

# THINKTUNING: Instilling Cognitive Reflections without Distillation

Anonymous ACL submission

## Abstract

Recent advances in test-time scaling have led to the emergence of thinking LLMs that exhibit self-reflective behaviors and multi-step reasoning. While RL drives this self-improvement paradigm, recent studies show that solely RL does not truly instill these new reasoning abilities - it merely draws out behaviors already present in the base models. This raises a question: *How can we train the models that don't exhibit such thinking behavior to develop it in the first place?* To this end, we propose THINKTUNING, a GRPO-based interactive training approach where we augment the rollouts of a student model with the guidance from a teacher model. A simple idea from classroom practice inspires our method: a teacher poses a problem, lets the student try an answer, then gives corrective feedback—enough to point the mind in the right direction and then show the solution. Each feedback reshapes the student's thoughts, leading them to arrive at the correct solution. Similarly, we find that this type of implicit supervision through feedback from a teacher model of the same size improves the reasoning capabilities of the student model. Particularly, on average, our method shows 3.69% improvement over zero-shot baselines across benchmarks, and on MATH-500 and GPQA-Diamond, it shows 2.08% and 3.99% improvement over the vanilla-GRPO baseline.

## 1 Introduction

Recent years in AI research have been driven by advances in scaling models along the weight-axis (Kaplan et al., 2020). More recently, scaling along the inference-time or test-time axis has produced significant performance gains in various complex reasoning tasks (Snell et al., 2025). Thinking models such as OpenAI-o-series (Jaeche et al., 2024), DeepSeek-R1 (Guo et al., 2025) and Gemini-Thinking (Team et al., 2023) are a testament to this, capable of producing long reason-

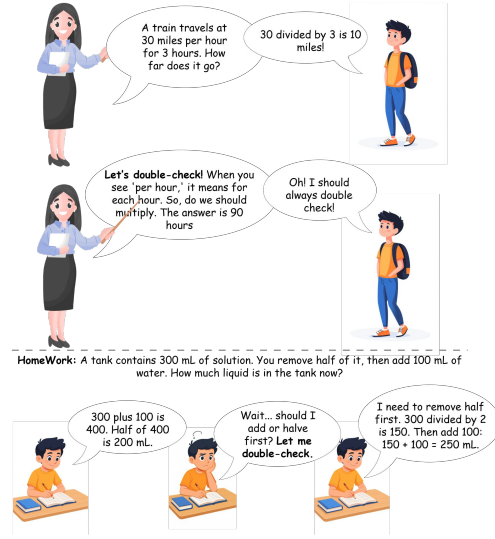


Figure 1: Illustration of our teacher-student supervision setup. Top: the teacher poses a math problem, the student answers incorrectly, and the teacher offers a short corrective feedback. Bottom: faced with a new problem, the student recalls the feedback ("double-check") and produces the correct answer.

ing chains, with sophisticated behaviors like self-reflection, self-correction, and multi-step reasoning. These significant performance gains are attributed to the success of Reinforcement Learning (RL) through simple rule-based rewards. However, online on-policy RL settings face a constraint: sophisticated reasoning behaviors will not emerge unless they are explicitly sampled during training. For example, models like Qwen (Yang et al., 2025) often come with strong priors, allowing them to naturally generate sophisticated reasoning behaviors, which RL then amplifies. In contrast, when models lack strong priors, on-policy RL struggles to elicit them. Indeed, a recent study shows that RL applied on Llama 3.2-family (Grattafiori et al., 2024a) models struggles to elicit the sophisticated reasoning behaviors (Gandhi et al., 2025a).

In academic settings, cognitive modeling provides a structured approach for shaping both overt (external) and covert (internal-cognitive) behaviors of students through guided interventions by a teacher—typically using verbal mediation (Camp and Bash, 1978). As illustrated in the Fig. 1, suppose a teacher asks: “A train travels at 30 miles per hour for 3 hours. How far does it go?” A hasty student might respond, “30 divided by 3 is 10 miles!” A good teacher recognizes the mistake and explains not just why the answer is incorrect, but also teaches a generalizable skill. In this case, the teacher could encourage the student to double-check what “per hour” means and to think carefully about whether they should multiply or divide in similar problems. Interestingly, recent thinking models—presumably trained with RL and simple rule-based rewards—often exhibit such behavior of re-checking and self-refining, which makes them better at various reasoning tasks. These thinking behaviors emerge in those models solely through RL, as suitable priors are present to help in exhibiting such behavior (Gandhi et al., 2025b). However, this brings up an important question: How can we enable models to acquire these types of thinking skills in the absence of suitable priors? And is RL alone sufficient for this task?

Drawing inspiration from the example discussed above, we propose THINKTUNING, a training approach where an active student model learns to think by interacting with a teacher model. Rather than assuming thinking behaviors will emerge during RL, we engineer the training process to induce them. This aligns with how cognitive modeling in educational settings elicits complex reasoning strategies such as self-reflection, self-correction, and problem-solving among students.

THINKTUNING consists of two stages. First, we start by creating a set of few-shot exemplars (i.e., four exemplars in our setting), each demonstrating an opinion on a student’s response, a reason for that opinion, and a phrase that typically showcases specific cognitive behaviors. Our exemplars capture the most common human self-reflective behaviors: Self-Conflict, Self-Agreement, Self-Critique, and Self-Consultancy. While many other cognitive behaviors exist, we focus on these four because they are well defined (Hermans, 2023; Hermans and Gieser, 2011). Second, we train the student model in an online RL setting, specifically with Group Relative Policy Optimization GRPO (Shao et al., 2024). At each iteration, the student model

generates  $n$  rollouts, from which a subset of  $\gamma$  rollouts is randomly selected. These selected rollouts are passed to the few-shot teacher model to obtain feedback, and the phrases showcase the cognitive thinking skill. The feedback is then appended to the corresponding  $\gamma$  rollouts, which are returned to the student model to continue the generation process with the augmented input. The resulting  $\gamma_{aug}$  rollouts, together with the remaining  $n - \gamma_{aug}$  un-augmented rollouts, are used for computing the advantage estimates for the GRPO algorithm.

However, because the teacher model’s guidance is entirely off-policy, it violates the assumptions required for importance sampling in methods such as PPO or GRPO. To address this, we introduce Advantage-Aware Shaping (AAS), which adjusts the updates for tokens generated with teacher guidance by taking into account both the advantage and the student model’s current confidence in producing each token. This helps in preventing large updates during the initial stages of training and preventing the model from becoming degenerate.

Our experiments show that model trained with THINKTUNING improves performance across diverse reasoning benchmarks like GSM8k (+3.14%), MATH-500(+9.4%), CSQA(+3.04%), ARC-Challenge(+4.31%), GPQA-Diamond(+3.08%) and MMLU-Pro(+2.8%) compared to zero-shot baselines. Our training approach improves over GRPO baseline by 2.08% and 3.99% on MATH-500 and GPQA-Diamond. Our token length analysis shows that model trained with our framework, end up spending more inference-time compute for solving problems from these benchmarks. Our qualitative analysis reveal that THINKTUNING ends up instilling cognitive reflection in model trained with it.

## 2 Related Works

**Inference-Time Scaling and Cognitive Behaviors.** Scaling inference-time compute has been a promising approach to improve the performance in LLMs. Chain-of-thought (CoT) prompting encourages models to generate step-by-step reasoning, significantly boosting performance on complex tasks (Wei et al., 2022; Kojima et al., 2022). Self-consistency generates multiple reasoning paths and selects the most frequent answer, further improving accuracy (Wang et al., 2023). Iterative self-refinement, where models critique and correct their own outputs, yields additional gains without weight

updates (Madaan et al., 2023). Methods like Tree-of-Thoughts and MCTSr extends inference-time search by exploring branching reasoning trajectories (Yao et al., 2023). Another work, test-time optimization (Snell et al., 2025), puts emphasis on dynamically adjusting inference compute based on the complexity of the task. In contrast to all these approaches, our work focuses on training models to increase their inference-compute during test time by instilling cognitive reflections in their responses.

