TIV: Thought Injection via Vectors for Efficient Reasoning in Large Reasoning Models

Yi Cao[1,2], Weijie Shi[3*], Wei-Jie Xu[4], Yucheng Shen[1], Yue Cui[5], Hanghui Guo[6], Shimin Di[7], Ziyi Liu[3*], Jiaming Li[8], Alexander Zhou[9], Jia Zhu[6*], Jiajie Xu[1,2*]

[1]School of Computer Science and Technology, Soochow University
[2]Key Laboratory of Data Intelligence and Advanced Computing, Soochow University
[3]Department of Computer Science and Engineering, The Hong Kong University of Science and Technology
[4]School of Artificial Intelligence, Nanjing University
[5]Alibaba Group
[6]Zhejiang Key Laboratory of Intelligent Education Technology and Application, Zhejiang Normal University
[7]School of Computer Science and Engineering, Southeast University
[8]ByteDance
[9]Department of Computing, The Hong Kong Polytechnic University
ycaosudaer@stu.suda.edu.cn, wshiah@connect.ust.hk, zyl@ust.hk, jiazhu@zjnu.edu.cn, xujj@suda.edu.cn

Abstract

Large Reasoning Models (LRMs) have recently demonstrated impressive performance across a range of reasoning tasks by generating intermediate thoughts. However, these models can suffer from *overthinking*—generating excessive tokens that contribute little to final accuracy while increasing inference cost. To mitigate this, we propose **TIV** (**T**hought **I**njection via **V**ectors), an innovative framework that compresses token-level reasoning into compact vectors without sacrificing performance. Rather than generating explicit thoughts, TIV injects learnable vectors into the post-attention hidden states of the final token across Transformer layers, enabling implicit and lightweight reasoning. We further introduce a two-stage reinforcement learning strategy: the first stage calibrates the model's reasoning distribution, and the second distills it into a vector-based policy optimized for both accuracy and brevity. Experiments on three reasoning benchmarks show that TIV preserves over 99% of the original accuracy while reducing output length by more than 65% on average, reaching up to 80% in some cases. Moreover, TIV consistently achieves superior trade-offs between accuracy and efficiency compared to existing methods, distinguishing itself as a state-of-the-art (SOTA) approach for efficient reasoning in LRMs.

1 Introduction

Recently, Large Language Models (LLMs) (Achiam et al. 2023; Grattafiori et al. 2024), built upon the Transformer architecture (Vaswani et al. 2017), have demonstrated strong performance across various tasks. However, they continue to struggle with complex reasoning problems. To address this limitation, Large Reasoning Models (LRMs) have emerged as an evolution of LLMs, using reinforcement learning (RL) to strengthen reasoning abilities. Notable examples include OpenAI-o1 (Jaech et al. 2024), DeepSeek-R1 (Guo et al. 2025), and MiniMax-M1 (Chen et al. 2025a), which perform well on challenging tasks like math and programming.

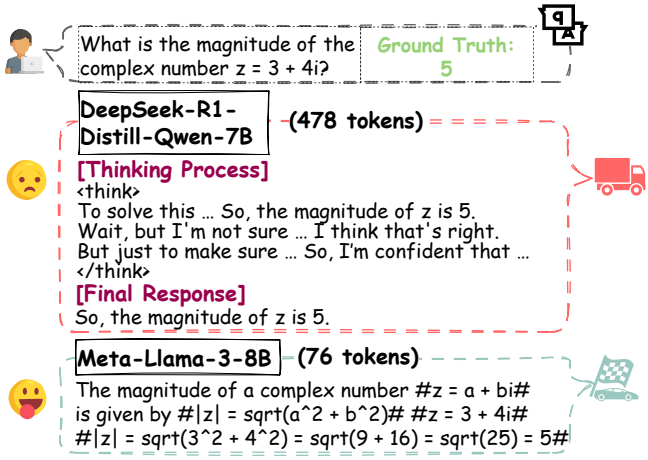

Figure 1: An example of *overthinking* phenomenon from the DeepSeek-R1-Distill-Qwen-7B model.

While effective, these reasoning models often engage in *overthinking*, generating excessively detailed and sometimes redundant solutions when they solve problems (Chen et al. 2025b; Sui et al. 2025). This leads to a significant increase in the number of output tokens, thereby incurring inefficient computational resource utilization. An illustrative example of this phenomenon is provided in Figure 1 for reference.

Existing efforts to shorten responses have typically relied on heuristic trade-offs, such as incorporating length-aware rewards during RL (Luo et al. 2025a; Shen et al. 2025) or enhancing reasoning without length constraints (Chen et al.

*Corresponding authors.

2025b). However, they treat reasoning as surface-level token sequences, not considering the potential to abstract reasoning into latent representations which still directly shape the subsequent computation and generation of final outputs.

In this paper, we introduce TIV, an innovative framework that replaces token-level reasoning with compact, learnable vectors injected into the hidden states, thereby bypassing the need for explicit thought generation. To ensure the generalizability of these injected vectors, we propose a specialized two-stage optimization strategy based on RL techniques.

Experiments show that TIV enhances reasoning efficiency while maintaining strong accuracy across benchmarks. On GSM8K, MATH 500, and AIME 2024, TIV cuts response length by 73.8%, 70.2%, and 56.7%, with minimal accuracy changes (+0.7%, -1.2%, -1.1%). This suggests that vector injection effectively balances accuracy and efficiency.

Our main contributions are as follows: (1) We rigorously analyze the attention decomposition and show that thoughts can be compactly encoded through vector injection; (2) We design a two-stage RL-based framework that stabilizes and distills reasoning into lightweight vector representations; (3) TIV preserves over 99% of original accuracy while reducing reasoning length by over 65%, surpassing prior approaches.

2 Related Works

2.1 Large Reasoning Models

With the support from RL (Sutton and Barto 1999), LRMs are advancing toward Artificial General Intelligence (AGI), with representative works such as OpenAI-o1 (Jaech et al. 2024), DeepSeek-R1 (Guo et al. 2025), and MiniMax-M1 (Chen et al. 2025a). These reasoning models simulate the way humans solve problems by generating detailed step-by-step Chain-of-Thought (CoT) (Wei et al. 2022). However, such reasoning often leads to the generation of excessive and unnecessary tokens, resulting in inefficient computation—a phenomenon referred to as *overthinking* (Chen et al. 2025b).

