A STATE-TRANSITION FRAMEWORK FOR EFFICIENT LLM REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

While Long Chain-of-Thought (CoT) reasoning significantly improves Large Language Models (LLMs) performance on complex reasoning tasks, the substantial computational and memory costs of generating long CoT sequences limit their efficiency and practicality. Existing studies usually enhance the reasoning efficiency of LLMs by compressing CoT sequences. However, this approach conflicts with test-time scaling, limiting the reasoning capacity of LLMs. In this paper, we propose an efficient reasoning framework that models the reasoning process of LLMs as a *state-transition* process. Specifically, we first apply a linear attention mechanism to estimate the LLM's reasoning state, which records the historical reasoning information from previous reasoning steps. Then, based on the query prompt and the reasoning state, the LLM can efficiently perform the current reasoning step and update the state. With the linear attention, each token in the current reasoning step can directly retrieve relevant historical reasoning information from the reasoning state, without explicitly attending to tokens in previous reasoning steps. In this way, the computational complexity of attention is reduced from quadratic to linear, significantly improving the reasoning efficiency of LLMs. In addition, we propose a state-based reasoning strategy to mitigate the over-thinking issue caused by noisy reasoning steps. Extensive experiments across multiple datasets and model sizes demonstrate that our framework not only improves the reasoning efficiency of LLMs but also enhances their reasoning performance.

1 INTRODUCTION

Chain-of-Thought (CoT) (Wei et al., 2022; Kojima et al., 2022) has become a core technique for enhancing the reasoning ability of large language models (LLMs) on complex tasks. Through prompting step-by-step reasoning, CoT enables LLMs to decompose complex problems into simpler subtasks, thus improving their problem-solving capabilities (Yao et al., 2023; Wang et al., 2023; Zhou et al., 2023). Recent studies, including OpenAI o1 (OpenAI et al., 2024), QwQ (Team, 2024), and DeepSeek-R1 (DeepSeek et al., 2025), demonstrate that scaling up CoT length can further enhance the reasoning abilities of LLMs. However, since most current LLMs are built on the Transformer architecture (Vaswani et al., 2017b), the computational complexity of their attention grows quadratically with context length, and the memory overhead of their KV-cache increases linearly with context length. Hence, generating long CoT substantially increase the computational and memory cost of LLMs, limiting their practical efficiency on complex reasoning tasks.

To improve the reasoning efficiency of LLMs, previous studies employ prompting (Han et al., 2024; Ma et al., 2025a), supervised fine-tuning (SFT) (Liu et al., 2024; Munkhbat et al., 2025), or reinforcement learning (RL) (Aggarwal et al., 2025; Shen et al., 2025) to encourage LLMs toward generating shorter CoT sequences. However, these methods often impair the reasoning ability of LLMs (Jin et al., 2024; Merrill et al., 2024), since CoT shortening conflicts with test-time scaling (OpenAI et al., 2024). To preserve the reasoning ability of LLMs, some studies (Ma et al., 2025b; Kang et al., 2025; Xia et al., 2025) express the CoT in more concise text (*e.g.*, by removing less important tokens or rewriting with GPT-4) to reduce its length. However, they risk losing critical reasoning information or reducing interpretability when simplifying long CoT (Wang et al., 2025b).

In this paper, we propose an efficient reasoning framework for LLMs, which models the reasoning process of LLMs as a state-transition process. We regard a long CoT as a sequence of reasoning

steps, where LLMs perform a specific thinking pattern in each step, such as *induction* or *reflection*. Notably, each reasoning step contains two types of information: substantial linguistic information to ensure its fluency, and limited reasoning information to support subsequent reasoning or answer generation (Zhang et al., 2025; Xia et al., 2025). Thus, our framework (Figure 1) first compresses the reasoning information from previously generated reasoning steps into a matrix, termed as the *reasoning state matrix*. Then, based on the query prompt and the state matrix, LLMs can efficiently generate the current reasoning step and updates the state matrix accordingly. Specifically, tokens in the current reasoning step can directly retrieve relevant historical reasoning information from the reasoning state, without explicitly attending to tokens in previous steps. In this way, we effectively reduce both the computational complexity of attention and the memory overhead of the KV-cache. Crucially, our framework does not shorten or simplify the CoT sequences generated by LLMs, thus preserving their reasoning ability and interpretability.

To efficiently obtain the state matrix, we design a Mixed Attention Module (**MAM**) to replace the original attention module in LLMs, which consists of a Softmax-Attention (**SA**) submodule and a Linear-Attention (**LA**) submodule. We use the original attention module of LLMs as our SA submodule, where each token can only attend to the tokens in the query prompt and those in its current reasoning step. In the LA submodule, we adopt a linear-attention mechanism (Katharopoulos et al., 2020) to capture the reasoning state of LLMs. Meanwhile, with the linear attention, each token can directly retrieve relevant historical reasoning information from the reasoning state. Compared with other methods, such as CNN (Krizhevsky et al., 2012) and Q-Former (Li et al., 2023), linear attention offers the following advantages in capturing the model's reasoning state: (1) As a variant of softmax attention, linear attention is naturally compatible with it, thereby reducing the risk of losing critical reasoning information during compression and ensuring the stability and efficiency of model training. (2) Recent studies (Yang et al., 2024b; 2025c) have demonstrated that the state-update process of linear attention is essentially a gradient-descent learning procedure (i.e., test-time training), revealing its strong potential for handling complex reasoning tasks (Zhu et al., 2025).

Another challenge is that LLMs often produce noisy reasoning steps, which may mislead subsequent reasoning steps and lead to the overthinking issue (Chen et al., 2025b; Yang et al., 2025b). To mitigate this issue, we propose a *state-based reasoning strategy*. Given the model's state-transition process is a gradient-descent process, we first apply the momentum method to accumulate gradients from completed reasoning steps, obtaining a global gradient. The global gradient indicates the global reasoning direction of LLMs. Thus, we employ it to guide LLMs in completing the current reasoning step, ensuring they do not significantly deviate from the global direction.

To validate the efficacy of our framework, we conduct experiments on *seven widely-used benchmark datasets*. Experimental results show that our framework not only improves the reasoning efficiency of LLMs but also enhances their reasoning performance. Extensive ablation studies further demonstrate the effectiveness of various components in our framework.

2 PRELIMINARY

We first present a brief background on linear attention, which lays the foundation for our proposed framework. Meanwhile, we outline the specific form of the CoT sequences in our framework.