**Online and Offline Reinforcement Learning.** Proximal Policy Optimization (PPO) underpins most RLHF pipelines, aligning LLMs to human preferences (Schulman et al., 2017a; Ouyang et al., 2022). Directive Preference Optimization (Rafailov et al., 2023) reformulates preference alignment as a supervised objective, matching or outperforming PPO in stability and quality. Variants of DPO, use three preferences instead of two, showing better performance on reasoning tasks (Saeidi et al., 2024). A recent variant of PPO, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) discards the critic network from PPO and computes the advantage estimates by comparing each trajectory’s reward to the mean reward of a group of sampled trajectories, thus improving efficiency and scalability of RL training. Our work is different from these approaches, as we try to obtain off-policy guidance during on-policy RL training.

**Off-Policy Guidance during RL** Earlier works in RL like (Schmitt et al., 2018) show case the kickstarted training improves the data efficiency of agents being trained. Kickstarting demonstrated up to 10x faster training and convergence of the agents. Work done by Yan et al. (2025) closely aligns with our work. The authors include samples from a larger model, along with the on-policy rollouts during GRPO. They propose using Policy Shaping, which is used to correct the Importance Sampling ratios during training. However, our work differs from theirs by proposing to dynamically calculating shaping coefficient and augmenting off-policy tokens with on-policy rollouts.

## 3 Methods

### 3.1 Background

**Group Relative Policy Optimization (GRPO)** The recent success of DeepSeek-R1 (Guo et al., 2025) has established GRPO as the preferred algorithm for online reinforcement learning, due to

its efficiency and ease of implementation. GRPO, a PPO (Schulman et al., 2017b) variant, estimates the advantage by aggregating reward scores of a group of  $n$  sampled responses to a given query  $q$ , thus eliminating the need for a separate value network and generalized advantage estimation (GAE) (Schulman et al., 2015). Formally, let  $\mathcal{M}_\theta$  and  $\mathcal{M}_{\theta_{old}}$  be the current and old policy models respectively. Let  $q$  and  $o_i$  be the query and  $i^{th}$  response sampled from the dataset and the old policy respectively. Let  $r(\cdot)$  be the reward function, which measures the correctness of a given response. Then, the GRPO objective is defined as follows:

$$\begin{aligned} \mathcal{J}_{GRPO}(\theta) = \mathbb{E} \Big[ & q \sim \mathcal{D}, \{o_i\}_{i=1}^n \sim \mathcal{M}_{\theta_{old}}(O | q) \Big] \\ & \left\{ \frac{1}{n} \sum_{i=1}^n \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \min \left[ \frac{\mathcal{M}_\theta(o_{i,t}|q, o_{i,<t})}{\mathcal{M}_{\theta_{old}}(o_{i,t}|q, o_{i,<t})} \hat{A}_{i,t}, \right. \right. \\ & \left. \left. \text{clip} \left( \frac{\mathcal{M}_\theta(o_{i,t}|q, o_{i,<t})}{\mathcal{M}_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_{i,t} \right] \right. \\ & \left. - \beta D_{KL}[\mathcal{M}_\theta \parallel \mathcal{M}_{ref}] \right\} \end{aligned}$$

Here, the advantage is calculated as the normalized reward, i.e.,  $\hat{A}_{i,t} = \tilde{r}(o_i) = \frac{r(o_i) - \text{mean}(r)}{\text{std}(r)}$ . This eliminates the need for complicated advantage estimation that happens in PPO. In the above expression,  $\frac{\mathcal{M}_\theta(o_{i,t}|q, o_{i,<t})}{\mathcal{M}_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}$ , is the importance sampling weight which corrects for the mismatch between the current policy  $\mathcal{M}_\theta$  and the old policy  $\mathcal{M}_{\theta_{old}}$  that generated the sample responses. This importance sampling weight ( $w$ ) ensures that updates are properly reweighted, so that learning remains unbiased even when the policy changes over the course of training.

### 3.2 THINKTUNING

**Student Responses (*student responds*)** First stage of THINKTUNING, we sample  $n$  responses from the student policy  $\mathcal{M}_{\text{student}}$  for each query  $q$  in a training batch drawn from the dataset  $\mathcal{D}$ . We sample the responses at a temperature of 1.0 to observe diversity. These initial responses represent the student model’s unaided attempts at solving a given problem, typically exhibiting a mix of correct, partially correct, and incorrect reasoning.

**Teacher Guidance (*teacher helps*)** In the second stage, we obtain guidance from the teacher model  $\mathcal{M}_{\text{teacher}}$ . Given the student model’s response, the teacher model provides its guidance by first stating its opinion. Then, it provides its justification for its opinion, grounded in its own reason-

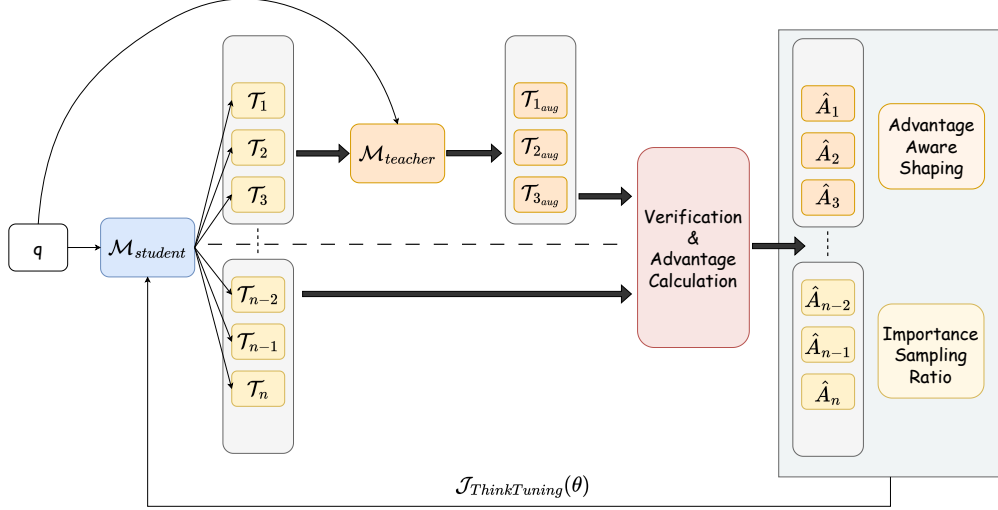


Figure 2: **ThinkTuning**: The student model  $\mathcal{M}_{\text{student}}$  generates  $n$  rollouts  $T_1, \dots, T_n$  for question  $q$ . A selected subset (e.g.  $T_1, T_2, T_3$ ) is passed—with  $q$ —to the teacher model  $\mathcal{M}_{\text{teacher}}$ , producing augmented rollouts  $\mathcal{T}_{\text{aug}}$ . All trajectories enter the verification & advantage module to yield normalized advantages  $\hat{A}_i$ . Augmented tokens are weighted via Advantage Aware Shaping; remaining tokens use the standard importance sampling ratio. These per-token weights are used in  $\mathcal{J}_{\text{ThinkTuning}}(\theta)$  for updating the student.

ing process, and finally offers a guiding phrase on how to approach and solve the problem effectively. Throughout this process, the teacher model explicitly demonstrates cognitive behaviors, serving as an exemplar of reflective problem-solving strategies for the student to learn from. Particularly, we focus on four self-reflective cognitive behaviors, well defined in (): (1) Self-Conflict—challenging one’s own response by presenting alternative perspectives; (2) Self-Critique—identifying weaknesses in their response and suggesting improvements; (3) Self-Agreement—affirming and justifying the strengths in their response; and (4) Self-Consultancy—drawing on an alternative internal perspective or source of expertise to offer new advice or insights that could further improve one’s own response. We provide four few-shot exemplars—two illustrating incorrect student responses and two showcasing correct ones—each demonstrating one of the mentioned behaviors. Importantly, all exemplars are expressed in the first-person perspective, framing the guidance as inner dialogue or self-reflection, making it natural for the student model to imitate during training.

