

# STEP-KTO: Optimizing Mathematical Reasoning through Stepwise Binary Feedback

Yen-Ting Lin<sup>\*†</sup> Di Jin<sup>\*</sup> Tengyu Xu<sup>\*</sup> Tianhao Wu<sup>‡§</sup> Sainbayar Sukhbaatar<sup>‡</sup>  
Chen Zhu<sup>\*</sup> Yun He<sup>\*</sup> Yun-Nung Chen<sup>†</sup> Jason Weston<sup>‡</sup> Yuandong Tian<sup>‡</sup>  
Arash Rahnama<sup>\*</sup> Sinong Wang<sup>\*</sup> Hao Ma<sup>\*</sup> Han Fang<sup>\*</sup>

<sup>\*</sup>Meta GenAI <sup>†</sup>National Taiwan University <sup>‡</sup>Meta FAIR <sup>§</sup>UC Berkeley  
ytl@ieee.org

## Abstract

Large language models (LLMs) have recently demonstrated remarkable success in mathematical reasoning. Despite progress in methods like chain-of-thought prompting and self-consistency sampling, these advances often focus on final correctness without ensuring that the underlying reasoning process is coherent and reliable. This paper introduces STEP-KTO, a training framework that combines process-level and outcome-level binary feedback to guide LLMs toward more trustworthy reasoning trajectories. By providing binary evaluations for both the intermediate reasoning steps and the final answer, STEP-KTO encourages the model to adhere to logical progressions rather than relying on superficial shortcuts. Our experiments on challenging mathematical benchmarks show that STEP-KTO significantly improves both final answer accuracy and the quality of intermediate reasoning steps. For example, on the MATH-500 dataset, STEP-KTO achieves a notable improvement in Pass@1 accuracy over strong baselines. These results highlight the promise of integrating stepwise process feedback into LLM training, paving the way toward more interpretable and dependable reasoning capabilities.

## 1 Introduction

Large language models (LLMs) have recently shown remarkable capabilities in reasoning-intensive tasks such as coding (Chen et al., 2021; Li et al., 2022; Rozière et al., 2023) and solving complex mathematical problems (Shao et al., 2024; Azerbayev et al., 2024). Prompting strategies like chain-of-thought prompting (Nye et al., 2021; Wei et al., 2022; Kojima et al., 2022; Adolphs et al., 2022) and self-consistency sampling (Wang et al., 2023) enhance these models’ final-answer accuracy by encouraging them to articulate intermediate reasoning steps. However, a significant issue remains: even when these methods boost final-answer correctness, the internal reasoning steps are often unreliable or logically inconsistent (Uesato et al., 2022; Lightman et al., 2024).

This discrepancy between correct final answers and flawed intermediate reasoning limits our ability to trust LLMs in scenarios where transparency and correctness of each reasoning stage are crucial (Lanham et al., 2023). For example, in mathematical problem-solving, a model might produce the right answer for the wrong reasons (Lyu et al., 2023; Zheng et al., 2024), confounding our understanding of its true capabilities (Turpin et al., 2023). To address this, researchers are increasingly emphasizing the importance of guiding models to produce not just correct final answers, but also verifiable and faithful step-by-step solution paths (Uesato et al., 2022; Shao et al., 2024; Setlur et al., 2024).

Prior work in finetuning has largely focused on outcome-level correctness, using outcome reward models to improve the probability of final-answer accuracy (Cobbe et al., 2021; Hosseini et al., 2024; Zhang et al., 2024). While effective, such an approach does not ensure that the intermediate reasoning steps are valid. Conversely, while process-level supervision through process reward models (PRMs) (Lightman et al., 2024; Wang et al., 2024; Luo et al., 2024) can guide models to follow correct reasoning trajectories, prior work has mainly used PRMs as a ranking method rather than a way to provide stepwise feedback. As a result, relying solely on process-level supervision may lead models to prioritize step-by-step correctness without guaranteeing a correct final outcome.

In this paper, we introduce Stepwise Kahneman-Tversky-inspired Optimization (STEP-KTO), a training framework that integrates both process-level and outcome-level binary feedback to produce coherent and correct reasoning steps alongside high-quality final answers. Our approach evaluates each intermediate reasoning step against known correct patterns using a PRM, while simultaneously leveraging a rule-based reward signal for the final answer. To fuse these signals, we employ a Kahneman-Tversky-inspired value function (Tversky and Kahneman, 2016; Ethayarajh et al., 2024) that emphasizes human-like risk and loss

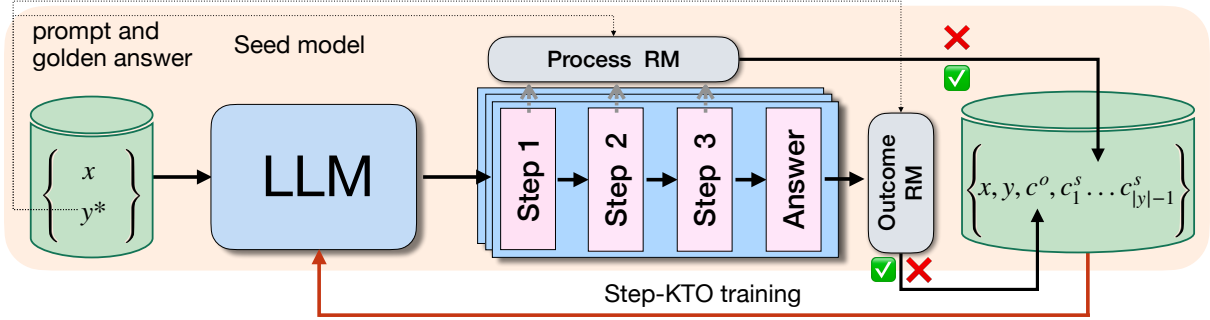


Figure 1: **STEP-KTO Training Process.** Given a dataset of math problems (left), a language model (LLM) produces both reasoning steps and a final answer. Each intermediate reasoning step is evaluated by a process reward model (Process RM), and the final answer is assessed by an outcome reward model (Outcome RM). The binary feedback signals from both levels (outcome-level correctness  $c^o$  and stepwise correctness  $c_h^s$ ) are recorded together with the input ( $x$ ) and the model’s response ( $y$ ) §2.1. These signals are then used to compute the STEP-KTO loss, guiding the LLM to not only produce correct final answers but also maintain coherent and correct reasoning steps §2.3. Through multiple iterations of this training process §2.4, the model progressively improves both its stepwise reasoning and final answer accuracy.

aversion, encouraging models to gradually correct their reasoning and avoid errors. The result is a training objective that aligns the entire reasoning trajectory with verified solutions while ensuring that final correctness remains a top priority.

Figure 1 illustrates the STEP-KTO pipeline. We start with a base LLM and repeatedly refine it through iterative training. At each iteration, the PRM provides step-level binary feedback that helps the model navigate correct solution paths, while the outcome-level binary feedback ensures that the final answer is correct. The Kahneman-Tversky-inspired value function transforms these binary signals into guidance that progressively reduces errors in the chain-of-thought. Over successive rounds, STEP-KTO yields systematically more accurate intermediate reasoning steps and steadily improves the final-answer accuracy.

We evaluate STEP-KTO on challenging mathematical reasoning benchmarks including MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024), AMC23 (MAA, 2023), and AIME24 (MAA, 2024). Our experiments show that incorporating both process-level and outcome-level signals leads to substantial improvements over state-of-the-art baselines that rely solely on final-answer supervision. For example, on MATH-500, STEP-KTO improves Pass@1 accuracy from 53.4% to 63.2%, while also producing more coherent and trustworthy step-by-step reasoning. Moreover, iterative training with STEP-KTO achieves cumulative gains, demonstrating that balancing process- and outcome-level feedback refines reasoning quality over time. In summary, our key contributions are:

- We propose STEP-KTO, a novel finetuning framework that combines process-level and outcome-level feedback, encouraging both correct final answers and faithful step-by-step reasoning.
- We show that iterative training with STEP-KTO yields consistent cumulative improvements, showing the effectiveness of combined process-level and outcome-level feedback in refining LLM reasoning.
- We demonstrate that STEP-KTO surpasses state-of-the-art baselines on multiple math reasoning tasks, delivering higher accuracy (63.2% vs 53.4% Pass@1 on MATH-500) and more reliable intermediate solutions.

