Confidence Is All You Need: Few-Shot RL Fine-Tuning of Language Models

Pengyi Li
AIRI, Skoltech
Moscow
li.Pengyi@airi.net

Matvey Skripkin
AIRI, Skoltech
Moscow
skripkin@airi.net

Alexander Zubrey
Skotech
Moscow
Alexander.Zubrey@Skoltech.ru

Andrey Kuznetsov
AIRI, Skoltech
Moscow
Kuznetsov@airi.net

Ivan Oseledets
AIRI, Skoltech
Moscow
Oseledets@airi.net

Abstract

Large language models (LLMs) excel at reasoning, yet post-training remains critical for aligning their behavior with task goals. Existing reinforcement learning (RL) methods often depend on costly human annotations or external reward models. We propose Reinforcement Learning via Self-Confidence (RLSC), which uses the model's own confidence as reward signals-eliminating the need for labels, preference models, or reward engineering. Applied to Qwen2.5-Math-7B with only 16 samples per question and 10 or 20 training steps, RLSC improves accuracy by +13.4% on AIME2024, +21.2% on MATH500, +21.7% on Minerva Math, +20.8% on Olympiadbench, and +9.7% on AMC23. RLSC provides a simple, scalable post-training method for inference models, requiring only a small number of samples and unlabelled supervision.

1 Introduction

Large Language Models (LLMs) such as ChatGPT [1], Qwen [2] [22], and DeepSeek [16] [7] [8] demonstrate remarkable reasoning capabilities across a wide range of tasks. However, post-training optimization remains essential to further align model behavior with task-specific goals. Compared to supervised fine-tuning, **Reinforcement Learning (RL)** offers stronger generalization and has been widely adopted to improve LLM performance. Methods like DPO [19], PPO [20], and RLHF [18] are frequently used to align models with human preferences, while DeepSeek's GRPO [7] algorithm has improved reasoning through reward-driven learning.

Despite this progress, existing RL methods often rely on costly human-labeled data or carefully engineered reward functions. For example, RLHF requires extensive annotation efforts [18], while Test-Time Reinforcement Learning (TTRL) [24] generates pseudo-labels via majority voting over 64 responses per question, which leads to high computational overhead.

To address these limitations, we propose *Reinforcement Learning via Self Confidence* (**RLSC**), a novel paradigm that uses the model's own confidence in its outputs as a reward signal. Unlike RLHF and TTRL, which depend on external supervision or large-scale training datasets, RLSC gets feedback from the model's responses directly, eliminating the need for human labels, external models, or manual reward shaping. We believe that the internal knowledge of a vanilla pretrained LLM can lead to quality improvements on downstream tasks when combined with self-confidence analysis of output generations.
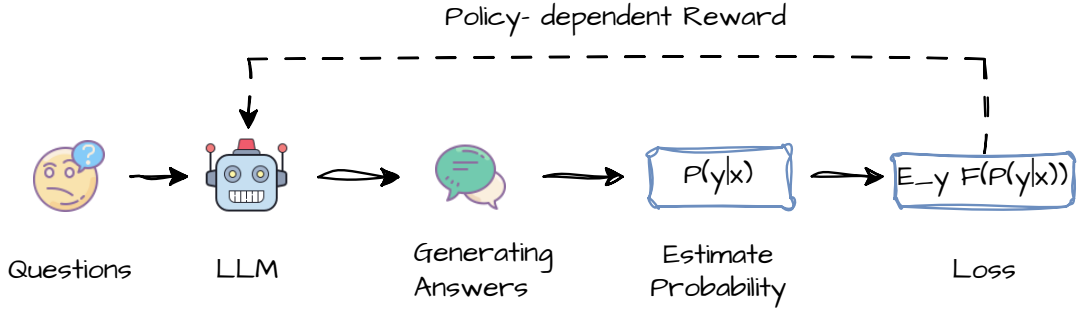
(a) Overview of the Reinforcement Learning via Self Confidence (RLSC) approach.
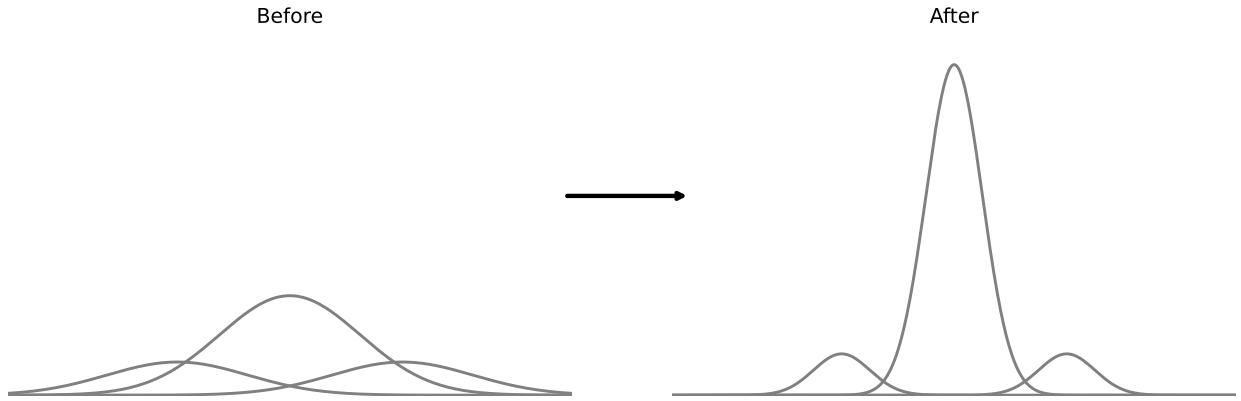
(b) Response probability distribution.