2.2 Efficient Reasoning in LRMs

Enhancing reasoning efficiency in LRMs has been a central research goal, with early efforts focusing on inference-time control and supervised fine-tuning. Inference-based methods adjust decoding strategies at test time to generate shorter outputs (Du et al. 2025), but they rely on heuristic signals that may not align with semantic reasoning quality, leading to performance degradation. Supervised fine-tuning methods train models to imitate pre-constructed shorter reasoning traces (Han et al. 2025), yet this often demands costly data curation and risk performance deterioration on account of the static and potentially suboptimal nature of the strategy.

Overall, the aforementioned approaches are limited in the ability to trade off between accuracy and efficiency. As a result, recent advances have turned to RL as a more principled method to improve reasoning efficiency (Sui et al. 2025).

For instance, OverThink (Chen et al. 2025b) introduces a self-training framework that uses principled metrics to select compact yet effective CoT solutions, and leverages Simple Preference Optimization (SimPO) (Meng, Xia, and Chen 2024) to balance reasoning depth with task performance.

Unlike OverThink, several RL approaches have received considerable attention for incorporating response length as a factor in their reward calculations. Notable works include O1-Pruner (Luo et al. 2025a) and DAST (Shen et al. 2025).

O1-Pruner decouples response generation from policy optimization. Specifically, it first builds a long-form response corpus offline, then uses the Proximal Policy Optimization (PPO) algorithm (Schulman et al. 2017) to train a compact policy. This policy is designed to strike a balance between informativeness and conciseness, leveraging a reward function that explicitly favors shorter yet accurate outputs.

DAST introduces an adaptive reasoning framework that adjusts response length according to question difficulty. It estimates target lengths dynamically and integrates model-based preference signals. Employing SimPO for fine-tuning, it achieves dynamic control over the depth of reasoning.

However, existing RL-based methods regard reasoning as surface-level text to be rewarded, truncated, or reweighted, not deeply probing into its potential internal representations. By restricting their attention to observable token sequences, they overlook the possibility that reasoning can be abstracted into latent representations, which could still directly guide the final response generation. Without access to or modeling of such representations, these approaches exert only rather coarse-grained control over the reasoning process in LRMs.

2.3 Vector-based Representations in LLMs

Vectors are increasingly used to encode control signals, external knowledge, or task semantics in LLMs. Methods like Prefix-Tuning (Li and Liang 2021), which prepends task-specific vectors, and In-Context Vectors (Liu et al. 2024), which modify intermediate activations, show that injecting information via vectors is a promising way to guide LLMs.

3 Problem Formulation

Consider that π_θ represents a LRM parameterized by θ. For a question $q \in \mathcal{Q}$, where $\mathcal{Q}$ is the space of all questions, π_θ generates a lengthy thinking process $t = \pi_\theta(q)$ composed of a sequence of intermediate tokens $\{t_1, t_2, \ldots, t_{n_{\text{think}}}\}$. This thought guides π_θ to produce a final answer $a = \pi_\theta(q, t) \in \mathcal{A}$, where $\mathcal{A}$ represents the space of all answers. In this context, we denote the combined token count of t and a as n_{token}.

However, redundant tokens could increase computational overhead. This motivates a compressed thinking process $\hat{t} = \{\hat{t}_1, \hat{t}_2, \ldots, \hat{t}_{\hat{n}_{\text{think}}}\}$ yielding answer $\hat{a}$, where $\hat{n}_{\text{token}} < n_{\text{token}}$, reducing token count, and $\hat{a}$ still matches the correct answer.

Formally, we seek to learn a mapping $\partial : \theta \to \hat{\theta}$ in order to solve the following multi-objective optimization problem:

$$\min_{\partial} \quad \mathbb{E}_{q\sim\mathcal{Q}}\big[\hat{n}_{\text{token}} \mid \pi_{\hat{\theta}}(q) = \hat{t}, \pi_{\hat{\theta}}(q, \hat{t}) = \hat{a}\big], \tag{1}$$

$$\max_{\partial} \quad \mathbb{E}_{q\sim\mathcal{Q}}\big[\mathbb{I}(\hat{a} = a^*) \mid \pi_{\hat{\theta}}(q) = \hat{t}, \pi_{\hat{\theta}}(q, \hat{t}) = \hat{a}\big], \tag{2}$$

where $\mathbb{I}(\cdot)$ is the indicator function, outputting 1 if true and 0 if false, with a^* as the ground-truth answer to question q.

4 Methodology

In this section, we introduce the TIV framework for efficient reasoning in LRMs. We begin by reviewing the Transformer

backbone to provide necessary architectural context (§4.1). Next, we show that the impact of intermediate reasoning on final response generation can be reduced to a injected vector (§4.2). Building upon this, we propose a two-stage RL-based optimization strategy that initially calibrates the reasoning distribution and subsequently distills it into compact vectors (§4.3). An overview of the pipeline is presented in Figure 2.

4.1 Preliminaries

LRMs are built upon the Transformer architecture (Vaswani et al. 2017). Each layer is composed of a Self-Attention (SA) mechanism that enables cross-token interaction and a Multi-Layer Perceptron (MLP) for token-wise transformation.

Self-Attention (SA). Given an input sequence $\mathbf{X} \in \mathbb{R}^{n\times d}$ (n tokens, hidden size d), the self-attention mechanism is:

$$\mathrm{Attn}(\mathbf{Q_X}, \mathbf{K_X}, \mathbf{V_X}) = \mathrm{softmax}(\frac{\mathbf{Q_X}\mathbf{K_X}^\top}{\sqrt{d}})\mathbf{V_X}, \quad (3)$$

where $\mathbf{Q_X}, \mathbf{K_X}, \mathbf{V_X} \in \mathbb{R}^{n\times d}$ are the query, key, and value matrices, respectively, derived from linear projections of $\mathbf{X}$.

Multi-Layer Perceptron (MLP). Acting on each token vector $\mathbf{x} \in \mathbb{R}^d$ (hidden size d) in parallel, the MLP applies:

$$\mathrm{MLP}(\mathbf{x}) = \sigma(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \quad (4)$$

where $\mathbf{W}_1 \in \mathbb{R}^{d\times d_{\text{ff}}}$, $\mathbf{W}_2 \in \mathbb{R}^{d_{\text{ff}}\times d}$, $\mathbf{b}_1 \in \mathbb{R}^{d_{\text{ff}}}$, $\mathbf{b}_2 \in \mathbb{R}^d$, and $\sigma(\cdot)$ represents a specific non-linear activation function.