## 2 Methodology

### 2.1 Problem Setup and Notation

We adopt the notation and setup similar to Setlur et al. (2024). Let  $\mathcal{D} = \{(x_i, \mathbf{y}_x^*)\}_i$  be a dataset of math problems, where each problem  $x \in \mathcal{X}$  has an associated ground-truth solution sequence  $\mathbf{y}_x^* = (s_1^*, s_2^*, \dots, s_{|y^*|}^*) \in \mathcal{Y}$ . A policy model  $\pi_\theta$ , parameterized by  $\theta$ , generates a response sequence  $\mathbf{y} = (s_1, s_2, \dots, s_{|y|})$  autoregressively given the problem  $x$ , where each step  $s_h$  is a reasoning step separated by a special token (e.g., “## Step”).

The correctness of the final answer  $\mathbf{y}$  can be automatically determined by a rule-based correctness function  $\text{Regex}(\mathbf{y}, \mathbf{y}_x^*) \in \{0, 1\}$ , which compares the model’s final derived answer to the ground-truth final answer (Hendrycks et al., 2021). The model’s final answer is explicitly denoted using a special format in the final step  $s_{|y|}$ , such as `boxed{·}`, al-

lowing the correctness function to easily extract and verify it. Our primary objective is to improve the expected correctness of the final answer:

$$\mathbb{E}_{\mathbf{x} \in \mathcal{D}, \mathbf{y} \sim \pi_\theta(\cdot | \mathbf{x})}[\text{Regex}(\mathbf{y}, \mathbf{y}_x^*)].$$

Ensuring a correct final answer does not guarantee logically sound intermediate reasoning. To address this, we incorporate a stepwise binary correctness signal  $\text{Prm}(\mathbf{x}, \mathbf{y}_x^*, s_h) \in \{0, 1\}$  for each reasoning step  $s_h$ . Unlike the final-answer correctness  $\text{Regex}$ , this signal directly measures whether each intermediate step is locally valid and aligns with proper problem-solving principles, without strictly mirroring the reference solution steps. We obtain these stepwise correctness evaluations by prompting an LLM (Llama-3.1-70B-Instruct) as our process reward model (PRM), following the structured template in Appendix D. In summary, we have two levels of binary signals:

- **Outcome feedback:**  $\text{Regex}(\mathbf{y}, \mathbf{y}_x^*) \in \{0, 1\}$  indicates if the final answer derived from  $\mathbf{y}$  is correct.
- **Stepwise feedback:**  $\text{Prm}(\mathbf{x}, \mathbf{y}_x^*, s_h) \in \{0, 1\}$  indicates if the intermediate reasoning step  $s_h$  is correct.

Our goal is to integrate both of these signals into the training objective of  $\pi_\theta$ . By doing so, we guide the model to produce not only correct final answers but also to maintain correctness, coherence, and reliability throughout its reasoning trajectory. This integrated approach will be formalized through the STEP-KTO framework.

## 2.2 KTO Background

KTO (Ethayarajh et al., 2024) aims to align a policy  $\pi_\theta$  with binary feedback using a Kahneman-Tversky-inspired value function (Tversky and Kahneman, 2016). Rather than maximizing the log-likelihood of preferred outputs or directly using reinforcement learning, KTO defines a logistic value function that is risk-averse for gains and risk-seeking for losses.

The original KTO loss focuses on the final-

answer level. Let:

$$r_\theta(x, y) = \log \frac{\pi_\theta(y | x)}{\pi_{\text{ref}}(y | x)}, \quad (1)$$

$$z_0 = \text{KL}(\pi_\theta(y' | x) \parallel \pi_{\text{ref}}(y' | x)), \quad (2)$$

$$v(x, y) = \begin{cases} \lambda_D \sigma(\beta(r_\theta(x, y) - z_0)) & \text{if } \text{Regex}(\mathbf{y}, \mathbf{y}_x^*) = 1, \\ \lambda_U \sigma(\beta(z_0 - r_\theta(x, y))) & \text{if } \text{Regex}(\mathbf{y}, \mathbf{y}_x^*) = 0. \end{cases} \quad (3)$$

Here,  $\pi_{\text{ref}}$  is a reference policy (typically the initial model checkpoint) that provides a baseline for comparison,  $\sigma$  is the logistic function,  $\beta > 0$  controls risk aversion, and  $\lambda_D, \lambda_U$  are weighting coefficients. The  $z_0$  term, where  $y'$  denotes an arbitrary output sequence, serves as a reference point to ensure balanced optimization. The KTO loss at the outcome level is:

$$L_{\text{KTO}}(\pi_\theta, \pi_{\text{ref}}) = \mathbb{E}_{x, y \sim D}[\lambda_y - v(x, y)], \quad (4)$$

where  $\lambda_y = \lambda_D$  if  $\text{Regex}(\mathbf{y}, \mathbf{y}_x^*) = 1$  and  $\lambda_y = \lambda_U$  if  $\text{Regex}(\mathbf{y}, \mathbf{y}_x^*) = 0$ .

## 2.3 STEP-KTO

While KTO ensures correctness of final answers, many reasoning tasks require validity at each intermediate step. We extend KTO by incorporating stepwise binary feedback  $\text{Prm}(\mathbf{x}, \mathbf{y}_x^*, s_h)$  to assess the quality of each reasoning step. We begin by defining an *implied reward* at the step level:

$$r_\theta(x, s_h) = \log \frac{\pi_\theta(s_h | x, s_{<h})}{\pi_{\text{ref}}(s_h | x, s_{<h})}.$$

This quantity can be viewed as the incremental advantage of producing step  $s_h$  under  $\pi_\theta$  compared to  $\pi_{\text{ref}}$ . It captures how much more (or less) reward is implied by choosing  $s_h$  over the reference model’s baseline likelihood, conditioned on the same context  $(x, s_{<h})$ . Next, we introduce a stepwise KL baseline:

$$z_0^{(\text{step})} = \text{KL}(\pi_\theta(s'_h | x, s'_{<h}) \parallel \pi_{\text{ref}}(s'_h | x, s'_{<h})).$$

Analogous to  $z_0$  at the outcome level,  $z_0^{(\text{step})}$  serves as a local reference point. It prevents the model from gaining reward merely by diverging from the reference and ensures that improvements are grounded in genuine reasoning quality. Given the

binary stepwise feedback  $\text{Prm}(\mathbf{x}, \mathbf{y}_x^*, s_h)$ , we define a value function that parallels the outcome-level case. If a step  $s_h$  is deemed stepwise-desirable, the model should increase its implied reward  $r_\theta(x, s_h)$  relative to  $z_0^{(step)}$  (Huang and Chen, 2024). Conversely, if  $s_h$  is stepwise-undesirable, the model is encouraged to lower that implied reward. Formally:

$$v^{(step)}(x, s_h) = \begin{cases} \lambda_D^{(step)} \sigma(\beta_{step}(r(x, s_h) - z_0^{(step)})) & \text{if } \text{Prm}(\mathbf{x}, \mathbf{y}_x^*, s_h) = 1, \\ \lambda_U^{(step)} \sigma(\beta_{step}(z_0^{(step)} - r(x, s_h))) & \text{if } \text{Prm}(\mathbf{x}, \mathbf{y}_x^*, s_h) = 0. \end{cases} \quad (5)$$

Here,  $\lambda_D^{(step)}$ ,  $\lambda_U^{(step)}$  and  $\beta_{step}$  mirror their outcome-level counterparts, controlling the strength of the reward or penalty at the granularity of individual steps. By leveraging these signals, the stepwise value function  $v^{(step)}$  directs the model’s distribution toward steps deemed correct and coherent, and away from those that are not. With these definitions, the stepwise loss is:

$$\mathcal{L}_{step}(\cdot) = \mathbb{E}_{\mathbf{x}, \mathbf{y}, s_h \sim D^{(step)}} [\lambda_y^{(step)} - v^{(step)}(x, s_h)]. \quad (6)$$

where  $\lambda_y^{(step)} = \lambda_D^{(step)}$  if  $\text{Prm}(\mathbf{x}, \mathbf{y}_x^*, s_h) = 1$  and  $\lambda_y^{(step)} = \lambda_U^{(step)}$  if  $\text{Prm}(\mathbf{x}, \mathbf{y}_x^*, s_h) = 0$ .