2.1 LINEAR ATTENTION

Softmax Attention (SA). Popular LLMs, such as Qwen 3 (Yang et al., 2025a) and Llama 4 (Meta, 2025), adopt a decoder-only Transformer architecture composed of repeated blocks of multi-head softmax attention followed by feed-forward networks (FFNs) (Vaswani et al., 2017a). Given the input sequence $\boldsymbol{X}=[x_1, \cdots, x_{|\boldsymbol{X}|}]$, the softmax attention can be formulated as follows:

$$\boldsymbol{o}_t = \frac{\sum_{i=1}^{t} \exp\left(\boldsymbol{q}_t \boldsymbol{k}_i^\top / \sqrt{d}\right) \boldsymbol{v}_i}{\sum_{i'=1}^{t} \exp\left(\boldsymbol{q}_t \boldsymbol{k}_{i'}^\top / \sqrt{d}\right)}; \quad \boldsymbol{q}_t, \boldsymbol{k}_t, \boldsymbol{v}_t = x_t \boldsymbol{W}_Q,\ x_t \boldsymbol{W}_K,\ x_t \boldsymbol{W}_V, \tag{1}$$

where $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, $\boldsymbol{W}_V$ are learnable weight matrices. The computational complexity of the softmax function increases quadratically with the length of the context sequence. Moreover, softmax

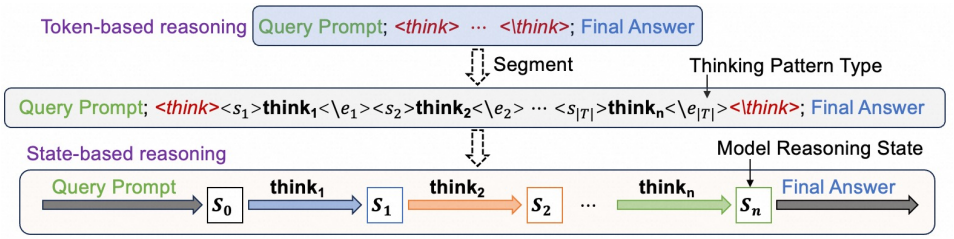

Figure 1: A comparison of traditional *token-based reasoning* with our *state-based reasoning*. $\textbf{think}_t$ denotes one reasoning step. In state-based reasoning, LLMs can efficiently generate $\textbf{think}_t$ only based on the query prompt and the reasoning state $\boldsymbol{S}_{t-1}$.

attention heavily depends on the growing KV-cache to recall historical information for sequence modeling, leading to substantial memory overheads, particularly in the long context setting.

Linear Attention (LA). Linear attention is a variant of softmax attention, designed to reduce its computational complexity and memory costs. Here, we first replaces the exponential function $\exp(\cdot)$ in softmax attention with a simpler kernel function $\phi(\cdot)$: $\exp(\boldsymbol{q}_t\boldsymbol{k}_i/\sqrt{d}) \rightarrow \phi(\boldsymbol{q}_t)\phi(\boldsymbol{k}_i)^\top$. Next, based on the associative property of matrix products, linear attention can be written as

$$\boldsymbol{o}_t = \frac{\sum_{i=1}^{t}\phi(\boldsymbol{q}_t)\phi(\boldsymbol{k}_i)^\top\boldsymbol{v}_i}{\sum_{i'=1}^{t}\phi(\boldsymbol{q}_t)\phi(\boldsymbol{k}_{i'})^\top} = \frac{\phi(\boldsymbol{q}_t)\sum_{i=1}^{t}\phi(\boldsymbol{k}_i)^\top\boldsymbol{v}_i}{\phi(\boldsymbol{q}_t)\sum_{i'=1}^{t}\phi(\boldsymbol{k}_{i'})^\top}. \tag{2}$$

Moreover, recent studies (Sun et al., 2023; Yang et al., 2024a) have shown that linear attention can perform well even when $\phi(\cdot)$ is set to the identity function and the normalizer is removed:

$$\boldsymbol{o}_t = \boldsymbol{q}_t\sum_{i=1}^{t}\boldsymbol{k}_i^\top\boldsymbol{v}_i = \boldsymbol{q}_t\boldsymbol{S}_t; \quad \boldsymbol{S}_t = \sum_{i=1}^{t}\boldsymbol{k}_i^\top\boldsymbol{v}_i, \tag{3}$$

where $\boldsymbol{S}_t$ is the state matrix storing historical information. By using $\boldsymbol{S}_t$, linear attention can perform sequence modeling in linear time while maintaining constant memory overhead.

The Test-Time Training (TTT) Perspective of LA. Recent works (Sun et al., 2024; Chen et al., 2025b) treat the state matrix $\boldsymbol{S}_t$ in linear attention as a fast-adapting parameter, updated at every token via lightweight gradient descent. They further introduce an online learning objective for linear attention and provide its closed-form gradient descent solution:

$$\begin{aligned} &\textbf{Objective:} \quad \mathcal{L}(\boldsymbol{S}) = -\langle \boldsymbol{S}\boldsymbol{k}_t, \boldsymbol{v}_t\rangle, \\ &\textbf{SGD update:} \quad \boldsymbol{S}_t = \boldsymbol{S}_{t-1} - \beta\nabla\mathcal{L}_t(\boldsymbol{S}_{t-1}) = \boldsymbol{S}_{t-1} + \beta\boldsymbol{v}_t\boldsymbol{k}_i^\top, \end{aligned} \tag{4}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product, and β is the learning rate. When β is set to 1, the state update process of linear attention is equivalent to training the state matrix $\boldsymbol{S}_t$ using the learning objective $\mathcal{L}(\boldsymbol{S})$. While current studies have yet to provide direct evidence that linear attention can improve the model's reasoning ability, its theoretical properties suggest significant potential (Zhu et al., 2025).

2.2 Long CoT Segmentation

A long CoT sample $(\boldsymbol{Q}, \boldsymbol{T}, \boldsymbol{A})$ usually comprises three parts: a query prompt $\boldsymbol{Q}$, a thinking sequence $\boldsymbol{T}$, and a final answer $\boldsymbol{A}$. Given the query prompt $\boldsymbol{Q}$, LLMs first perform complex reasoning ($\boldsymbol{T}$), and then generates the final answer $\boldsymbol{A}$. During reasoning, LLMs engage in various thinking patterns (*e.g.*, *reflection* and *result verification*) and switch between them using common transitional tokens (*e.g.*, "*Wait*", "*Hmm*") (Yang et al., 2025b; Wang et al., 2025c). Recent work (Wang et al., 2025a) further shows that LLMs usually transition thinking patterns at some high-entropy tokens, such as "*Alternatively*" and "*Maybe*". Here, we refer to the completion of a thinking pattern by LLMs as a *reasoning step*. Notably, when performing the current reasoning step, LLMs only consider the reasoning information (*e.g.*, conclusion) of previous completed steps without their linguistic information (*e.g.*, grammar).