4.2 TIV: Thought Injection via Vectors

Global Attention Decomposition. We first ask: is there a function $\eth : \mathbf{D}_\eth \to \mathbb{R}^d$ such that, for any question matrix $\mathbf{Q} \in \mathbb{R}^{|\mathbf{Q}|\times d}$, thought matrix $\mathbf{T} \in \mathbb{R}^{|\mathbf{T}|\times d}$, and query vector $\mathbf{x} \in \mathbb{R}^d$, there exist scalars λ, μ that satisfy the following:

$$\begin{aligned}&\mathrm{Attn}\big(\mathbf{x}, \big[\mathbf{Q}^\top\ \mathbf{T}^\top\big]^\top, \big[\mathbf{Q}^\top\ \mathbf{T}^\top\big]^\top\big)\\ &= \lambda\cdot\mathrm{Attn}(\mathbf{x},\mathbf{Q},\mathbf{Q}) + \mu\cdot\eth(\mathbf{x},\mathbf{T},\mathbf{T}),\end{aligned} \quad (5)$$

where $\mathrm{Attn}(\cdot)$ represents the self-attention mechanism?

To explore this, we use the attention mechanism formula:

$$\begin{aligned}&\mathrm{Attn}\big(\mathbf{x}, \big[\mathbf{Q}^\top\ \mathbf{T}^\top\big]^\top, \big[\mathbf{Q}^\top\ \mathbf{T}^\top\big]^\top\big)\\ &= \mathrm{softmax}\Big(\begin{bmatrix}\frac{\mathbf{x}\mathbf{Q}^\top}{\sqrt{d}} & \frac{\mathbf{x}\mathbf{T}^\top}{\sqrt{d}}\end{bmatrix}\Big)\big[\mathbf{Q}^\top\ \mathbf{T}^\top\big]^\top,\end{aligned} \quad (6)$$

where $\begin{bmatrix}\frac{\mathbf{x}\mathbf{Q}^\top}{\sqrt{d}} & \frac{\mathbf{x}\mathbf{T}^\top}{\sqrt{d}}\end{bmatrix} \in \mathbb{R}^{|\mathbf{Q}|+|\mathbf{T}|}$. Expanding $\mathrm{softmax}(\cdot)$:

$$\begin{aligned}&\mathrm{softmax}\Big(\begin{bmatrix}\frac{\mathbf{x}\mathbf{Q}^\top}{\sqrt{d}} & \frac{\mathbf{x}\mathbf{T}^\top}{\sqrt{d}}\end{bmatrix}\Big)\\ &= \begin{bmatrix}\frac{\exp(\frac{(\mathbf{x}\mathbf{Q}^\top)_1}{\sqrt{d}})}{\mathcal{Z}_1+\mathcal{Z}_2} \cdots \frac{\exp(\frac{(\mathbf{x}\mathbf{Q}^\top)_i}{\sqrt{d}})}{\mathcal{Z}_1+\mathcal{Z}_2} \cdots \frac{\exp(\frac{(\mathbf{x}\mathbf{Q}^\top)_{|\mathbf{Q}|}}{\sqrt{d}})}{\mathcal{Z}_1+\mathcal{Z}_2}\\ \frac{\exp(\frac{(\mathbf{x}\mathbf{T}^\top)_1}{\sqrt{d}})}{\mathcal{Z}_1+\mathcal{Z}_2} \cdots \frac{\exp(\frac{(\mathbf{x}\mathbf{T}^\top)_i}{\sqrt{d}})}{\mathcal{Z}_1+\mathcal{Z}_2} \cdots \frac{\exp(\frac{(\mathbf{x}\mathbf{T}^\top)_{|\mathbf{T}|}}{\sqrt{d}})}{\mathcal{Z}_1+\mathcal{Z}_2}\end{bmatrix}\\ &= \begin{bmatrix}\frac{\exp(\frac{\mathbf{x}\mathbf{Q}^\top}{\sqrt{d}})}{\mathcal{Z}_1+\mathcal{Z}_2} & \frac{\exp(\frac{\mathbf{x}\mathbf{T}^\top}{\sqrt{d}})}{\mathcal{Z}_1+\mathcal{Z}_2}\end{bmatrix},\end{aligned} \quad (7)$$

with $\mathcal{Z}_1 = \sum_{i=1}^{|\mathbf{Q}|}\exp(\frac{(\mathbf{x}\mathbf{Q}^\top)_i}{\sqrt{d}})$, $\mathcal{Z}_2 = \sum_{i=1}^{|\mathbf{T}|}\exp(\frac{(\mathbf{x}\mathbf{T}^\top)_i}{\sqrt{d}})$, and i represents the i-th element of the corresponding vector. Consequently, the attention output can be decomposed as:

$$\begin{aligned}&\mathrm{Attn}\big(\mathbf{x}, \big[\mathbf{Q}^\top\ \mathbf{T}^\top\big]^\top, \big[\mathbf{Q}^\top\ \mathbf{T}^\top\big]^\top\big)\\ &= \frac{\exp(\frac{\mathbf{x}\mathbf{Q}^\top}{\sqrt{d}})\mathbf{Q}}{\mathcal{Z}_1+\mathcal{Z}_2} + \frac{\exp(\frac{\mathbf{x}\mathbf{T}^\top}{\sqrt{d}})\mathbf{T}}{\mathcal{Z}_1+\mathcal{Z}_2}\\ &= \frac{\mathcal{Z}_1\cdot\mathrm{softmax}(\frac{\mathbf{x}\mathbf{Q}^\top}{\sqrt{d}})\mathbf{Q}}{\mathcal{Z}_1+\mathcal{Z}_2} + \frac{\mathcal{Z}_2\cdot\mathrm{softmax}(\frac{\mathbf{x}\mathbf{T}^\top}{\sqrt{d}})\mathbf{T}}{\mathcal{Z}_1+\mathcal{Z}_2}.\end{aligned} \quad (8)$$

Based on the above dissection, we take $\eth, \lambda, \mu$ as follows:

$$\eth(\varrho_q, \varrho_k, \varrho_v) \equiv \mathrm{softmax}(\frac{\varrho_q\varrho_k^\top}{\sqrt{d}})\varrho_v, \quad (9)$$

$$\lambda := \frac{\mathcal{Z}_1}{\mathcal{Z}_1+\mathcal{Z}_2}, \quad \mu := \frac{\mathcal{Z}_2}{\mathcal{Z}_1+\mathcal{Z}_2}. \quad (10)$$

