# HAPO: Training Language Models to Reason Concisely via History-Aware Policy Optimization

**Chengyu Huang***, **Zhengxin Zhang***, **Claire Cardie**
Department of Computer Science, Cornell University
ch2263@cornell.edu, zz865@cornell.edu, ctc9@cornell.edu

### Abstract

While scaling the length of responses at test-time has been shown to markedly improve the reasoning abilities and performance of large language models (LLMs), it often results in verbose outputs and increases inference cost. Prior approaches for efficient test-time scaling, typically using universal budget constraints or query-level length optimization, do not leverage historical information from previous encounters with the same problem during training. We hypothesize that this limits their ability to progressively make solutions more concise over time. To address this, we present History-Aware Policy Optimization (HAPO), which keeps track of a history state (e.g., the minimum length over previously generated correct responses) for each problem. HAPO employs a novel length reward function based on this history state to incentivize the discovery of correct solutions that are more concise than those previously found. Crucially, this reward structure avoids overly penalizing shorter incorrect responses with the goal of facilitating exploration towards more efficient solutions. By combining this length reward with a correctness reward, HAPO jointly optimizes for correctness and efficiency. We use HAPO to train DeepSeek-R1-Distill-Qwen-1.5B, DeepScaleR-1.5B-Preview, and Qwen-2.5-1.5B-Instruct, and evaluate HAPO on several math benchmarks that span various difficulty levels. Experiment results demonstrate that HAPO effectively induces LLMs' concise reasoning abilities, producing length reductions of 33-59% with accuracy drops of only 2-5%.

**Code** — https://github.com/HCY123902/HAPO

## Introduction

Recent advances in large language models (LLMs) have highlighted the effectiveness of test-time scaling (DeepSeek-AI et al. 2025; Jaech et al. 2024; Zeng et al. 2025; Luo et al. 2025c,b; Qwen 2025), which enables LLMs to develop longer and more sophisticated reasoning behaviors such as self-reflection and verification, substantially improving their performance across a wide range of tasks. While generating long reasoning chains can significantly boost models' accuracy, it induces verbosity, increases inference cost, and incurs substantial computational and memory overhead due to the quadratic attention complexity and linear key-value (KV) cache growth that are inherent in transformer architectures (Vaswani et al. 2017). Notably, even for very simple problems like "What is the answer to 2 plus 3?", these reasoning models tend to engage in unnecessarily lengthy reasoning and produce solutions that span hundreds of tokens when only a few steps would suffice (Kumar et al. 2025a; Chen et al. 2024; Sui et al. 2025). This inefficiency, often referred to as *overthinking*, limits the practical deployment of reasoning models in real-world applications.

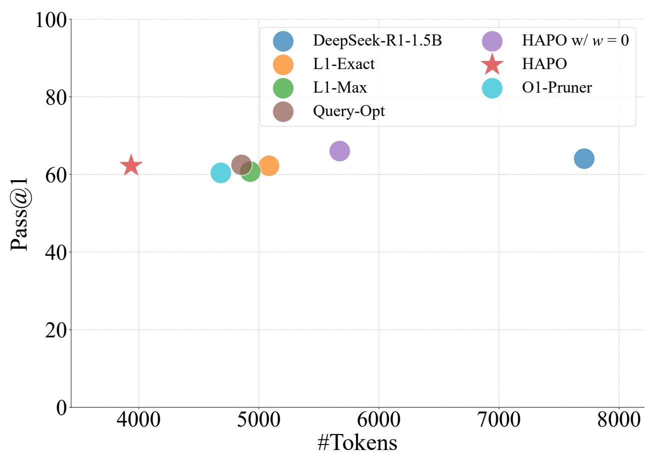



Figure 1: Results averaged across GSM8K, MATH500, and AIME2024. Pass@1 is the average accuracy across multiple sampled responses per query; #Tokens is the average number of tokens in the responses. Compared with the base model DeepSeek-R1-1.5B, HAPO significantly reduces response length while preserving accuracy, giving a better tradeoff than other baselines.

There have been initial explorations into limiting the generation length of reasoning models, either by enforcing a fixed, universal budget constraint to be applied to each question (i.e., universal budget forcing) (Fu et al. 2024; Muennighoff et al. 2025; Han et al. 2024; Aggarwal and Welleck 2025; Hou et al. 2025), or by dynamically optimizing the reasoning length for each question (Luo et al. 2025a; Arora and Zanette 2025; Yeo et al. 2025; She et al. 2025; Qu et al. 2025) (i.e., query-level optimization). *Universal budget forcing methods* intervene when a predefined token limit

*These authors contributed equally.

is reached. For reinforcement learning (RL) based methods, this usually results in a non-positive reward. For solutions that use supervised-finetuning (SFT) or few-shot prompting, it implies forcing a response length cutoff by inserting an END-OF-THINKING token. For example, S1 (Muennighoff et al. 2025) introduces a length control mechanism that injects special tokens (e.g., FINAL ANSWER) when responses are too long. Despite reducing output length with respect to a base model, such fixed token budgets cannot address the overthinking problem; instead they unnecessarily constrain exploratory reasoning on harder questions and still allow excessive token usage on simpler ones.

In contrast, *query-level optimization methods* learn a different budget for each query. For example, Arora and Zanette (2025) compare the lengths of the responses to the same query in each training step and reward shorter responses while penalizing longer ones. Although these methods alleviate the overthinking problem to some extent, they cannot leverage historical information derived from previous encounters with the same query during training, i.e., across multiple training steps. This limits their ability to progressively make solutions more concise over time. For example, suppose responses sampled initially were considerably longer than those sampled in a subsequent encounter with the same problem. We argue that in this case, the model has become more efficient and, therefore, should receive a positive length reward in the later encounter. Unfortunately, existing approaches—including both universal budget forcing and query-level optimization methods—lack the mechanisms to utilize such historical signals.

In this paper, we propose **History-Aware Policy Optimization (HAPO)**. As illustrated in the left side of Figure 2, we introduce a history state $h_i$ that records the minimum length of the correct responses observed in previous encounters with problem $x_i$. We dynamically adjust the length reward (Figure 2 right) based on $h_i$, assigning higher rewards to correct responses that are shorter than $h_i$. This encourages the model to generate correct solutions that are more concise than previously known ones, which is a different incentive from prior approaches. Additionally, unlike existing methods, we do not overly penalize short incorrect responses, under the assumption that they may represent exploratory attempts toward more concise solutions. HAPO jointly optimizes for both accuracy and efficiency by combining the length reward with an accuracy reward.