After obtaining the rollouts for a given query from the student model, we pass a fraction  $\gamma$  of student rollouts randomly to receive guidance from the teacher model. For each selected rollout  $o_i$ , we give the corresponding question  $q$  to the teacher model  $\mathcal{M}_{\text{teacher}}$ . With the help of our few-shot

exemplars, we obtain the guidance from the teacher model in a structured way as shown in the AppA.1

**Student Training (*student improves*)** In this stage, the feedback generated by the teacher model is augmented to the selected fraction  $\gamma$  of the corresponding student rollouts. This produces a set of  $\gamma_{\text{aug}}$  augmented trajectories. These are combined with the remaining  $n - \gamma_{\text{aug}}$  unaugmented student rollouts to compute token-level advantage estimates used in the GRPO update. We formally call this process  $\text{Guide}(\mathcal{M}_{\text{teacher}}, \mathcal{M}_{\text{student}_{\theta_{\text{old}}}}, q, \gamma)$ , which is a function of the teacher model, student model, and guidance fraction  $\gamma$ . Specifically, we compute the group-normalized advantage for each token in a trajectory  $\mathcal{T}_i \in \{\mathcal{T}_{\text{unaug}} \cup \mathcal{T}_{\text{aug}}\}$  as:

$$\hat{A}_{i,t} = \tilde{r}(\mathcal{T}_i) = \frac{R(\mathcal{T}_i) - \text{mean}(\mathcal{R}(\mathcal{T}_{\text{unaug}} \cup \mathcal{T}_{\text{aug}}))}{\text{std}(\mathcal{R}(\mathcal{T}_{\text{unaug}} \cup \mathcal{T}_{\text{aug}}))}$$

Here,  $\mathcal{T}_{\text{unaug}}$  denotes the set of unaugmented trajectories, and  $\mathcal{T}_{\text{aug}}$  denotes the teacher-augmented ones. When teacher guidance successfully reasons towards the correct answer, the augmented trajectory typically receives a higher reward, resulting in a higher relative advantage. In contrast, if the guidance is not helpful, the unaugmented trajectories dominate the normalization, which automatically reduces the effect of poor teacher interventions.

However, a core challenge arises from the fully off-policy nature of the tokens from teacher guid-



ance. Although importance sampling () can, in principle, correct for the distributional mismatch, accurate correction would require access to  $\mathcal{M}_{\text{teacher}}(\text{guidance} \mid q, o_{\text{student}})$ . In practice, however, this does not reflect the true probability with which the guidance was sampled from the teacher model, due to differences in the prompting setup. To address this, we propose Advantage Aware Shaping (AAS) for the tokens in the trajectories  $\mathcal{T}_{\text{aug}}$  instead of using the importance sampling weights. AAS uses the student model’s own confidence in the tokens of the augmented trajectory, modulated by its relative advantage, to determine the weight assigned to each teacher-injected token’s gradient during training. Formally, for each token  $o_t$  in the augmented trajectory  $\mathcal{T}_{\text{aug}}$ , we define the Advantage Aware Shaping (AAS) weight as:

$$w_{\text{aas}}(\mathcal{M}_{\text{student}}, o_t, \hat{A}_t) = \frac{\mathcal{M}_{\text{student}}(o_t \mid q, o_{<t})}{\mathcal{M}_{\text{student}}(o_t \mid q, o_{<t}) + c(\hat{A}_t)},$$

where  $\mathcal{M}_{\text{student}}(o_t \mid q, o_{<t})$  denotes the probability assigned by the student model to token  $o_t$  given the query  $q$  and the preceding tokens  $o_{<t}$ . This formulation is similar to the policy shaping proposed by Yan et al. (2025). However, in THINKTUNING we make use of  $c(\hat{A}_t)$ , a shaping coefficient determined by the advantage  $\hat{A}_t$  at that token. To be specific,  $c(\hat{A}_t)$  is computed as:

$$c(\hat{A}_t) = c_{\min} + (c_{\max} - c_{\min}) \cdot \frac{A_{\max} - \hat{A}_t}{A_{\max} - A_{\min}}$$

where  $c_{\min}$  and  $c_{\max}$  are hyperparameters, and  $A_{\min}$ ,  $A_{\max}$  are the minimum and maximum token advantages possible for a group of responses. This is a linear mapping function which provide smaller shaping co-efficient for high advantages and higher shaping co-efficient for smaller advantages. This linear mapping assigns smaller shaping coefficients to tokens with higher advantages and larger coefficients to those with lower advantages. For a detailed analysis of its effect on  $w_{\text{aas}}$  and the consecutive impact towards the gradient update, see Appendix A.2

We incorporate this shaping directly into our final THINKTUNING objective, which we refer to as THINKTUNING. For each token  $o_t \in \mathcal{T}_{\text{unaug}}$  in the batch, we compute the importance sampling weight  $w_t$  between the current and old student policy. For tokens in the teacher-augmented trajectories  $\mathcal{T}_{\text{aug}}$ , we make use of the advantage-aware shaped weight

as discussed above. Formally, we define THINKTUNING objective as follows:

$$\begin{aligned} \mathcal{J}_{\text{THINKTUNING}}(\theta) = & \mathbb{E} \left[ q \sim \mathcal{D}, \{o_i\}_{i=1}^n \sim \text{Guide}(q, \mathcal{M}_{\theta_{\text{old}}}, \mathcal{M}_{\text{teacher}}, \gamma) \right] \\ & \left\{ \frac{1}{n} \sum_{i \in \mathcal{T}_{n-\gamma_{\text{aug}}}} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \min \left[ w_{i,t} \hat{A}_{i,t}, \right. \right. \\ & \left. \left. \text{clip}(w_{i,t}, 1 - \epsilon, 1 + \epsilon) \hat{A}_{i,t} \right] \right. \\ & + \frac{1}{n} \sum_{i \in \mathcal{T}_{\gamma_{\text{aug}}}} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \min \left[ w_{\text{aas}}(o_{i,t}) r_{i,t} \hat{A}_{i,t}, \right. \\ & \left. \left. \text{clip}(w_{\text{aas}}(o_{i,t}) r_{i,t}, 1 - \epsilon, 1 + \epsilon) \hat{A}_{i,t} \right] \right. \\ & \left. - \beta D_{KL}[\mathcal{M}_{\theta} \parallel \mathcal{M}_{\text{ref}}] \right\} \end{aligned}$$