Token-level Attention Decomposition. Next, we extend the global analysis to a token-level form. Does there exist a function $\wp : \mathbf{D}_\wp \to \mathbb{R}^d$ such that, for any question matrix $\mathbf{Q} \in \mathbb{R}^{|\mathbf{Q}|\times d}$, thought matrix $\mathbf{T} \in \mathbb{R}^{|\mathbf{T}|\times d}$, and query vector $\mathbf{x} \in \mathbb{R}^d$, there exist scalars $\alpha, \{\beta_i\}_{i=1}^{|\mathbf{T}|}$ satisfy the following:

$$\begin{aligned}&\mathrm{Attn}\big(\mathbf{x}, \big[\mathbf{Q}^\top\ \mathbf{T}^\top\big]^\top, \big[\mathbf{Q}^\top\ \mathbf{T}^\top\big]^\top\big)\\ &= \alpha\cdot\mathrm{Attn}(\mathbf{x},\mathbf{Q},\mathbf{Q}) + \sum_i \beta_i\cdot\wp(\mathbf{x},\mathbf{T}_i,\mathbf{T}_i),\end{aligned} \quad (11)$$

where $\mathrm{Attn}(\cdot)$ represents the self-attention mechanism, and $\mathbf{T}_i$ denotes the specific i-th token in the thinking process $\mathbf{T}$?

To deal with this, we start by defining $\mathbf{T}_{1:i} \in \mathbb{R}^{i\times d}$ as the matrix formed by the first i rows of $\mathbf{T}$. By recursively using the decomposition from the global attention case, we obtain:

$$\begin{aligned}&\mathrm{Attn}\big(\mathbf{x}, \big[\mathbf{Q}^\top\ \mathbf{T}_{1:|\mathbf{T}|}^\top\big]^\top, \big[\mathbf{Q}^\top\ \mathbf{T}_{1:|\mathbf{T}|}^\top\big]^\top\big)\\ &= \lambda_1\cdot\mathrm{Attn}\big(\mathbf{x}, \big[\mathbf{Q}^\top\ \mathbf{T}_{1:|\mathbf{T}|-1}^\top\big]^\top, \big[\mathbf{Q}^\top\ \mathbf{T}_{1:|\mathbf{T}|-1}^\top\big]^\top\big)\\ &\quad + \mu_1\cdot\eth(\mathbf{x},\mathbf{T}_{|\mathbf{T}|},\mathbf{T}_{|\mathbf{T}|})\\ &= \lambda_1\lambda_2\cdot\mathrm{Attn}\big(\mathbf{x}, \big[\mathbf{Q}^\top\ \mathbf{T}_{1:|\mathbf{T}|-2}^\top\big]^\top, \big[\mathbf{Q}^\top\ \mathbf{T}_{1:|\mathbf{T}|-2}^\top\big]^\top\big)\\ &\quad + (\lambda_1\mu_2)\cdot\eth(\mathbf{x},\mathbf{T}_{|\mathbf{T}|-1},\mathbf{T}_{|\mathbf{T}|-1})\\ &\quad + \mu_1\cdot\eth(\mathbf{x},\mathbf{T}_{|\mathbf{T}|},\mathbf{T}_{|\mathbf{T}|})\\ &= \dots\\ &= (\prod_i \lambda_i)\cdot\mathrm{Attn}(\mathbf{x},\mathbf{Q},\mathbf{Q})\\ &\quad + \sum_i(\mu_{|\mathbf{T}|-i+1}\prod_{j=1}^{|\mathbf{T}|-i}\lambda_{|\mathbf{T}|-i-j+1})\cdot\eth(\mathbf{x},\mathbf{T}_i,\mathbf{T}_i).\end{aligned} \quad (12)$$

Based on the above dissection, we take $\wp, \alpha, \beta_i$ as follows:

$$\wp \equiv \eth, \quad (13)$$

$$\alpha := \prod_i \lambda_i, \quad \beta_i := \mu_{|\mathbf{T}|-i+1}\prod_{j=1}^{|\mathbf{T}|-i}\lambda_{|\mathbf{T}|-i-j+1}. \quad (14)$$

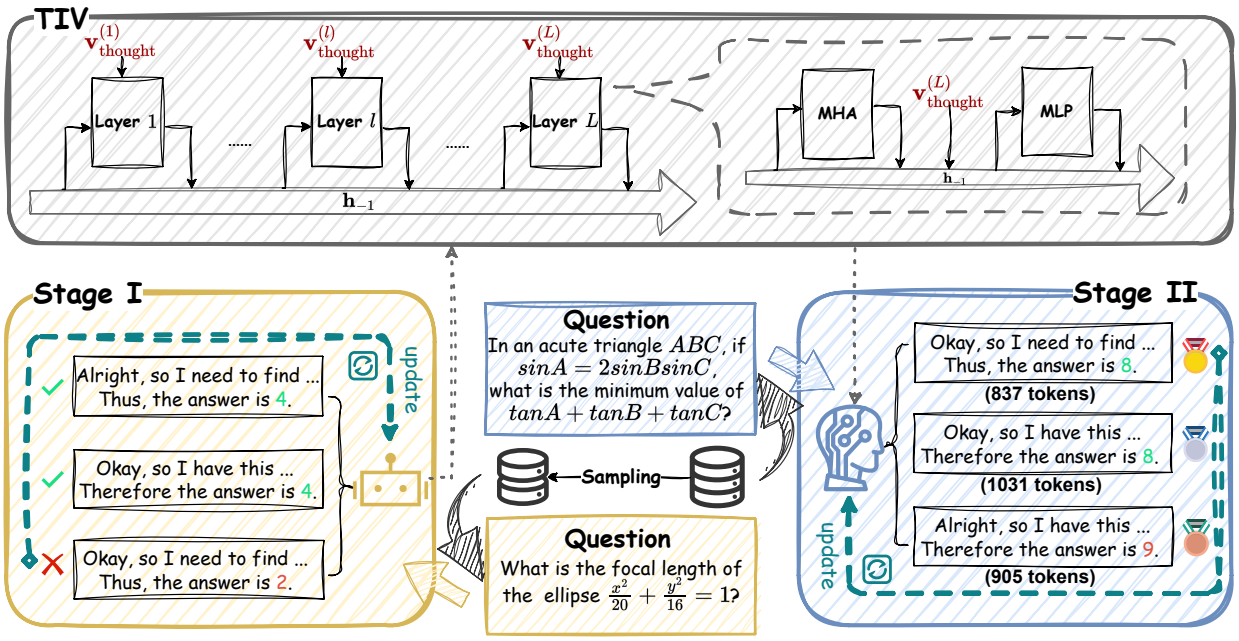

Figure 2: Overview of our framework. Stage I calibrates reasoning in the model without considering output length. In Stage II, thought vectors are injected into the calibrated model to optimize for both reasoning accuracy and efficiency.

