR	12	4	0.550	53.1	45.7	43.4	8.4	37.7
Fast Quiet-STaR	12	4	0.550	**59.0**	**52.5**	**50.7**	**10.0**	**43.1**
Quiet-STaR	8	4	0.305	49.1	42.2	39.3	7.2	34.5
Fast Quiet-STaR	8	4	0.305	56.9	51.1	49.0	9.8	41.7
Fast Quiet-STaR-NTP	1	1	**0.028**	55.0	50.1	49.2	9.6	41.0
Performance Δ	-	-	-	**+9.1**	**+8.5**	**+13.8**	**+4.7**	**+9.0**
Qwen2.5-7B								
Pre-Trained	1	1	**0.026**	70.1	60.7	52.4	11.6	48.7
Quiet-STaR	16	8	0.633	**77.6**	**68.1**	**66.5**	**17.7**	**57.5**
Quiet-STaR	12	4	0.481	72.4	61.8	59.7	17.1	52.8
Fast Quiet-STaR	12	4	0.481	74.3	64.5	63.9	17.6	55.1
Quiet-STaR	8	4	0.269	70.2	60.3	54.9	11.9	49.3
Fast Quiet-STaR	8	4	0.269	74.5	63.4	59.3	16.9	53.5
Fast Quiet-STaR-NTP	1	1	**0.026**	74.9	65.8	60.3	16.5	54.4
Performance Δ	-	-	-	**+4.8**	**+5.1**	**+7.9**	**+4.9**	**+5.7**

Table 2: Comparison of generation latency(in seconds) between different methods. For prefix length 256 and generate length 128, we use a prompt of 256 tokens and let the model generate 128 tokens after the prompt. AVG ACC represents the average accuracy on PIQA, SIQA, CommonsenseQA and GSM8K.

Prefix Length	256		512		AVG
Generate Length	128	256	256	512	ACC
Pre-Trained	**3.2**	**7.3**	**8.8**	**17.1**	**32.0**
Quiet-STaR 16-8	52.7	116.9	167.0	326.4	39.2
Quiet-STaR 12-4	40.6	92.9	102.4	288.6	37.7
Fast Quiet-STaR 12-4	40.6	92.9	102.4	288.6	43.1
Quiet-STaR 8-4	33.0	65.9	82.4	184.4	34.5
Fast Quiet-STaR 8-4	33.0	65.9	82.4	184.4	41.7
Fast Quiet-STaR-NTP	**3.2**	**7.3**	**8.8**	**17.1**	**41.0**

lengths, and S_{ans} represents the candidate set of answers (e.g. $S_{ans} = \{A, B, C, D, E\}$ for CommonsenseQA). This evaluation and training setup is consistent with the Quiet-STaR (Zelikman et al., 2024). To further assess the effectiveness of Fast Quiet-STaR in general reasoning tasks, we also introduce two more general-purpose evaluation benchmarks: SIQA (Sap et al., 2019) and PIQA (Bisk et al., 2020).

Implementation details. All training experiments are conducted on 8 H800 GPUs. For Quiet-STaR, we train for 100 steps, and for Fast Quiet-STaR, we select the last checkpoint in the previous stage for initialization and train for another 50 steps for each training stage. See Appendix A for more details.

4.2 Main Results

We evaluate Quiet-STaR, Fast Quiet-STaR, and Fast Quiet-STaR NTP on four benchmarks: PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), CommonsenseQA (Talmor et al., 2018), and GSM8K (Cobbe et al., 2021) (Table 1). Under **equal TTFT**, Fast Quiet-STaR consistently outperforms Quiet-STaR, exhibiting stable performance even as the number of thought tokens decreases—unlike Quiet-STaR, which degrades significantly. For Mistral 7B, multi-stage training further boosts performance: Fast Quiet-STaR with 8 tokens surpasses the 16-token variant by 1.8% while cutting inference time to 41.3%. Compared to pre-trained baselines, Fast Quiet-STaR NTP achieves notable gains without added compute, improving average accuracy by 9% on Mistral 7B and 5.7% on Qwen2.5 7B. These results validate the effectiveness of incorporating a curriculum learning strategy within the Quiet-STaR framework to simultaneously improve both model efficiency and performance.