Following Wang et al. (2025a), we first extract high-entropy transition tokens from the long CoT training set, which occur at the start of a sentence. Next, we use these tokens to segment the thinking

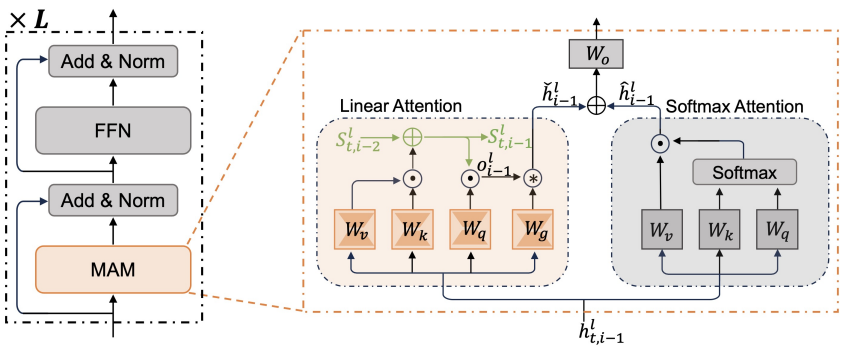

Figure 2: In our framework, the original softmax attention module in LLMs is replaced with our mixed attention module (MAM), thus improving reasoning efficiency and reducing memory cost.

sequence $\boldsymbol{T}$ in each training sample into a sequence of reasoning steps. Meanwhile, we cluster all reasoning steps in the entire training set, with each cluster corresponding to a thinking pattern (*i.e.*, type). Finally, for each thinking pattern, we introduce two special tokens to enclose its corresponding reasoning steps, as shown in Figure 1. With the reconstructed training set, the trained LLMs can generate more structured thinking sequences. Meanwhile, these special tokens allow us to track and control the reasoning processes of LLMs more accurately.

3 Our Framework

In this section, we provide a detailed description for our proposed framework. In our framework, we first design a mixed attention module (**MAM**) to replace the original attention module in LLMs, as illustrated in Figure 2. Using MAM, we can model the reasoning process of LLMs as a state-transition process (see Section 3.1). Then, we further introduces our state-based reasoning strategy in Section 3.2. Finally, our training strategy is presented in Section 3.3.

3.1 Mixed Attention Module

Our framework aims to improve the reasoning efficiency of LLMs without sacrificing reasoning ability by modeling their reasoning process as a state-transition process. To achieve this, we design a MAM to *replace* the softmax attention module in LLMs, which consists of a Softmax Attention (**SA**) submodule and a Linear Attention (**LA**) submodule. To avoid the performance loss caused by this replacement, we use the original softmax attention module of LLMs as our SA submodule. However, in the SA submodule, each token can only attend to the tokens in the query prompt $\boldsymbol{Q}$ and the previously generated tokens in its reasoning step. By doing so, we reduce the computational complexity of attention from quadratic $O(\mathcal{C}^2)$ to linear $O(\mathcal{C})$ and the memory usage of the KV-cache from linear $O(\mathcal{C})$ to constant $O(1)$, where $\mathcal{C}$ denotes the context length. This significantly improves the reasoning efficiency of our model, especially in complex scenarios requiring longer thinking sequences. Moreover, the LA submodule applies a linear attention mechanism to obtain the LLM's reasoning state matrix, which records the reasoning information from previously completed reasoning steps. Therefore, each token in the current reasoning step can access relevant historical information from the state matrix without attending directly to tokens in previous reasoning steps.

Given our LLM generates each token in every reasoning step using the same procedure, we take the generation of the i-th token $x_{t,i}$ in the t-th reasoning step as an example. Specifically, at the l-th layer of our model, we first feed $h^{(l-1)}_{t,i-1}$, the output feature of the input token $x_{t,i-1}$ at the (l-1)-th layer, into the MAM of the current layer. Then, $h^{(l-1)}_{t,i-1}$ is passed to the two submodules in MAM:

In the SA submodule, the KV-cache retains only the KV vectors of the query prompt tokens and ones of the previously generated tokens in the current reasoning step, while those of tokens in completed reasoning steps have been removed. Through the softmax attention mechanism (seen Eq. 1), the input token $x_{t,i-1}$ attends to tokens retained in the KV-cache, yielding an updated feature $\hat{h}^{(l)}_{t,i-1}$.

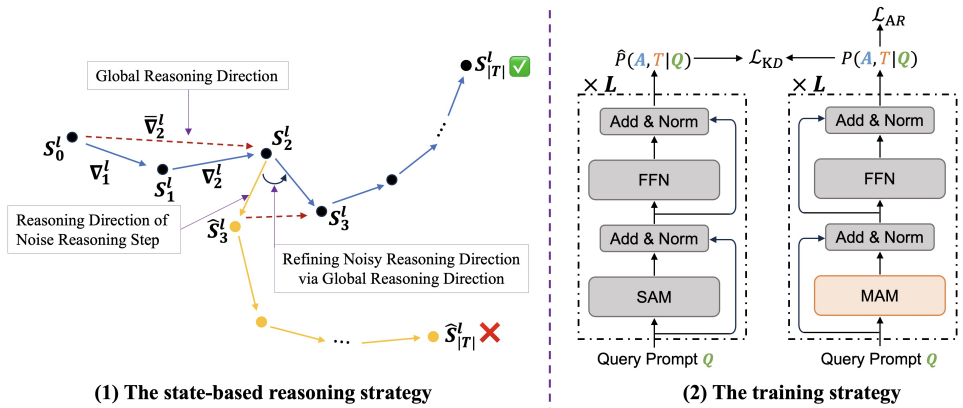

Figure 3: Illustration of our reasoning and training strategies. SAM is the softmax attention module.