Combining the stepwise objective with the outcome-level KTO loss (Eq. 4) yields the final STEP-KTO objective:

$$\mathcal{L}_{STEP-KTO}(\pi_\theta, \pi_{ref}) = \mathcal{L}_{KTO}(\pi_\theta, \pi_{ref}) + \mathcal{L}_{step}(\pi_\theta, \pi_{ref}). \quad (7)$$

This composite loss encourages the model to produce not only correct final answers but also to refine each intermediate step. By jointly optimizing outcome-level and stepwise-level feedback, STEP-KTO ensures that the model’s entire reasoning trajectory—from the earliest steps to the final solution—is both correct and coherent.

## 2.4 Iterative Training

We train our models using an iterative procedure inspired by previous alignment methods that refine a model’s parameters over multiple rounds (Zelikman et al., 2022; Yuan et al., 2024; Pang et al., 2024; Prasad et al., 2024). For Llama-3.3-70B-Instruct, we use it directly as our seed model  $M_0$ . For Llama-3.1 models, we first perform supervised finetuning on the training data before using them as  $M_0$ . Starting from  $M_0$ , we refine it iteratively to obtain  $M_1, M_2, \dots, M_T$  using the following procedure:

1. **Candidate Generation:** For each problem  $x \in \mathcal{D}$ , we sample 8 candidate solutions  $\mathbf{y}^k \sim \pi_{M_t}(\cdot | x)$  using temperature  $T = 0.7$  and nucleus sampling with  $p = 0.95$  (Holtzman et al., 2020). This stochastic decoding strategy encourages diverse candidate solutions, aiding both positive and negative sample selection.
2. **Outcome Assessment:** We evaluate each candidate  $\mathbf{y}^k$  against the ground-truth solution  $\mathbf{y}_x^*$  using the outcome correctness function  $\text{Regex}(\mathbf{y}^k, \mathbf{y}_x^*)$ . If no sampled solutions are correct, we include the ground-truth solution  $\mathbf{y}_x^*$  as a positive sample, as suggested by Pang et al. (2024). If all sampled solutions are correct, we discard this problem in the current iteration to prioritize learning from problems where the model can still improve.
3. **Stepwise Evaluation:** For the selected solutions, we apply the stepwise correctness function  $\text{Prm}(\mathbf{x}, \mathbf{y}_x^*, s_h)$  to assess the quality of each reasoning step. This yields a set of binary signals indicating whether each intermediate step aligns with desirable reasoning patterns.
4. **Dataset Construction:** We aggregate these annotated samples into  $\mathcal{D}_{M_t} = \{(\mathbf{x}, \mathbf{y}, c^{out}, c_1^{step}, \dots, c_{S-1}^{step}) | \mathbf{y} \in \mathcal{D}\}$ , where  $c^{out} = \text{Regex}(\mathbf{y}, \mathbf{y}_x^*)$  is the outcome-level correctness, and  $c_h^{step} = \text{Prm}(\mathbf{x}, \mathbf{y}_x^*, s_h)$  are the stepwise correctness indicators for the  $S - 1$  intermediate steps of the solution  $\mathbf{y}$ .<sup>1</sup>
5. **Parameter Update:** Using  $\mathcal{D}_{M_t}$ , we update the model parameters according to the chosen alignment objective—either our STEP-KTO loss or a baseline method (e.g., IRPO).
6. **Iteration:** We repeat this process for  $T$  iterations, each time producing a new model  $M_{t+1}$  refined from  $M_t$ .

While KTO and STEP-KTO does not inherently require balanced positive and negative samples, we impose this constraint for fairness when comparing against pairwise preference-based baselines like DPO. Specifically, we randomly sample at most two pairs per problem per iteration, ensuring a consistent number of training examples across different alignment strategies. This controlled sampling regime facilitates direct comparisons between

<sup>1</sup>At each iteration  $t$ , the dataset  $\mathcal{D}_{M_t}$  is constructed specifically from  $M_t$ . Thus,  $M_1$  is trained on the dataset derived from seed model  $M_0$  shared by all methods,  $M_2$  on the dataset derived from  $M_1$  specifically for method testing, and so forth.

methods and clarifies the impact of stepwise and outcome-level feedback on the model’s refinement process.

### 3 Experiments

#### 3.1 Task and Datasets

We evaluate our approach on established math reasoning benchmarks from high school competitions, testing the model’s ability to solve challenging problems across various domains and difficulties. All problems require a final answer, typically a number, simplified expression (e.g.,  $\frac{\pi}{2}$ ,  $1 \pm \sqrt{19}$ ), or short text (e.g., “east”).

- **MATH-500:** A 500-problem subset of the MATH dataset (Hendrycks et al., 2021), selected as in Lightman et al. (2024). It covers diverse subjects (e.g., Algebra, Geometry, Pre-calculus) for a broad evaluation of mathematical reasoning.
- **AMC23:** A test set of 40 problems from the American Mathematics Competitions (AMC 12, 2023)<sup>2</sup>. These problems are known for their subtlety and depth, providing a stringent reasoning test.
- **AIME24:** A test set of 30 problems from the American Invitational Mathematics Examination (AIME, 2024)<sup>3</sup>, typically requiring intricate multi-step reasoning and posing a higher-level challenge.

Following standard mathematical LLM evaluation practices (Hendrycks et al., 2021), we extract final answers from model outputs using regular expressions and verify their mathematical equivalence to ground-truth solutions with SYMPY<sup>4</sup>, accommodating minor stylistic differences. We report Pass@1 (accuracy of a single greedy completion from  $\pi_\theta$ ) and Maj@8 (accuracy from the majority answer among 8 solutions sampled at  $T = 0.7$  (Ackley et al., 1985; Ficler and Goldberg, 2017; Wang et al., 2023))<sup>5</sup>. These metrics provide a comprehensive assessment on challenging mathematical reasoning tasks, reflecting direct accuracy (Pass@1) and sampled robustness (Maj@8).

<sup>2</sup><https://github.com/QwenLM/Qwen2.5-Math/blob/main/evaluation/data/amc23/test.jsonl>

<sup>3</sup><https://github.com/QwenLM/Qwen2.5-Math/blob/main/evaluation/data/aime24/test.jsonl>

<sup>4</sup><https://github.com/sympy/sympy>

<sup>5</sup>Varying temperature ( $T = 0.5 - 1.0$ ) had limited impact on Maj@8 in pilot experiments.

In addition to these evaluation benchmarks, all experiments are conducted using a large-scale prompt set,  $\mathcal{S}_{\text{Numina}}$ , referred to as NuminaMath (LI et al., 2024). NuminaMath comprises a broad range of math problems and their solutions, totaling 438k examples, spanning difficulty levels from elementary to high school competition standards. To ensure the integrity of final answers, we remove subsets of synthetic questions and Orca Math problems (Mitra et al., 2024), as their correctness are not verified by human.

#### 3.2 Baseline Methods

We evaluate our proposed STEP-KTO against several strong baseline approaches for mathematical reasoning. All methods are trained using offline iterative optimization, with online preference learning left as future work:

- **RFT (Rejection Finetuning)** (Yuan et al., 2023): Performs supervised finetuning exclusively on solutions with correct final answers, relying on outcome-level filtering without explicit preference signals.
- **IRPO (Iterative Reasoning Preference Optimization)** (Pang et al., 2024): An iterative DPO (Rafailov et al., 2023) variant using outcome-level pairwise preferences, stabilized by an NLL loss, but lacks stepwise feedback.
- **KTO (Kahneman-Tversky Optimization)** (Ethayarajh et al., 2024): Employs an outcome-level, Kahneman-Tversky-inspired value function (see §2.2) for alignment, focusing on risk aversion but not incorporating stepwise signals.
- **SimPO and IPO** (Meng et al., 2024; Azar et al., 2024): DPO variants that utilize simplified outcome-level preference mechanisms for more stable optimization, without targeting stepwise correctness or advanced reasoning performance.
- **Step-DPO** (Lai et al., 2024): A DPO variant that optimizes stepwise preferences instead of outcome-level ones for granular supervision, but requires significant data processing and rejection sampling for intermediate steps.