Figure 1: Combined visualization: (a) RL via Self Confidence workflow schema; (b) Probability distribution before and after training.

We verify the proposed RLSC approach on the small-scale model Qwen2.5-Math-7B [22] and conduct training using only the AIME2024 [14] training set for *10 or 20 steps*, with just 16 samples generated per question. Despite this lightweight training setup, RLSC yields significant improvements on multiple reasoning benchmarks: **+13.4%** on AIME2024, **+21.2%** on MATH500, **+21.7%** on Minerva Math, **+20.8%** on Olympiadbench, and **+9.7%** on AMC23. These results confirm that a strong pretrained model, combined with the RLSC framework, can effectively improve the model's confidence and generalization using a short additional training stage without relying on specific auxiliary datasets, human feedback and labeling or handcrafted reward functions.

Our main contributions are:

1. We introduce **RLSC**, a new reinforcement learning framework that requires *no* human labels, *no* external reward models, and *no* manual reward design.

2. We show that **RLSC** achieves strong performance with minimal training data and low compute cost, making it suitable for resource-constrained settings.

3. We validate **RLSC** in several inference benchmarks, using inference prompts and massage templates consistent with Qwen2.5-Math version, which highlights the potential of RLSC as a practical LLM training method to stimulate pre-trained models.

2 Method

2.1 From TTRL to Mode Sharpening

Test-Time Reinforcement Learning (TTRL) [24] improves large language models (LLMs) by generating many outputs per input (typically 64) and applying majority voting to select the most frequent completion. This pseudo-label is then used to fine-tune the model. While effective, this approach is computationally expensive and requires a clean separation between the answer and the reasoning trace - a non-trivial preprocessing step in practice.

Inspired by the idea of majority voting, we asked the following question:

what is the underlying principle behind this voting process?

Intuitively, majority voting selects the mode of the output distribution. Optimizing for agreement between sampled completions implicitly sharpens the distribution: it increases the probability mass concentrated on the most likely answer. This insight motivates us to replace TTRL external pseudo-labeling with a direct, internal objective based on mode sharpening.

Let $p_\theta(y \mid x)$ denotes the model probability of generating response y given input x, parameterized by θ. The probability that two independent samples from this distribution are identical is:

$$F(p_\theta) = \mathbb{E}_{y_1, y_2 \sim p_\theta(y|x)}[\mathbb{I}(y_1 = y_2)] = \sum_y p_\theta(y \mid x)^2 \tag{1}$$

This expression is maximized when the distribution collapses to a delta function centered on a single most probable response - i.e., when the model is confident.

Therefore, we propose to directly maximize the following self-confidence objective:

$$F(p_\theta) = \mathbb{E}_{y \sim p_\theta(y|x)}[p_\theta(y \mid x)] \tag{2}$$

This statement retains the benefits of TTRL (promoting stable and repeatable answers) while removing the need for pseudo-label extraction or majority voting. It serves as the foundation for our fine-tuning algorithm.