In the LA submodule, after completing the first $t-1$ reasoning steps, our model state is updated to $\boldsymbol{S}_{t-1}^{(l)}$. Then, we use this state as the initial state $\boldsymbol{S}_{t,0}^{(l)}=\boldsymbol{S}_{t-1}^{(l)}$ for the current reasoning step and update it token-by-token, yielding the current model state $\boldsymbol{S}_{t,i-1}^{(l)}=\boldsymbol{S}_{t,0}^{(l)}+\sum_{j=1}^{i-1}\boldsymbol{k}_j^{(l)\top}\boldsymbol{v}_j^{(l)}$. Next, we utilize the query vector $\boldsymbol{q}_{i-1}^{(l)}$ of the input token x_{i-1} to extract the relevant historical reasoning information $\boldsymbol{o}_{i-1}^{(l)}$ from $\boldsymbol{S}_{t,i-1}^{(l)}$: $\boldsymbol{o}_{i-1}^{(l)}=\boldsymbol{q}_{i-1}^{(l)}\boldsymbol{S}_{t,i-1}^{(l)}$. In the current reasoning step, the model usually relies more on historical reasoning information to generate token in the early stage, and this dependence gradually decreases as more tokens are generated. Therefore, we obtain the output of this submodule via a gating mechanism: $\check{h}_{t,i-1}^{(l)}=\sigma(\boldsymbol{W}_g h_{t,i-1}^{(l-1)})*\boldsymbol{o}_{i-1}^{(l)}$, where $\sigma(\cdot)$ denotes the sigmoid function and $\boldsymbol{W}_g$ is a learnable weight (see Figure 2).

Finally, we combine the outputs of the two submodules and apply a linear output layer to yield the final output of MAM: $\tilde{h}_{t,i-1}^{(l)}=\boldsymbol{W}_o(\check{h}_{t,i-1}^{(l)}+\hat{h}_{t,i-1}^{(l)})$. As shown in Figure 2, the two submodules have identical structures, except that the LA submodule incorporates an additional gating weight $\boldsymbol{W}_g$. To control its parameter size, we implement the LA submodule via the LoRA strategy (Hu et al., 2022).

Subsequently, we further process $\tilde{h}_{t,i-1}^{(l)}$ using the FFN module to produce the output $h_{t,i-1}^{(l)}$ for the l-th layer of our model. By repeating the above process L times, we obtain the final output feature $h_{t,i-1}^{(L)}$ of the input token x_{t-1}, where L denotes the number of layers in our model. Next, $h_{t,i-1}^{(L)}$ is fed into the model's prediction layer to produce the predicted distribution $p(x)$ of the next token. Finally, our model generates the i-th token x_i of the t-th inference step according to $x_i=\arg\max_x p(x)$.

Following the above procedure, our model sequentially generates tokens in the current reasoning step until reaching its end token $\langle\backslash e_t\rangle$, which corresponds to the thinking pattern of the step (see Figure 1). After generating $\langle\backslash e_t\rangle$, the state matrix of our model at the l-th layer is updated to $\boldsymbol{S}_{t,n_t}^{(l)}$, which we denote as $\boldsymbol{S}_t^{(l)}$ and use as the initial state for the next reasoning step, where n_i denotes the number of tokens in the i-th reasoning step. Meanwhile, we clear the KV vectors of all tokens in the current inference step from the KV-cache.

3.2 The State-Based Reasoning Strateg

During reasoning, LLMs often produce noisy reasoning steps that may mislead subsequent ones, thus resulting in overthinking problems (Chen et al., 2025b; Yang et al., 2025b). In our framework, such noisy reasoning step can deviate the model's state transitions from the correct reasoning trajectory, resulting in erroneous results (see Figure 3(a)). To mitigate this issue, we propose a state-based reasoning strategy, which guides model reasoning with a global reasoning direction to reduce the bias from noisy reasoning steps.

In the reasoning process, the state transition of our model at the l-th layer can be formalized as: $\boldsymbol{S}_0^l\to\boldsymbol{S}_1^l\to\cdots\to\boldsymbol{S}_{|\boldsymbol{T}|}^l$, where $\boldsymbol{S}_0^l$ denotes the model state after the LA submodule processes the token

in the query prompt $\boldsymbol{Q}$. Considering that the state transition process in the line attention is a gradient descent process (see Section 2.1), we represent the total gradient contributed by each reasoning step as $[\nabla_1^l, \nabla_2^l, \cdots, \nabla_{|\boldsymbol{T}|}^l]$, where $\nabla_t^l = \boldsymbol{S}_t^l - \boldsymbol{S}_{t-1}^l$ indicates the reasoning direction of the t-th step. The reasoning direction of a noisy reasoning step often deviates substantially from those of other steps.

Therefore, we first aggregate the reasoning directions of all previous steps into a global reasoning direction $\bar{\nabla}_{t-1}^l$ using momentum accumulation: $\bar{\nabla}_{t-1}^l = (1-\frac{1}{t-1})\bar{\nabla}_{t-2}^l + \frac{1}{t-1}\nabla_{t-1}^l = \frac{1}{t-1}\sum_{i=1}^{t-1}\nabla_i^l$, where $\bar{\nabla}_0^l$ is initialized as a zero matrix. Then, once the t-th reasoning step is completed, we employ the global direction $\bar{\nabla}_{t-1}^l$ to correct its reasoning direction ∇_t^l: $\hat{\nabla}_t^l = (1-\alpha)\nabla_t^l + \alpha\bar{\nabla}_{t-1}^l$, where $\alpha = \max\{\alpha_{\max}, \frac{t}{|\boldsymbol{T}|}\}$ and $|\boldsymbol{T}|$ denotes the maximum number of reasoning steps (default: 40). Since more prior reasoning steps yield a more accurate global reasoning direction, we linearly increase the correction coefficient α up to a predefined threshold $\alpha_{\max}$. In this way, our model can fully explore diverse reasoning directions during the early stages of reasoning and then gradually converge toward the global reasoning direction. Finally, we use the corrected reasoning direction $\hat{\nabla}_t^l$ to update the model state, $\boldsymbol{S}_t^l = \boldsymbol{S}_{t-1}^l + \hat{\nabla}_t^l$, alleviating the negative impact of the noisy step on subsequent steps.

To enhance the diversity of thinking patterns during reasoning, we apply special markers indicating thinking patterns to guide the model toward adopting different thinking patterns across consecutive steps. By doing so, we can further enhance the robustness and overall performance of our model.