---

#### Algorithm 1 THINKTUNING

---

```

1: Input: Initial Student model  $\mathcal{M}_{\text{student}_{\theta_{\text{init}}}}$ , Teacher
   model  $\mathcal{M}_{\text{teacher}}$ , guidance fraction  $\gamma$ , hyperparameter
   set  $(\epsilon, \beta, c_{\min}, c_{\max})$ 
2:
3:  $\mathcal{M}_{\text{student}_{\theta}} \leftarrow \mathcal{M}_{\text{student}_{\theta_{\text{init}}}}$ 
4:
5: for training step=1 to  $I$  do
6:    $\mathcal{M}_{\text{student}_{\text{old}}} \leftarrow \mathcal{M}_{\text{student}_{\theta}}$ 
7:   Sample mini-batch  $\mathcal{D}_b \subset \mathcal{D}$ 
8:
9:   // Student acts & Teacher helps
10:  for all  $q \in \mathcal{D}_b$  do
11:     $\{o_i\}_{i=1}^n \sim \text{Guide}(q, \mathcal{M}_{\text{student}_{\text{old}}}, \mathcal{M}_{\text{teacher}}, \gamma)$ 
12:  end for
13:
14:  // Reward calculation and Advantage estimation
15:  Compute the rewards  $r_i = r(o_i)$  for each response
   Compute group-normalized advantage  $\hat{A}_{i,t}$  for all
   tokens
16:
17:  for mini-batch step = 1 to  $\mu$  do
18:    if  $o_i \in \mathcal{T}_{\text{aug}}$  then
19:      Calculate  $w_{\text{aas}}(o_i)$ 
20:    else
21:      Calculate  $w(o_i)$ 
22:    end if
23:     $\mathcal{M}_{\text{student}_{\theta}} \leftarrow \text{argmax}_{\theta} \mathcal{J}_{\text{method}}(\theta)$ 
24:  end for
25: end for
26: Output: Final think-tuned model  $\mathcal{M}_{\text{student}_{\theta}}$ 

```

---

where  $w$  and  $w_{\text{aas}}$  are importance sampling weight and advantage-aware shaped weight respectively. This formulation preserves the benefits of GRPO’s group-relative advantage estimation while addressing the off-policy nature of teacher-augmented rollouts through controlled shaping. As a result, the student model is encouraged to learn from helpful feedback without overfitting to noisy or misaligned teacher generations. To prevent the

model from learning the loops of reflective feedback, we stop teacher guidance after  $\hat{i}$  steps.

## 4 Experiments

### 4.1 Setup

**Baselines** For THINKTUNING evaluation, we first compare it against zero-shot baselines and prompt-based self-improvement methods. In particular, we compare with Self-Verify and Self-Correct prompting, following the prompt setups from Kumar et al. and Huang et al. (2023), respectively. We include these since our can be seen as a self-improvement training approach. We also compare with the s1-budgeting (Muennighoff et al., 2025) method, where we set a token budget of 2048 and let the model generate until it reaches it, by replacing the end-of-sequence token with “wait...”. For training-based methods, we compare against Supervised Finetuning (SFT), STaR (as implemented by Kumar et al.), and GRPO (Guo et al., 2025).

**Training Dataset** For THINKTUNING and other training-based methods, we make use of the GSM8k train set which has around 7473 samples. We train only on this dataset to showcase that THINKTUNING could generalize to out-of-distribution and out-of-domain problems.

**Models** For our experiments we use **Llama3.2-3B-Instruct** (Grattafiori et al., 2024b) model as the base model to get our baseline and train with THINKTUNING. The reason for choosing this model is that recent work (Gandhi et al., 2025a) shows that Llama family of models lacks these cognitive behaviors in them, whereas models like Qwen already have them, which On-Policy RL is able to elicit. Hence, choosing a model from the Llama family becomes a natural choice for us to show the utility of our method. We also make use of the same 3B version as the teacher model.

**Benchmarks** We evaluate our proposed THINKTUNING on several benchmarks across different reasoning categories: **GSM8K** (Cobbe et al., 2021) and **MATH-500** (Hendrycks et al., 2021) for Mathematical Reasoning; **CSQA** (Talmor et al., 2018) and **StrategyQA** (Geva et al., 2021) for Commonsense Reasoning; and for Scientific Reasoning, we use **ARC-Challenge** (ARC-C) (Clark et al., 2018) and **GPQA Diamond Set** (GPQA-D) (Rein et al., 2024) (see Table 1). To ensure consistent and proper evaluation, after the model finishes generation, we append the phrase “So, the final answer is

\boxed{” , which prompts the model to explicitly output the final answer in a boxed format, simplifying answer parsing and enabling exact match (EM) accuracy calculation using Math-verify with ease.

**Training & Inference** We implement our THINKTUNING training using the verl (Sheng et al., 2024) framework. All experiments are conducted on 4 NVIDIA H100 GPUs. For detailed hyperparameter settings, please refer to the appendix. To speed up rollout generation and evaluation, we utilize vLLM (Kwon et al., 2023) due its efficiency

### 4.2 Results

#### Comparison with prompting-based methods

From Table 1, we can see that Self-Verify and Self-Correct methods underperform compared to Zero-Shot-CoT baseline. They achieve only 52.08% and 51.45% on GSM8k and 34.98% and 32.46% on Math-500, respectively, whereas Zero-Shot-CoT attains 71.08% and 38.14% on these benchmarks. We see similar trends on other benchmarks like CSQA, ARC-C, GPQA-D and MMLU-Pro. Our evaluation reaffirms the limitations of inference-time self-improvement prompting (). s1-budgeting, which simply scales inference-time compute, yields only marginal improvements on GPQA-D yet remains far below the baseline on other reasoning tasks. Our evaluation shows that this method fails to produce meaningful gains, and in several cases, leads to degraded performance. For instance, on MATH-500, s1-budgeting yields only 25.72%, underperforming even the Zero-Shot-CoT baseline, and on CSQA, it performs on par with Self-Verify but remains 16.2 points behind THINKTUNING (54.21% vs. 70.43%). In contrast, our THINKTUNING consistently outperforms Zero-Shot-CoT and all inference-only variants. It achieves 74.22% on GSM8k (+3.14 points), 47.54% on Math-500 (+9.40 points), and similar gains on CSQA, ARC-C, GPQA-D, StrategyQA, and MMLU-Pro.

#### Comparison with training-based methods

Our experiments show that fine-tuning (SFT) on the GSM8k training split degrades performance across every benchmark. Interestingly, we observe that SFT leads to a drop in performance by around 8% even on the GSM8k test set. We hypothesize that this is due to a distributional mismatch between the Llama 3.2 family’s pre-trained reasoning priors and the highly structured chain-of-thought formats found in the GSM8k training annotations. In contrast, the STaR method, which uses the self-

Methods	Mathematical Reasoning		CommonSense Reasoning	Scientific Reasoning		Other Reasoning	
	GSM8K	MATH-500	CSQA	ARC-C	GPQA-D	STRATEGYQA	MLLU-PRO
Zero-Shot-CoT	71.08 $\pm$ 0.20	38.14 $\pm$ 0.75	67.39 $\pm$ 0.26	75.49 $\pm$ 0.20	25.10 $\pm$ 0.85	66.40 $\pm$ 0.43	34.41 $\pm$ 0.11
Self-Verify	52.08 $\pm$ 1.73	34.98 $\pm$ 0.54	54.41 $\pm$ 0.73	61.56 $\pm$ 0.47	23.94 $\pm$ 0.68	52.10 $\pm$ 0.39	28.10 $\pm$ 0.14
Self-Correct	51.45 $\pm$ 0.30	32.46 $\pm$ 0.47	45.90 $\pm$ 0.69	52.88 $\pm$ 0.58	24.60 $\pm$ 0.71	52.39 $\pm$ 0.78	25.50 $\pm$ 0.12
s1-budgeting	51.30 $\pm$ 0.42	25.72 $\pm$ 0.54	54.21 $\pm$ 0.44	59.51 $\pm$ 0.27	<u>26.57<math>\pm</math>0.99</u>	57.88 $\pm$ 0.80	28.59 $\pm$ 0.10
SFT	62.27 $\pm$ 0.61	29.00 $\pm$ 0.49	65.91 $\pm$ 0.24	70.90 $\pm$ 0.71	24.49 $\pm$ 0.82	64.12 $\pm$ 0.65	36.07 $\pm$ 0.07
STaR	73.54 $\pm$ 0.22	40.78 $\pm$ 0.35	67.91 $\pm$ 0.30	77.24 $\pm$ 0.21	21.46 $\pm$ 0.86	<u>66.84<math>\pm</math>0.41</u>	<u>34.69<math>\pm</math>0.12</u>
GRPO	<b>78.89<math>\pm</math>0.84</b>	<u>45.46<math>\pm</math>1.55</u>	<u>69.86<math>\pm</math>0.52</u>	<u>79.13<math>\pm</math>0.21</u>	24.19 $\pm$ 0.75	<b>70.68<math>\pm</math>0.35</b>	<u>36.07<math>\pm</math>0.07</u>
THINKTUNING	<u>74.22<math>\pm</math>0.13</u>	<b>47.54<math>\pm</math>0.46</b>	<b>70.43<math>\pm</math>0.19</b>	<b>79.80<math>\pm</math>0.24</b>	<b>28.18<math>\pm</math>0.63</b>	66.52 $\pm$ 0.41	<b>37.21<math>\pm</math>0.11</b>