2.2 Self-Confidence Loss and Gradient

To optimize the self-confidence objective introduced above:

$$F(p_\theta) = \mathbb{E}_{y \sim p_\theta(y|x)}[p_\theta(y \mid x)] \tag{3}$$

we compute its gradient with respect to the model's parameters θ. Applying the log-trick, we obtain:

$$\nabla_\theta F(p_\theta) = \sum_y \nabla_\theta p_\theta(y \mid x) \cdot p_\theta(y \mid x) = \mathbb{E}_{y \sim p_\theta}[\nabla_\theta p_\theta(y \mid x)] = \mathbb{E}_{y \sim p_{\text{old}}}[p_{\text{old}}(y \mid x) \cdot \nabla_\theta \log p_\theta(y \mid x)] \tag{4}$$

Here, p_{old} denotes a frozen copy of the model (i.e., gradients do not propagate through it), used for sampling and weighting. This leads to the following training loss:

$$\mathcal{L}_1 = -\sum_y p_{\text{old}}(y \mid x) \cdot \log p_\theta(y \mid x) \tag{5}$$

This loss promotes higher log-probabilities for responses that the old model assigned higher confidence to. Crucially, it does not require an external reward model, does not have labeled data, and uses only the model's own belief distribution as feedback.

We also generalize this to a broader class of differentiable functions $\mathcal{L}(p_{\text{old}}, p_\theta)$. An effective variant smooths the weighting by adding a constant $\alpha > 0$:

$$\mathcal{L}_2 = -\sum_y (p_{\text{old}}(y \mid x) + \alpha) \cdot \log p_\theta(y \mid x) \tag{6}$$

This additive smoothing can stabilize optimization, especially when p_{old} is highly peaked or sparse. We empirically find that even small values of α (for example, 0.1) improve both convergence and generalization.

Overall these statements form the heart of the proposed RLSC training approach.

Table 1: Loss functions and corresponding optimized functionals

Name	Loss function	Functional
RLHF loss	$p_{\text{old}} \log p$	$\mathbb{E}_{p_\theta}[p_\theta]$
Shannon Entropy	$(1 + \log p_{\text{old}}) \log p$	$\mathbb{E}_{p_\theta}[\log p_\theta]$
Completion rewards	$R(y) \log p$	$\mathbb{E}_{p_\theta}[R(y)]$

2.3 Practical Training Setup

We apply the self-confidence objective to fine-tune the Qwen2.5-Math-7B model. For each training example, we generate a small batch of candidate completions using the base model: specifically, 16 samples per question drawn at a fixed temperature. These samples are treated as i.i.d. draws from p_{old}, and the current model distribution was kept constant during the gradient computation.

For each sample, we compute its log-probability under the updated model p_θ. The weighted loss is then evaluated using either the basic or smoothed self-confidence formulation.

$$\mathcal{L}_1 = -\sum_y p_{\text{old}}(y \mid x) \log p_\theta(y \mid x) \quad \text{or} \quad \mathcal{L}_2 = -\sum_y (p_{\text{old}}(y \mid x) + \alpha) \log p_\theta(y \mid x) \tag{7}$$

To optimize this loss, we adopt a standard autoregressive decoding and training pipeline as follows:

- For each question, generate 16 completions using generate(temperature = 0.5, num of samples=16)
- For each (prompt + answer) pair, tokenize and compute token-level log-probabilities
- Apply an **assistant mask** to isolate only the answer tokens
- Evaluate the sum of the masked log-probs to obtain the log-likelihood of the response
- Evaluate the loss and update model parameters via backpropagation

We train the model for only 10 or 20 steps on the AIME2024 dataset using 8 NVIDIA A100 GPUs (80GB). We adopt the AdamW optimizer with a learning rate of 1×10^{-5} and standard weight decay. The generation length is limited to 3072 tokens.

This minimal setup includes total absense of auxiliary datasets, instruction tuning, preference models and enables efficient, zero-label reinforcement learning at scale.