3.3 Model training

We train our model using the long CoT training set constructed in Section 2.1. To improve training efficiency while preserving the original reasoning ability of LLMs, we fine-tune only the parameters of the newly added LA submodule and ones of the special tokens corresponding to thinking patterns. As shown in Figure 3(b), we jointly optimize our model with two loss terms: (1) the autoregressive loss $\mathcal{L}_{\mathrm{AR}}$ of our model on the training samples, and (2) the knowledge distillation loss $\mathcal{L}_{\mathrm{KD}}$ between the base model (the original LLM with softmax attention module) and our proposed model. Finally, our training objective is defined as $\mathcal{L} = \mathcal{L}_{\mathrm{AR}} + \beta\mathcal{L}_{\mathrm{KD}}$, where β denotes a hyperparameter.

Specifically, we first input a long CoT sample into our model to obtain the predicted probability distribution $P(\boldsymbol{A}, \boldsymbol{T}|\boldsymbol{Q})$ over the thinking sequence $\boldsymbol{T}$ and the final answer $\boldsymbol{A}$ conditioned on the query prompt $\boldsymbol{Q}$. To maintain the consistency between training and testing, we apply a customized mask matrix in the SA submodule to ensure that each token attends only the tokens in the query prompt $\boldsymbol{A}$ and those from its corresponding reasoning step. Next, our autoregressive loss term $\mathcal{L}_{\mathrm{AR}}$ is defined as follows: $\mathcal{L}_{\mathrm{AR}} = -\log P(\boldsymbol{A}, \boldsymbol{T}|\boldsymbol{Q})$.

Meanwhile, the same long CoT sample is fed into the base model to produce the predicted probability distribution $\hat{P}(\boldsymbol{A}, \boldsymbol{T}|\boldsymbol{Q})$. Notably, our model is built upon the base model by replacing its softmax attention module with our MAM. In the base model, each token attends to all previous tokens. Subsequently, we use the Kullback–Leibler (KL) divergence between $P(\boldsymbol{A}, \boldsymbol{T}|\boldsymbol{Q})$ and $\hat{P}(\boldsymbol{A}, \boldsymbol{T}|\boldsymbol{Q})$ as our distillation loss term $\mathcal{L}_{\mathrm{KD}} = \mathbf{KL}(\hat{P}(\boldsymbol{A}, \boldsymbol{T}|\boldsymbol{Q})||P(\boldsymbol{A}, \boldsymbol{T}|\boldsymbol{Q}))$. The $\mathcal{L}_{\mathrm{KD}}$ loss term is designed to effectively train our LA submodule for capturing global reasoning information.

4 Experiments

4.1 Experiment Settings

Implementation Details. Our experiments are primarily conducted on the Qwen-2.5 series models (Hui et al., 2024). Meanwhile, we extract 95K high-quality samples from OpenR1-Math-220K to construct our training set using the method described in Section 2.1. We initialize Qwen-2.5 using its corresponding distilled version of DeepSeek-R1 (DeepSeek et al., 2025), and then train it on the training set to obtain **our base model**. Finally, our framework is built upon this base model.

Baselines. We compare our framework with two types of baselines: efficient reasoning methods and KV-cache reduction approaches. In the first category, **LightThinker** (Zhang et al., 2025) uses learnable special tokens to compress the reasoning information of completed reasoning steps, improving the reasoning efficiency of LLMs. **INFTYTHINK** (Yan et al., 2025) compresses the information from previous reasoning steps into a concise summary. In the second category, **H2O** (Zhang et al.,

Method	GSM8K			MATH-500			AMC23			AIME24			AIME25			AVG.	
	Acc↑	Tok	ReL↓	Acc↑	Tok	ReL↓	Acc↑	Tok	ReL↓	Acc↑	Tok	ReL↓	Acc↑	Tok	ReL↓	Acc↑	ReL↓
Qwen2.5-1.5B Series																	
Ours (Base)	80.1	2086	76.2	78.8	3958	139.4	62.5	6392	242.9	20.0	13765	536.8	16.7	12684	504.7	51.5	300.0
+H2O	76.1	2116	71.7	75.4	3835	128.5	55.0	6230	209.7	13.3	13921	462.2	10.0	12802	428.6	46.0	260.1
+SepLLM	77.3	2230	75.8	76.0	3890	130.3	57.5	6573	220.2	13.3	13690	452.6	13.3	12690	423.9	47.5	260.6
LightThinker	78.8	2109	69.6	77.6	4060	134.8	60.5	6312	214.6	16.7	13827	461.6	13.3	12934	434.6	50.1	263.0
INFTYTHINK	79.5	2503	89.3	78.0	4750	170.2	62.5	7605	263.5	16.7	16318	566.2	16.7	15020	501.0	50.7	318.0
Ours	**82.1**	**2116**	**69.7**	**81.2**	**3812**	**125.8**	**67.5**	**6460**	**213.2**	**26.7**	**13610**	**440.13**	**23.3**	**12910**	**416.0**	**56.2**	**253.0**
Qwen2.5-7B Series																	
Ours (Base)	89.4	2649	96.4	87.4	4253	161.6	82.5	6704	242.9	40.0	12901	522.1	30.0	13204	548.1	65.9	314.2
+H2O	83.5	2730	92.0	82.6	4389	147.1	77.5	6843	209.7	30.0	13274	448.9	23.3	13204	443.7	59.4	268.3
+SepLLM	84.9	2582	92.3	84.2	4244	144.3	80.0	6511	220.2	33.3	13100	484.7	23.3	12972	435.9	61.1	275.5
LightThinker	85.6	2779	90.0	84.8	4321	145.2	80.0	6790	214.6	33.3	13307	449.0	26.7	13250	443.9	62.1	268.5
INFTYTHINK	87.9	3237	117.0	86.2	5050	184.8	82.5	8061	254.3	36.7	15094	508.7	30.0	15448	530.5	64.7	319.0
Ours	**90.9**	**2603**	**87.2**	**90.0**	**4196**	**140.6**	**85.0**	**6665**	**213.2**	**43.3**	**13017**	**434.8**	**33.3**	**13191**	**440.6**	**68.3**	**263.3**

Table 1: The results of our model and baselines on mathematical reasoning benchmarks.

2023) reduces KV-cache by only retaining the KV vectors for a small set of heavy-hitter tokens and recent tokens with a greedy eviction strategy. **SapLLM** (Chen et al., 2025a) only stores the KV vectors of the initial, neighboring, and separator tokens into the KV-cache of LLMs. These baselines are trained using the LoRA strategy and have the same number of trainable parameters as our model.