#### 3.3 Main Results

Table 1 presents our main results, comparing STEP-KTO with various baseline methods and commercial systems across the MATH-500, AMC23, and AIME24 benchmarks. We report both Pass@1 and Maj@8 accuracy, as described in §3. Overall,

Method	MATH-500		AMC23		AIME24	
	Pass@1	Maj@8	Pass@1	Maj@8	Pass@1	Maj@8
<i>Llama-3.1-8B-Instruct</i>						
Seed model $M_0$	53.4	55.0	35.0	37.5	3.3	6.7
Rejection Finetuning $M_3$	53.8	56.0	30.0	32.5	10.0	6.7
IRPO $M_3$	55.4	59.2	35.0	40.0	6.7	6.7
KTO $M_3$	60.6	61.6	35.0	32.5	<b>16.7</b>	<b>16.7</b>
STEP-KTO (ours) $M_3$	<b>63.2</b>	<b>64.6</b>	<b>47.5</b>	<b>47.5</b>	<b>16.7</b>	<b>16.7</b>
<i>Llama-3.1-70B-Instruct</i>						
Seed model $M_0$	74.6	76.2	40.0	60.0	13.3	16.7
Rejection Finetuning $M_1$	74.8	73.6	55.0	60.0	13.3	13.3
IRPO $M_1$	74.4	74.8	55.0	57.5	10.0	13.3
KTO $M_1$	75.6	77.2	55.0	65.0	13.3	13.3
STEP-KTO (ours) $M_1$	<b>76.2</b>	<b>78.4</b>	<b>60.0</b>	<b>67.5</b>	<b>16.7</b>	<b>20.0</b>
<i>Llama-3.3-70B-Instruct <math>M_0</math></i>						
Seed model $M_0$	75.8	77.6	57.5	60.0	26.7	30.0
Rejection Finetuning $M_1$	77.4	78.4	60.0	65.0	20.0	23.3
IRPO $M_1$	78.6	80.8	55.0	57.5	23.3	26.7
KTO $M_1$	78.6	79.8	60.0	65.0	20.0	23.3
STEP-KTO (ours) $M_1$	<b>79.6</b>	<b>81.6</b>	<b>70.0</b>	<b>75.0</b>	<b>30.0</b>	<b>33.3</b>
<i>Commercial Models</i>						
Llama-3.1-8B-Instruct	51.4	55.2	15.0	27.5	3.3	3.3
Llama-3.1-70B-Instruct	64.8	70.4	37.5	47.5	10.0	30.0
Llama-3.1-405B-Instruct	68.8	74.4	47.5	52.5	30.0	26.6
O1	94.8	-	-	-	78.0	-
O1-Mini	90.0	-	90.0	90.0	33.3	46.7
Gemini 1.5 Pro	79.4	83.0	75.0	82.5	26.7	26.7
GPT-4o	73.0	76.4	57.5	70.0	10.0	16.7
Claude 3.5 Sonnet	70.0	74.4	62.5	67.5	23.3	26.7
Grok-Beta	67.0	72.2	50.0	52.5	10.0	13.3

Table 1: **Math problem solving performance** comparing Llama models of different sizes and proprietary models. Results show accuracy on MATH-500, AMC23, and AIME24 test sets using both greedy decoding (Pass@1) and majority voting over 8 samples (Maj@8). Models highlighted in blue are 8B parameter models, green are 70B parameter models, and gray are commercial models.

STEP-KTO consistently outperforms the baselines that rely solely on outcome-level correctness, such as KTO, IRPO, SimPO, and IPO, as well as simpler methods like RFT.

For instance, on MATH-500 with the 8B Llama-3.1-Instruct model, STEP-KTO achieves a Pass@1 of 63.2%, improving from the baseline KTO model’s 60.6% and substantially surpassing IRPO and RFT. On AMC23, STEP-KTO attains a Pass@1 of 47.5%, outperforming baselines by a notable margin. On AIME24, where problems require especially intricate multi-step reasoning, STEP-KTO sustains its advantage, demonstrating that the stepwise supervision is particularly valuable for more challenging tasks. Scaling to the 70B further improves results. Llama-3.1-70B-Instruct with STEP-KTO reaches a Pass@1 of 76.2% on MATH-500 and continues to excel on AMC23 (60.0%) and AIME24 (16.7%). Llama-3.3-70B-Instruct with STEP-KTO model pushes performance higher still, with STEP-KTO achieving 79.6% on MATH-500, 70.5% on AMC23, and 29.6% on AIME24. Although larger models also benefit from outcome-only alignment techniques, STEP-KTO still deliv-

ers consistent gains, indicating that even powerful models trained on extensive data can be further improved by targeting intermediate reasoning quality. Compared to strong proprietary models, STEP-KTO-enhanced Llama models remain competitive and close the performance gap. For example, while GPT-4o achieves a respectable 73.0% Pass@1 on MATH-500, O1 series pushes this accuracy to 90.0% and higher but requires a substantially larger inference budget. In contrast, our STEP-KTO-enhanced Llama-3.1-70B-Instruct model attains 76.2% Pass@1 on MATH-500 using only a 5k-token budget.

### 3.4 Iterative Training

Table 2 illustrates how model performance evolves over multiple iterative training rounds ( $M_1, M_2, M_3$ ) when starting from the same seed model  $M_0$  (Llama-3.1-8B-Instruct). We compare STEP-KTO against other iterative methods such as IRPO, KTO, and Rejection Finetuning.

Overall, STEP-KTO not only achieves higher final performance but also improves more consistently across iterations. For instance, on MATH-

Method	MATH-500		AMC23		AIME24	
	Pass@1	Maj@8	Pass@1	Maj@8	Pass@1	Maj@8
<i>Llama-3.1-8B-Instruct</i>						
Seed model $M_0$	53.4	55.0	35.0	37.5	3.3	6.7
IPO $M_1$	52.6	55.8	22.5	30.0	3.3	3.3
SimPO $M_1$	55.8	57.2	25.0	25.0	6.7	10.0
Step-DPO $M_1$	56.8	58.4	27.5	30.0	6.7	10.0
Rejection Finetuning $M_1$	55.0	57.0	30.0	35.0	10.0	10.0
Rejection Finetuning $M_2$	54.0	56.2	22.5	20.0	3.3	6.7
Rejection Finetuning $M_3$	53.8	56.0	30.0	32.5	10.0	6.7
IRPO $M_1$	58.2	59.6	35.0	35.0	10.0	10.0
IRPO $M_2$	57.2	62.4	32.5	40.0	6.7	10.0
IRPO $M_3$	55.4	59.2	35.0	40.0	6.7	6.7
KTO $M_1$	56.2	55.6	32.5	32.5	6.7	10.0
KTO $M_2$	59.4	62.8	35.5	35.0	<b>16.7</b>	<b>16.7</b>
KTO $M_3$	60.6	61.6	35.0	32.5	<b>16.7</b>	<b>16.7</b>
STEP-KTO (ours) $M_1$	59.4	60.6	22.5	32.5	13.3	10.0
STEP-KTO (ours) $M_2$	<b>63.6</b>	63.0	40.0	40.0	13.3	<b>16.7</b>
STEP-KTO (ours) $M_3$	63.2	<b>64.6</b>	<b>47.5</b>	<b>47.5</b>	<b>16.7</b>	<b>16.7</b>

Table 2: **Iterative training performance** comparing different methods on Llama-3.1-8B-Instruct model. Results show accuracy across multiple iterations ( $M_1$ ,  $M_2$ ,  $M_3$ ) of training on MATH-500, AMC23, and AIME24 test sets using both greedy decoding (Pass@1) and majority voting over 8 samples (Maj@8).

500, STEP-KTO progresses from 59.4% Pass@1 at  $M_1$  to 63.2% at  $M_3$ , surpassing the gains observed by IRPO and KTO at the same checkpoints. Similarly, on AMC23 and AIME24, STEP-KTO shows steady iterative improvements, reflecting the cumulative value of integrating both process- and outcome-level feedback. In contrast, Rejection Finetuning (RFT) and IRPO exhibit less stable gains across iterations, with performance sometimes plateauing or even regressing at later rounds. KTO does improve over iterations, but not as robustly as STEP-KTO, highlighting that stepwise feedback adds tangible benefits beyond what outcome-level optimization alone can achieve.

These results underscore the importance of iterative refinement. While simply applying preference-based or rejection-based finetuning may yield some initial improvements, STEP-KTO’s combined stepwise and outcome-level guidance drives steady, sustained enhancements in mathematical reasoning quality, iteration after iteration.