Mapping Thought to Vector. If we let the query vector $\mathbf{x}$ in Equation (11) correspond to the token used to generate the final answer, then the attention contribution of the entire thinking process $\mathbf{T}$ can be equivalently expressed as follows:

$$\mathbf{v}_{\text{thought}} := \sum_i \beta_i \cdot \wp(\mathbf{x}, \mathbf{T}_i, \mathbf{T}_i) \in \mathbb{R}^d. \tag{15}$$

Here, $\mathbf{v}_{\text{thought}}$ is accordingly defined as the *thought vector*—a concise and semantically meaningful representation of the thought that integrates seamlessly with the attention output. Constructed at the token level, it can also represent partially redundant segments. By strategically injecting $\mathbf{v}_{\text{thought}}$ at the right place, we can achieve the effect of explicit reasoning.

Vector Injection. In autoregressive decoders, the hidden state of the final token, denoted as $\mathbf{h}_{-1}$, aggregates causal attention over all preceding tokens. Therefore, we propose to inject the thought vector $\mathbf{v}_{\text{thought}}$ to $\mathbf{h}_{-1}$ to simulate thinking:

$$\mathbf{h}_{-1}^{(l)} \leftarrow \mathbf{h}_{-1}^{(l)} + \mathbf{v}_{\text{thought}}^{(l)}, \tag{16}$$

where $\mathbf{v}_{\text{thought}}^{(l)}$ is a vector injected into layer l, and $\mathbf{h}_{-1}^{(l)}$ is the post-attention hidden state of the last token in layer l.

4.3 Optimization Strategy

As Equation (15) shows, the thought vector $\mathbf{v}_{\text{thought}}$ is conditioned on the query vector $\mathbf{x}$ used for generating the final answer, and thus tends to fluctuate across inputs. To derive a more stable and generalizable representation, we propose a two-stage optimization strategy based on Group Relative Policy Optimization (GRPO) (Shao et al. 2024): the first stage calibrates the model's reasoning distribution, and the second stage distills it into compact vector representations.

Data Preparation. Let $\mathcal{D}_{\text{train}} = \{(q_i, a_i^*)\}_i$ represent the full training dataset, where q_i is a question and a_i^* is its ground-truth answer. The calibration subset $\mathcal{D}_{\text{cal}}$ is formed by uniformly sampling up to 1% or 50 samples from $\mathcal{D}_{\text{train}}$. Despite randomness, the inherent diversity of $\mathcal{D}_{\text{train}}$ ensures that $\mathcal{D}_{\text{cal}}$ effectively represents various question types.

Stage I: Reasoning Calibration. For the same question, valid reasoning paths may vary in length, causing gradient conflict and leaving the model π_θ uncertain about how much to reason. Thus, we calibrate π_θ on $\mathcal{D}_{\text{cal}}$ to align with the downstream task and stabilize reasoning for compression.

Given question q_i, we sample rollouts $\{(t_{i,j}, a_{i,j})\}_j$ from π_θ , where $t_{i,j}$ is the thinking process and $a_{i,j}$ is the answer. The reward and group-relative advantage are as follows:

$$r_{i,j}^{\text{cal}} = \mathbb{I}(a_{i,j} = a_i^*) \xrightarrow[\text{in } r_{i,\cdot}^{\text{cal}}]{\text{grouped}} A_{i,j}^{\text{cal}}, \tag{17}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function (1 for true, 0 for false). The loss function for each rollout is then defined as:

$$\mathcal{L}_{i,j}^{\text{cal}}(\theta) = \min\big(w_{i,j}^{\text{cal}} A_{i,j}^{\text{cal}}, \text{clip}(w_{i,j}^{\text{cal}}, 1-\epsilon, 1+\epsilon) A_{i,j}^{\text{cal}}\big) - \beta^{\text{cal}} \mathbb{D}_{\text{KL}}\big(\pi_\theta(t_{i,j}, a_{i,j}|q_i) \| \pi_{\text{ref}}(t_{i,j}, a_{i,j}|q_i)\big), \tag{18}$$

where $w_{i,j}^{\text{cal}} = \frac{\pi_\theta(t_{i,j}, a_{i,j}|q_i)}{\pi_{\theta_{\text{old}}}(t_{i,j}, a_{i,j}|q_i)}$ denotes the importance weight between current and previous policy parameters θ and θ_{old}, $\text{clip}(\cdot, 1-\epsilon, 1+\epsilon)$ clamps within $[1-\epsilon, 1+\epsilon]$, $\mathbb{D}_{\text{KL}}$ denotes the KL divergence, $\beta^{\text{cal}} > 0$ is the KL coefficient, and π_{ref} is a slowly-updated reference policy for regularization.

In the meanwhile, we track the average number of generated tokens in correct trajectories, denoted as $\bar{n}_{\text{token}}$, to guide length-aware optimization in the subsequent stage.

Stage II: Vector-based Compression. Building upon the calibrated reasoning distribution, we compress π_θ by injecting thought vectors as specified in Equation (16), and fine-tune the resulting policy $\pi_{(\theta,\mathbf{v})}$ on $\mathcal{D}_{\text{train}}$ with a length-aware reward and its corresponding group-relative advantage:

$$r_{i,j}^{\text{cmp}} = \begin{cases} 1 - \eta \frac{n_{\text{token},i,j}}{\bar{n}_{\text{token}}}, & \text{if } a_{i,j} = a_i^*, \\ 0, & \text{if } a_{i,j} \neq a_i^*, \end{cases} \xrightarrow[\text{in } r_{i,\cdot}^{\text{cmp}}]{\text{grouped}} A_{i,j}^{\text{cmp}}, \quad (19)$$

where $n_{\text{token},i,j}$ is the total number of tokens generated in the j-th rollout for question q_i and $\eta > 0$ is a hyper-parameter for the length penalty. The loss function for each rollout is:

$$\mathcal{L}_{i,j}^{\text{cmp}}(\theta, \mathbf{v}) = \min(w_{i,j}^{\text{cmp}} A_{i,j}^{\text{cmp}}, \text{clip}(w_{i,j}^{\text{cmp}}, 1-\epsilon, 1+\epsilon) A_{i,j}^{\text{cmp}}) - \beta^{\text{cmp}} \mathbb{D}_{\text{KL}}\big(\pi_{(\theta,\mathbf{v})}(t_{i,j}, a_{i,j}|q_i) \| \pi_{\text{ref}}(t_{i,j}, a_{i,j}|q_i)\big), \quad (20)$$

where $w_{i,j}^{\text{cmp}} = \frac{\pi_{(\theta,\mathbf{v})}(t_{i,j},a_{i,j}|q_i)}{\pi_{(\theta,\mathbf{v})_{\text{old}}}(t_{i,j},a_{i,j}|q_i)}$ represents the importance weight, and $\beta^{\text{cmp}} > 0$ indicates the KL coefficient.