Dataset & Metric. We evaluate our framework on seven benchmarks, including five mathematical reasoning benchmarks (GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021), AMC23 (AMC, 2023), AIME24 (AIME, 2024), AIME25 (AIME, 2025)), a scientific reasoning benchmark (GPQA_Diamond (Rein et al., 2024)), and a code reasoning benchmark (HumanEval (Chen et al., 2021)). We use greedy decoding to generate the output of the target LLM. We report three metrics to assess the performance and efficiency of various methods: (1) *Accuracy* (**Acc**) denotes the percentage of correct answers generated by the target model; (2) *Token Number* (**Tok**) refers to the average length of the CoT sequence; (3) *Reasoning Latency* (**ReL**) is defined as the average inference time per sample. Moreover, we measure reasoning latency on a single NVIDIA A100 GPU with a batch size of 1, while enabling FlashAttention-2 (Dao, 2023) with BF16 (Kalamkar et al., 2019).

4.2 Main Results

The experimental results on mathematical benchmarks are presented in Table 1. As shown in the *AVG.* column, our framework outperforms all baselines in reasoning efficiency and attains the best overall performance. After carefully analyzing these results, we draw several conclusions:

First, our model significantly outperforms the base model in reasoning speed, particularly on benchmarks requiring longer CoT. On the AIME24 benchmark, our model achieved a $16-18\%$ improvement in reasoning speed over the base model. This is mainly attributed to the lower computational complexity of our MAM relative to the softmax attention module of the base model. Furthermore, our model consistently achieves superior performance over the base model across all benchmarks. These results highlight the strong potential of linear attention mechanisms for efficient reasoning.

Second, the baselines (except INFTYTHINK) and our model generate CoT sequences with similar length on each benchmark, as they are trained on the same dataset. While these baselines achieve higher reasoning efficiency than the base model, they exhibit worse reasoning performance. For instance, on the AIME25 benchmark, LightThinker improves reasoning efficiency by 10–12% over the base model, but its performance decreases by 3–6 points. These results suggest that naive KV-cache compression methods risk losing important reasoning information, thus limiting model performance.

Third, unlike other baselines, INFTYTHINK uses text summaries of previous reasoning steps to record their reasoning information. While enhancing interpretability, this method reduces reasoning efficiency because it requires generating longer CoT sequences. Moreover, such discrete summarization may omit part of the implicit reasoning information within the model, leading to a slight performance drop for INFTYTHINK compared to the base model. However, we effectively avoid these two issues by utilizing the model's reasoning states to record reasoning information, enabling our model to outperform INFTYTHINK in both efficiency and performance.

Method	GSM8K	MATH-500	AMC23	AIME24	AIME25	AVG.
Ours	**82.1**	**81.2**	**67.0**	**26.7**	**23.3**	**56.2**
(1) *w/o* The state-based reasoning strategy	79.3	78.8	65.0	20.0	20.0	52.6
(2) *w/o* Thinking patterns diversity	80.9	80.0	65.0	23.3	20.0	53.8
(3) *w/o* The distillation loss $\mathcal{L}_{\mathrm{KD}}$	77.0	77.6	62.5	20.0	16.7	50.8
(4) *w/o* The model reasoning state	76.4	76.2	62.5	20.0	16.7	46.8
(5) *w/o* The gating mechanism	80.1	79.4	65.0	23.3	23.3	54.2

Table 2: Ablation results of our model on mathematical reasoning benchmarks.

4.3 Ablation Study

We further conduct extensive ablation studies by removing different components from our framework to investigate their different impacts. As shown in Table 2, we only compare our model with the following variants in terms of performance, as they share the same reasoning efficiency.

w/o The state-based reasoning strategy. In this variant, we remove the state-based reasoning strategy from our framework. We observe an average performance drop of 3.54 points across all benchmarks for this variant. These results suggest that our reasoning strategy can indeed enhance the robustness of LLMs to noisy reasoning steps, improving their overall performance. Moreover, this also indicates that, with suitable processing, our model states can intuitively reflect the quality of reasoning steps, which offers an interesting insight for future exploration (e.g., process-reward-based RL).

w/o Thinking patterns diversity. We guide LLMs to use different thinking patterns in adjacent reasoning steps via corresponding special tokens, thus ensuring diversity of the thinking patterns in the reasoning process. To test its effectiveness, we remove this operation from our framework. As shown in **Line (2)**, this variant exhibits a consistent decline in performance across all benchmarks. These findings imply that more diverse thinking patterns can enhance LLMs' exploration ability, thereby improving the accuracy of their generated answers.

The distillation loss $\mathcal{L}_{\mathrm{KD}}$. As described in Section 3.3, we use $\mathcal{L}_{\mathrm{KD}}$ to train our LA submodule so that it can better capture global reasoning information. In this variant, $\mathcal{L}_{\mathrm{KD}}$ is excluded from our training objective. From **Line (3)** in Table 2, we note that this variant results in a performance drop of 3.6–6.6 points across all benchmarks. The main reason is that the autoregressive loss $\mathcal{L}_{\mathrm{AR}}$ may struggle to train the LA submodule to capture the global correlations between reasoning steps.

w/o The model reasoning state. In this variant, we replace the model states in our framework with a sliding window of size 64 to store historical reasoning information. Although this approach is also compatible with the original softmax attention in LLMs, this variant yields the most significant performance drop compared with other variants. This is mainly because this approach lacks reasoning capability as strong as that of linear attention.

w/o The gating mechanism. This variant removes the gating mechanism from our LA submodule and directly sums the outputs of the two submodules in the MAM. This change leads to a slight performance drop across all benchmarks. This confirms that our gating mechanism can more precisely control the amount of historical reasoning information each token requires.

4.4 Further Analysis

Analysis of Computational and Memory Costs. We conduct experiments to further compare the computational and memory efficiency of our model and the base model across varying CoT lengths. The experimental results are presented in Figure 4(a). Although our model exhibits similar reasoning efficiency to the base model for shorter CoT, it significantly surpasses the base model once the CoT length exceeds 4K. In particular, when the CoT length reaches 32K, our model achieves over 40% faster reasoning speed than the base model. Moreover, our model maintains a nearly constant memory usage across varying CoT lengths, whereas that of the base model increases linearly with CoT length. Theoretically, our model's advantages in computational and memory efficiency would become even more significant when FlashAttention-2 is disabled.