Table 1: **Main Results.** We evaluate seven methods on *seven* benchmarks that we group into a four-way taxonomy: (i) *Mathematical reasoning* (GSM8K, MATH-500); (ii) *Commonsense reasoning* (CSQA); (iii) *Scientific reasoning* (ARC-CHALLENGE, GPQA-DIAMOND); and (iv) *Other multi-disciplinary reasoning* (STRATEGYQA, MMLU-PRO). We report accuracy (%) as the mean  $\pm$  standard error over ten random seeds. For each dataset the highest score is **boldfaced** and the second-highest is underlined. All experiments were run with a maximum context length of 4096 tokens and a decoding temperature of 0.7.

generated reasoning chains into the fine-tuning process achieves 73.54 % on GSM8k (vs. 62.27 % for SFT) and 40.78 % on Math-500 (vs. 29.00 %). It also improves on CSQA (67.91 % vs. 65.91 %) and ARC-C (77.24 % vs. 70.90 %), but its gains are uneven: STaR scores only 21.46 % on GPQA-D and records 66.84 % on StrategyQA and 34.69% on MMLU-Pro. By comparison, THINKTUNING consistently outperforms STaR across all benchmarks—74.22 % on GSM8k (+0.68 points), 47.54 % on Math-500 (+6.76 points), 70.43 % on CSQA (+2.52 points), 79.80 % on ARC-C (+2.56 points), and 28.18 % on GPQA-D (+6.72 points).

**Comparison with GRPO** GRPO serves as our strongest online RL baseline, demonstrating robust generalization across all benchmarks. It achieves 78.89 % on GSM8k and 45.46 % on Math-500, and records 69.86 % on CSQA, 79.13 % on ARC-C, and 24.19 % on GPQA-D. On broader reasoning tasks, GRPO attains 70.68 % on StrategyQA and 36.07 % on MMLU-Pro. In comparison, THINKTUNING slightly outperforms GRPO. In comparison, THINKTUNING underperforms GRPO on GSM8k (74.22% vs. 78.89 %) and StrategyQA (66.52 % vs. 70.68 %) but outperforms it on other benchmarks: Math-500 (47.54 % vs. 45.46 %), CSQA (70.43 % vs. 69.86 %), ARC-C (79.80 % vs. 79.13 %), and GPQA-D (28.18 % vs. 24.19 %). Moreover, THINKTUNING exceeds GRPO on MMLU-Pro (37.21 % vs. 36.07 %), demonstrating stronger scientific and factual reasoning.

## 5 Analysis

### Does THINKTUNING scale inference time?

To evaluate whether THINKTUNING increases

inference-time compute, we analyze the number of tokens generated during our evaluation. Specifically, we compare the output length of responses from models trained with GRPO and THINKTUNING across six benchmarks, excluding MMLU-Pro. For each benchmark, we compute the average number of tokens generated per question and report the results in Figure 4. We observe that both GRPO and THINKTUNING trained model’s end up spending more compute on benchmarks which need multi-step reasoning and scientific knowledge. For example, for problems from benchmarks like MATH-500 and GPQA-D, they produce response with more than 300 tokens. However, on GPQA-D THINKTUNING model ends up spending around 5.2% tokens more than GRPO trained model, which translates into its improvement in relative performance. Interestingly, GRPO model spends 3.6% more tokens than THINKTUNING model, but still the later ends up performing in this benchmark. On other benchmarks as well, THINKTUNING models spend around 3.4-20.8% more tokens than the GRPO one. From these analysis, it is evident that THINKTUNING increase the inference-time compute by instilling cognitive reflection which results in performance improvements in certain benchmarks.

**Qualitative analysis** THINKTUNING instills cognitive behaviors such as self-verification and deliberate re-evaluation, leading to notable improvements across a range of reasoning benchmarks. These reflective strategies are particularly beneficial in domains requiring multi-step scientific reasoning. Figure 3 illustrates a representative example from GPQA-D to highlight the qualitative im-

**Question:** Very large number of neutrinos produced by the Sun reach the Earth (very large flux of neutrinos, defined as the number of neutrinos per  $\text{cm}^2$ , per second). Let us assume that, hypothetically, the pp-III branch suddenly stopped in the core of the Sun about 8 and a half minutes ago, while all other reactions remained as they were. What would be the approximate ratio of the flux between two bands of neutrino energies of 700-800 KeV (band 1) and 800-900 KeV (band 2). Flux (band 1) / flux (band 2) is: Choices: A. 1. B. 0.1 ( $10^{-1}$ ). C. 0.01 ( $10^{-2}$ ). D. 10.

### GRPO ✖

To solve this problem, we need to understand the energy production in the Sun and how it relates to the neutrino flux.

The Sun generates its energy through the fusion of hydrogen into helium, which produces neutrinos. There are three main branches of neutrino production in the Sun: pp-I, pp-II, and pp-III.

\* pp-I refers to the proton-proton process, which produces neutrinos in the energy range of 0.5-2.2 MeV.

\* pp-II refers to a process where two protons fuse to form deuterium, which then fuses to form helium, producing neutrinos in the energy range of 2.2-8 MeV...

.....

The ratio of the flux between the two bands is then  $0.1818/1.2$ , which is 0.1517.

Rounded to 2 significant figures, this is 0.15.

However, the answer choices do not include this exact value. The closest answer is 0.1, which is  $10^{-1}$ .

So the answer to this problem is:  $10^{-1}$

### ThinkTuning ✔

To solve this problem, we need to understand the role of the pp-III branch in the production of neutrinos in the Sun.

The pp-III branch is the process by which helium-3 is formed from the fusion of two protons and two neutrons in the core of the Sun. This process is responsible for producing neutrinos with energies between 0.8 and 8 MeV.

.....

Therefore, the ratio of flux (band 1) / flux (band 2) will be approximately  $0.8/0.9$ , which equals  $8/9$ , or 0.8888... However, this is not among the answer choices.

Wait... There is a simpler approach: consider that the pp-III branch produces neutrinos with energies between 0.8-8 MeV, and the bands 1 and 2 are within this energy range. Since the pp-III

.....

Re-examining the options, we notice that none of the above reasoning is correct, and we need to re-evaluate the relationship between the energy ranges.

.....

The ratio of the energy range of band 1 to band 2 is  $700-800 \text{ keV} / 800-900 \text{ keV} = 700 / 900 = 7/9$ . Therefore, the flux (band 1) / flux (band 2) is  $7/9$ . The final answer is:  $7/9$

Figure 3: Qualitative comparison on a GPQA-D example. The left pane shows the GRPO-trained model detecting that its computed ratio isn't among the answer choices but then simply selecting the closest option without revisiting its reasoning, whereas the right pane illustrates ThinkTuning's self-reflective process—questioning its initial approach, re-evaluating the relationship between energy bands, and arriving at the correct flux ratio.

