RoiRL: Efficient, Self-Supervised Reasoning with Offline Iterative Reinforcement Learning

Aleksei Arzhantsev
Criteo AI Lab, Ecole Polytechnique
Paris, France
a.arzhantsev@criteo.com

Otmane Sakhi
Criteo AI Lab
Paris, France
o.sakhi@criteo.com

Flavian Vasile
Criteo AI Lab
Paris, France
f.vasile@criteo.com

Abstract

Reinforcement learning (RL) is central to improving reasoning in large language models (LLMs) but typically requires ground-truth rewards. Test-Time Reinforcement Learning (TTRL) removes this need by using majority-vote rewards, but relies on heavy online RL and incurs substantial computational cost. We propose RoiRL: **R**easoning with **o**ffline **i**terative **R**einforcement **L**earning, a family of lightweight offline learning alternatives that can target the same regularized optimal policies. Unlike TTRL, RoiRL eliminates the need to maintain a reference model and instead optimizes *weighted log-likelihood* objectives, enabling stable training with significantly lower memory and compute requirements. Experimental results show that RoiRL trains to $2.5\times$ faster and consistently outperforms TTRL on reasoning benchmarks, establishing a scalable path to self-improving LLMs without labels.

1 Introduction

Reasoning [18] is at the core of large language model (LLM) capabilities, improving performance on mathematical problem solving [7], commonsense inference [18, 15], and agentic applications [19]. Recent advances have demonstrated that reasoning ability can be enhanced not only by scaling model size and data but also by explicitly training models to generate and evaluate chains of thought [4]. Reinforcement learning (RL) [14] has played a particularly important role in this direction: RL aligns models generations with outcome quality, improving their ability to solve complex tasks.

However, RL-based approaches require access to *ground-truth rewards*, mostly in the form of correctness labels (e.g., for math problems). This reliance can limit their scalability, since ground-truth supervision is costly and often unavailable. To circumvent this bottleneck, recent work has leveraged *majority-vote* as a weak supervision signal: instead of relying on external labels, the model itself generates multiple candidate solutions, and majority voting [17] is used to estimate correctness. This idea has proven highly effective at inference time, where increasing the number of sampled solutions substantially improves accuracy through test-time scaling [17].

Building on this observation, Test-Time Reinforcement Learning (TTRL) [20] has been proposed as a mechanism for turning majority-vote feedback, originally used in test time scaling, into a training signal. By repeatedly generating candidate chains of thought (CoT) [18] with their respective answers, evaluating them with majority voting, and updating the model parameters online, TTRL enables reasoning improvement without ground-truth labels. Empirically, this approach has demonstrated strong gains on reasoning benchmarks, validating the potential of self-generated feedback.

Despite its promise, TTRL faces two critical limitations. First, it is *computationally expensive*. The method requires maintaining a reference model and computing its logits at every training step. Combined with repeated chain-of-thought (CoT) sampling during training, this quickly saturates memory and makes the approach increasingly difficult to scale to larger models or longer runs. Second, its *online nature introduces instability*. Performance is highly sensitive to hyperparameter

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Efficient Reasoning.

choices, as also reported in [20]. These issues make TTRL challenging to deploy in practice and limit its applicability as a general recipe for scalable reasoning improvements.

Inspired by offline RL approaches [9, 10], we introduce RoiRL (**R**easoning with **o**ffline **i**terative **R**einforcement **L**earning), a lightweight alternative that preserves the benefits of self-generated rewards while overcoming the limitations of TTRL. Our method optimizes simple *weighted log-likelihood objectives* in an iterative offline loop, eliminating the need for online RL or maintaining a reference model. This design improves stability, reduces memory overhead, and scales efficiently with model size. On small-scale models and modest compute budgets, RoiRL trains faster and more efficiently, while consistently surpassing TTRL across reasoning benchmarks.

Contributions. We introduce a family of *offline weighted log-likelihood* objectives, that can target and solve the same underlying problem of TTRL without requiring online RL, nor maintaining a reference model. We demonstrate empirically that RoiRL, which builds on these simple objectives, achieves superior performance and scalability, offering a practical path toward self-improving LLMs without reliance on true labels.