5 Experiments

5.1 Experimental Setup

Models and Datasets. We focus on two LRMs, namely DeepSeek-R1-Distill-Qwen-1.5B and DeepSeek-R1-Distill-Qwen-7B (Guo et al. 2025), both of which have exhibited remarkable reasoning capabilities. These models are trained on the DeepScaleR dataset (Luo et al. 2025b), containing around 40,000 math problems sourced from diverse competitions and resources, including AIME (1983-2023), AMC, Omni-Math (Gao et al. 2025), and STILL (Min et al. 2024).

Benchmarks. We comprehensively evaluate performance on three benchmarks, ordered by increasing difficulty:

- **GSM8K** (Cobbe et al. 2021): A collection of 1319 grade-school-level math problems for entry-level reasoning.
- **MATH 500** (Lightman et al. 2024): A dataset comprising 500 high-school competition math problems.
- **AIME 2024**: A highly challenging dataset containing 30 Olympiad-level math problems from the American Invitational Mathematics Examination held in 2024.

Evaluation Metrics. We evaluate the performance of the trained models utilizing the following three key metrics:

- **Accuracy**: We measure the proportion of problems for which the model generates a correct final answer.
- **Length**: We calculate the average number of tokens contained within the solutions generated by the model.
- **AES**: To assess the trade-off between accuracy and computational cost, we apply the Accuracy-Efficiency Score (AES) from O1-Pruner (Luo et al. 2025a), calculated as:

$$\text{AES} = \begin{cases} \xi_1 \Delta\text{Acc} - \xi_3 \Delta\text{Length}, & \text{if } \Delta\text{Acc} \geq 0, \\ \xi_2 \Delta\text{Acc} - \xi_3 \Delta\text{Length}, & \text{if } \Delta\text{Acc} < 0, \end{cases} \quad (21)$$

where ΔAcc and ΔLength represent the relative changes in accuracy and output length compared to the original reasoning model. ξ_1, ξ_2, and ξ_3 weight accuracy gains, accuracy losses, and length reductions, respectively.

Implementation Details. All RL experiments are built on the VeRL framework (Sheng et al. 2025). Training is conducted with a batch size of 128, a constant learning rate of 1e-6, and 4 rollouts per prompt, running on eight NVIDIA A100 GPUs (80GB VRAM per GPU, 640GB in total).

Following O1-Pruner (Luo et al. 2025a), we set the AES weights to ξ_1 at 3, ξ_2 at 5, and ξ_3 at 1. For the GSM8K and MATH 500 datasets, we generate one deterministic response per problem. For the 30-problem AIME 2024 dataset, we generate 16 independent responses per problem and report the average to minimize randomness. Decoding processes are executed using the vLLM framework (Kwon et al. 2023).

5.2 Baselines

We compare TIV with the following baseline approaches:

- **NoThink**: A special control token "`</think>`", appended immediately after the prompt, instructs the model to bypass the generation of intermediate thoughts and transition directly into the final response stage.
- **DPO** (Rafailov et al. 2023): For each problem, we generate multiple responses, form preference pairs with the shortest correct response as preferred and the longest as dispreferred, and fine-tune the model using these pairs.
- **RFT** (Reject Fine-Tuning): We sample responses from *Think* and *NoThink* models, select the correct response from *NoThink* if it is better, otherwise select the correct *Think* response, and fine-tune the model with these.
- **OverThink** (Chen et al. 2025b): Long-thinking response is labeled as negative, and the first two correct attempts as positive. Then, fine-tuning is performed using SimPO (Meng, Xia, and Chen 2024) to optimize the model.
- **O1-Pruner** (Luo et al. 2025a): Responses are generated offline, with length and accuracy baselines set. Fine-tuning is performed using PPO (Schulman et al. 2017).
- **DAST** (Shen et al. 2025): A response-length budget is estimated from sampled outputs, preference data is constructed, and fine-tuning is performed using SimPO.

To ensure a fair comparison, we implement all the baselines, utilizing the DeepScaleR dataset where training is required.

5.3 Main Results

Table 1 shows results on three math reasoning benchmarks. Our method, TIV, consistently achieves the best balance of accuracy and efficiency for both the 1.5B and 7B models. On average, TIV retains over 99% of the original accuracy while reducing output length by more than 65% compared to the original model (*Think* baseline). In specific cases, such as GSM8K with the 7B model, the output length is compressed by over 80%. Additionally, TIV achieves the highest AES scores across all benchmarks, demonstrating its ability to preserve reasoning quality while eliminating redundancy. These results highlight the superior effectiveness of vector-based reasoning compression over token-level generation.

5.4 Ablation Study

Table 2 presents the ablation study conducted on the R1-1.5B model. Our full approach, which effectively integrates

Model	Method	GSM8K			MATH 500			AIME 2024		
		Acc↑	Length↓	AES↑	Acc↑	Length↓	AES↑	Acc↑	Length↓	AES↑
R1-1.5B	*Think*	*75.3*	*635.9*	-	*82.0*	*4972.2*	-	*29.0*	*16239.6*	-
	NoThink	*73.8*	*268.0*	-	*68.2*	*763.5*	-	*9.8*	*3943.6*	-
	DPO	74.6	532.8	0.12	83.8	3782.6	0.31	30.2	14529.2	0.23
	RFT	72.4	705.3	-0.30	73.6	4426.7	-0.40	24.8	15017.5	-0.65
	O1-Pruner	71.3	293.8	0.27	83.6	3262.0	0.40	28.5	13926.8	0.06
	DAST	73.6	386.0	0.28	**84.4**	2480.3	0.59	26.5	10437.9	-0.07
	OverThink	73.6	467.0	0.15	82.6	4209.0	0.18	27.9	15152.1	-0.12
	TIV (Ours)	**75.9**	**236.4**	**0.65**	82.4	**1283.3**	**0.76**	**30.8**	**6605.4**	**0.78**
R1-7B	*Think*	*87.4*	*725.7*	-	*91.2*	*3886.1*	-	*52.9*	*13187.2*	-
	NoThink	*86.2*	*263.0*	-	*76.0*	*589.6*	-	*18.1*	*1982.2*	-
	DPO	85.2	423.8	0.29	**92.6**	2648.3	0.36	51.9	11140.9	0.06
	RFT	85.7	385.4	0.37	85.6	2560.2	0.03	48.7	12752.0	-0.36
	O1-Pruner	87.1	459.4	0.35	87.4	2680.3	0.10	48.5	12433.1	-0.36
	DAST	86.2	488.4	0.26	90.6	2289.8	0.38	45.0	9686.5	-0.48
	OverThink	85.8	453.3	0.28	90.4	2575.6	0.29	**52.5**	11181.5	0.11
	TIV (Ours)	**87.9**	**110.2**	**0.87**	88.6	**1313.6**	**0.52**	48.5	**6057.0**	**0.12**