We use HAPO to train three open-source reasoning models, including DeepSeek-R1-Distill-Qwen-1.5B (DeepSeek-AI et al. 2025), DeepScaleR-1.5B-Preview, and Qwen-2.5-1.5B-Instruct, and evaluate the models on math problems from three standard datasets—GSM8K (Cobbe et al. 2021), MATH500 (Hendrycks et al. 2021), and AIME2024. Overall we find that HAPO effectively elicits LLMs' concise reasoning abilities: when averaged across the three benchmarks, HAPO produces a 49% length reduction with only 2% accuracy drop for DeepSeek-R1-Distill-Qwen-1.5B; a 33% length reduction with 5% accuracy drop for DeepScaleR-1.5B-Preview; and a 59% length reduction with 2% accuracy drop for Qwen-2.5-1.5B-Instruct. Using DeepSeek-R1-Distill-Qwen-1.5B as the base model, we compare HAPO with baselines from universal budget forcing (the L1 models (Aggarwal and Welleck 2025)) and query-level optimization (Query-Opt (Arora and Zanette 2025)) paradigms. Here, our results show that HAPO enables a better overall length-correctness trade-off. Notably, HAPO uses 19% fewer tokens than the most competitive baseline while achieving similar accuracy (see Figure 1).

## Related Work

**Test-time scaling** Prior work (Wei et al. 2022; Snell et al. 2025; Wu et al. 2024) has shown that increasing test-time computation can improve the performance of large language models (LLMs), particularly in reasoning-intensive tasks such as math problem solving and code generation. Wei et al. (2022) demonstrates that generating intermediate reasoning steps (i.e., chain-of-thought (CoT)) can effectively elicit the reasoning capabilities of LLMs, inspiring a wave of follow-up research. Many subsequent methods focus on parallel scaling, where multiple responses are sampled from the model and aggregated into a final answer, either through an external reward model or an internal voting mechanism. Representative approaches include Best-of-N sampling (Wu et al. 2024), beam search (Snell et al. 2025), majority voting (Wang et al. 2023) and its weighted variant (Li et al. 2023), as well as tree-based search methods (Yao et al. 2023; Xin et al. 2025; Besta et al. 2024).

Another line of work focuses on scaling the *length* of the response rather than the *number* of responses. It aims to enable LLMs to engage in more thorough reasoning and support self-correction and verification (Kumar et al. 2025b). With the success of OpenAI's o1 (Jaech et al. 2024) model, more recent efforts have been made along this line. Notable efforts include DeepSeek's R1 models (DeepSeek-AI et al. 2025), Qwen's QwQ-series models, and other work that reproduces long CoT capabilities in smaller-scale models (Zeng et al. 2025; Muennighoff et al. 2025). A common feature of how most of these models are trained is the use of reinforcement learning (RL) algorithms (Schulman et al. 2017; Shao et al. 2024) and rule-based rewards. This training paradigm encourages models to generate increasingly long CoT in order to arrive at the correct answer.

**Efficient reasoning** While test-time scaling with long CoT significantly improves accuracy, it comes at the cost of computational inefficiency. In particular, reasoning models often produce verbose and unnecessary reasoning when solving simple problems—a phenomenon commonly referred to as *overthinking* (Sui et al. 2025). Empirically, we observe that reasoning models such as DeepSeek-R1-Distill-Qwen-1.5B can generate over 6K tokens when solving problems from the MATH dataset (Hendrycks et al. 2021), whereas human-written solutions only have around 200 tokens on average.

A variety of methods have been proposed to address this issue. For example, some approaches (Geiping et al. 2025; Hao et al. 2024; Cheng and Durme 2024; Su et al. 2024; Shen et al. 2025a,c) replace discrete vocabulary tokens with latent tokens in the continuous embedding space to perform reasoning. Other methods remain in the vocabulary space and focus on prompt optimization or additional training. Xu et al. (2025) designs a prompt that explicitly encourages conciseness at each reasoning step, while Xia et al. (2025); Han et al. (2024); Shen et al. (2025b) employ offline training algorithms to teach LLMs to generate concise yet correct responses. However, such offline training typically requires carefully curated supervision data, which can be expensive and labor-intensive to obtain.

In contrast, online RL training integrates length control directly within the reward signal. While the design of such length rewards varies across prior work, they generally fall into two categories: (1) Universal rewards, which compare response length against a predefined, universal budget and penalize responses exceeding this budget (Hou et al. 2025; Aggarwal and Welleck 2025), or utilize a universal penalty function (Yeo et al. 2025); (2) Query-level rewards, which are computed according to LLMs' performance specific to each query. Many query-level rewards are comparison-based; some methods compare the lengths of responses to the same query within a batch, penalizing relatively longer ones (Team et al. 2025; Arora and Zanette 2025), while others compare the current response length against that of a reference model or ground truth (Luo et al. 2025a; She et al. 2025). We argue that universal rewards are suboptimal due to their lack of adaptivity, necessitating a manual specification of budget estimates or penalty functions. Although query-level rewards seem more promising, current designs need improvement. For instance, in-batch comparison may not achieve a global optimum (potentially rewarding long responses if all outputs in a batch are long), while comparison against a fixed reference can lead to rapid reward saturation once the model consistently outperforms the reference.

In this work, we instead follow the idea of utilizing past information from training history (Zhang et al. 2023; Shinn et al. 2023; Le et al. 2025), to dynamically adjust the length reward per query, thereby providing the most up-to-date stimulus for the LLMs to compress their responses according to their own performance in the previous training steps.

## History-Aware Length Reward

In this section, we will first define our task setting and then describe our history-aware length reward.

### Task Setup

Denote a $\theta$ parameterized LLM as $p_\theta$. Using the GRPO algorithm[1] (Shao et al. 2024) we train $p_\theta$ on a dataset $\mathcal{D} = \{(x_i, a_i^*)\}_{i=1}^N$, where $x_i$ is a query (e.g., a math problem) and $a_i^*$ is its ground truth answer (e.g, a number). For each query $x_i$ in a batch, a set of responses $Y_i = \{y_0, y_1, ...\}$ is sampled from the current LLM $p_\theta$. Following the procedure in (DeepSeek-AI et al. 2025), we extract a candidate final answer $a_i$ from each response $y_i$. We train $p_\theta$ for multiple epochs; thus the model encounters each $x_i$ multiple times and we use $Y_i^{(j)}$ to refer to the set of responses generated by $p_\theta$ in its $j$-th encounter with $x_i$. Computation of the reward associated with $x_i$ is described next.