### 3.5 Comparison with Step-DPO

Step-DPO (Lai et al., 2024) also targets intermediate steps but relies on computationally intensive rejection sampling for error correction. STEP-KTO contrasts by efficiently combining stepwise and outcome signals for global solution coherence. Empirically, Step-DPO achieved 56.8% Pass@1 on MATH-500 ( $M_1$ ), whereas STEP-KTO reached 59.4%. Our Step-DPO implementation used Llama-3.3-70B-Instruct for error identifi-

cation and rejection sampling (filtering unsolved after 8 attempts), underscoring STEP-KTO’s advantage in sustained improvement via integrated optimization.

### 3.6 Preference Optimization Variants

Table 2 compares STEP-KTO against baselines over iterative training from the 8B  $M_0$ . On MATH-500 ( $M_1$ ), STEP-KTO (59.4% Pass@1) outperformed IPO (52.6%), SimPO (55.8%), IRPO (58.2%), and KTO (56.2%). While its initial  $M_1$  gains on AMC23 and AIME24 were comparable or more modest, STEP-KTO demonstrated stronger subsequent improvements. By  $M_3$ , STEP-KTO achieved 47.5% Pass@1 on AMC23, surpassing all baselines, and tied for the highest Pass@1 (16.7%) on AIME24, highlighting the value of integrating stepwise and outcome-level signals.

### 3.7 Evaluating Reasoning Quality

8B Model	Stepwise Errors in Correct Solutions	
	KTO	STEP-KTO
$M_0$	27.3%	27.3%
$M_1$	24.6%	<b>22.9%</b>
$M_2$	22.6%	<b>20.8%</b>
$M_3$	21.1%	<b>19.9%</b>

Table 3: **Reasoning Quality Analysis** comparing the ratio of solutions that arrive at correct final answers despite containing erroneous intermediate steps on the MATH-500.

To assess the internal consistency of solutions with correct final answers, we evaluate the propor-

tion of solutions that, despite having correct final answer  $\text{Regex}(\mathbf{y}, \mathbf{y}_x^*) = 1$ , contain at least one erroneous intermediate step. We use the ProcessBench (Zheng et al., 2024) as our evaluation framework, which is prompted to identify the earliest error in the generated solution  $\mathbf{y}$ , as detailed in its benchmark construction. Additionally, we utilize the critique capabilities of the QwQ-32B-Preview model (Qwen, 2024) to identify the first error in the reasoning. We prompt QwQ using the prompt detailed in Appendix D. We then measure the percentage of correctly answered problems where QwQ identifies at least one erroneous intermediate step.

Table 3 shows the percentage of correctly answered solutions containing errors in reasoning steps, starting from the initial 8B seed model  $M_0$ , which produces reasoning steps containing errors in 27.3% of its correctly answered solutions on the MATH-500 test set. Both STEP-KTO and KTO reduce the prevalence of such errors across iterations, with STEP-KTO showing a greater and more consistent reduction from 27.3% at  $M_0$  to 19.9% at  $M_3$ , compared to KTO’s more modest improvement to 21.1%.

## 4 Related Work

**Outcome-Oriented Methods.** Many efforts refine LLMs using only final outputs. Instruction tuning (Ouyang et al., 2022; Touvron et al., 2023) and outcome-level feedback via Reinforcement Learning from Human Feedback (RLHF) (e.g., InstructGPT (Ouyang et al., 2022)) or direct preference optimization (DPO (Rafailov et al., 2023), KTO (Ethayarajh et al., 2024), SimPO (Meng et al., 2024), IPO (Azar et al., 2024)) improve final answer accuracy using human or synthetic labels. AI-generated feedback (RLAIF (Lee et al., 2023)) or predefined rules (Constitutional AI (Bai et al., 2022b)) aim to reduce human annotation. While refinements like CGPO (Xu et al., 2024) offer richer signals, they primarily evaluate entire outputs. A key limitation is that correct final answers do not guarantee sound intermediate reasoning (Wu et al., 2024), potentially yielding untrustworthy solution paths (Turpin et al., 2023; Lanham et al., 2023).

**Process-Level Feedback and Verification.** Process Reward Models (PRMs) (Lightman et al., 2024; Uesato et al., 2022; Xiong et al., 2024; Luo et al., 2024) focus on stepwise correctness, assigning local feedback to guide models toward logically consistent solutions. This is prevalent in math

reasoning, supported by datasets like PRM800K (Lightman et al., 2024), CriticBench (Lin et al., 2024), and ProcessBench (Zheng et al., 2024) that facilitate step-level evaluations. PRM-based techniques influence decoding (Li et al., 2023; Chuang et al., 2024; Wang et al., 2024), re-ranking (Cobbe et al., 2021), filtering (Dubey et al., 2024; Shao et al., 2024), and iterative loops like STaR (Zelikman et al., 2022) and ReST (Gülçehre et al., 2023; Singh et al., 2024). Synthetic feedback helps scale annotations (Wang et al., 2024; Lightman et al., 2024; Chiang and Lee, 2024; Huang and Chen, 2024). Yet, focusing solely on process may not yield correct final answers, as local rewards can be exploited or chains may fail to converge (Gao et al., 2024).

**Integrating Outcome- and Process-Level Signals.** Recognizing the limitations of supervising only outcomes or processes, recent studies combine both signals. FactTune (Tian et al., 2024) and FactAlgin (Huang and Chen, 2024) integrate PRMs with factuality evaluators for alignment, enhancing factual accuracy. In math reasoning, Uesato et al. (2022) and Shao et al. (2024) also leveraged combined step and outcome feedback. While the principle of multi-granularity supervision is broadly applicable, especially to math reasoning, these combined approaches can still face challenges in scaling, balancing feedback types, and avoiding premature performance plateaus (Bai et al., 2022a; Xu et al., 2023; Singh et al., 2024).

## 5 Conclusion

This work proposes STEP-KTO, a training framework that leverages both outcome-level and process-level binary feedback to guide large language models toward more coherent, interpretable, and dependable reasoning. By integrating stepwise correctness signals into the alignment process, STEP-KTO improves the quality of intermediate reasoning steps while maintaining or even enhancing final answer accuracy. Our experiments on challenging mathematical reasoning benchmarks demonstrate consistent gains in performance, particularly under iterative training and for complex reasoning tasks. These findings underscore the value of aligning not only final outcomes but also the entire reasoning trajectory. We envision STEP-KTO as a stepping stone toward more reliable reasoning in LLMs.



## Limitations

Despite STEP-KTO’s promise, several limitations persist. First, outcome-level feedback can be noisy; for instance, automated math answer verification may misjudge valid but unconventional representations, limiting training signal precision. Second, STEP-KTO currently presumes access to ground-truth solutions for outcome and (implicitly) for guiding stepwise correctness. Generating meaningful stepwise feedback is challenging without high-quality reference reasoning or in domains with inherently ambiguous intermediate steps. Learning from weaker signals or pure preferences remains an open area. Finally, our experiments assume some baseline correctness. If initial outcomes are consistently incorrect and intermediate steps are invalid, STEP-KTO’s ability to bootstrap performance is uncertain. Such scenarios might require complementary techniques like curriculum learning or stronger initialization before stepwise feedback becomes effective.

## Acknowledgements

We thank the anonymous reviewers for their helpful comments. This paper’s writing received minor language-polishing suggestions from ChatGPT. In addition, parts of our experimental code were drafted or refactored with assistance from Meta AI; all final implementations were manually reviewed and verified by the authors.

## References

- David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. 1985. [A learning algorithm for boltzmann machines](#). *Cogn. Sci.*, 9(1):147–169.
- Leonard Adolphs, Kurt Shuster, Jack Urbanek, Arthur Szlam, and Jason Weston. 2022. [Reason first, then respond: Modular generation for knowledge-infused dialogue](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 7112–7132. Association for Computational Linguistics.
- Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Rémi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. [A general theoretical paradigm to understand learning from human preferences](#). In *International Conference on Artificial Intelligence and Statistics, 2-4 May 2024, Palau de Congressos, Valencia, Spain*, volume 238 of *Proceedings of Machine Learning Research*, pages 4447–4455. PMLR.
- Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. [Llemma: An open language model for mathematics](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022a. [Training a helpful and harmless assistant with reinforcement learning from human feedback](#). *arXiv preprint arXiv:2204.05862*.
- Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, and 32 others. 2022b. [Constitutional AI: harmlessness from AI feedback](#). *CoRR*, abs/2212.08073.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. [Evaluating large language models trained on code](#). *CoRR*, abs/2107.03374.
- Cheng-Han Chiang and Hung-yi Lee. 2024. [Merging facts, crafting fallacies: Evaluating the contradictory nature of aggregated factual claims in long-form generations](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2734–2751, Bangkok, Thailand. Association for Computational Linguistics.
- Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R. Glass, and Pengcheng He. 2024. [DoLa: Decoding by contrasting layers improves factuality in large language models](#). In *The Twelfth International Conference on Learning Representations*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *CoRR*, abs/2110.14168.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 82 others. 2024. [The llama 3 herd of models](#). *CoRR*, abs/2407.21783.

- Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. [Model alignment as prospect theoretic optimization](#). In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.
- Jessica Fidler and Yoav Goldberg. 2017. [Controlling linguistic style aspects in neural language generation](#). In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, Copenhagen, Denmark. Association for Computational Linguistics.
- Jiaxuan Gao, Shusheng Xu, Wenjie Ye, Weilin Liu, Chuyi He, Wei Fu, Zhiyu Mei, Guangju Wang, and Yi Wu. 2024. [On designing effective RL reward at training time for LLM reasoning](#). *CoRR*, abs/2410.15115.
- Çağlar Gülçehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. 2023. [Reinforced self-training \(rest\) for language modeling](#). *CoRR*, abs/2308.08998.
- 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](#). In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. [The curious case of neural text degeneration](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
- Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron C. Courville, Alessandro Sordani, and Rishabh Agarwal. 2024. [V-star: Training verifiers for self-taught reasoners](#). *CoRR*, abs/2402.06457.
- Chao-Wei Huang and Yun-Nung Chen. 2024. [FactAlign: Long-form factuality alignment of large language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16363–16375, Miami, Florida, USA. Association for Computational Linguistics.
- Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. [Large language models are zero-shot reasoners](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.
- Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangu Peng, and Jiaya Jia. 2024. [Step-dpo: Step-wise preference optimization for long-chain reasoning of llms](#). *CoRR*, abs/2406.18629.
- Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamile Lukosiute, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, and 11 others. 2023. [Measuring faithfulness in chain-of-thought reasoning](#). *CoRR*, abs/2307.13702.
- Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbone, and Abhinav Rastogi. 2023. [RLAIF: scaling reinforcement learning from human feedback with AI feedback](#). *CoRR*, abs/2309.00267.
- Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. [Numinamath](#). [<https://huggingface.co/AI-MO/NuminaMath-CoT>]([https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina\\_dataset.pdf](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf)).
- Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. [Inference-time intervention: Eliciting truthful answers from a language model](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d’Autume, Igor Babuschkin, Xinyun Chen, Posen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, and 7 others. 2022. [Competition-level code generation with alphacode](#). *Science*, 378(6624):1092–1097.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. [Let’s verify step by step](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Zicheng Lin, Zhibin Gou, Tian Liang, Ruilin Luo andG Haowei Liu, and Yujia Yang. 2024. [Criticbench: Benchmarking llms for critique-correct reasoning](#). In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 1552–1587. Association for Computational Linguistics.
- Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024. [Improve mathematical reasoning in language models by automated process supervision](#). *CoRR*, abs/2406.06592.

- Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. [Faithful chain-of-thought reasoning](#). In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, IJCNLP 2023 -Volume 1: Long Papers, Nusa Dua, Bali, November 1 - 4, 2023*, pages 305–329. Association for Computational Linguistics.
- MAA. 2023. American mathematics competitions (amc).
- MAA. 2024. American invitational mathematics examination (aime).
- Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. [SimpO: Simple preference optimization with a reference-free reward](#). *CoRR*, abs/2405.14734.
- Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. 2024. [Orca-math: Unlocking the potential of slms in grade school math](#). *CoRR*, abs/2402.14830.
- Maxwell I. Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. [Show your work: Scratchpads for intermediate computation with language models](#). *CoRR*, abs/2112.00114.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.
- Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. 2024. [Iterative reasoning preference optimization](#). *CoRR*, abs/2404.19733.
- Archiki Prasad, Weizhe Yuan, Richard Yuanzhe Pang, Jing Xu, Maryam Fazel-Zarandi, Mohit Bansal, Sainbayar Sukhbaatar, Jason Weston, and Jane Yu. 2024. [Self-consistency preference optimization](#). *Preprint*, arXiv:2411.04109.
- Qwen. 2024. Qwq-32b preview. <https://qwenlm.github.io/blog/qwq-32b-preview/>. Accessed: 2024-06-17.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. [Direct preference optimization: Your language model is secretly a reward model](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton-Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, and 6 others. 2023. [Code llama: Open foundation models for code](#). *CoRR*, abs/2308.12950.
- Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2024. [Rewarding progress: Scaling automated process verifiers for LLM reasoning](#). *CoRR*, abs/2410.08146.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [Deepseekmath: Pushing the limits of mathematical reasoning in open language models](#). *CoRR*, abs/2402.03300.
- Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, and 1 others. 2024. [Beyond human data: Scaling self-training for problem-solving with language models](#). *Transactions on Machine Learning Research*. Expert Certification.
- Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D Manning, and Chelsea Finn. 2024. [Fine-tuning language models for factuality](#). In *The Twelfth International Conference on Learning Representations*.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. [Llama: Open and efficient foundation language models](#). *CoRR*, abs/2302.13971.
- Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. [Language models don’t always say what they think: Unfaithful explanations in chain-of-thought prompting](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Amos Tversky and Daniel Kahneman. 2016. *Advances in Prospect Theory: Cumulative Representation of Uncertainty*, pages 493–519. Springer International Publishing, Cham.
- Jonathan Uesato, Nate Kushman, Ramana Kumar, H. Francis Song, Noah Y. Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. [Solving math word problems with process- and outcome-based feedback](#). *CoRR*, abs/2211.14275.
- Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. [Math-shepherd: Verify and reinforce LLMs](#)

- step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand. Association for Computational Linguistics.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. [Self-consistency improves chain of thought reasoning in language models](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- Tianhao Wu, Janice Lan, Weizhe Yuan, Jiantao Jiao, Jason Weston, and Sainbayar Sukhbaatar. 2024. [Thinking llms: General instruction following with thought generation](#). *CoRR*, abs/2410.10630.
- Wei Xiong, Chengshuai Shi, Jiaming Shen, Aviv Rosenberg, Zhen Qin, Daniele Calandriello, Misha Khalman, Rishabh Joshi, Bilal Piot, Mohammad Saleh, Chi Jin, Tong Zhang, and Tianqi Liu. 2024. [Building math agents with multi-turn iterative preference learning](#). *CoRR*, abs/2409.02392.
- Jing Xu, Andrew Lee, Sainbayar Sukhbaatar, and Jason Weston. 2023. [Some things are more CRINGE than others: Preference optimization with the pairwise cringe loss](#). *CoRR*, abs/2312.16682.
- Tengyu Xu, Eryk Helenowski, Karthik Abinav Sankararaman, Di Jin, Kaiyan Peng, Eric Han, Shao-liang Nie, Chen Zhu, Hejia Zhang, Wenxuan Zhou, Zhouhao Zeng, Yun He, Karishma Mandyam, Arya Talabzadeh, Madian Khabsa, Gabriel Cohen, Yuan-dong Tian, Hao Ma, Sinong Wang, and Han Fang. 2024. [The perfect blend: Redefining RLHF with mixture of judges](#). *CoRR*, abs/2409.20370.
- Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. [Self-rewarding language models](#). In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.
- Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. 2023. [Scaling relationship on learning mathematical reasoning with large language models](#). *CoRR*, abs/2308.01825.
- Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. [STaR: Bootstrapping reasoning with reasoning](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024. [Generative verifiers: Reward modeling as next-token prediction](#). *CoRR*, abs/2408.15240.
- Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024. [Processbench: Identifying process errors in mathematical reasoning](#). *arXiv preprint arXiv:2412.06559*.