2 Preliminaries

We assume access to a strong base LLM, typically pre-trained or instruction-tuned , which we denote by the policy $\pi_0 = \pi_{\theta_0}$, . Given a prompt x, the model generates a chain-of-thought c leading to the answer y, sampled as $\{c, y\} \sim \pi_0(\cdot \mid x)$. Alongside this model, we consider a collection of reasoning tasks for which ground-truth answers are unavailable. These tasks are represented as a prompt dataset $\mathcal{P}_n = \{x_i\}_{i\in[n]}$, where each x_i corresponds to a question or input prompt and n is the dataset size. Crucially, the dataset contains no labels, reflecting the realistic setting where large collections of problems are readily available, while solutions are not.

Test Time Reinforcement Learning. Given $\mathcal{P}_n$, TTRL [20] provides a learning algorithm to improve the reasoning abilities of π_{θ_0}. For each reasoning task x_i, TTRL attributes an approximate ground truth label $\tilde{y}_i$ to x_i on the fly. For each optimization step t, π_{θ_t} generates $k > 2$ candidates $\{c_i^\ell, y_i^\ell\}_{\ell\in[k]}$, defined with their respective CoT and answers. The answers $\{y_i^\ell\}_{\ell\in[k]}$ are used to define the approximate ground truth label $\tilde{y}_i$ as the label with the majority vote i.e. $\tilde{y}_i^k(\theta_t) = \texttt{maj}_{\ell\in[k]}(y_i^\ell)$. Rewards are then naturally attributed to the generated candidates and constructed as $\tilde{r}_k(y, x_i, \theta_t) = \mathbb{1}\left[y = \tilde{y}_i^k(\theta_t)\right]$, augmenting data with rewards and enabling the use of RL algorithms to train and optimize the parametrised LLM policy π_θ. For instance, TTRL optimizes the KL regularized expected reward using GRPO [13]:

$$\max_\theta \left\{ \sum_{i=1}^{n} \mathbb{E}_{(c,y)\sim\pi_\theta(\cdot|x_i)} \left[\tilde{r}_k(y, x_i, \theta)\right] - \beta\, \mathrm{KL}(\pi_\theta, \pi_0|x_i) \right\}. \tag{1}$$

with $\mathrm{KL}(\pi_\theta, \pi_0|x)$ the KL divergence between $\pi_\theta(\cdot|x)$ and $\pi_0(\cdot|x)$, and $\beta > 0$ a regularization parameter, solving for an LLM that optimizes for the consistency of the generated answer while staying close to the original model.

Non-stationary rewards. The particularity of the optimization problem in Equation (1) is that the rewards are non-stationary and depend on the current policy π_θ we are optimizing. In step t, the reward of an answer y is positive when it matches the majority vote at k: $\tilde{y}^k(\theta_t)$. This means that the reward shifts when the majority vote changes in the optimization. This subtlety makes this approach differ from only distilling the majority voter back into the model.

3 Self Supervised Reasoning with Iterative Offline Reinforcement Learning

TTRL optimizes the KL-regularized reward maximization objective of Equation (1) using GRPO [13] in an online setting. While effective, this procedure is computationally demanding: it requires maintaining a reference model in memory, repeatedly sampling potentially long answers during training, and computing their logits under both current and reference policies. This saturates GPU memory and limit scalability. In addition, the reliance on online RL makes the method highly sensitive to hyperparameter choices, leading to instability and unreliable performance in practice [20].

This raises a natural question: *can we achieve the same objective with a procedure as simple and stable as supervised fine-tuning?* Building on offline RL methods [10], we answer affirmatively with RoiRL (**R**easoning with **o**ffline **i**terative **R**einforcement **L**earning), an iterative, offline approach that address these limitations. In each iteration $m \geq 1$, RoiRL alternates between two steps:

(1) Generation: From the current policy π_{m-1}, we sample k candidate solutions $\{c_i^\ell, y_i^\ell\}_{\ell \in [k]}$ for each prompt $x_i \in \mathcal{P}$. These candidates are scored with a majority-vote reward, $\tilde{r}_k(y_i^\ell, x_i, \theta_{m-1}) = \mathbb{1}\left[y_i^\ell = \tilde{y}_i^k(\theta_{m-1})\right]$ with $\tilde{y}_i^k(\theta_{m-1}) = \texttt{maj}_{\ell \in [k]}(y_i^\ell)$. This produces an offline dataset:

$$\mathcal{D}_{m-1} = \left\{ x_i, \left\{ c_i^\ell, y_i^\ell, \tilde{r}_k(y_i^\ell, x_i, \theta_{m-1}) \right\}_{\ell \in [k]} \right\}_{i \in [n]}$$

(2) Offline Update: Using $\mathcal{D}_{m-1}$, we approximate and solve the weighted log-likelihood objective

$$\theta_m = \arg\max_\theta \left\{ \sum_{i=1}^n \mathbb{E}_{(c,y) \sim \pi_{m-1}(\cdot|x_i)} \left[g_m(\tilde{r}_k(y, x_i, \theta_{m-1})) \log \pi_\theta(c, y|x_i) \right] \right\}, \tag{2}$$

with $g_m : \mathbb{R} \to \mathbb{R}$ an increasing reward transform. We then update the policy as $\pi_m \leftarrow \pi_{\theta_m}$.