We further analyze generation latency, a more general metric for evaluating speed. For prefix lengths of 256 and 512, the models generate 128/256 and 256/512 tokens, respectively (Table 2). Fast Quiet-STaR NTP significantly reduces latency, achieving just 6% of the end-to-end generation time of Quiet-STaR 16-8 (for 256-128), on par with the

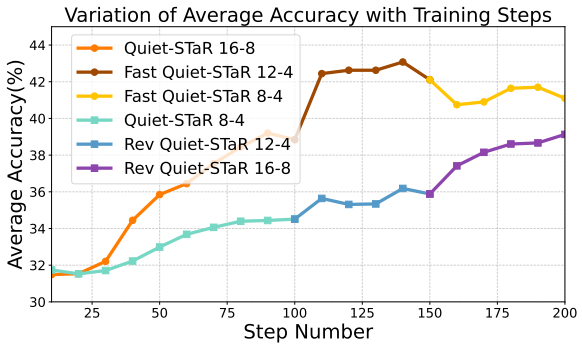

Figure 3: Comparison of the average accuracy of Fast Quiet-STaR and Rev Quiet-STaR with training steps.

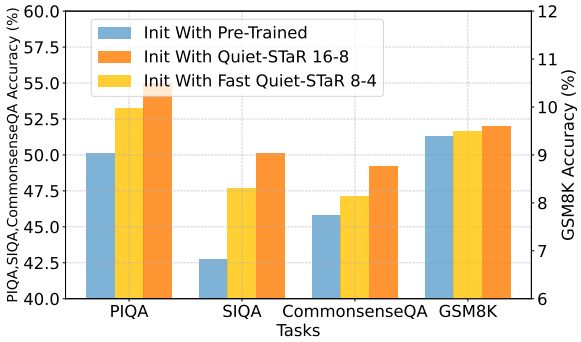

Figure 4: Comparison of Fast Quiet-STaR NTP under different initialization schemes.The left vertical axis corresponds to the average accuracy on PIQA, SIQA, and CommonsenseQA, while the right vertical axis indicates the accuracy on GSM8K.

pre-trained baseline. Additionally, it yields accuracy gains of 0.8% and 9%, respectively. These results highlight Fast Quiet-STaR NTP effectiveness in addressing both the latency of Quiet-STaR and the poor performance of standard pre-trained models.

4.3 Experimental Analysis

We choose Mistral 7B (Jiang et al., 2023) for our analytical experiments, which is consistent with Quiet-STaR(Zelikman et al., 2024).

4.3.1 Ablation Studies

Curriculum Learning. To evaluate the effectiveness of our easy-to-hard multi-stage curriculum learning training strategy, we experiment with an alternative where we reverse the entire training process. Specifically, we start from the Quiet-STaR 8-4 model and follow a "8-4 → 12-4 → 16-8" training sequence. At each stage, the model is initialized with the weights obtained from the previous stage. We refer to this series of progressively trained models as Rev Quiet-STaR. We compare the average performance of Rev Quiet-STaR and Fast Quiet-STaR across four benchmarks: PIQA, SIQA, CommonsenseQA, and GSM8K (Figure 3). Experimental results indicate that the multi-stage training method, progressing from difficult to easier, does not lead to performance improvements. Notably, Rev Quiet-STaR 16-8 even underperforms Fast Quiet-STaR 8-4, despite utilizing a larger number of thought tokens.

Reinforcement Learning Initialization. To study the impact of initialization, we compare Fast Quiet-STaR 8-4 with two alternatives: the pre-trained model and Quiet-STaR 16-8. We evaluate all approaches on four benchmarks—PIQA, SIQA, CommonsenseQA, and GSM8K—summarized in Figure 4. Results show that Fast Quiet-STaR 8-4 yields the best performance, followed by Quiet-STaR 16-8, and then the pre-trained model. We attribute this to Fast Quiet-STaR 8-4's ability to generate a compact yet informative thought trace, which is conducive to further improving efficiency and expanding the reasoning paradigm to NTP. In contrast, the pre-trained model lacks an explicit thought trace prior; Quiet-STaR 16-8 provides detailed thought traces, which rely on a longer reasoning process, which may lead to a larger span of reasoning paradigm difficulty when learning, thus affecting the overall training performance.