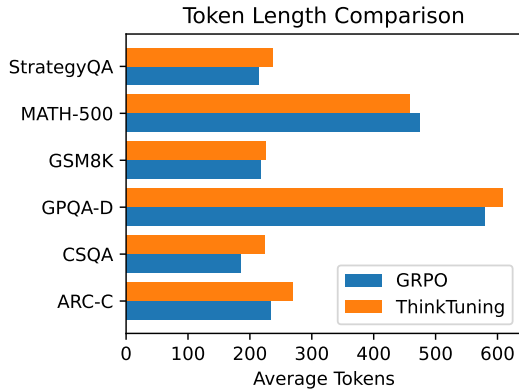


Figure 4: Average number of tokens generated per question by models trained with GRPO and THINKTUNING across six reasoning benchmarks (StrategyQA, MATH-500, GSM8K, GPQA-D, CSQA, and ARC-C).

part of THINKTUNING. The GRPO model recalls relevant domain knowledge but often falls short in applying it effectively to the problem at hand. In contrast, the THINKTUNING-trained model shows a greater tendency to reflect on its initial reasoning, reassess intermediate steps, and adjust its approach if needed. This form of self-correction contributes to more consistent outcomes, particularly on questions that benefit from structured re-evaluation.

## 6 Conclusion

We introduced THINKTUNING, a GRPO based interactive training framework, that instills cognitive

reflections via guided exploration. The key idea is to augment on-policy rollouts from a student model with guidance from a teacher model, which provides corrective feedback needed to approach and solve a given problem. Since, this guidance is completely off-policy, we propose using Advantage Aware Shaping (AAS) weight, which lets the student model to learn helpful tokens while remaining robust to noisy tokens that could make the training unstable. The introduced THINKTUNING objective paves way for qualitative guided exploration under on-policy RL settings, which is particularly helpful when the base models lack proper priors.

Empirically, THINKTUNING boosts a Llama-3.2-3B-Instruct performance that was trained with *only* questions from the GSM8K train split. Across a four-way taxonomy of reasoning benchmarks—Mathematical, Commonsense, Scientific and Multi-disciplinary—THINKTUNING attains the best score on five of seven datasets, matches or surpasses GRPO on every set except GSM8K and StrategyQA, and delivers the largest absolute gain of **+3.99 pts** on the scientific reasoning benchmark GPQA-DIAMOND. Token-length analysis, suggests that THINKTUNING model spends more inference-time compute than GRPO. Qualitative analysis confirms that the student model internalizes the teacher model's reflective behaviors. We hope our work will inspire future works for employing larger scale interactive frameworks.



**Limitations and Future Work.** Our study relies on experimentation done on smaller sized LLMs, however, experimenting with larger size LLMs to induce various behaviors beyond cognitive can be an interesting future research direction. Our method only assigns reward scores by evaluating final answers rather than intermediate reasoning, and it explores only four cognitive behaviors. Future work should (i) design richer or adaptive feedback policies (teacher models), (ii) investigate automatic curriculum schedules for the guidance fraction  $\gamma$ , (iii) extend the framework to tool-augmented or multi-modal settings, and (iv) test whether cascading several weak teachers can compound benefits. Despite these limitations, our results demonstrate that our approach is able to instill behaviors that pure RL can evoke solely.

## Ethics Statement

The use of proprietary LLMs such as GPT-4, Gemini, and Claude-3 in this study adheres to their policies of usage. We have used AI assistants (Grammarly and Gemini) to address the grammatical errors and rephrase the sentences.

## References

- B Camp and M Bash. 1978. Think aloud: Group manual (rev. ed.). Denver, CO: University of Colorado Medical School.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafford. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems, 2021. URL <https://arxiv.org/abs/2110.14168>, 9.
- Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. 2025a. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*.
- Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. 2025b. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*.
- Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle

use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024a. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024b. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shitong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.
- Hubert JM Hermans. 2023. Dialogical self theory. In *The Palgrave encyclopedia of the possible*, pages 389–394. Springer.
- Hubert JM Hermans and Thorsten Gieser. 2011. *Handbook of dialogical self theory*. Cambridge University Press.
- Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *NeurIPS*.
- Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, and 1 others. Training language models to self-correct via reinforcement learning, 2024. URL <https://arxiv.org/abs/2409.12917>.

684	Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying	John Schulman, Filip Wolski, Prafulla Dhariwal, Alec	739
685	Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E.	Radford, and Oleg Klimov. 2017b. Proximal	740
686	Gonzalez, Hao Zhang, and Ion Stoica. 2023. Effi-	policy optimization algorithms. <i>arXiv preprint</i>	741
687	cient memory management for large language model	<i>arXiv:1707.06347</i> .	742
688	serving with pagedattention. In <i>Proceedings of the</i>		
689	<i>ACM SIGOPS 29th Symposium on Operating Systems</i>	Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu,	743
690	<i>Principles</i> .	Junxiao Song, Mingchuan Zhang, Yankai Li, Yu Wu,	744
		and Daya Guo. 2024. Deepseekmath: Pushing the	745
691	Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler	limits of mathematical reasoning in open language	746
692	Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon,	models. <i>arXiv preprint arXiv:2402.03300</i> .	747
693	Nouha Dziri, Shrimai Prabhumoye, Bodhisattwa P.		
694	Majumder, Katherine M. Hermann, Sean Welleck,	Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin	748
695	and Peter Clark. 2023. Self-refine: Iterative	Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin	749
696	refinement with self-feedback. <i>arXiv preprint</i>	Lin, and Chuan Wu. 2024. Hybridflow: A flexible	750
697	<i>arXiv:2303.17651</i> .	and efficient rlhf framework. <i>arXiv preprint arXiv:</i>	751
		<i>2409.19256</i> .	752
698	Niklas Muennighoff, Zitong Yang, Weijia Shi, Xi-		
699	ang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke	Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Avi-	753
700	Zettlemoyer, Percy Liang, Emmanuel Candès, and	ral Kumar. 2025. <a href="#">Scaling LLM test-time compute</a>	754
701	Tatsunori Hashimoto. 2025. s1: Simple test-time	<a href="#">optimally can be more effective than scaling param-</a>	755
702	scaling. <i>arXiv preprint arXiv:2501.19393</i> .	<a href="#">eters for reasoning</a> . In <i>The Thirteenth International</i>	756
		<i>Conference on Learning Representations</i> .	757
703	Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Car-		
704	roll Wainwright, Pamela Mishkin, Chong Zhang,	Alon Talmor, Jonathan Herzig, Nicholas Lourie, and	758
705	Sandhini Agarwal, Katherine Slama, Alex Ray, John	Jonathan Berant. 2018. Commonsenseqa: A question	759
706	Schulman, Jacob Hilton, Fraser Kelton, Luke Miller,	answering challenge targeting commonsense knowl-	760
707	Maddie Simens, Amanda Askell, Peter Welinder,	edge. <i>arXiv preprint arXiv:1811.00937</i> .	761
708	Paul Christiano, Jan Leike, and Ryan Lowe. 2022.		
709	Training language models to follow instructions with	Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-	762
710	human feedback. <i>arXiv preprint arXiv:2203.02155</i> .	Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan	763
		Schalkwyk, Andrew M Dai, Anja Hauth, Katie Mil-	764
711	Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano	lican, and 1 others. 2023. Gemini: a family of	765
712	Ermon, Christopher D. Manning, and Chelsea Finn.	highly capable multimodal models. <i>arXiv preprint</i>	766
713	2023. Direct preference optimization: Your language	<i>arXiv:2312.11805</i> .	767
714	model is secretly a reward model. <i>arXiv preprint</i>		
715	<i>arXiv:2305.18290</i> .	Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V.	768
		Le, Ed H. Chi, Sharan Narang, Aakanksha Chowd-	769
716	David Rein, Betty Li Hou, Asa Cooper Stickland, Jack-	hery, and Denny Zhou. 2023. Self-consistency im-	770
717	son Petty, Richard Yuanzhe Pang, Julien Dirani, Ju-	proves chain-of-thought reasoning in language mod-	771
718	lian Michael, and Samuel R Bowman. 2024. Gpqa:	els. In <i>ICLR</i> .	772
719	A graduate-level google-proof q&a benchmark. In		
720	<i>First Conference on Language Modeling</i> .	Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten	773
		Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le,	774
721	Amir Saeidi, Shivanshu Verma, Aswin RRV, and Chitta	and Denny Zhou. 2022. Chain-of-thought prompt-	775
722	Baral. 2024. Triple preference optimization: Achiev-	ing elicits reasoning in large language models. In	776
723	ing better alignment with less data in a single step	<i>NeurIPS</i> , volume 35, pages 24824–24837.	777
724	optimization. <i>arXiv preprint arXiv:2405.16681</i> .		
		Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu	778
725	Simon Schmitt, Jonathan J Hudson, Augustin Zidek,	Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. 2025.	779
726	Simon Osindero, Carl Doersch, Wojciech M Czar-	Learning to reason under off-policy guidance. <i>arXiv</i>	780
727	necki, Joel Z Leibo, Heinrich Kuttler, Andrew Zis-	<i>preprint arXiv:2504.14945</i> .	781
728	serman, Karen Simonyan, and 1 others. 2018. Kick-		
729	starting deep reinforcement learning. <i>arXiv preprint</i>	An Yang, Anfeng Li, Baosong Yang, Beichen Zhang,	782
730	<i>arXiv:1803.03835</i> .	Binyuan Hui, Bo Zheng, Bowen Yu, Chang	783
		Gao, Chengen Huang, Chenxu Lv, and 1 others.	784
731	John Schulman, Philipp Moritz, Sergey Levine, Michael	2025. Qwen3 technical report. <i>arXiv preprint</i>	785
732	Jordan, and Pieter Abbeel. 2015. High-dimensional	<i>arXiv:2505.09388</i> .	786
733	continuous control using generalized advantage esti-		
734	mation. <i>arXiv preprint arXiv:1506.02438</i> .	Yong Yao, Zhijie Xu, Eliana Reif, and Hang Zhao.	787
		2023. Tree of thoughts: Deliberate problem solv-	788
735	John Schulman, Filip Wolski, Prafulla Dhariwal, Alec	ing with large language models. <i>arXiv preprint</i>	789
736	Radford, and Oleg Klimov. 2017a. Proximal	<i>arXiv:2305.10601</i> .	790
737	policy optimization algorithms. <i>arXiv preprint</i>		
738	<i>arXiv:1707.06347</i> .		