[1]We explore other RL approaches such as PPO (Schulman et al. 2017) in the Appendix.

### Reward Computation and History Update

The HAPO reward function for the $j$-th occurrence of $x_i$ is comprised of two parts, an accuracy reward, $ra_i^{(j)}$, and a length reward, $rl_i^{(j)}$.

**Accuracy reward** We employ a *binary accuracy reward* that compares the extracted answer $a_i$ against the ground truth $a_i^*$: $ra_i^{(j)} = 1$ if $a_i^{(j)} = a_i^*$, and equals 0 otherwise.

However, models trained with only this rule-based correctness reward tend to generate verbose responses even for simple questions. As a result, we design a novel length reward that uses the training history to encourage conciseness while preserving correctness. The motivation behind our length reward is that if $p_\theta$ gives a correct response $y_i$ to query $x_i$, then in subsequent training if $p_\theta$ encounters $x_i$ again, it should be positively rewarded if the new response is still correct but is shorter than $y_i$; otherwise (i.e., the new response to $x_i$ is longer than $y_i$ or is incorrect), it needs to be penalized. To compute the length reward, we maintain a *history reference length* $h_i$ for each query $x_i$ in the training set. $h_i$ is initialized as `Null` = MAXLENGTH (i.e., set to max length for update but treated as `Null` for reward computation) and is updated whenever a shorter correct response for $x_i$ is produced during training. Details of the length reward computation are described below and the overall procedure is illustrated in Figure 2 (left).

**Length reward design** The length reward will be computed for *each* response $y_i^{(j)} \in Y_i^{(j)}$. At a high level, we desire a length reward with the following characteristics. (1) If $|y_i^{(j)}| < h_i$ and $a_i^{(j)}$ is correct, we should assign a positive reward, since $y_i^{(j)}$ is a more concise solution than any previous correct one. (2) If $|y_i^{(j)}| > h_i$ and $y_i^{(j)}$ is correct, then we should assign a negative reward. This is because $y_i^{(j)}$ is suboptimal with the presence of a shorter correct response. (3) If $y_i^{(j)}$ is incorrect but $|y_i^{(j)}| < h_i$, we hypothesize that $p_\theta$ is exploring a *potentially* correct and shorter response and so assign a neutral reward of 0. (4) If $|y_i^{(j)}| > h_i$ and $y_i^{(j)}$ is incorrect, then, similar to (2), we want a negative reward since $y_i^{(j)}$ is suboptimal. (5) In contrast to prior work (Yeo et al. 2025; Shen et al. 2025b), we also want the reward to decrease smoothly as response length increases, even for incorrect responses, as long as $h_i$ is not `Null`. (6) The length reward should ensure that any correct response has an overall reward $r_i^{(j)}$ that is strictly higher than any incorrect one. (7) Lastly, if the reference $h_i$ is still `Null`, then this means $p_\theta$ has not generated any correct response in the training history, and therefore we give the current response a reward of 0 since no comparison can be made.

Figure 2 (right) depicts our length reward function that ranges from -1 to 1, rewards correct responses shorter than $h_i$, penalizes all responses longer than $h_i$, and smoothly decreases as $|y_i^{(j)}|$ increases. It relies on the cosine function in $f$ to meet characteristic (5) as well as a clipping cutoff