Table 1: Performance comparison of different methods for efficient reasoning in LRMs across three mathematical benchmarks. "Acc" indicates the average accuracy (%), "Length" represents the average number of generated tokens, and "AES" denotes the Accuracy-Efficiency Score. Notably, due to the limited number of problems in AIME 2024, we sample 16 responses per problem and to compute the average. Arrows "↑" and "↓" indicate that higher or lower values are preferred, respectively. Bold values denote the best performance. R1-1.5B and R1-7B refer to DeepSeek-R1-Distill-Qwen-1.5B and 7B, respectively.

Method	GSM8K			MATH 500			AIME 2024		
	Acc↑	Length↓	AES↑	Acc↑	Length↓	AES↑	Acc↑	Length↓	AES↑
TIV (Ours)	75.9	236.4	0.65	82.4	1283.3	0.76	30.8	6605.4	0.78
w/o thought injection	74.3	333.7	0.41	80.4	1553.3	0.59	25.8	6726.2	0.03
w/o length penalty	74.4	452.2	0.23	81.6	2811.1	0.41	30.8	9021.0	0.63
w/o calibration	72.3	357.8	0.24	79.0	1898.3	0.44	24.2	7655.0	-0.30

Table 2: Ablation results on the R1-1.5B model.

thought injection, length penalty, and calibration, achieves the highest AES across all benchmarks. Results further confirm the complementary, synergistic roles of each module in preserving reasoning quality while minimizing verbosity.

5.5 More Analysis

Effect of Rollout Count. As shown in Figure 3, increasing rollouts from 2 to 5 yields clear improvements in both reward and response compression. Beyond 5 rollouts, however, the performance gain plateaus, with only marginal additional benefits. Based on this, TIV achieves near-optimal performance with as few as 5 rollouts per prompt, reducing computational overhead while maintaining training quality.

Effect of Vector Count. To evaluate the representational capacity for vector-based reasoning, we change the number of injected vectors while maintaining the injection point of TIV unchanged. Figure 4 reveals that just a single vector can achieve comparable reward and response compression as multiple vectors, indicating that TIV effectively encodes

Figure 3: Reward and response length with different rollout counts in compression training. "rollout.n": rollout count.

rich reasoning semantics into a compact representation for efficient reasoning with minimal parameter overhead.

Injection Point: MHA vs. MHA and MLP. Motivated by MHA's cross-token interactions and MLP's token-wise transformations, we explore performing injection after MLP.

Injection Point	GSM8K			MATH 500			AIME 2024		
	Acc↑	Length↓	AES↑	Acc↑	Length↓	AES↑	Acc↑	Length↓	AES↑
MHA & MLP	76.1	217.3	0.69	82.4	1266.2	0.76	32.5	6532.0	0.96
only MHA	75.9	236.4	0.65	82.4	1283.3	0.76	30.8	6605.4	0.78

Table 3: Performance comparison for different injection points.

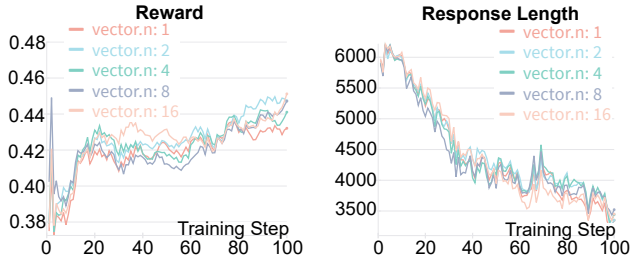

Figure 4: Reward and response length with different vector counts in compression training. "vector.n": vector count.

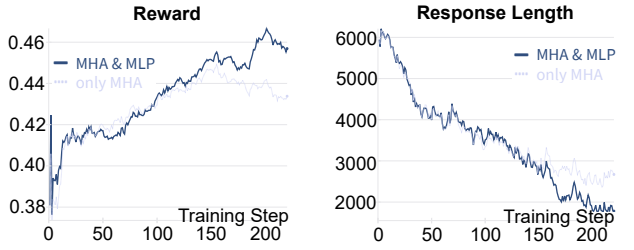

Figure 5: Reward and response length with different injection points in compression training. "MHA & MLP" refers to injecting vectors after both MHA and MLP; "only MHA" refers to injecting vectors only after MHA (TIV method).

Figure 5 and Table 3 demonstrate that this brings slight but consistent improvements over MHA-only injection.

Computational Overhead Verification. To ensure TIV's efficiency does not incur runtime cost, we measure its inference latency. Table 4 shows that TIV increases TTFT by less than 2% and TPOT by less than 4%, maintaining baseline responsiveness while supporting token-efficient reasoning.

Generalization to OOD Scenario. To investigate TIV's generalization to out-of-distribution (OOD) tasks, we assess on the MMLU benchmark (Hendrycks et al. 2021), which includes 14K multiple-choice questions across 57 subjects not seen during training. Table 5 reveals that TIV reduces response length while maintaining or improving accuracy. On R1-1.5B, it increases accuracy by 15.8% with a 62.5% reduction in length; On R1-7B, it preserves accuracy (+0.2%) while cutting length by 56.0%. These results demonstrate that TIV generalizes effectively across different domains.

Case Study. To understand how TIV operates in practice, we examine key cases from our benchmark datasets. Baseline models often generate verbose reasoning traces, marked by redundant question restatements, trivial step commentary, and unhelpful self-corrections, which inflate response length without enhancing quality. In contrast, TIV replaces such token-level reasoning with a injected vector encoding core reasoning. This design enables the model to produce concise answers while preserving essential logical structure.