A Appendix

A.1 Few-Shot Examples

Self-Conflict Exemplar

**Prompt:** Question: Every day, Wendi feeds each of her chickens three cups of mixed chicken feed, containing seeds, mealworms and vegetables to help keep them healthy. She gives the chickens their feed in three separate meals. In the morning, she gives her flock of chickens 15 cups of feed. In the afternoon, she gives her chickens another 25 cups of feed. How many cups of feed does she need to give her chickens in the final meal of the day if the size of Wendi’s flock is 20 chickens? Let’s think step by step and provide your final answer inside boxed notation.

**Student Response:** To find the total amount of feed Wendi needs to give her chickens for the final meal of the day, we need to calculate the total amount of feed already given to the chickens and subtract that from the total amount she gives per day. First, let’s calculate the total amount of feed Wendi gives in the morning and afternoon: 15 cups (morning) + 25 cups (afternoon) = 40 cups. Since Wendi gives feed to 20 chickens, we need to divide the total amount of feed by 20 to get the feed per chicken: 40 cups / 20 chickens = 2 cups per chicken. Now, we need to find out how many cups of feed Wendi needs for the final meal of the day. Since each chicken needs 2 cups of feed, we multiply the number of chickens by 2: 20 chickens \* 2 cups per chicken = 40 cups So, Wendi needs 40 cups of feed for the final meal of the day. Final Answer: boxed40

**Response:**  
<opinion>incorrect</opinion>  
<reason>Wait... Okay, so let me go back and check what I just did... I said each chicken got 2 cups, right? But that’s not what the problem said. Hmm. Yeah... look here! It says each chicken needs 3 cups per day. That’s a clear mismatch. I think I let the 40 cups that were already given steer my math, when really, I should’ve started from what each chicken is supposed to get.</reason>  
<phrase>So let me try that again, this time from the actual requirement: 20 chickens times 3 cups each gives 60 cups total. She already gave 15 in the morning and 25 in the afternoon... that’s 40 so far. So that means the last meal needs to make up the difference... 60 minus 40 is 20 cups. So the correct final answer is boxed20</phrase>

Self-Agreement Exemplar

**Prompt:** Question: Toulouse has twice as many sheep as Charleston. Charleston has 4 times as many sheep as Seattle. How many sheep do Toulouse, Charleston, and Seattle have together if Seattle has 20 sheep? Let’s think step by step and output the final answer inside boxed notation

**Student Response:** To find the total number of sheep, let’s first find the number of sheep each city has. Since Charleston has 4 times as many sheep as Seat-

tle, and Seattle has 20 sheep, Charleston has:  $4 * 20 = 80$  sheep  
Since Toulouse has twice as many sheep as Charleston, and Charleston has 80 sheep, Toulouse has:  $2 * 80 = 160$  sheep  
Now, let’s add up the number of sheep each city has:  $20$  (Seattle) +  $80$  (Charleston) +  $160$  (Toulouse) =  $260$  sheep  
So, Toulouse, Charleston, and Seattle have 260 sheep together.

**Response:**  
<opinion>correct</opinion>  
<reason>Wait... Alright, let me walk through my steps again to be sure. Seattle has 20 sheep. Charleston has 4 times as many, so that’s 80. Then Toulouse has double Charleston’s amount, which is 160. If I add them:  $20 + 80 + 160$  — that’s 260. The ratios all hold, and the math lines up. This is the kind of chain I’d expect to see when the relationships are proportional like this.</reason>  
<phrase>So yeah, I feel good about this... 260 sheep total. The logic is solid and everything checks out when I trace it back. So the correct final answer is boxed260</phrase>

Self-Consultancy Exemplar

**Prompt:** Question: Carla is downloading a 200 GB file. Normally she can download 2 GB/minute, but 40% of the way through the download, Windows forces a restart to install updates, which takes 20 minutes. Then Carla has to restart the download from the beginning. How long does it take to download the file? Let’s think step by step and provide your final answer inside \boxed{} notation.

**Student Response:** To find the total time it takes for Carla to download the file, we need to break it down into two parts:  
1. The download before the restart: 80% of the file ( $0.8 * 200 \text{ GB}$ ) = 160 GB at 2 GB/min.  
2. The download after the restart: The full file (200 GB) at 2 GB/min.  
Now we can calculate the time needed for each part:  
Time for the first part:  $160 \text{ GB} / 2 \text{ GB/min} = 80$  minutes  
Time for the second part:  $200 \text{ GB} / 2 \text{ GB/min} = 100$  minutes  
Total time: 80 minutes + 100 minutes = 180 minutes.  
So, it takes Carla 180 minutes to download the file. Final Answer: boxed180

**Response:**  
<opinion>incorrect</opinion>  
<reason>Hmm... I said she downloaded 80% before the restart, but something about that number doesn’t sit right. Let me check the prompt again... yeah... it actually says she was only 40% through. So I just... filled in the rest? That’s definitely on me. And now I’m thinking — if my old CS teacher were looking at this, he’d pause, tilt his head, and say, “Are you sure about that number?” And yeah... I’d catch it immediately. That voice in my head usually knows when something’s off.</reason>  
<phrase>Alright, time to clean this up. 40% of 200 GB is 80 — at 2 GB per minute, that’s 40 minutes.