## A Implementation Details

We use AdamW ( $\beta_1 = 0.9$ ,  $\beta_2 = 0.95$ , weight decay = 0.1) with a linear warmup for the first 100 steps and a cosine decay schedule that reduces the learning rate to  $0.1 \times$  its initial value. The starting learning rate is  $1.0 \times 10^{-6}$ , and we apply global norm gradient clipping of 1.0. The effective global batch size is set to approximately one million tokens, and we train for about 2000 steps, periodically evaluating our models during training on the hold-out test set from MATH (Hendrycks et al., 2021)<sup>6</sup> to select the best checkpoint for each method. For IRPO, we use an NLL weight of 0.2. We set  $\beta = 0.1$  for all methods. All training jobs are run on 64 H100 GPUs.

## B Decontamination

To prevent data leakage between training and test sets, we perform standard decontamination by normalizing text (converting to lowercase and removing non-alphanumeric characters) and checking for exact string matches between test questions and training prompts (Dubey et al., 2024). We remove any matching examples from the training data. This process is applied to all datasets in our evaluation. Even if mild contamination were present, we expect any resulting performance inflation to be small and consistent across all conditions, leaving the relative comparisons between our methods largely unaffected.

## C Details of API Usage for Proprietary Models

In our experiments, we evaluated several proprietary models via their respective APIs: O1 (metrics are self-reported), O1-Mini (o1-mini-2024-09-12, MATH-500 is self-reported<sup>7</sup>), Gemini 1.5 Pro (gemini-1.5-pro-002), GPT-4o (gpt-4o-2024-08-06), Claude 3.5 Sonnet (claude-3-5-sonnet-20241022), and Grok-Beta. These experiments took place on November 15 and 16, 2024. For each model, questions were used directly as user prompts. For greedy decoding, we set the temperature to 0.0 to ensure deterministic outputs, except for o1 models where we used temperature 1.0 due to API restrictions (only temperature 1.0 is allowed) and took the first sample. For sampling, we set the temperature to 0.7 and performed 8 generations per question to enable majority voting.

### Response Generation for Proprietary Models

**User:**

Please answer the following question step-by-step. Once you have the final answer, place it on a new line as: The final answer is  $\boxed{\text{answer}}$ \$.  
Question: {{ question }}

<sup>6</sup>MATH-500 questions are excluded.

<sup>7</sup>Numbers from <https://github.com/openai/simple-evals>

## D Prompts

Prompt templates<sup>8</sup> for generating solutions are given below in Appendix D.

### Response Generation Template from (Dubey et al., 2024)

**User:**

Solve the following math problem efficiently and clearly:

For simple problems (2 steps or fewer):  
Provide a concise solution with minimal explanation.

For complex problems (3 steps or more):  
Use this step-by-step format:

## Step 1: [Concise description]  
[Brief explanation and calculations]

## Step 2: [Concise description]  
[Brief explanation and calculations]

...

Regardless of the approach, always conclude with:

Therefore, the final answer is:  $\boxed{\text{answer}}$ . I hope it is correct.

Where [answer] is just the final number or expression that solves the problem.

Problem: {{ question }}

Prompt for Llama-3.1-70B-Instruct to provide stepwise feedback on candidate solutions  $y$ . The model analyzes each step  $s_h$  of a potential solution against the correct answer  $y^*$ , evaluating the reasoning and accuracy of each step. The feedback is structured in JSON format with fields for step number, reflection on the reasoning, and a binary decision on whether the step contributes positively to reaching the solution.

<sup>8</sup>The prompt template was from <https://huggingface.co/datasets/meta-llama/Llama-3.1-70B-Instruct-evals>

## Generation Prompt for Stepwise Feedback

### User:

Please analyze the following problem and its potential solution step-by-step. Provide feedback on each step and determine if it contributes positively to reaching the correct solution.

```
<problem>
{{ problem }}
</problem>
```

```
<correct solution>
{{ answer }}
</correct solution>
```

```
<potential answer>
{% for step in steps %}
<step {{ loop.index }}>
{{ step }}
</step {{ loop.index }}>
{% endfor %}
</potential answer>
```

Analyze your **potential solution** as if you had originally generated it. Carefully review each step, considering its reasoning, accuracy, and execution. Assess whether the step contributes positively to reaching the correct solution. Where necessary, refine the step to address any flaws or gaps. Use the correct answer as a ground truth reference to guide your analysis.

Provide your output in JSON format, where each element represents a step of the solution. Use the fields below:

- **step**: The step order number in the reasoning process.
- **reflection**: A concise evaluation of the accuracy of the reasoning in this step (point out why it helps or hinders the solution).
- **decision**: The evaluation of the step, either "positive" or "negative".

The expected output format follows:

```
```json
[
  {
    "step": 1,
    "reflection": "[evaluation of step 1 reasoning]",
    "decision": "positive"
  },
  {
    "step": 2,
    "reflection": "[evaluation of step 2 reasoning]",
    "decision": "negative"
  },
  ...\\
]
```
```

### # Notes

- Assign **"negative"** only to steps that are clearly incorrect and prevent the solution from progressing.
- Use the correct answer as one of the ground truth references.
- Ensure the reflections remain concise, with the total explanation under 200 words across all steps combined.

**User:**

The following is a math problem and a solution (split into paragraphs, enclosed with tags and indexed from 0):

```
<problem>
{{ problem }}
</problem>
```

```
<solution>
{% for step in steps %}
<step {{ loop.index }}>
{{ step }}
</step {{ loop.index }}>
{% endfor %}
</solution>
```

Your task is to review and critique the solution paragraph by paragraph. Once you identify an error in a paragraph, return the index of the paragraph where the earliest error occurs. Otherwise, return the index of -1 (which typically denotes "not found").

Please put your final answer (i.e., the index) in `\boxed{{}}`.



## E Qualitative Examples

We analyze several examples from Llama-3.3-70B-Instruct STEP-KTO  $M_1$  on MATH-500 to understand how Step-KTO helps improve mathematical reasoning. The examples demonstrate three key scenarios where Step-KTO provides effective feedback: (1) when all steps and the final answer are correct, (2) when intermediate steps contain errors but lead to the correct final answer, and (3) when both intermediate steps and the final answer are incorrect.

### Step-KTO Generation Example 1: Final Answer is Right and All Steps are Right

problem: For a point  $P$ , let  $d_1$ ,  $d_2$  and  $d_3$  represents the distances from  $P$  to the planes  $x - z = 0$ ,  $x - 2y + z = 0$ , and  $x + y + z = 0$ . Let  $S$  be the set of points  $P$  such that  $d_1^2 + d_2^2 + d_3^2 = 36$ . Find the region of the volume enclosed by  $S$ .

answer: "288  $\pi$ ",  
subject: "Precalculus",  
level: 4,  
unique\_id: "test/precalculus/1056.json"

Model prediction:

## Step 1: Understand the problem and the equation given

The problem provides an equation  $d_1^2 + d_2^2 + d_3^2 = 36$  where  $d_1$ ,  $d_2$ , and  $d_3$  represent the distances from a point  $P$  to the planes  $x - z = 0$ ,  $x - 2y + z = 0$ , and  $x + y + z = 0$  respectively. We need to find the region of the volume enclosed by the set  $S$  of points  $P$  that satisfy this equation.

## Step 2: Calculate the distance of a point to each plane

The distance  $d$  from a point  $(x_0, y_0, z_0)$  to a plane  $Ax + By + Cz + D = 0$  is given by the formula  $d = \frac{|Ax_0 + By_0 + Cz_0 + D|}{\sqrt{A^2 + B^2 + C^2}}$ . Applying this formula to each plane, we get:

$$\begin{aligned} - d_1 &= \frac{|x - z|}{\sqrt{1^2 + (-1)^2}} = \frac{|x - z|}{\sqrt{2}} \\ - d_2 &= \frac{|x - 2y + z|}{\sqrt{1^2 + (-2)^2 + 1^2}} = \frac{|x - 2y + z|}{\sqrt{6}} \\ - d_3 &= \frac{|x + y + z|}{\sqrt{1^2 + 1^2 + 1^2}} = \frac{|x + y + z|}{\sqrt{3}} \end{aligned}$$

## Step 3: Substitute the distance formulas into the given equation

Substituting  $d_1$ ,  $d_2$ , and  $d_3$  into the equation  $d_1^2 + d_2^2 + d_3^2 = 36$ , we get:  
 $\left(\frac{x - z}{\sqrt{2}}\right)^2 + \left(\frac{x - 2y + z}{\sqrt{6}}\right)^2 + \left(\frac{x + y + z}{\sqrt{3}}\right)^2 = 36$