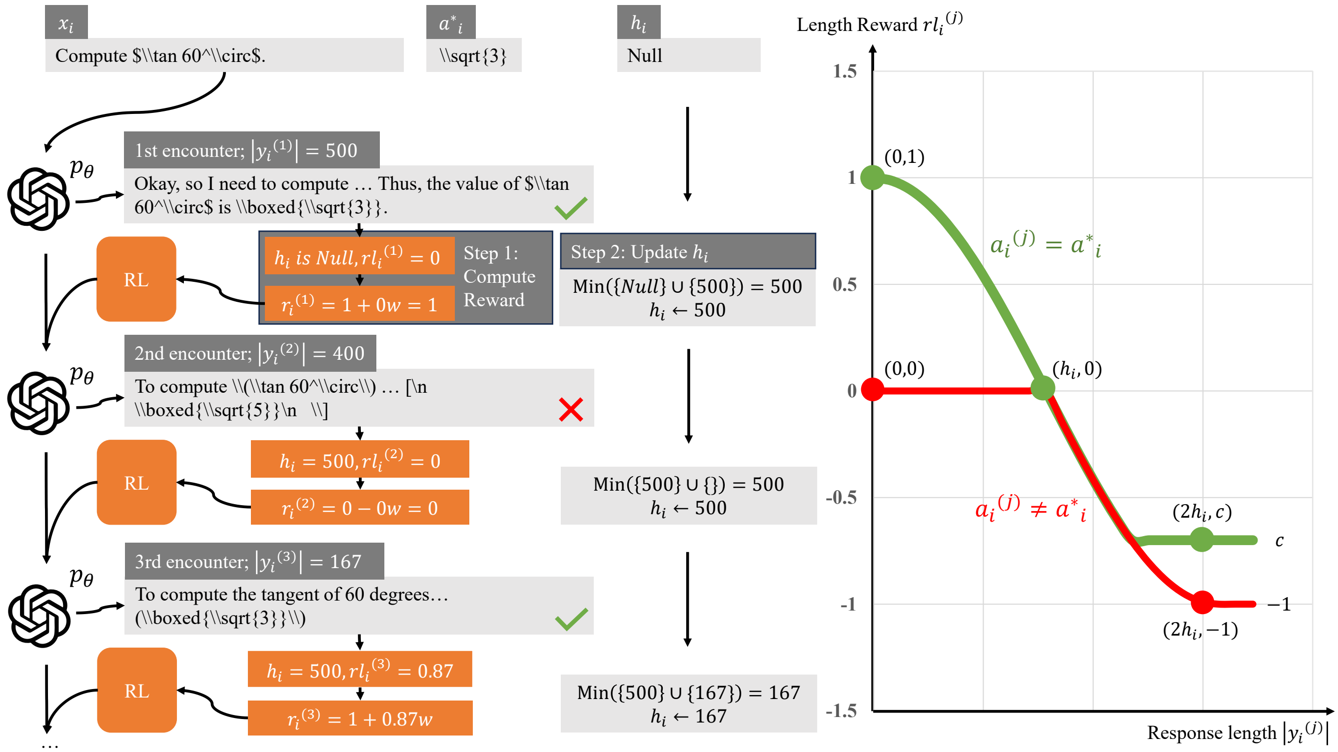



Figure 2: **Left**: Illustration of the HAPO reward computation and history state ($h_i$) update over three encounters with problem $x_i$, assuming one response is sampled per encounter ($|Y_i^{(j)}| = 1$). Initially ($j = 1$, $h_i = \texttt{Null}$). The first correct response $y_i^{(1)}$ (length 500) receives a zero length reward ($rl = 0$) and an overall reward of 1 ($r = 1$), and updates $h_i$ to 500. In the second encounter ($j = 2$), an incorrect response $y_i^{(2)}$ (length 400) receives $rl = 0$ and $r = 0$, and does not update $h_i$. In the third encounter ($j = 3$), a correct response $y_i^{(3)}$ (length 167), being shorter than the current $h_i = 500$, receives a positive length reward ($rl = 0.87$) and overall reward ($r \geq 1$), and updates $h_i$ to 167. **Right**: The HAPO length reward function $rl_i^{(j)}$ plotted against response length $|y_i^{(j)}|$. **Green curve**: reward for correct responses ($a_i^{(j)} = a_i^*$), incentivizing lengths shorter than $h_i$; **Red curve**: reward for incorrect responses ($a_i^{(j)} \neq a_i^*$). Both curves are centered at $(h_i, 0)$. The initial case where $h_i$ is $\texttt{Null}$ is omitted for visual clarity.

$c \in (-1, 0)$ for the $a_i^{(j)} = a_i^*$ case to meet characteristic (6):

$$rl_i^{(j)} = \begin{cases} max\left(f(|y_i^{(j)}|, h_i), c\right) & \text{if } a_i^{(j)} = a_i^* \\ min\left(f(|y_i^{(j)}|, h_i), 0\right) & \text{if } a_i^{(j)} \neq a_i^* \\ 0 & \text{if } h_i \text{ is } \texttt{Null} \end{cases}$$

where $f(|y_i^{(j)}|, h_i) = cos\left(min(\frac{\pi}{2}\frac{|y_i^{(j)}|}{h_i}, \pi)\right)$.

**Final reward function** The final reward combines the length reward with the accuracy reward:

$$r_i^{(j)} = 1(a_i^{(j)} = a_i^*) + w \cdot rl_i^{(j)}$$

$w$ is a hyperparameter within the range $[0, 1]$ that controls the weight of the length component of the overall reward.

**History state update** After computing the reward for each response in $Y_i^{(j)}$, we update the history state $h_i$. Specifically, we extract the lengths of all correct responses in $Y_i^{(j)}$ as $L_i^{(j)} = \{|y_i^{(j)}| | y_i^{(j)} \in Y_i^{(j)} \wedge a_i^{(j)} = a_i^*\}$. We then update $h_i$ using an aggregation function $aggre$ (i.e., $h_i \leftarrow aggre(\{h_i\} \cup L_i^{(j)})$). In this work, we set $aggre$ to the minimum function, so $h_i$ tracks the shortest correct response observed thus far. Other aggregation choices, such as taking the mean or median, are also possible.

## Experiment Setup

**Training Details** We train our models on 2,000 examples from the DeepScaleR-Preview-Dataset (Luo et al. 2025c) and validate the models on another 500. The original dataset contains 40,000 math problems drawn from a variety of sources, spanning a difficulty spectrum from grade-school level (e.g., GSM8K (Cobbe et al. 2021)) to competitive mathematics (e.g., AMC, AIME). For the base models, we experiment with DeepSeek-R1-Distill-Qwen-1.5B (**DeepSeek-R1-1.5B** for short), DeepScaleR-1.5B-Preview (**DeepScaleR-1.5B** for short), and Qwen-2.5-1.5B-Instruct (**Qwen-2.5-1.5B-Inst** for short).

Both DeepSeek-R1-1.5B and DeepScaleR-1.5B are reasoning models, where the latter is an improved version of the former, with higher accuracy and shorter responses on reasoning tasks. In contrast, Qwen-2.5-1.5B-Inst is not a reasoning model, and is not trained specifically to generate long reasoning traces to solve complex questions. To apply HAPO to Qwen-2.5-1.5B-Inst, we curate an easier training set that comprises 2,000 examples from the train split of MATH dataset. This is to ensure that the model can still generate correct answers and receive positive length rewards for a reasonable number of queries during training. All models are trained for 5 epochs using the GRPO algorithm with HAPO hyperparameters set to $w = 1.0$ and $c = -0.7$. More training details are in the Appendix.

**Evaluation** We evaluate the trained models on GSM8K (Cobbe et al. 2021), MATH500 (Hendrycks et al. 2021), and AIME2024, which represent grade-school, intermediate, and competition-level math problems, respectively. Following DeepSeek-AI et al. (2025), we use a sampling temperature of 0.6, top-$p$ of 0.95, and a maximum context length of 32,768 tokens. For GSM8K and MATH, we generate 4 responses per query, and for AIME2024, we generate 32 responses per query. As in prior work, we report **Pass@1** (i.e., accuracy averaged across multiple sampled responses) and the average **#Tokens** in the responses for each benchmark.

**Baselines** We compare our method against two baselines: (1) the untrained base model and (2) an ablation baseline, denoted as **HAPO w/** $w = 0$. In the ablation baseline, the length reward $rl$ is removed from the total reward $r_i^{(j)}$ by setting its weight $w$ to 0, such that the model is trained to optimize solely for correctness. This setup allows us to isolate the impact of length reward in HAPO.

Since many existing approaches build upon DeepSeek-R1-1.5B and have released code or model checkpoints, we additionally compare HAPO in this setting against several representative methods from two paradigms: universal budget forcing and query-level optimization.