Table 3: Performance comparison between Quiet-STaR NTP and Fast Quiet-STaR NTP. CSQA represents for CommonsenseQA, and Performance Δ represents the difference between Fast Quiet-STaR NTP and Quiet-STaR NTP.

Method	PIQA	SIQA	CSQA	GSM8K	AVG
Quiet-STaR NTP	49.1	44.3	42.5	7.3	38.1
Fast Quiet-STaR NTP	55.0	50.1	49.2	9.6	41.0
Performance Δ	**+5.9**	**+5.8**	**+6.7**	**+2.3**	**+5.2**

Fast Quiet-STaR NTP Without Curriculum Learning. To evaluate the effectiveness of the curriculum learning procedure "16-8 → 12-4 → 8-4 → NTP", we omit the intermediate stages. Specifically, we directly initialize the pre-trained model with Quiet-STaR 16-8 and use its log-likelihood loss as the reference for computing rewards in reinforcement learning, resulting in Quiet-STaR NTP. As shown in Table 3, this shortcut results in a 5.2% drop in average accuracy compared to Fast Quiet-STaR NTP obtained through the full curriculum. These results underscore the critical role of the curriculum learning process in enhancing the overall

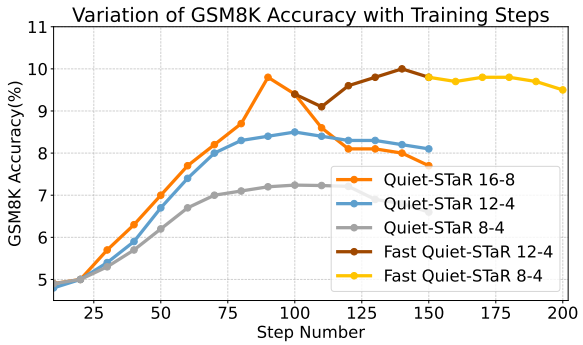

Figure 5: Comparison of the accuracy of Quiet-STaR and Fast Quiet-STaR on GSM8K during training.

Table 4: Zero-shot performance on Fast Quiet-STaR and Pre-Trained applied to chain-of-thought on GSM8K. Performance Δ represents the difference between Fast Quiet-STaR NTP and Pre-Trained.

Method	maj@2	maj@3	maj@4	maj@5	maj@6
Pre-Trained	28.5	32.6	37.5	40.3	43.3
Fast Quiet-STaR NTP	36.0	40.6	45.8	49.2	52.4
Performance Δ	**+7.5**	**+8.0**	**+8.3**	**+8.9**	**+9.1**

performance of the model.

4.3.2 Data Efficiency

To ensure that Fast Quiet-STaR's performance gains do not stem from an increase in data volume, we track GSM8K accuracy throughout training (Figure 5). Quiet-STaR trains for 150 steps, with performance peaking around step 100 and declining thereafter—consistent with prior findings (Zelikman et al., 2024). In contrast, Fast Quiet-STaR achieves strong results with just 20–40 additional steps. These results suggest that the gains arise from the progressive learning mechanism of multi-stage training, not from greater data exposure.

4.3.3 Performance on Generation Tasks

To evaluate the performance of Fast Quiet-STaR on generative tasks, we compare Fast Quiet-STaR NTP with the original pre-trained model under the Next Token Prediction (NTP) inference paradigm on the GSM8K dataset. Specifically, we adopt the Chain-of-Thought (CoT) reasoning approach and measure accuracy using majority voting over 6 samples (cot-maj@6), with results as show in table 4. Experimental results show that as the number of votes increases, the performance advantage of Fast Quiet-STaR NTP over the pretrained model becomes more pronounced. On the cot-maj@6 metric, Fast Quiet-STaR NTP achieves an accuracy improvement from 43.3% to 52.4%, demonstrating its effectiveness in complex reasoning tasks. These results demonstrate that Fast Quiet-STaR can further enhance inference performance on top of CoT reasoning and **Fast Quiet-STaR is complementary to CoT**, rather than redundant.