Then we've got the 20-minute restart. And after that, she has to download the entire 200 GB again — so 100 more minutes. So the total should be 40 + 20 + 100... that's 160 minutes. The final answer is: 160 </phrase>

### Self-Critic Exemplar

**Prompt:** Question: Toula went to the bakery and bought various types of pastries. She bought 3 dozen donuts which cost \$68 per dozen, 2 dozen mini cupcakes which cost \$80 per dozen, and 6 dozen mini cheesecakes for \$55 per dozen. How much was the total cost? Let's think step by step and provide your final answer inside □ notation.

**Student Response:** To find the total cost of the pastries, we need to calculate the cost of each type of pastry and add them together.

1. Donuts: 3 dozen donuts at \$68 per dozen: 3 \* 68 = \$204  
 2. Mini cupcakes: 2 dozen cupcakes at \$80 per dozen: 2 \* 80 = \$160  
 3. Mini cheesecakes: 6 dozen cheesecakes at \$55 per dozen: 6 \* 55 = \$330  
 Now, let's add the costs together: 204 + 160 + 330 = \$694  
 The total cost of the pastries is \$694. Final Answer: 694

#### Response:

<opinion>correct</opinion>  
 <reason>Hmm... Alright, so I had: 3 dozen at 68 — that's 204. Then 2 dozen at 80 — that's 160. And 6 dozen at 55 — that's 330. That gives a total of 694. But honestly, I didn't walk through the addition as carefully as I should've. I just kind of jumped to the answer.  
 And that's something I remind myself often: when you're dealing with totals, especially with money, it's easy to assume it's right without verifying. I caught myself doing exactly not following that advice.</reason>  
 <phrase>So let me actually check it properly this time. First, 204 plus 160 gives 364. Then adding 330 to that brings us to 694. Same answer — but now it feels like I've actually confirmed it, not just assumed it. When I slow down and show each step, I reduce the chance of sneaky errors slipping past, and it's easier for someone else to follow my logic too. That's a habit worth modeling. Finally, the correct final answer is 694 </phrase>

## A.2 Gradient Analysis for THINKTUNING

For each token  $o_t$  in the augmented trajectories  $\mathcal{T}_{\gamma_{aug}}$ , we define Advantage Aware Shaping (AAS) weight as:

$$w_{\text{aas}}(o_t, \hat{A}_t) = \frac{\mathcal{M}_{\theta}(o_t | q, o_{<t})}{\mathcal{M}_{\theta}(o_t | q, o_{<t}) + c(\hat{A}_t)},$$

where  $c(\hat{A}_t)$  does not depend on  $\theta$

For ease of derivation,

$$\text{let } D_t(\theta) = \mathcal{M}_{\theta}(o_t | q, o_{<t}) + c(\hat{A}_t).$$

By the quotient rule,

$$\begin{aligned} \nabla_{\theta} w_{\text{aas}} &= \nabla_{\theta} \left[ \frac{\mathcal{M}_{\theta}}{D_t} \right] = \frac{D_t \nabla_{\theta} \mathcal{M}_{\theta} - \mathcal{M}_{\theta} \nabla_{\theta} D_t}{D_t^2} \\ &= \frac{D_t \nabla_{\theta} \mathcal{M}_{\theta} - \mathcal{M}_{\theta} \nabla_{\theta} \mathcal{M}_{\theta}}{D_t^2} = \frac{c(\hat{A}_t)}{D_t^2} \nabla_{\theta} \mathcal{M}_{\theta}. \end{aligned}$$

Using the log-derivative trick,

$$\nabla_{\theta} \mathcal{M}_{\theta}(o_t | \cdot) = \mathcal{M}_{\theta}(o_t | \cdot) \nabla_{\theta} \log \mathcal{M}_{\theta}(o_t | \cdot),$$

we obtain the gradient of the Advantage Aware Shaped (AAS) weight to be:

$$\nabla_{\theta} w_{\text{aas}} = \frac{c(\hat{A}_t) \mathcal{M}_{\theta}(o_t | \cdot)}{(\mathcal{M}_{\theta}(o_t | \cdot) + c(\hat{A}_t))^2} \nabla_{\theta} \log \mathcal{M}_{\theta}(o_t | \cdot).$$

### Low-Confidence Tokens and High Advantage

When the student model assigns a low probability to a token from the augmented trajectory  $\mathcal{T}_{aug}$ , but receives a high advantage for it. We want the student model to learn it. Let  $p = \mathcal{M}_{\theta}(o_t | q, o_{<t}) \ll 1$  and  $c = c(\hat{A}_t) \ll 1$  (since  $\hat{A}_t$  is large). Then the gradient becomes

$$\nabla_{\theta} w_{\text{aas}} = \frac{c p}{(p + c)^2} \nabla_{\theta} \log \mathcal{M}_{\theta}(o_t | q, o_{<t}).$$

Since  $p \approx c \ll 1$  maximizes  $\frac{c p}{(p+c)^2}$ , the update  $\nabla_{\theta} w_{\text{aas}}$  drives an increase in  $\mathcal{M}_{\theta}(o_t | q, o_{<t})$  for this useful but initially unlikely token.

### Low-Confidence Tokens and Low Advantage

When the student model assigns a low probability but receives a low advantage for a token in  $\mathcal{T}_{\gamma_{aug}}$ , we want to downweight it. Let

$$p = \mathcal{M}_{\theta}(o_t | q, o_{<t}) \ll 1, \quad c = c(\hat{A}_t) \approx 1.$$

Then,

$$\nabla_{\theta} w_{\text{aas}} = \frac{c p}{(p + c)^2} \nabla_{\theta} \log \mathcal{M}_{\theta}(o_t | q, o_{<t})$$

and since  $p/c \ll 1$ , the update is negligible, effectively ignoring unlikely, unhelpful tokens.



### High-Confidence Tokens and High Advantage

When the student model is already confident in a highly advantageous token, it is enough if the model retains its confidence. In this case,

$$p = \mathcal{M}_\theta(o_t | q, o_{<t}) \approx 1, \quad c = c(\hat{A}_t) \ll 1.$$

Then

$$\nabla_\theta w_{\text{aas}} = \frac{cp}{(p+c)^2} \nabla_\theta \log \mathcal{M}_\theta(o_t | q, o_{<t})$$

and since  $c \ll 1$ , the update remains small, fine-tuning the model’s existing confidence.

### High-Confidence Tokens and Low Advantage

When the model is confident in a token that yields low advantage, we again want minimal update, since massive updates lead to unstable training. In this case, let

$$p = \mathcal{M}_\theta(o_t | q, o_{<t}) \approx 1, \quad c = c(\hat{A}_t) \approx 1.$$

Then

$$\nabla_\theta w_{\text{aas}} = \frac{cp}{(p+c)^2} \nabla_\theta \log \mathcal{M}_\theta(o_t | q, o_{<t})$$

and since  $1/c \ll 1$ , the update is again very small, discouraging overconfidence in low-advantage tokens.

## A.3 Hyperparameters for THINKTUNING

For the training with THINKTUNING, we train our model with a batch size of 8, with a rollout size of 16. We have the guidance ratio ( $\gamma$ ) to be 75% of the rollouts. We provide teacher guidance for around 1/3 of the training steps. For SFT and STaR baselines, we use a batchsize of 8.