For universal budget forcing, we compare against **L1-Exact** and **L1-Max** (Aggarwal and Welleck 2025). Both methods employ GRPO with a token budget penalty alongside the accuracy reward. **L1-Exact** trains models to produce responses with exactly the target budget, while **L1-Max** treats the budget as an upper bound. To ensure a fair comparison—since the original models were trained on a significantly larger dataset—we rerun these methods using our own training data. For each benchmark, the L1-Exact budget is set to the average response length of HAPO, while L1-Max is given twice that average. Although exploring multiple token budgets for L1 baselines may improve their length-correctness trade-offs, we avoid this to limit compute and human interventions, and maintain fairness.

For query-level optimization, we include the method proposed by Arora and Zanette (2025), denoted as **Query-Opt**. This method compares the lengths of responses for the same query in each training batch, rewarding relatively shorter responses and penalizing longer ones. Additionally, we evaluate **O1-Pruner** (Luo et al. 2025a), which uses the base model's responses to each query as a reference, to compute length and correctness rewards.

# Results and Analysis

We first present the main results on three models: DeepSeek-R1-1.5B, DeepScaleR-1.5B, and Qwen-2.5-1.5B-Inst. We then focus on DeepSeek-R1-1.5B for further experiments and analysis, including comparisons with prior work, a case study on MATH500, and an inspection of training dynamics. Finally, we examine the impact of training set size on the model's performance.

## Main Results

**HAPO achieves substantial length reduction while preserving most of the accuracy.** As shown in Table 1, on DeepSeek-R1-1.5B, compared with the base model, HAPO reduces response length by 68%, 54%, and 44% on GSM8K, MATH500, and AIME2024, respectively. The corresponding changes in accuracy are a 5% drop, a 2% gain, and a 3% drop. Overall, HAPO achieves a 49% reduction in response length with only a 2% decrease in accuracy.

On DeepScaleR-1.5B, HAPO reduces response length by 34%, 29%, and 34% across the three benchmarks, with accuracy drops of 3%, 4%, and 7%, respectively. The gains here are less pronounced, likely because DeepScaleR-1.5B has already been fine-tuned to reduce response length from the original DeepSeek-R1-1.5B model. This can leave less room for further length compression.

On Qwen-2.5-1.5B-Inst, HAPO again proves effective, reducing response length by 38%, 51%, and 62%, while keeping accuracy drops within 3%, 1%, and 1% on the three benchmarks. These suggest that HAPO is broadly effective, even for models not originally optimized for reasoning.

Consistent with the findings in (Arora and Zanette 2025), we observe a reduction in token usage even when the model is trained solely with the correctness reward (i.e., HAPO w/ $w = 0$). We hypothesize that this may be due to the relatively low difficulty of questions in the training set, though further investigation is needed to confirm the underlying cause. Compared to this baseline, HAPO produces responses that are, on average, 31% shorter and 4% less accurate on DeepSeek-R1-1.5B across all benchmarks. The corresponding reductions for DeepScaleR-1.5B and Qwen-2.5-1.5B-Inst are (26%, 5%) and (18%, 2%), respectively. These suggest that HAPO achieves a more favorable length-accuracy trade-off than optimizing for correctness alone.

## Comparison with Prior Work

**HAPO achieves a superior length-accuracy trade-off compared with prior work.** In Table 2, we compare HAPO against L1-Exact, L1-Max, Query-Opt, and O1-Pruner.

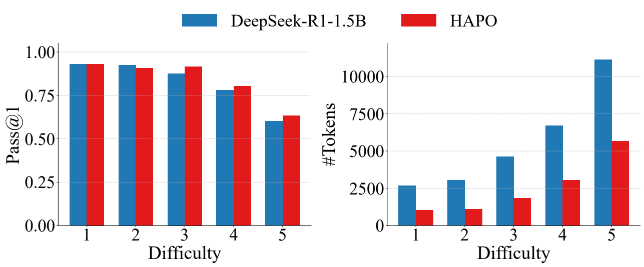
On MATH500, HAPO produces the shortest responses, using 9% fewer tokens than the second place. It attains an accuracy that is similar to the prior methods, except for O1-Pruner, whose accuracy is much lower. On AIME2024, HAPO again gives the most concise responses, using 19% fewer tokens than the second place. In terms of accuracy, it is superior or on par with prior methods, except that O1-Pruner's accuracy stands out higher. On GSM8K, HAPO is less effective than Query-Opt and generates longer responses, though it still outperforms the other methods.

| Models | GSM8K | | MATH500 | | AIME2024 | | Average | |
|---|---|---|---|---|---|---|---|---|
| | Pass@1 ↑ | #Tokens ↓ | Pass@1 ↑ | #Tokens ↓ | Pass@1 ↑ | #Tokens ↓ | Pass@1 ↑ | #Tokens ↓ |
| DeepSeek-R1-1.5B | 83.65 | 2066 | 79.15 | 6449 | 29.38 | 14615 | 64.06 | 7710 |
| HAPO w/ $w = 0$ | **84.40** | 1381 | **82.15** | 4939 | **31.15** | 10701 | **66.02** | 5674 |
| HAPO | 79.08 | **661** | 81.05 | **2978** | 26.56 | **8171** | 62.23 | **3937** |
| DeepScaleR-1.5B | 86.55 | 1702 | **88.05** | 3765 | 37.92 | 8826 | 70.84 | 4764 |
| HAPO w/ $w = 0$ | **86.81** | 1509 | 88.00 | 3462 | **40.10** | 8094 | **71.64** | 4355 |
| HAPO | 83.59 | **1122** | 84.20 | **2681** | 31.15 | **5813** | 66.31 | **3205** |
| Qwen-2.5-1.5B-Inst | 70.43 | 322 | 53.50 | 799 | **3.54** | 3923 | 42.49 | 1681 |
| HAPO w/ $w = 0$ | **72.95** | 301 | **54.75** | 589 | 2.71 | 1643 | **43.47** | 844 |
| HAPO | 67.90 | **200** | 52.25 | **390** | 2.81 | **1493** | 40.99 | **694** |

Table 1: Main results on GSM8K, MATH500, and AIME2024.

| Models | GSM8K | | MATH500 | | AIME2024 | | Average | |
|---|---|---|---|---|---|---|---|---|
| | Pass@1 ↑ | #Tokens ↓ | Pass@1 ↑ | #Tokens ↓ | Pass@1 ↑ | #Tokens ↓ | Pass@1 ↑ | #Tokens ↓ |
| DeepSeek-R1-1.5B | 83.65 | 2066 | 79.15 | 6449 | 29.38 | 14615 | 64.06 | 7710 |
| L1-Exact (U) | 79.31 | 963 | **82.70** | 3366 | 24.69 | 10927 | 62.23 | 5085 |
| L1-Max (U) | 80.33 | 1239 | 81.55 | 3404 | 20.30 | 10144 | 60.73 | 4929 |
| Query-Opt (Q) | 79.22 | **478** | 81.45 | 3284 | 26.67 | 10805 | 62.45 | 4856 |
| O1-Pruner (Q) | 76.99 | 497 | 71.80 | 3264 | **32.42** | 10290 | 60.40 | 4684 |
| HAPO w/ $w = 0$ | **84.40** | 1381 | 82.15 | 4939 | 31.15 | 10701 | **66.02** | 5674 |
| HAPO | 79.08 | 661 | 81.05 | **2978** | 26.56 | **8171** | 62.23 | **3937** |

Table 2: Comparison with prior work on DeepSeek-R1-1.5B. (U) denotes universal budget forcing methods; (Q) denotes query-level optimization methods.