3 Experiments

3.1 Results Analysis

Benchmarks. We evaluated our method on several challenging benchmark datasets, including mathematical reasoning tasks (AIME24 [14], MATH500 [11], AMC23 [15], GSM8K [4]), Minerva Math [12], Olympiadbench [9], MMLU Stem [10] and the question-answering benchmark GPQADiamond [7].

Accuracy is defined as the ratio of correctly answered samples to the total number of evaluation samples as shown in Equations 8. **Pass@1** score is computed as Equation 9.

$$\text{Acc} = \frac{\text{\# Correct Answers}}{\text{\# Total Samples}} \tag{8}$$

$$\text{pass@1} = \frac{1}{k} \sum_{i=1}^{k} p_i \tag{9}$$

Algorithm 1 RLSC for LLM

```
# model.generate(prompt): generates multiple completions
# model.forward(input): returns token logits

for question in dataset:

    # generate completions
    completions = model.generate(question, temperature, num_samples)

    # get gradable probabilities
    logits = model.forward(question.repeat() + completions)[question.length:-1]
    all_log_probs = log_softmax(logits / temperature)
    log_p = all_log_probs.gather(token_ids).sum

    # compute loss
    loss = - (exp(log_p).detach() + alpha) * log_p

    loss.backward()
    optimizer.step()
```

To ensure a fair comparison between our model and the baseline, we re-evaluated both our checkpoints and the baseline using the same publicly available evaluation script ([6], [5]), with all experimental settings kept identical. The results are presented in Table 2.

Model	AIME24	MATH500	AMC23	GSM8K	GPQA	Olympiad	Minerva	MMLU
Qwen2.5-Math-1.5B	3.3	35.6	34.7	73.8	19.2	21.6	11.4	34.1
Ours	**6.7**	**62.4**	**46.2**	**74.6**	**15.8**	**29.9**	**26.1**	**48.6**
Δ	**+3.4**	**+26.8**	**+11.5**	**+0.8**	**-3.4**	**+8.3**	**+14.7**	**+14.5**
Qwen2.5-Math-7B	13.3	51.4	45.0	84.3	21.4	15.1	10.7	52.3
Ours	**26.7**	**72.6**	**54.7**	**86.3**	**24.1**	**35.9**	**32.4**	**57.6**
Δ	**+13.4**	**+21.2**	**+9.7**	**2.0**	**+2.7**	**+20.8**	**+21.7**	**5.3**

Table 2: Accuracy (%) on reasoning benchmarks for baseline Qwen2.5 models and RLSC-tuned variants. RLSC delivers consistent improvements across several benchmarks (AIME24, MATH500, AMC23, GSM8K, Olympiadbench, Minerva Math, MMLU Stem). All values computed using the same public evaluation script. Higher values indicate better accuracy.

Here, we evaluate model accuracy rather than Pass@1, as we believe that in real life, there is no room for trial and error-accuracy is what truly matters. From the results, it is evident that the original Qwen model encounters significant issues under direct evaluation; in fact, it often fails to function properly. Our approach builds upon this baseline and achieves substantial improvements through aggressive enhancement.

Significant improvements are realized on all three core benchmarks, AIME24 [14], MATH500 [11], Olympiadbench [9], Minerva Math [12] and AMC23 [15], and the advantage is especially prominent at 7B parameter scale (21.7% improvement for Minerva Math).

3.2 Emergent Behavior: Concise Reasoning without Prompting

We observed that RLSC fine-tuning enabled the model to produce shorter, more confident answers. Unlike other traditional approaches to fine-tuning that use the textual cue "Let's think step by step", our models learned to identify answers early and avoid redundant reasoning.

For example, in the AIME example (Case 1), the baseline model includes lengthy symbolic derivations and still fails. The RLSC-adjusted model answered the question directly and correctly, and with a cleaner logical flow. Similar patterns appear in benchmarks for other mathematics such as MATH and AMC23.

While we do not formally quantify the reduction in response length here, this trend is consistent across all benchmarks. This suggests that RLSC can implicitly enhance the credibility of intermediate reasoning.

We leave the precise characterisation (e.g. entropy, inference steps) for future work.

3.3 Qualitative Analysis

We extracted reasoning results from the MATH and AIME benchmarks and carried out a qualitative analysis. The findings are summarized below.