4.3.4 Analysis of long-term inference savings

The training process of Fast Quiet-STaR is highly efficient. The entire training pipeline (from Quiet-STaR 16-8 → Fast Quiet-STaR 12-4 → Fast Quiet-STaR 8-4 → Fast Quiet-STaR NTP) requires only 0.5M tokens for continue pre-training. This process takes just 54 minutes on 8 H800 GPUs. We use prompts containing 256 tokens and allow the model to generate 128 tokens for demonstration and analysis purposes. On a single H800 GPU, the inference latency of Fast Quiet-STaR-NTP is 3.2 seconds, compared to 52.7 seconds for Quiet-STaR 16-8(more scenarios with inference latency are in Table (**?**)). For 67 end-to-end inference runs, Quiet-STaR 16-8 requires approximately 59 minutes in total, whereas Fast Quiet-STaR-NTP completes the same task in just 4 minutes—yielding an overall speedup of 55 minutes. In other words, the time saved from just 67 inference runs is sufficient to offset the entire training cost of Fast Quiet-STaR-NTP. More importantly, in real-world deployments, models are typically required to perform millions of inference runs—far exceeding 67—where our approach would demonstrate substantial advantages in total inference cost.

4.3.5 Thought token analysis

We visualize the thought tokens generated by Quiet-STaR 8-4 and Fast Quiet-STaR 8-4 at key positions—tokens most informative for final predictions—on the GSM8K dataset to examine their internal reasoning behavior. As shown in Figure 6, Quiet-STaR 8-4 produces relatively unstructured thoughts, indicating incomplete acquisition of the Quiet-STaR reasoning paradigm. In contrast, Fast Quiet-STaR 8-4 demonstrates more abstract and goal-directed reasoning behavior. These observations indicate that the incorporation of a curriculum learning strategy—progressing from easier to more difficult—enables Fast Quiet-STaR to gradually acquire the ability to perform effective reasoning under resource constraints.

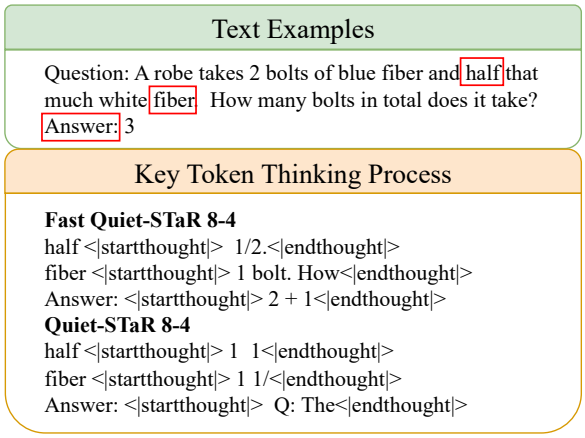

Figure 6: Examples of the text and its thought process at key tokens.

5 Conclusion

In this paper, we proposed Fast Quiet-STaR, an efficient extension of the Quiet-STaR reasoning paradigm that maintains the core benefits of fine-grained token-level reasoning while significantly reducing inference overhead. By leveraging a curriculum learning-based training strategy that progressively reduces the number of thought tokens, Fast Quiet-STaR enables models to develop compact yet effective reasoning abilities. Furthermore, through reinforcement learning-based fine-tuning, we extend this paradigm to the standard Next Token Prediction setting, eliminating the need for explicit thought-token generation during inference. Experiments on Mistral 7B and Qwen2.5 7B across four benchmark datasets show that Fast Quiet-STaR achieves substantial gains over Quiet-STaR under the same number of thought tokens, and Fast Quiet-STaR NTP outperforms the pre-trained model and performs on par with Quiet-STaR. These results highlight Fast Quiet-STaR as a practical solution for enhancing reasoning capabilities in LLMs.

Limitations

Despite the notable improvements in inference efficiency and performance achieved by Fast Quiet-STaR, several limitations remain. First, the evaluation in this study primarily focuses on mathematical and logical reasoning tasks, leaving its generalization capability to other domains yet to be thoroughly validated. Second, this method is only for the Quiet-STaR reasoning method.

Acknowledgments

This work was supported by National Natural Science Foundation of China (Nos. 62525103, 62271281, 62441235).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024a. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024b. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Dongyuan Li, Ying Zhang, Zhen Wang, Shiyin Tan, Satoshi Kosugi, and Manabu Okumura. 2024. Active learning for abstractive text summarization via llm-determined curriculum and certainty gain maximization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8959–8971.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2024. Can language models learn to skip steps? *arXiv preprint arXiv:2411.01855*.