(a) Pass@1 by difficulty. (b) #Tokens by difficulty.

Figure 3: Comparison between HAPO and DeepSeek-R1-1.5B in terms of Pass@1 and #Tokens across questions of varying difficulty levels.

On average, HAPO's responses are shorter than L1-Exact, L1-Max, Query-Opt, and O1-Pruner by 23%, 20%, 19%, and 16% respectively, while maintaining similar accuracy.

## Case Study on MATH500

While Table 2 provides a high-level overview of HAPO's performance across benchmarks, we conduct a more fine-grained case study on MATH500 to examine how HAPO performs across questions of varying difficulty levels. We obtain the difficulty labels from the original MATH dataset, which range from level 1 (easiest) to level 5 (hardest), and analyze Pass@1 accuracy and response length at each difficulty level. As shown in Figure 3, on easier questions (levels 1 and 2), HAPO largely preserves accuracy while reducing response length by 61% and 64%, respectively. On more challenging questions (levels 3 to 5), HAPO slightly improves accuracy while still achieving a response length reduction of at least 50%. These results indicate that **HAPO consistently reduces response length while maintaining accuracy across varying levels of question difficulties**.

## Results on Out of Domain Benchmarks

We evaluate HAPO on two out-of-domain benchmarks to assess its generalizability. GPQA (Rein et al. 2024) comprises multiple-choice questions on biology, physics, and chemistry, while LiveCodeBench (Jain et al. 2025) includes a variety of coding-related problems.

As shown in Table 3, **training on math problems with HAPO enables the model to reason more concisely in other domains, albeit to a lesser extent**. On GPQA, HAPO reduces response length by 27% with a 2% drop in accuracy. On LiveCodeBench, it achieves a 34% reduction in length while slightly improving accuracy.

Compared with prior work, HAPO achieves the best length-correctness tradeoffs on LiveCodeBench, with the shortest responses and similar accuracy. On GPQA, the results are mixed. HAPO produces shorter responses but lower accuracy than Query-Opt and O1-Pruner, and it underperforms the L1 methods. Further investigation into these domain-specific trade-offs is left for future work.

| | GPQA | | CodeBench | | Average | |
|---|---|---|---|---|---|---|
| Models | Pass@1 ↑ | #Tokens ↓ | Pass@1 ↑ | #Tokens ↓ | Pass@1 ↑ | #Tokens ↓ |
| DeepSeek-R1-1.5B | 34.34 | 10296 | 16.42 | 15070 | 25.38 | 12683 |
| L1-Exact (U) | **36.70** | 6658 | 16.23 | 10965 | **26.47** | 8812 |
| L1-Max (U) | 33.43 | **6168** | 16.01 | 10578 | 24.72 | **8373** |
| Query-Opt (Q) | 35.86 | 8228 | **17.05** | 11456 | 26.46 | 9842 |
| O1-Pruner (Q) | 34.85 | 7944 | 16.34 | 10609 | 25.60 | 9277 |
| HAPO w/ $w = 0$ | 33.33 | 8611 | 16.67 | 11098 | 25.00 | 9855 |
| HAPO | 32.32 | 7563 | 16.84 | **9933** | 24.58 | 8748 |

Table 3: Results on GPQA and LiveCodeBench.

| | GSM8K | | MATH500 | | AIME2024 | | Average | |
|---|---|---|---|---|---|---|---|---|
| Models | Pass@1 ↑ | #Tokens ↓ | Pass@1 ↑ | #Tokens ↓ | Pass@1 ↑ | #Tokens ↓ | Pass@1 ↑ | #Tokens ↓ |
| DeepSeek-R1-1.5B | 83.65 | 2066 | 79.15 | 6449 | 29.38 | 14615 | 64.06 | 7710 |
| HAPO ($\vert\mathcal{D}\vert = 2$k) | 79.08 | 661 | 81.05 | 2978 | 26.56 | 8171 | 62.23 | 3937 |
| w/ $\vert\mathcal{D}\vert = 1k$ | 82.70 | 1108 | 83.55 | 3463 | 25.00 | 10408 | 63.75 | 4993 |
| w/ $\vert\mathcal{D}\vert = 500$ | 84.06 | 1371 | 82.05 | 5194 | 29.90 | 10700 | 65.34 | 5755 |
| w/ $\vert\mathcal{D}\vert = 100$ | 83.89 | 1785 | 80.80 | 5887 | 30.94 | 13518 | 65.21 | 7063 |

Table 4: Pass@1 (accuracy) and #Tokens (response length) when training on smaller datasets.

## Training Dynamics

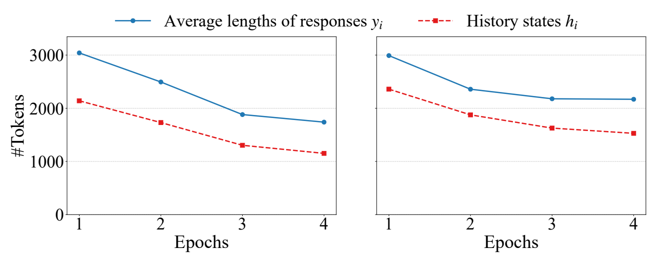
$h_i$ is a key component in HAPO. We inspect how the average response length $|y_i|$ and historical minimum length $h_i$ change as training progresses.