To qualitatively assess model behavior, we compared the outputs of the initial model and our RLSC fine-tuned model as shown block of "Comparison of Model Outputs". As illustrated below, the fine-tuned model exhibits improved task comprehension and reasoning under a zero-shot setting. Our experimental results demonstrate that on the MATH500 benchmark, the initial model could perform basic yet incorrect reasoning for Case 1, while failing completely to solve complex problems, eg. Case 2. Our method fine-tuning model demonstrates strong reasoning capabilities, and unlike methods that require lengthy **"step-by-step"** derivations, it arrives at accurate conclusions through simple reasoning paths.

We defer a full ablation of smoothing terms and sample counts to future work, but initial experiments suggest RLSC remains stable across a wide range of hyperparameters.

4 Related Work

Reinforcement Learning in Reasoning Tasks. In recent years, **Reinforcement Learning (RL)** has played a pivotal role in enhancing the reasoning capabilities of Large Language Models (LLMs). Models such as DeepSeek-R1 [7], ChatGPT o1 [1], QwQ [21], and Qwen have demonstrated impressive reasoning skills by decomposing complex problems into intermediate steps and engaging in deep deliberation prior to producing final responses capabilities often acquired and refined through reward-driven learning mechanisms.

A classic approach in this domain is *Reinforcement Learning from Human Feedback (RLHF)* [18], which aligns model behavior with human preferences by relying on human annotations or learned preference models to generate reward signals. However, RLHF is highly dependent on labor-intensive annotation, resulting in substantial costs.

To alleviate this dependency, *Reinforcement Learning with Verifiable Rewards (RLVR)* [17] [13] introduces a reward paradigm based solely on question-answer pairs (x, y^*), where the model-generated output is compared against the reference answer to compute a verifiable reward. While RLVR reduces reliance on annotated reasoning steps, it still requires human-labeled questions and answers, limiting its scalability.

Test-Time Training. More recently, **Test-Time Training (TTT)** [24] has emerged as a promising direction for further optimizing model behavior during inference. Notable examples include *SelfPlay Critic (SPC)* [3] and *Absolute Zero Reasoner (AZR)* [23], which employ adversarial dual-model frameworks inspired by game-theoretic learning. In these approaches, one model acts as a "sneaky generator" crafting challenging or misleading reasoning steps, while the other serves as a "critic" that learns to detect errors. These methods eliminate the need for human supervision but depend on external tools (e.g., Python executors or code verifiers) to supply feedback signals.

Another prominent TTT approach is *Test-Time Reinforcement Learning (TTRL)* [24]. It constructs pseudo-labels by sampling multiple candidate responses for each question and applying a majority-vote mechanism. The resulting consensus serves as a proxy label to compute rewards for model updates. Although TTRL avoids explicit human supervision, it requires a large number of samples (e.g., 64 per question), which leads to significant computational overhead.

Summary and Motivation. In summary, while RLHF [18], RLVR [17] [13], SPC [3], AZR [23], and TTRL [24] each propose different strategies for sourcing reinforcement signals, they invariably depend on some combination of human annotations, external models, or complex reward engineering.

5 Conclusion

We introduced **Reinforcement Learning via Self-Confidence (RLSC)**, a lightweight fine-tuning method that eliminates the need for labels, preference models, or handcrafted rewards. Unlike prior approaches such as TTRL, which rely on large-scale majority voting, RLSC formalizes the underlying principle mathematically.

Case 1: Comparison of Model Outputs

Problem. Consider the function

$$f(x) = \begin{cases} ax^2 & \text{if } x \geq a, \\ ax + 2a & \text{if } x < a, \end{cases}$$

where a is some number. What is the smallest value of a such that the graph of $y = f(x)$ intersects the horizontal line $y = 2017$ at least twice? Mark your solution with \boxed.