Model	Method	TTFT	TPOT
R1-1.5B	*Think*	*0.0297s*	*0.0257s*
	TIV (Ours)	0.0299s (+0.7%)	0.0267s (+3.9%)
R1-7B	*Think*	*0.0299s*	*0.0264s*
	TIV (Ours)	0.0304s (+1.7%)	0.0270s (+2.3%)

Table 4: Inference latency with and without vector injection. "TTFT": Time to First Token; "TPOT": Time per Output Token. Measured using the prompt "What is one plus one?" with a 128-token output on a single A100 GPU (80GB).

Model	Method	MMLU	
		Acc↑	Length↓
R1-1.5B	*Think*	*29.1*	*1714.2*
	TIV (Ours)	33.7 (+15.8%)	643.3 (-62.5%)
R1-7B	*Think*	*58.5*	*1138.5*
	TIV (Ours)	58.6 (+0.2%)	500.6 (-56.0%)

Table 5: Performance on the MMLU benchmark.

6 Conclusion

This paper introduces TIV, a novel framework designed to enhance the reasoning efficiency of LRMs. TIV employs attention decomposition and injects learnable vectors into the post-attention hidden states of the final token. We further propose a two-stage RL strategy that calibrates and distills the reasoning behavior into compact vector representations.

Extensive experiments on multiple reasoning benchmarks confirm that TIV significantly reduces response length while maintaining high accuracy, introduces negligible inference overhead, and generalizes well to out-of-distribution tasks.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant Nos. 62272334, 62572335, 62572336, 62577050) and the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; et al. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*.

Chen, A.; Li, A.; Gong, B.; Jiang, B.; Fei, B.; Yang, B.; Shan, B.; Yu, C.; Wang, C.; Zhu, C.; Xiao, C.; et al. 2025a. MiniMax-M1: Scaling Test-Time Compute Efficiently with Lightning Attention. *arXiv preprint arXiv:2506.13585*.

Chen, X.; Xu, J.; Liang, T.; He, Z.; Pang, J.; Yu, D.; Song, L.; Liu, Q.; Zhou, M.; Zhang, Z.; Wang, R.; Tu, Z.; Mi, H.; and Yu, D. 2025b. Do NOT Think That Much for 2+3=? On the Overthinking of Long Reasoning Models. In *The Forty-second International Conference on Machine Learning*.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; et al. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*.

Du, A.; Gao, B.; Xing, B.; Jiang, C.; Chen, C.; Li, C.; Xiao, C.; Du, C.; Liao, C.; Tang, C.; Wang, C.; Zhang, D.; Yuan, E.; Lu, E.; et al. 2025. Kimi k1.5: Scaling Reinforcement Learning with LLMs. *arXiv preprint arXiv:2501.12599*.

Gao, B.; Song, F.; Yang, Z.; Cai, Z.; Miao, Y.; Dong, Q.; Li, L.; Ma, C.; Chen, L.; Xu, R.; Tang, Z.; et al. 2025. Omni-MATH: A Universal Olympiad Level Mathematic Benchmark for Large Language Models. In *The Thirteenth International Conference on Learning Representations*.

Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

Guo, D.; Yang, D.; Zhang, H.; et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*.

Han, T.; Wang, Z.; Fang, C.; Zhao, S.; Ma, S.; and Chen, Z. 2025. Token-Budget-Aware LLM Reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021. Measuring Massive Multitask Language Understanding. In *The Ninth International Conference on Learning Representations*.

Jaech, A.; Kalai, A.; Lerer, A.; Richardson, A.; El-Kishky, A.; Low, A.; Helyar, A.; Madry, A.; Beutel, A.; Carney, A.; Iftimie, A.; Karpenko, A.; Passos, A. T.; et al. 2024. OpenAI o1 System Card. *arXiv preprint arXiv:2412.16720*.

Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C. H.; Gonzalez, J. E.; Zhang, H.; and Stoica, I. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Li, X. L.; and Liang, P. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Lightman, H.; Kosaraju, V.; Burda, Y.; Edwards, H.; Baker, B.; Lee, T.; Leike, J.; Schulman, J.; Sutskever, I.; and Cobbe, K. 2024. Let's Verify Step by Step. In *The Twelfth International Conference on Learning Representations*.

Liu, S.; Ye, H.; Xing, L.; and Zou, J. 2024. In-context Vectors: Making In Context Learning More Effective and Controllable Through Latent Space Steering. In *The Forty-first International Conference on Machine Learning*.

Luo, H.; Shen, L.; He, H.; Wang, Y.; Liu, S.; Li, W.; Tan, N.; Cao, X.; and Tao, D. 2025a. O1-Pruner: Length-Harmonizing Fine-Tuning for O1-Like Reasoning Pruning. In *The Second AI for Math Workshop @ ICML 2025*.

Luo, M.; Tan, S.; Wong, J.; Shi, X.; et al. 2025b. DeepScaleR: Surpassing O1-Preview with a 1.5B Model by Scaling RL. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2. Notion Blog.

Meng, Y.; Xia, M.; and Chen, D. 2024. SimPO: Simple Preference Optimization with a Reference-Free Reward. In *Advances in Neural Information Processing Systems*.

Min, Y.; Chen, Z.; Jiang, J.; Chen, J.; Deng, J.; Hu, Y.; Tang, Y.; Wang, J.; Cheng, X.; et al. 2024. Imitate, Explore, and Self-Improve: A Reproduction Report on Slow-thinking Reasoning Systems. *arXiv preprint arXiv:2412.09413*.

Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C. D.; Ermon, S.; and Finn, C. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *Advances in Neural Information Processing Systems*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*.

Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; et al. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*.

Shen, Y.; Zhang, J.; Huang, J.; et al. 2025. DAST: Difficulty-Adaptive Slow-Thinking for Large Reasoning Models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*.

Sheng, G.; Zhang, C.; Ye, Z.; Wu, X.; Zhang, W.; Zhang, R.; Peng, Y.; Lin, H.; and Wu, C. 2025. HybridFlow: A Flexible and Efficient RLHF Framework. In *Proceedings of the Twentieth European Conference on Computer Systems*.

Sui, Y.; Chuang, Y.-N.; Wang, G.; Zhang, J.; Zhang, T.; Yuan, J.; Liu, H.; Wen, A.; et al. 2025. Stop Overthinking: A Survey on Efficient Reasoning for Large Language Models. *Transactions on Machine Learning Research*.

Sutton, R. S.; and Barto, A. G. 1999. Reinforcement Learning. *Journal of Cognitive Neuroscience*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; et al. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; ichter, b.; Xia, F.; Chi, E.; et al. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*.