On the training set (Figure 4, left), both $|y_i|$ and $h_i$ steadily decrease as training progresses. In fact, the reductions in $|y_i|$ and $h_i$ reinforce each other: a decrease in average response length increases the likelihood of encountering an unprecedentedly short response, which in turn lowers $h_i$. A lower $h_i$ then acts as a stronger incentive for the model to produce even shorter responses in the next epoch, further reducing $|y_i|$. This mutually reinforcing trend is also observed on the validation set (Figure 4, right). However, the average response length curve begins to plateau around epoch 4, suggesting potential overfitting with continued training.

## Training on Smaller Datasets for More Epochs

Prior works such as (Fatemi et al. 2025) suggest that training LLMs on a small set of occasionally solvable examples can effectively reduce their response length. Inspired by this, we check whether reducing our training set size will yield similar or even better performance. We reduce the training set size from the original 2k to 1k, 500, and 100. In each setting, we train the model for 10 epochs to ensure a sufficient number of iterations while preventing overfitting. We keep other hyperparameters the same. Table 4 shows the results.

Compared with the original setting, training on fewer examples leads to higher accuracy but longer responses. The overall performance becomes much closer to that of the base model. This is expected: with a smaller training set, the model is exposed to a more limited range of question types, increasing the likelihood of encountering unfamiliar questions during evaluation. As discussed previously, the concise reasoning abilities learned by HAPO do generalize to unseen questions, but to a limited extent. Consequently, the model deviates less from the base model.



(a) Train Set Trajectories (b) Valid Set Trajectories

Figure 4: Trajectories of average response length $|y_i|$ and $h_i$ on the training set (left) and validation set (right), computed at the *end* of each epoch. For the training set, the value of $|y_i|$ at each epoch is obtained by averaging the lengths of responses sampled from $p_\theta$ during that epoch across the entire training set. Similarly, $h_i$ is averaged over all training examples, excluding `Null` values. Note that $h_i$ is measured *after* the update; that is, in epoch $j$, $h_i = \text{aggre}(L_i^{(1)}, \cdots, L_i^{(j)})$, which includes the latest length $L_i^{(j)}$.

## Conclusion

In conclusion, we propose History-Aware Policy Optimization (HAPO), a training method that enables large language models to reason more concisely by leveraging historical information from past interactions. Experimental results demonstrate that HAPO significantly reduces response length while maintaining accuracy, achieving a more favorable length-correctness trade-off than prior approaches. Through the use of training history, HAPO opens new directions for building more adaptive, efficient, and capable models in reasoning tasks.

## References


Aggarwal, P.; and Welleck, S. 2025. L1: Controlling How Long A Reasoning Model Thinks With Reinforcement Learning. *arXiv preprint arXiv:2503.04697, 2025*.

Arora, D.; and Zanette, A. 2025. Training Language Models to Reason Efficiently. *arXiv preprint arXiv:2502.04463, 2025*.

Besta, M.; Blach, N.; Kubicek, A.; Gerstenberger, R.; Podstawski, M.; Gianinazzi, L.; Gajda, J.; Lehmann, T.; Niewiadomski, H.; Nyczyk, P.; and Hoefler, T. 2024. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. In *Proceedings of the AAAI Conference on Artificial Intelligence 2024*.

Chen, X.; Xu, J.; Liang, T.; He, Z.; Pang, J.; Yu, D.; Song, L.; Liu, Q.; Zhou, M.; Zhang, Z.; et al. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Cheng, J.; and Durme, B. V. 2024. Compressed Chain of Thought: Efficient Reasoning through Dense Representations. *arXiv preprint arXiv:2412.13171, 2024*.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168, 2021*.

DeepSeek-AI; Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; and et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948, 2025*.

Fatemi, M.; Rafiee, B.; Tang, M.; and Talamadupula, K. 2025. Concise reasoning via reinforcement learning. *arXiv preprint arXiv:2504.05185*.

Fu, Y.; Chen, J.; Zhu, S.; Fu, Z.; Dai, Z.; Qiao, A.; and Zhang, H. 2024. Efficiently Serving LLM Reasoning Programs with Certaindex. *arXiv preprint arXiv:2412.20993*.

Geiping, J.; McLeish, S.; Jain, N.; Kirchenbauer, J.; Singh, S.; Bartoldson, B. R.; Kailkhura, B.; Bhatele, A.; and Goldstein, T. 2025. Scaling up Test-Time Compute with Latent Reasoning: A Recurrent Depth Approach. *arXiv preprint arXiv:2502.05171, 2025*.

Han, T.; Wang, Z.; Fang, C.; Zhao, S.; Ma, S.; and Chen, Z. 2024. Token-Budget-Aware LLM Reasoning. *arXiv preprint arXiv:2412.18547, 2024*.

Hao, S.; Sukhbaatar, S.; Su, D.; Li, X.; Hu, Z.; Weston, J.; and Tian, Y. 2024. Training Large Language Models to Reason in a Continuous Latent Space. *arXiv preprint arXiv:2412.06769, 2024*.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring Mathematical Problem Solving With the MATH Dataset. In *Advances in Neural Information Processing Systems*.

Hou, B.; Zhang, Y.; Ji, J.; Liu, Y.; Qian, K.; Andreas, J.; ; and Chang, S. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296, 2025*.

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

Jain, N.; Han, K.; Gu, A.; Li, W.-D.; Yan, F.; Zhang, T.; Wang, S.; Solar-Lezama, A.; Sen, K.; and Stoica, I. 2025. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code. In *International Conference on Learning Representations*.

Kumar, A.; Roh, J.; Naseh, A.; Karpinska, M.; Iyyer, M.; Houmansadr, A.; and Bagdasarian, E. 2025a. OverThink: Slowdown Attacks on Reasoning LLMs. *arXiv e-prints*, arXiv–2502.

Kumar, A.; Zhuang, V.; Agarwal, R.; Su, Y.; Co-Reyes, J. D.; Singh, A.; Baumli, K.; Iqbal, S.; Bishop, C.; Roelofs, R.; Zhang, L. M.; McKinney, K.; Shrivastava, D.; Paduraru, C.; Tucker, G.; Precup, D.; Behbahani, F.; and Faust, A. 2025b. Training Language Models to Self-Correct via Reinforcement Learning. In *International Conference on Learning Representations*.

Le, H.; Do, D.; Nguyen, D.; and Venkatesh, S. 2025. Reasoning Under 1 Billion: Memory-Augmented Reinforcement Learning for Large Language Models. *arXiv preprint arXiv:2504.02273*.