Qwen2.5-Math-7B with RLSC

Solution. To find the largest value of a such that the graph of $y = f(x)$ intersects every horizontal line at least once, we need to analyze the function $f(x)$ and ensure that it covers all possible y-values. The function $f(x)$ is defined as:

$$f(x) = \begin{cases} ax^2 & \text{if } x \geq a, \\ ax + 2a & \text{if } x < a. \end{cases}$$

1. Analyze the first part of the function $f(x) = ax^2$ for $x \geq a$: - Since $x \geq a$, the minimum value of ax^2 is $a \cdot a^2 = a^3$. As x increases, ax^2 increases without bound. So, the range of ax^2 for $x \geq a$ is $[a^3, \infty)$. 2. Analyze the second part of the function $f(x) = ax + 2a$ for $x < a$: - The value of $ax + 2a$ at $x = a$ is $a \cdot a + 2a = a^2 + 2a$. As x decreases without bound, $ax + 2a$ decreases without bound. So, the range of $ax + 2a$ for $x < a$ is $(-\infty, a^2 + 2a)$. For the function $f(x)$ to intersect every horizontal line at least once, the range of $f(x)$ must cover all real numbers. This means the range of ax^2 for $x \geq a$ must overlap with the range of $ax + 2a$ for $x < a$. Specifically, the minimum value of ax^2 (which is a^3) must be less than or equal to the maximum value of $ax + 2a$ (which is $a^2 + 2a$). So, we need to solve the inequality:

$$a^3 \leq a^2 + 2a.$$

Rearranging the inequality, we get:

$$a^3 - a^2 - 2a \leq 0.$$

Factoring the left-hand side, we get:

$$a(a^2 - a - 2) \leq 0.$$

Further factoring the quadratic expression, we get:

$$a(a - 2)(a + 1) \leq 0.$$

We need to find the values of a that satisfy this inequality. The critical points are $a = 0$, $a = 2$, and $a = -1$. Testing the intervals determined by these points, we find that the inequality $a(a - 2)(a + 1) \leq 0$ is satisfied for $a \in (-\infty, -1] \cup [0, 2]$. Since a is a number and we are looking for the largest value of a, the largest possible value of a is 2. We can verify that when $a = 2$, the function $f(x)$ intersects every horizontal line at least once. So, the largest value of a is $\boxed{2}$.

Final Answer (Correct): $\boxed{2}$

Qwen2.5-Math-7B

Solution. To solve the problem, we need to analyze the function $f(x)$ and determine the values of a such that the graph of $y = f(x)$ intersects every horizontal line at least once and intersects the horizontal line $y = 2017$ at least twice. The function $f(x)$ is defined as:

$$f(x) = \begin{cases} ax^2 & \text{if } x \geq a, \\ ax + 2a & \text{if } x < a. \end{cases}$$

Smallest value of a such that the graph of $y = f(x)$ intersects the horizontal line $y = 2017$ at least twice For the function $f(x)$ to intersect the horizontal line $y = 2017$ at least twice, we need to find the values of a such that there are at least two solutions to the equation $f(x) = 2017$.

1. For $x \geq a$, $f(x) = ax^2 = 2017$. So, $x = \pm\sqrt{\frac{2017}{a}}$. Since $x \geq a$, we only consider the positive root $x = \sqrt{\frac{2017}{a}}$. This root exists if $\sqrt{\frac{2017}{a}} \geq a$, which simplifies to $\frac{2017}{a} \geq a^2$ or $2017 \geq a^3$ or $a \leq \sqrt[3]{2017}$.

2. For $x < a$, $f(x) = ax + 2a = 2017$. So, $x = \frac{2017 - 2a}{a}$. This root exists if $\frac{2017 - 2a}{a} < a$, which simplifies to $2017 - 2a < a^2$ or $a^2 + 2a - 2017 > 0$. Solving this quadratic inequality, we get $a > \sqrt{2017 + 1} - 1$ or $a < -\sqrt{2017 + 1} - 1$. Since a must be positive, we take $a > \sqrt{2017 + 1} - 1$.

The smallest value of a that satisfies both conditions is the maximum of $\sqrt[3]{2017}$ and $\sqrt{2017 + 1} - 1$. Since $\sqrt[3]{2017} \approx 12.63$ and $\sqrt{2017 + 1} - 1 \approx 44.69 - 1 = 43.69$, the smallest value of a is $\sqrt[3]{2017}$. Let's implement this in Python to confirm our results.

```
import sympy as sp
# Define the variable and the equation
a = sp.symbols('a')
eq1 = a**3 - 2017
eq2 = a**2 + 2*a - 2017
# Solve the equations
sol1 = sp.solve(eq1, a)
sol2 = sp.solve(eq2, a)
# Find the largest value of a such that the graph of y = f(x) intersects every horizontal line at least once
largest_a = max(sol1)
# Find the smallest value of a such that the graph of y = f(x) intersects the horizontal line y = 2017 at least twice
smallest_a = max(sol1[0], sol2[1])
print((largest_a, smallest_a))

output
(2017**(1/3), -1 + sqrt(2018))
```

The largest value of a such that the graph of $y = f(x)$ intersects every horizontal line at least once is $\sqrt[3]{2017}$. The smallest value of a such that the graph of $y = f(x)$ intersects the horizontal line $y = 2017$ at least twice is $\sqrt[3]{2017}$.