Analysis of Hyper-Parameters. We also investigate the impact of the two key hyper-parameters, β and $\alpha_{\max}$, on the performance of our model. As illustrated in Figure 4(b)–(c), our model exhibits

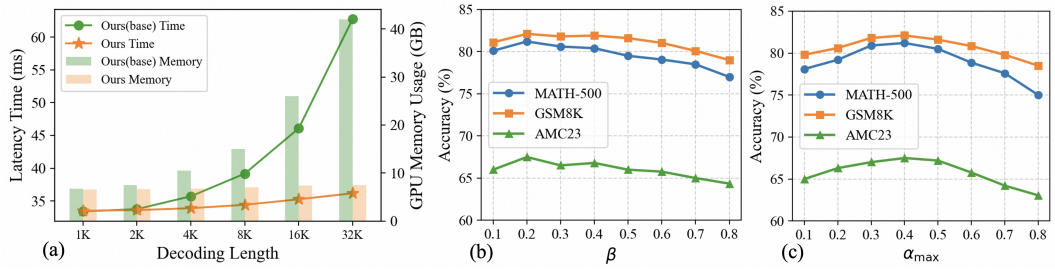

Figure 4: (a) shows the computational and memory efficiency of our model and the base model. (b) and (c) present our model's performance with different values of hyper-parameters β and $\alpha_{\max}$, respectively. These experiments are conducted on Qwen2.5-1.5B.

low sensitivity to these two hyper-parameters. Meanwhile, our model attains the best performance when β and $\alpha_{\max}$ are set to 0.2 and 0.4, respectively.

In Appendix A, we further analyze both the domain generalization capability of our framework and the gating mechanism within our LA submodule.

5 Related Work

Efficient Reasoning. Although long CoT enhances the reasoning ability of LLMs, it introduces significant computational overhead. To alleviate this challenge, existing methods for CoT compression can be roughly divided into three categories. **The first category** leverages prompting (Han et al., 2024; Ma et al., 2025a), supervised fine-tuning (SFT) (Liu et al., 2024; Munkhbat et al., 2025), and reinforcement learning (RL) (Aggarwal et al., 2025; Shen et al., 2025) to encourage LLMs to generate shorter CoT sequences. Since such CoT reductions conflict with test-time scaling (OpenAI et al., 2024), these methods often impair the reasoning ability of LLMs (Jin et al., 2024; Merrill et al., 2024). **The second category** compresses information from multiple text tokens into a single hidden vector, converting longer discrete CoT sequences into shorter continuous ones to improve the reasoning efficiency of LLMs (Hao et al., 2024; Su et al., 2025). However, such methods typically require altering the input/output patterns of LLMs, increasing their vulnerability to catastrophic forgetting and limiting their performance. **The third category** expresses the CoT in more concise text to reduce its length while preserving the reasoning ability of LLMs (Ma et al., 2025b; Kang et al., 2025; Xia et al., 2025). Nevertheless, when simplifying long CoT, these methods risk losing critical reasoning information or reducing interpretability (Wang et al., 2025b).

Linear Attention. To address the quadratic computational cost of softmax attention, many linear attention methods have been proposed to improve training and inference efficiency. Early studies primarily investigated kernel functions for linear attention (Kasai et al., 2021; Peng et al., 2021). Subsequently, several studies introduced gating mechanisms into linear attention to more effectively control the information in the model state (Yang et al., 2024a; Qin et al., 2024; Lan et al., 2025). Recent studies has provided theoretical evidence for the Test-Time Training (TTT) property of linear attention (Yang et al., 2024b; Sun et al., 2024; Liu et al., 2025; Chen et al., 2025b). They focus on exploring online learning objectives for linear attention to enhance its sequence modeling capability.

6 Conclusion

In this paper, we propose an efficient reasoning framework that models the reasoning process of LLMs as a state-transition process to enhance their reasoning efficiency. In our framework, we design an MAM to replace the original attention module in LLMs. The MAM uses a linear attention mechanism to capture the model's reasoning state that stores reasoning information from previous reasoning steps. By utilizing the model state, we can significantly reduce the computational complexity and memory cost of attention, thereby improving the reasoning efficiency of LLMs. Furthermore, we design a state-based reasoning strategy to avoid the overthinking issue caused by noisy reasoning steps. Extensive experiments across multiple benchmarks and model sizes demonstrate the efficiency and robustness of our framework.

REFERENCES

Pranjal Aggarwal, Sean Welleck, Pranjal Aggarwal, and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv:2503.04697*, 2025.

AIME. American invitational mathematics examination (aime) aime 2024-i & ii, 2024. URL https://huggingface.co/datasets/Maxwell-Jia/AIME_2024.

AIME. American invitational mathematics examination (aime) 2025-i & ii, 2025. URL https://huggingface.co/datasets/opencompass/AIME2025.

AMC. American mathematics competitions, 2023. URL https://artofproblemsolving.com/wiki/index.php/AMC_Problems_and_Solutions.

Guoxuan Chen, Han Shi, Jiawei Li, Yihang Gao, Xiaozhe Ren, Yimeng Chen, Xin Jiang, Zhenguo Li, Weiyang Liu, and Chao Huang. Sepllm: Accelerate large language models by compressing one segment into one separator. In *Proceedings of ICML*, 2025a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv:2107.03374*, 2021.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. In *Proceedings of ICML*, 2025b.

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

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv:2307.08691*, 2023.

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

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. *arXiv:2412.18547*, 2024.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv:2412.06769*, 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. In *Proceedings of NIPS*, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *Proceedings of ICLR*, 2022.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2.5 Technical Report. *arXiv:2409.12186*, 2024.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. The impact of reasoning step length on large language models. In *Proceedings of ACL*, 2024.

Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, et al. A study of BFLOAT16 for deep learning training. *arXiv:1905.12322*, 2019.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of AAAI*, 2025.

Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A Smith. Finetuning pretrained transformers into rnns. In *Proceedings of EMNLP*, 2021.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of ICML*, 2020.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Proceedings of NIPS*, 2022.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of NIPS*, 2012.

Disen Lan, Weigao Sun, Jiaxi Hu, Jusen Du, and Yu Cheng. Liger: Linearizing Large Language Models to Gated Recurrent Structures. In *Proceedings of ICML*, 2025.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of ICML*, 2023.

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. Longhorn: State space models are amortized online learners. In *Proceedings of ICLR*, 2025.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. Can language models learn to skip steps? In *Proceedings of NIPS*, 2024.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. Reasoning models can be effective without thinking. *arXiv:2504.09858*, 2025a.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. Cot-valve: Length-compressible chain-of-thought tuning. *arXiv:2502.09601*, 2025b.

William Merrill, Ashish Sabharwal, William Merrill, and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *Proceedings of ICLR*, 2024.