Li, Y.; Lin, Z.; Zhang, S.; Fu, Q.; Chen, B.; Lou, J.-G.; and Chen, W. 2023. Making Language Models Better Reasoners with Step-Aware Verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

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. *arXiv preprint arXiv:2501.12570, 2025*.

Luo, M.; Tan, S.; Huang, R.; Patel, A.; Ariyak, A.; Wu, Q.; Shi, X.; Xin, R.; Cai, C.; Weber, M.; Zhang, C.; Li, L. E.; Popa, R. A.; and Stoica, I. 2025b. DeepCoder: A Fully Open-Source 14B Coder at O3-mini Level. https://pretty-radio-b75.notion.site/DeepCoder-A-Fully-Open-Source-14B-Coder-at-O3-mini-Level-%1cf81902c14680b3bee5eb349a512a51. Notion Blog.

Luo, M.; Tan, S.; Wong, J.; Shi, X.; Tang, W. Y.; Roongta, M.; Cai, C.; Luo, J.; Li, L. E.; Popa, R. A.; and Stoica, I. 2025c. 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.

Muennighoff, N.; Yang, Z.; Shi, W.; Li, X. L.; Fei-Fei, L.; Hajishirzi, H.; Zettlemoyer, L.; Liang, P.; Candès, E.; and Hashimoto, T. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393, 2025*.

Qu, Y.; Yang, M. Y. R.; Setlur, A.; Tunstall, L.; Beeching, E. E.; Salakhutdinov, R.; and Kumar, A. 2025. Optimizing Test-Time Compute via Meta Reinforcement Fine-Tuning. *arXiv preprint arXiv:2503.07572, 2025*.

Qwen. 2025. QwQ-32B: Embracing the Power of Reinforcement Learning.

Rein, D.; Hou, B. L.; Stickland, A. C.; Petty, J.; Pang, R. Y.; Dirani, J.; Michael, J.; and Bowman, S. R. 2024. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. In *First Conference on Language Modeling*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347, 2017*.

Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y.; Wu, Y.; and Guo, D. 2024. DeepSeek-Math: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300, 2024*.

She, J.; Li, Z.; Huang, Z.; Li, Q.; Xu, P.; Li, H.; and Ho, Q. 2025. Hawkeye: Efficient Reasoning with Model Collaboration. *arXiv preprint arXiv:2504.00424, 2025*.

Shen, X.; Wang, Y.; Shi, X.; Wang, Y.; Zhao, P.; and Gu, J. 2025a. Efficient Reasoning with Hidden Thinking. *arXiv preprint arXiv:2501.19201, 2025*.

Shen, Y.; Zhang, J.; Huang, J.; Shi, S.; Zhang, W.; Yan, J.; Wang, N.; Wang, K.; ; and Lian, S. 2025b. DAST: Difficulty-Adaptive Slow-Thinking for Large Reasoning Models. *arXiv preprint arXiv:2503.04472, 2025*.

Shen, Z.; Yan, H.; Zhang, L.; Hu, Z.; Du, Y.; and He, Y. 2025c. CODI: Compressing Chain-of-Thought into Continuous Space via Self-Distillation. *arXiv preprint arXiv:2502.21074, 2025*.

Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K. R.; and Yao, S. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*.

Snell, C.; Lee, J.; Xu, K.; and Kumar, A. 2025. Scaling LLM Test-Time Compute Optimally Can be More Effective than Scaling Parameters for Reasoning. In *International Conference on Learning Representations*.

Su, D.; Zhu, H.; Xu, Y.; Jiao, J.; Tian, Y.; and Zheng, Q. 2024. Token Assorted: Mixing Latent and Text Tokens for Improved Language Model Reasoning. *arXiv preprint arXiv:2502.03275, 2025*.

Sui, Y.; Chuang, Y.-N.; Wang, G.; Zhang, J.; Zhang, T.; Yuan, J.; Liu, H.; Wen, A.; Zhong, S.; Chen, H.; and Hu, X. 2025. Stop Overthinking: A Survey on Efficient Reasoning for Large Language Models. *arXiv preprint arXiv:2503.16419, 2025*.

Team, K.; Du, A.; Gao, B.; Xing, B.; Jiang, C.; Chen, C.; Li, C.; Xiao, C.; Du, C.; Liao, C.; and et al. 2025. Kimi k1.5: Scaling Reinforcement Learning with LLMs. *arXiv preprint arXiv:2501.12599, 2025*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; ; and Zhou, D. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 24824—-24837.

Wu, Y.; Sun, Z.; Li, S.; Welleck, S.; and Yang, Y. 2024. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. In *MATH-AI: The 4th Workshop on Mathematical Reasoning and AI*.

Xia, H.; Li, Y.; Leong, C. T.; Wang, W.; and Li, W. 2025. TokenSkip: Controllable Chain-of-Thought Compression in LLMs. *arXiv preprint arXiv:2502.12067, 2025*.

Xin, H.; Ren, Z. Z.; Song, J.; Shao, Z.; Zhao, W.; Wang, H.; Liu, B.; Zhang, L.; Lu, X.; Du, Q.; Gao, W.; Zhu, Q.; Yang, D.; Gou, Z.; Wu, Z. F.; Luo, F.; and Ruan, C. 2025. Deepseek-prover-v1.5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. In *International Conference on Learning Representations*.

Xu, S.; Xie, W.; Zhao, L.; and He, P. 2025. Chain of Draft: Thinking Faster by Writing Less. *arXiv preprint arXiv:2502.18600, 2025*.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T. L.; Cao, Y.; and Narasimhan, K. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*.

Yeo, E.; Tong, Y.; Niu, M.; Neubig, G.; and Yue, X. 2025. Demystifying Long Chain-of-Thought Reasoning in LLMs. *arXiv preprint arXiv:2502.03373, 2025*.

Zeng, W.; Huang, Y.; Liu, Q.; Liu, W.; He, K.; Ma, Z.; and He, J. 2025. SimpleRL-Zoo: Investigating and Taming Zero Reinforcement Learning for Open Base Models in the Wild. *arXiv preprint arXiv:2503.18892, 2025*.

Zhang, D.; Chen, L.; Zhang, S.; Xu, H.; Zhao, Z.; and Yu, K. 2023. Large Language Models Are Semi-Parametric Reinforcement Learning Agents. In *Advances in Neural Information Processing Systems*.