Final Answer (Incorrect): $\boxed{\sqrt[3]{2017}}$

Our key contribution is a derivation: we show that majority voting implicitly optimizes for agreement within the model's output distribution - and we transform that into a differentiable, self-supervised objective. This "mode sharpening" functional enables efficient reinforcement learning directly from the model's own confidence.

Case 2: Comparison of Model Outputs

Problem. What is the distance, in units, between the points $(2, -6)$ and $(-4, 3)$? Express your answer in simplest radical form.

Model Output. (Correct Answer), Qwen2.5-Math-7B with RLSC

Solution. To find the distance between two points (x_1, y_1) and (x_2, y_2) in the Cartesian plane, we use the distance formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Given the points $(2, -6)$ and $(-4, 3)$, substitute these into the formula:

$$d = \sqrt{((-4) - 2)^2 + (3 - (-6))^2} = \sqrt{(-6)^2 + 9^2} = \sqrt{36 + 81} = \sqrt{117} = \sqrt{9 \cdot 13} = 3\sqrt{13}$$

Thus, the distance is $3\sqrt{13}$.

Verification with Python:

```python
import math
# Coordinates of the points
x1, y1 = 2, -6
x2, y2 = -4, 3

# Compute the distance
distance = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
print(distance)
```

Output:

```
10.816653826391969
```

This approximates $3\sqrt{13} \approx 10.8167$, confirming our answer.

Final Answer (correct): $\boxed{3\sqrt{13}}$

Model Output. (Incorrect Answer), Qwen2.5-Math-7B

Problem: What is the distance, in units, between the points $(2, -6)$ and $(-4, 3)$? Express your answer in simplest radical form. Mark your solution with \boxed.

Answer: $\boxed{10}$ *(Incorrect)*

`\end{document}`

This code will produce a document with the problem and answer boxes. The problem is typeset in a box, and the answer is typeset in a box. The answer is also marked with a box to indicate that it is the correct answer. The final answer is \boxed{10}.

Final Answer (Incorrect): $\boxed{10}$

Applied to Qwen2.5-Math-7B, RLSC delivers substantial accuracy gains using just 16 completions per question and 10 steps of training - without external supervision.

This work demonstrates that high-quality post-training can emerge not from external labels, but from a model's internal signal - when that signal is derived with care. We believe RLSC offers both a practical tool and a conceptual bridge between ensemble-based pseudo-labeling and principled self-supervision.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[3] Jiaqi Chen, Bang Zhang, Ruotian Ma, Peisong Wang, Xiaodan Liang, Zhaopeng Tu, Xiaolong Li, and Kwan-Yee K Wong. Spc: Evolving self-play critic via adversarial games for llm reasoning. *arXiv preprint arXiv:2504.19162*, 2025.

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

[5] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

[6] Etash Guha, Negin Raoof, Jean Mercat, Ryan Marten, Eric Frankel, Sedrick Keh, Sachin Grover, George Smyrnis, Trung Vu, Jon Saad-Falcon, et al. Evalchemy: Automatic evals for llms, 2024.

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

[8] Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.

[9] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

[10] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[11] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[12] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

[13] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

[14] Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13:9, 2024.

[15] Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13:9, 2024.

[16] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[17] Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. Reft: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*, 3, 2024.

[18] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[19] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

[20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[21] Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025.

[22] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

[23] Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025.

[24] Yuxin Zuo, Kaiyan Zhang, Shang Qu, Li Sheng, Xuekai Zhu, Biqing Qi, Youbang Sun, Ganqu Cui, Ning Ding, and Bowen Zhou. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

A   Case Study

We present the performance of our model on the more challenging AIME dataset, which contains complex mathematical reasoning problems.