AI Meta. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. *https://ai. meta. com/blog/llama-4-multimodal-intelligence/, checked on*, 2025.

Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. Self-training elicits concise reasoning in large language models. *arXiv:2502.20122*, 2025.

Aaron OpenAI, Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv:2412.16720*, 2024.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. In *Proceedings of ICLR*, 2021.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. In *Proceedings of COLM*, 2024.

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 *Proceedings of COLM*, 2024.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, Zhaoxiang Liu, and Shiguo Lian. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv:2503.04472*, 2025.

DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. Token assorted: Mixing latent and text tokens for improved language model reasoning. *arXiv:2502.03275*, 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: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:2307.08621*, 2023.

Qwen Team. Qwq: Reflect deeply on the boundaries of the unknown. 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of NIPS*, 2017a.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of NIPS*, 2017b.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv:2506.01939*, 2025a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of ICLR*, 2023.

Yibo Wang, Li Shen, Huanjin Yao, Tiansheng Huang, Rui Liu, Naiqiang Tan, Jiaxing Huang, Kai Zhang, and Dacheng Tao. R1-Compress: Long Chain-of-Thought Compression via Chunk Compression and Search. *arXiv:2505.16838*, 2025b.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, et al. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv:2501.18585*, 2025c.

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. In *Proceedings of NIPS*, 2022.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv:2502.12067*, 2025.

Yuchen Yan, Yongliang Shen, Yang Liu, Jin Jiang, Mengdi Zhang, Jian Shao, and Yueting Zhuang. Inftythink: Breaking the length limits of long-context reasoning in large language models. *arXiv:2503.06692*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv:2505.09388*, 2025a.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Zheng Lin, Li Cao, and Weiping Wang. Dynamic Early Exit in Reasoning Models. *arXiv:2504.15895*, 2025b.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of ICML*, 2024a.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Proceedings of NIPS*, 2024b.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *Proceedings of ICLR*, 2025c.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Proceedings of NIPS*, 2023.

Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. Lightthinker: Thinking step-by-step compression. *arXiv:2502.15589*, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Proceedings of NIPS*, 2023.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *Proceedings of ICLR*, 2023.

Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, et al. A survey on latent reasoning. *arXiv:2507.06203*, 2025.

A Further Analysis

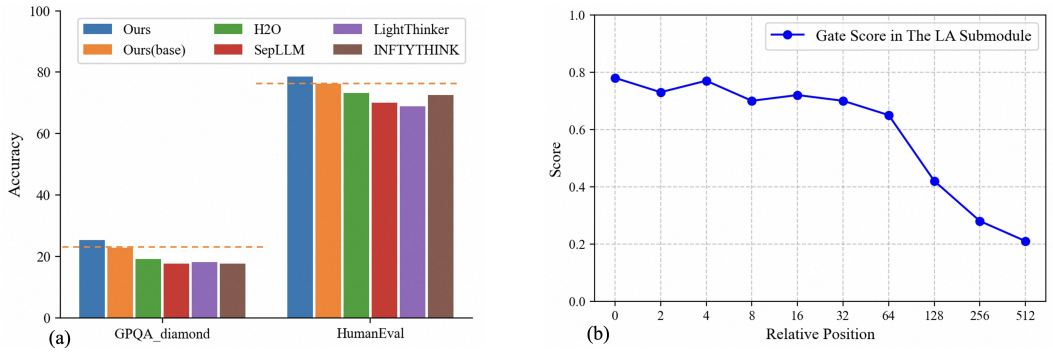

Figure 5: (a) shows the performance of our model and the baselines on the GPQA Diamond and HumanEval benchmarks. (b) illustrates the average gating scores in the LA submodule for tokens at different positions within each reasoning step. These experiments are conducted on Qwen2.5-7B.

Analysis of Domain Generalization. As stated in Section 4.1, our training set contains only mathematical reasoning data. To examine the domain generalization of our framework, we further test our model on the scientific reasoning benchmark GPQA_Diamond and the code reasoning benchmark HumanEval. As shown in Figure 5(a), our model consistently outperforms all baselines on these two benchmarks. These results demonstrate that our model state can effectively store domain-agnostic important historical reasoning information, which ensures the accuracy of subsequent reasoning.

Analysis of Gating Scores. Here, we further investigate how much historical reasoning information is required by tokens at different positions within each reasoning step. To achieve this, we calculate the average gating scores of these tokens in the LA submodule. As illustrated in Figure 5(b), tokens occurring earlier in a reasoning step generally require more historical reasoning information from the model state than those occurring later. The main reason is that later tokens can directly obtain more information from earlier tokens via the SA sublayer, reducing their reliance on the model state.

B Implementation Details

We train both our model and the baselines for 2 epochs on the training set, with a batch size of 32. The learning rate is set to $2e{-}4$, with a warmup ratio of 0.03. The learning rate follows a cosine decay schedule to reach zero. We set the learnable LoRA parameters for both our model and the baselines to approximately 66.5M on Qwen2.5-1.5B and 147.3M on Qwen2.5-7B, respectively. To control training cost, we limit the maximum length of long CoT samples in the training set to 16,384 tokens. Meanwhile, we train our model and the baselines on 8 NVIDIA A100 GPUs, each with 80GB of memory.

C Ethics Statement

This work adheres to the ICLR Code of Ethics. In this study, no human subjects or animal experimentation was involved. All datasets used, including OpenR1-Math-220K, GSM8K, MATH-500,

AMC23, AIME24, AIME25, GPQA_Diamond, and HumanEval, were sourced in compliance with relevant usage guidelines, ensuring no violation of privacy. We have taken care to avoid any biases or discriminatory outcomes in our research process. No personally identifiable information was used, and no experiments were conducted that could raise privacy or security concerns. We are committed to maintaining transparency and integrity throughout the research process.

D Reproducibility Statement

We have made every effort to ensure that the results presented in this paper are reproducible. The experimental setup, including training steps, model configurations, and hardware details, is described in detail in the paper. Additionally, we use publicly available datasets, such as OpenR1-Math-220K, GSM8K, MATH-500, and HumanEval, to ensure consistent and reproducible evaluation results. We believe these measures will enable other researchers to reproduce our work and further advance the field.

E LLM Usage

In the preparation of this paper, we use Large Language Models (LLMs) solely to aid in writing and polishing the text, including improving clarity, grammar, and readability. LLMs are not used for generating scientific content, experimental design, analysis, or conclusions. All technical ideas, experiments, and results reported in this paper are entirely the work of the authors.