The resulting optimization routine is summarized in Algorithm 1. Equation (2) can be interpreted as a *weighted supervised fine-tuning loss* on generated answers, in contrast to the unstable online updates of TTRL. RoiRL is more stable and alleviates the need for π_0, making it significantly more scalable.

Algorithm 1: **R**easoning with **o**ffline **i**terative **R**einforcement **L**earning (RoiRL)

Input: Policy π_θ, reward transforms $(g_m)_{m \in \mathbb{N}}$, prompt dataset $\mathcal{P} = \{x_i\}_{i \in [n]}$, number of candidates k.
Initialize: $\theta = \theta_0$, $\pi_0 = \pi_{\theta_0}$
for $m = 1, 2, \ldots$ **do**
  Construct offline dataset $\mathcal{D}_{m-1}$ with π_{m-1}
  Update parameters θ_m by solving Equation (2)
  Set $\pi_m \leftarrow \pi_{\theta_m}$

Connection to KL-Regularized Objectives. Unlike TTRL, which explicitly enforces KL regularization, RoiRL leverages a sequence $(g_m)_{m \in \mathbb{N}}$ of reward transform $g_m : \mathbb{R} \to \mathbb{R}, \forall m$ that implicitly control the reward influence. At iteration m, the analytical solution of Algorithm 1 takes the form:

$$\forall (c, y), x, \quad \pi_m(c, y|x) \propto \left(\prod_{j=1}^m g_j(\tilde{r}_k(y, x, \theta_{j-1}) \right) \pi_0(c, y|x). \tag{3}$$

For example, choosing $g_j(r) = g_\beta(r) = \exp(r/\beta), \forall j$ yields

$$\forall (c, y), x, \quad \pi_m(c, y|x) \propto \exp\left(\frac{1}{\beta} \sum_{j=1}^m \tilde{r}_k(y, x, \theta_{j-1}) \right) \pi_0(c, y|x),$$

which closely mirrors the closed-form solution of KL-regularized RL objectives widely used in preference alignment [8, 11] and reasoning [13] for example. A proof of the analytical solution is provided in Appendix A.2. In particular, the following proposition connects RoiRL and TTRL.

Proposition 3.1. *For any $\beta > 0$, there exists a choice of the reward transforms $(g_m)_{m \in \mathbb{N}}$ such that Equation (1) and Algorithm 1 admit the same solution.*

This result is proven in details in Appendix A.3, and shows that RoiRL can target the same theoretical objective as TTRL, while being more stable, scalable, and practically implementable. Moreover, by flexibly choosing g, and thus controlling the reward influence on the updates, RoiRL extends beyond TTRL to encompass a broader family of objectives, including known regularized objectives [16].

4 Experiments

Experimental Setup. We design our experimental setting to compare the training and generalization performance of RoiRL against TTRL. We evaluate the learning approaches on three mathematical

Table 1: Results are reported for training on unlabeled problems from MATH500 *Train* and evaluating on all datasets. RoiRL outperforms TTRL in most cases. For each decoding strategy, the second-best result is underlined, the best result is in bold, and marked with $\star$ when it beats π_0 with $\texttt{maj}_{128}$.