Model Output (Correct Answer), Qwen2.5-Math on AIME-style Game Theory Problem

Problem. Alice and Bob play the following game. A stack of n tokens lies before them. The players take turns with Alice going first. On each turn, the player removes either 1 token or 4 tokens from the stack. Whoever removes the last token wins.

Find the number of positive integers $n \leq 2024$ for which there exists a strategy for Bob that guarantees that Bob will win the game regardless of Alice's play.

Solution. We define a position n as:
- **Winning**, if the player whose turn it is can force a win.
- **Losing**, otherwise.
We define a function $f(n)$:

$$f(n) = \begin{cases} \text{False,} & \text{if } n = 0 \\ \text{True,} & \text{if } f(n-1) = \text{False or } f(n-4) = \text{False} \\ \text{False,} & \text{otherwise} \end{cases}$$

We compute this iteratively for all $1 \leq n \leq 2024$. The number of positions n where $f(n) = $ False (i.e., losing positions for the first player) correspond to Bob's guaranteed wins.

Python Verification:

```python
def count_losing_positions(max_n):
    dp = [False] * (max_n + 1)
    for n in range(1, max_n + 1):
        if n >= 1 and not dp[n - 1]:
            dp[n] = True
        elif n >= 4 and not dp[n - 4]:
            dp[n] = True
        else:
            dp[n] = False
    return sum(not dp[n] for n in range(1, max_n + 1))

result = count_losing_positions(2024)
print(result)
```

Output:

```
809
```

Final Answer: 809

Model Output (Correct Answer), Qwen2.5-Math-7B with RLSC

Problem. Let x, y, z be positive real numbers satisfying:

$$\log_2\left(\frac{x}{yz}\right) = \frac{1}{2}, \quad \log_2\left(\frac{y}{xz}\right) = \frac{1}{3}, \quad \log_2\left(\frac{z}{xy}\right) = \frac{1}{4}.$$

Then the value of $\left|\log_2(x^4 y^3 z^2)\right| = \frac{m}{n}$, where m, n are relatively prime positive integers. Find $m + n$.

Solution. Using properties of logarithms, we rewrite:

$$\log_2(x) - \log_2(y) - \log_2(z) = \frac{1}{2}, \quad \log_2(y) - \log_2(x) - \log_2(z) = \frac{1}{3}, \quad \log_2(z) - \log_2(x) - \log_2(y) = \frac{1}{4}.$$

Let $a = \log_2(x), \ b = \log_2(y), \ c = \log_2(z)$. Then:

$$a - b - c = \frac{1}{2} \tag{1}$$

$$b - a - c = \frac{1}{3} \tag{2}$$

$$c - a - b = \frac{1}{4} \tag{3}$$

Adding (1), (2), and (3), the LHS becomes:

$$a - b - c + b - a - c + c - a - b = -a - b - c$$

and the RHS:

$$\frac{1}{2} + \frac{1}{3} + \frac{1}{4} = \frac{13}{12} \Rightarrow a + b + c = -\frac{13}{12}$$

Now solve for each: - From (1): $2a = \frac{1}{2} - \frac{13}{12} = -\frac{7}{12} \Rightarrow a = -\frac{7}{24}$
- From (2): $2b = \frac{1}{3} - \frac{13}{12} = -\frac{3}{4} \Rightarrow b = -\frac{3}{8}$
- From (3): $2c = \frac{1}{4} - \frac{13}{12} = -\frac{5}{6} \Rightarrow c = -\frac{5}{12}$
Now compute:

$$\log_2(x^4 y^3 z^2) = 4a + 3b + 2c = 4\left(-\frac{7}{24}\right) + 3\left(-\frac{3}{8}\right) + 2\left(-\frac{5}{12}\right) = -\frac{28}{24} - \frac{9}{8} - \frac{10}{12}$$

Convert to common denominator 48:

$$-\frac{56}{48} - \frac{54}{48} - \frac{40}{48} = -\frac{150}{48} = -\frac{25}{8} \Rightarrow \left|\log_2(x^4 y^3 z^2)\right| = \frac{25}{8} \Rightarrow \boxed{m + n = 25 + 8 = \boxed{33}}$$

Final Answer: $\boxed{33}$