Marwa Naïr, Kamel Yamani, Lynda Said Lhadj, and Riyadh Baghdadi. 2024. Curriculum learning for small code language models. *arXiv preprint arXiv:2407.10194*.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, and 244 others. 2024. Openai o1 system card. *Preprint*, arXiv:2412.16720.

Rui Pan, Yinwei Dai, Zhihao Zhang, Gabriele Oliaro, Zhihao Jia, and Ravi Netravali. 2025. Specreason: Fast and accurate inference-time compute via speculative reasoning. *arXiv preprint arXiv:2504.07891*.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. 2023. Openwebmath: An open dataset of high-quality mathematical web text. *arXiv preprint arXiv:2310.06786*.

Du Phan, Matthew Douglas Hoffman, David Dohan, Sholto Douglas, Tuan Anh Le, Aaron Parisi, Pavel Sountsov, Charles Sutton, Sharad Vikram, and Rif A Saurous. 2023. Training chain-of-thought via latent-variable inference. *Advances in Neural Information Processing Systems*, 36:72819–72841.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. *arXiv preprint arXiv: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, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. *arXiv preprint arXiv:1906.02361*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yizhe Xiong, Hui Chen, Zijia Lin, Sicheng Zhao, and Guiguang Ding. 2023. Confidence-based visual dispersal for few-shot unsupervised domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11621–11631.

Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. 2020. Curriculum learning for natural language understanding. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 6095–6104.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Yuanhao Yue, Chengyu Wang, Jun Huang, and Peng Wang. 2024. Distilling instruction-following abilities of large language models with task-aware curriculum planning. *arXiv preprint arXiv:2405.13448*.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. 2024.

Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*.

Jinghan Zhang, Xiting Wang, Fengran Mo, Yeyang Zhou, Wanfu Gao, and Kunpeng Liu. 2025. Entropy-based exploration conduction for multi-step reasoning. *arXiv preprint arXiv:2503.15848*.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. 2024. Chain of preference optimization: Improving chain-of-thought reasoning in llms. *Advances in Neural Information Processing Systems*, 37:333–356.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*.

A Implementation details

Our experimental settings are consistent with Quiet-STaR (Zelikman et al., 2024). Specifically, we employ the AdamW optimizer with a warm-up step count of 20, a weight decay of 0.001, and a batch size of 8. For Quiet-STaR, we train for 100 steps, and for Fast Quiet-STaR, we select the last checkpoint in the previous stage for initialization and train for 50 steps for each stage. The learning rates are adjusted slightly depending on the model: we use a learning rate of 1e-6 for Mistral (Jiang et al., 2023) and 8e-6 for Qwen2.5 (Qwen et al., 2025). During training, we perform sampling with a temperature of T=1. For evaluation, we adopt greedy decoding to ensure deterministic outputs. All training experiments are conducted on eight H800 GPUs. For measuring the Time to First Token (TTFT), we utilize a single H800 GPU and fix the context length to 256. TTFT is defined as the elapsed time between the moment the model receives the full input sequence and the generation of the first token.

We report the version numbers of used packages in Table 5.

Package	Version	Package	Version
PyTorch	2.1.0	transformers	4.46.0
deepspeed	0.10.0	tokenizers	0.13.3
datasets	2.14.3		

Table 5: Versions of used packages.

B License for Scientific Artifacts

The Open web math (Paster et al., 2023) is licensed under ODC-By 1.0 License[1]. The Mistral model (Jiang et al., 2023) and Qwen2.5 model (Qwen et al., 2025) is licensed under Apache License 2.0 license[2]. The evaluation datasets (Cobbe et al., 2021; Talmor et al., 2018; Sap et al., 2019; Bisk et al., 2020) are subject to the MIT license[3]. All usages of scientific artifacts in this paper obey the corresponding licenses.

[1]https://spdx.org/licenses/ODC-By-1.0.html
[2]https://choosealicense.com/licenses/apache-2.0/
[3]https://choosealicense.com/licenses/mit/