Decode	Model	Qwen2.5-Math-1.5B				Phi4-mini-reasoning-4B				Llama-3.2-3B-Instruct			
		MATH500		AMC	AIME	MATH500		AMC	AIME	MATH500		AMC	AIME
		Train	*Test*			*Train*	*Test*			*Train*	*Test*		
$\texttt{maj}_1$	Base (π_0)	0.244	0.239	0.170	0.036	0.210	0.160	0.071	0.000	0.256	0.295	0.141	0.050
	TTRL	0.307	0.298	0.214	0.026	0.272	0.225	0.090	0.000	0.361	**0.394**	0.159	0.043
	RoiRL g_I	**0.686**	0.587	0.337	**0.083**	**0.660***	**0.511**	**0.246**	**0.016***	0.395	0.376	**0.198**	**0.060**
	RoiRL g_β	0.670	**0.604**	**0.340**	0.070	0.533	0.344	0.125	0.000	**0.487**	0.256	0.090	0.020
$\texttt{maj}_{10}$	Base (π_0)	0.572	0.520	0.445	0.100	0.420	0.350	0.157	0.000	0.495	0.480	0.253	0.033
	TTRL	0.625	0.560	0.469	0.066	0.483	0.460	0.193	0.000	**0.510**	0.490	**0.313**	0.167
	RoiRL g_I	**0.712**	**0.690***	**0.518***	0.133	**0.720***	**0.680***	**0.421***	**0.067***	0.508	0.520	**0.313**	**0.200***
	RoiRL g_β	0.685	0.650	0.469	**0.200**	0.543	0.560	0.277	0.033	0.508	**0.530***	0.229	0.100
$\texttt{maj}_{128}$	Base (π_0)	0.717	0.680	0.506	0.233	0.563	0.560	0.289	0.000	0.543	0.520	0.361	0.167

reasoning benchmarks: MATH500 [6], AMC [6], and AIME2024 [2]. The MATH500 dataset is further divided into 400 training (MATH500 *Train*) and 100 test (MATH500 *Test*) problems. All training algorithms are run on the unlabeled problems from the *Train* split of MATH500, which defines the problem dataset $\mathcal{P}_n$. Using ground-truth labels, we then measure accuracy on the MATH500 *Train* split, the MATH500 *Test* split, AMC, and AIME2024 to assess the generalization capabilities of the learning methods. For base models, we use three reasoning-oriented LLMs of diverse sizes: Qwen2.5-Math-1.5B [3], Phi-4-mini-reasoning-4B [1], and Llama-3.2-3B-Instruct [5]. These models already demonstrate good reasoning capabilities and differ sufficiently in design to enable robust validation of our learning approaches across architectures and training paradigms [12]

Training. Both TTRL and RoiRL use the majority vote signal $\tilde{r}_k$ as a training signal to improve the base model. For each problem x_i, we generate $k = 10$ candidates, with which the majority vote signal $\tilde{r}_k$ is defined. TTRL is implemented using GRPO [13] with the KL regularizer $\beta = 0.1$, that we compare to two flavors of RoiRL: the first one uses an exponential function $g_\beta : x \to \exp(x/\beta)$ with $\beta = 0.1$ to mimic TTRL's behavior, and the second uses the identity function $g_I : x \to x$, reducing the offline update to simple *supervised finetuning* on good generated answers. Implementation details are developed in Appendix B.1.1.

Decoding strategies and baselines. We evaluate our learning methods based on their ability to improve the base model π_0. Since training uses the majority-vote signal with $k = 10$, improving a single sampled answer ($k = 1$) is relatively straightforward, as the model effectively distills the majority vote. To assess performance beyond this distilled signal, we compare models using $k = 1$ ($\texttt{maj}_1$) and $k = 10$ ($\texttt{maj}_{10}$). We also report the base model with $k = 128$ ($\texttt{maj}_{128}$) to evaluate whether the learned model can surpass this strong but costly baseline.

Results. Table 1 shows that RoiRL outperforms the online RL-based TTRL approach in the majority of cases while being up to $2.5\times$ faster (see Appendix B.1.3). RoiRL with g_β improves over TTRL despite both targeting a KL-regularized objective, while the g_I variant achieves the best overall results, suggesting that alternative reward transforms beyond KL may be more effective. RoiRL is a self-improving method and not merely a distillation of majority voting: $\texttt{maj}_1$ decoding with obtained models can surpass the base model's $\texttt{maj}_{10}$ decoding, and $\texttt{maj}_{10}$ decoding with obtained models can outperform the base model with costly $\texttt{maj}_{128}$ decoding, even on unseen problems. Extended results with training curves and developed discussions are provided in Appendix B.

5 Conclusion

We proposed RoiRL, a simple and scalable approach for self-supervised reasoning in LLMs that converts majority-vote signals into efficient offline updates. Unlike online RL approaches such as TTRL, RoiRL requires no reference model, achieves greater stability and speed, and consistently improves accuracy across benchmarks, demonstrating that lightweight offline reinforcement learning is sufficient for self-improvement in reasoning tasks. Future work will extend this approach by evaluating RoiRL on larger LLMs to further validate scalability, exploring alternative ground-truth estimation strategies beyond majority vote, and studying the impact of different reward transforms, which we found to substantially influence performance.

References

[1] Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. Phi-4 technical report, 2024.