## Step 4: Simplify the equation

Expanding and simplifying, we have:

$$\frac{(x - z)^2}{2} + \frac{(x - 2y + z)^2}{6} + \frac{(x + y + z)^2}{3} = 36$$

## Step 5: Further simplification and rearrangement

Multiplying through by 6 to clear the denominators gives:

$$3(x - z)^2 + (x - 2y + z)^2 + 2(x + y + z)^2 = 216$$

## Step 6: Expand the terms

Expanding each term yields:

$$3(x^2 - 2xz + z^2) + (x^2 - 4xy + 4y^2 + z^2 + 2xz - 4yz) + 2(x^2 + 2xy + 2xz + y^2 + 2yz + z^2) = 216$$

## Step 7: Combine like terms

Combining like terms, we get:

$$3x^2 - 6xz + 3z^2 + x^2 - 4xy + 4y^2 + z^2 + 2xz - 4yz + 2x^2 + 4xy + 4xz + 2y^2 + 4yz + 2z^2 = 216$$

## Step 8: Simplify further

This simplifies to:

$$6x^2 + 6y^2 + 6z^2 + 0xy + 0xz + 0yz = 216$$

## Step 9: Divide by 6

Dividing the entire equation by 6 gives:

$$x^2 + y^2 + z^2 = 36$$

## Step 10: Interpret the result

The equation  $x^2 + y^2 + z^2 = 36$  represents a sphere with radius 6 centered at the origin  $(0, 0, 0)$ .

The final answer is:  $\boxed{288\pi}$

## Step-KTO Generation Example 2: Final Answer is Right but Step 2 is Wrong

problem: For how many real values of  $\sqrt{120-\sqrt{x}}$  is an integer?  
answer: "11"  
subject: "Algebra"  
level: 5  
unique\_id: "test/algebra/1282.json"

Model prediction:

## Step 1: Understand the problem and identify the condition for  $\sqrt{120-\sqrt{x}}$  to be an integer. We need  $\sqrt{120-\sqrt{x}}$  to be an integer, which means  $120-\sqrt{x}$  must be a perfect square.

## Step 2: List all perfect squares less than or equal to 120.

The perfect squares less than or equal to 120 are: 0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, and 121. However, since we are considering  $120-\sqrt{x}$ , the relevant perfect squares are from 0 to 120, which are 0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, and 121 is not included because it exceeds 120.

## Step 3: Calculate the possible values of  $\sqrt{x}$ .

For each perfect square  $y$  from the list, we calculate  $\sqrt{x} = 120 - y$ . This gives us the possible values of  $\sqrt{x}$  as: 120, 119, 116, 111, 104, 95, 84, 71, 56, 39, 20.

## Step 4: Consider the validity of each possible value of  $\sqrt{x}$ .

Since  $\sqrt{x}$  cannot be negative and must be real, all calculated values in step 3 are valid because they are all positive or zero. However, we should also consider that  $\sqrt{x}$  itself must be real, meaning  $x \geq 0$ . Given that  $\sqrt{x} = 120 - y$ , the smallest value of  $\sqrt{x}$  is when  $y = 100$ , giving  $\sqrt{x} = 20$ . For  $y = 121$ ,  $\sqrt{x}$  would be negative, which is not valid for real  $x$ . Thus, we only consider perfect squares up to 100.

## Step 5: Count the number of valid values of  $\sqrt{x}$  that make  $\sqrt{120-\sqrt{x}}$  an integer.

From step 3 and considering the correction in step 4, we see that for each perfect square up to 100 (inclusive), we get a valid  $\sqrt{x}$ . Thus, we count the number of perfect squares from 0 to 100, which are 11 (0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100).

The final answer is:  $\boxed{11}$

QwQ Critique:  
 $\boxed{2}$

The error in the solution occurs in the 2nd paragraph, where the reasoning incorrectly states "The perfect squares less than or equal to 120 are: 0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, and 121." It's incorrect because 121 is greater than 120, not less than or equal to, and shouldn't be included in the list.

### Step-KTO Generation Example 3: Final Answer is Wrong

Problem: Find the greatest integer less than  $(\sqrt{7} + \sqrt{5})^6$ .

(Do not use a calculator!)

answer: "13535"

subject: "Intermediate Algebra"

level: 3

unique\_id: "test/intermediate\_algebra/607.json"

Model prediction:

## Step 1: Understand the problem and identify the key elements

We need to find the greatest integer less than  $(\sqrt{7} + \sqrt{5})^6$  without using a calculator. This involves expanding the given expression and simplifying it to a form where we can estimate its value.

## Step 2: Apply the Binomial Theorem to expand the expression

The Binomial Theorem states that for any non-negative integer  $n$ ,  $((a + b)^n = \sum_{k=0}^n \binom{n}{k} a^{n-k} b^k$ ). Applying this to our problem with  $a = \sqrt{7}$ ,  $b = \sqrt{5}$ , and  $n = 6$ , we get  $((\sqrt{7} + \sqrt{5})^6 = \binom{6}{0}(\sqrt{7})^6 + \binom{6}{1}(\sqrt{7})^5(\sqrt{5}) + \binom{6}{2}(\sqrt{7})^4(\sqrt{5})^2 + \binom{6}{3}(\sqrt{7})^3(\sqrt{5})^3 + \binom{6}{4}(\sqrt{7})^2(\sqrt{5})^4 + \binom{6}{5}(\sqrt{7})(\sqrt{5})^5 + \binom{6}{6}(\sqrt{5})^6$ .

## Step 3: Calculate each term of the expansion

Calculating each term:

$$- \binom{6}{0}(\sqrt{7})^6 = 1 \cdot 7^3 = 343$$

$$- \binom{6}{1}(\sqrt{7})^5(\sqrt{5}) = 6 \cdot 7^2 \cdot 7 \cdot \sqrt{5} = 6 \cdot 49 \cdot 7 \cdot \sqrt{5} = 2058\sqrt{5}$$

$$- \binom{6}{2}(\sqrt{7})^4(\sqrt{5})^2 = 15 \cdot 7^2 \cdot 5 = 15 \cdot 49 \cdot 5 = 3675$$

$$- \binom{6}{3}(\sqrt{7})^3(\sqrt{5})^3 = 20 \cdot 7 \cdot 7 \cdot 5 \cdot \sqrt{5} = 20 \cdot 49 \cdot 5 \cdot \sqrt{5} = 4900\sqrt{5}$$

$$- \binom{6}{4}(\sqrt{7})^2(\sqrt{5})^4 = 15 \cdot 7 \cdot 5^2 = 15 \cdot 7 \cdot 25 = 2625$$

$$- \binom{6}{5}(\sqrt{7})(\sqrt{5})^5 = 6 \cdot \sqrt{7} \cdot 5^2 \cdot \sqrt{5} = 6 \cdot \sqrt{7} \cdot 25 \cdot \sqrt{5} = 150\sqrt{35}$$

$$- \binom{6}{6}(\sqrt{5})^6 = 1 \cdot 5^3 = 125$$

## Step 4: Combine like terms and simplify

Combining like terms, we notice that terms with  $\sqrt{5}$  and  $\sqrt{35}$  will contribute to non-integer values, but we can simplify the expression by recognizing that the integer part of the expansion will come from the terms without square roots (since they are multiplied by integers) and the terms with square roots will contribute to the fractional part. The terms without square roots are  $(343 + 3675 + 2625 + 125 = 6768)$ .

## Step 5: Estimate the value of the expression with square roots

To estimate the value of the terms with square roots, we note that  $(2058\sqrt{5} \approx 2058 \cdot 2.236 = 4599.33)$ ,  $(4900\sqrt{5} \approx 4900 \cdot 2.236 = 10943.4)$ , and  $(150\sqrt{35} \approx 150 \cdot 5.916 = 886.4)$ . Summing these approximations gives us an estimate of the non-integer contributing terms:  $(4599.33 + 10943.4 + 886.4 \approx 15429.13)$ .

## Step 6: Combine the integer and non-integer parts

Adding the integer part (from step 4) to the estimated value of the non-integer part (from step 5), we get  $(6768 + 15429.13 \approx 22197.13)$ .

## Step 7: Determine the greatest integer less than the result

The greatest integer less than  $(22197.13)$  is  $(22197)$ .

The final answer is:  $\boxed{22197}$