[2] American Invitational Mathematics Examination. Aime 2024 dataset. https://artofproblemsolving.com/contests/aime, 2024. Accessed: 2025-09-08.

[3] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report, 2023.

[4] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[5] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, and Llama Team. The llama 3 herd of models, 2024.

[6] 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. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[7] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.

[8] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.

[9] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2019.

[10] Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24(1):1607–1612, Jul. 2010.

[11] 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. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[12] Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Wei Koh, and Luke Zettlemoyer. Spurious rewards: Rethinking training signals in rlvr, 2025.

[13] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.

[14] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[15] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[16] Chaoqi Wang, Yibo Jiang, Chenghao Yang, Han Liu, and Yuxin Chen. Beyond reverse KL: Generalizing direct preference optimization with diverse divergence constraints. In *The Twelfth International Conference on Learning Representations*, 2024.

[17] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[18] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.

[19] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[20] Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. TTRL: Test-Time Reinforcement Learning, 2025.

A Technical Discussions and Proofs

A.1 Useful Lemma

Our main theoretical result is based on the use of the following Lemma.

Lemma A.1. *Let g a positive function. The solution of the weighted likelihood objective of the form:*

$$\arg\max_{\pi} \left\{ \mathbb{E}_{(c,y)\sim\pi_0(\cdot|x)} \left[g(x,c,y) \log \pi(c,y|x) \right] \right\}$$

can be computed analytically and is:

$$\forall x, (y,c)\,, \pi^\star(c,y|x) \propto g(x,c,y)\pi_0(c,y|x)\,.$$

Proof. We prove this Lemma below for completeness. The objective is a constrained optimization problem, that can be solved by Lagrange multipliers. For a fixed input x, we want to maximize:

$$J(\pi) = \sum_{c,y} \pi_0(c,y|x) g(x,c,y) \log \pi(c,y|x)$$

subject to the normalization constraint:

$$\sum_{c,y} \pi(c,y|x) = 1$$

Setting up the Lagrangian:

$$\mathcal{L}(\pi,\lambda) = \sum_{c,y} \pi_0(c,y|x) g(x,c,y) \log \pi(c,y|x) - \lambda \left(\sum_{c,y} \pi(c,y|x) - 1 \right)$$

Taking the partial derivative with respect to $\pi(c,y|x)$ and setting it to zero:

$$\frac{\partial \mathcal{L}}{\partial \pi(c,y|x)} = \frac{\pi_0(c,y|x) g(x,c,y)}{\pi(c,y|x)} - \lambda = 0$$

Solving for $\pi(c,y|x)$:

$$\pi(c,y|x) = \frac{\pi_0(c,y|x) g(x,c,y)}{\lambda}$$

Using the normalization constraint $\sum_{c,y} \pi(c,y|x) = 1$:

$$\sum_{c,y} \frac{\pi_0(c,y|x) g(x,c,y)}{\lambda} = 1$$

Therefore:

$$\lambda = \sum_{c,y} \pi_0(c,y|x) g(x,c,y)$$

Substituting back:

$$\pi^\star(c,y|x) = \frac{\pi_0(c,y|x) g(x,c,y)}{\sum_{c',y'} \pi_0(c',y'|x) g(x,c',y')}$$

This shows that $\pi^\star(c,y|x) \propto g(x,c,y)\pi_0(c,y|x)$, completing the proof. □

A.2 Analytical Solution of `RoiRL`

We remind the reader of the `RoiRL` algorithm below:

`RoiRL` leverages a set $(g_m)_{m\in\mathbb{N}}$ of increasing reward transform $g_m : \mathbb{R} \to \mathbb{R}\,, \forall m$ that implicitly control the reward influence. At iteration m, the analytical solution of Algorithm 1 takes the form:

$$\forall (c,y), x_i\,, \quad \pi_m(c,y|x_i) \propto \left(\prod_{j=1}^{m} g_j(\tilde{r}_k(y,x_i,\theta_{j-1})) \right) \pi_0(c,y|x_i)\,.$$

Algorithm: Reasoning with **o**ffline **i**terative **R**einforcement **L**earning (RoiRL)

Input: Policy π_θ, transforms $(g_m)_{m\in\mathbb{N}}$, prompt dataset $\mathcal{P} = \{x_i\}_{i\in[n]}$, number of candidates k.
Initialize: $\theta = \theta_0$, $\pi_0 = \pi_{\theta_0}$
for $m = 1, 2, \dots$ **do**
  Construct offline dataset $\mathcal{D}_{m-1}$ with π_{m-1}
  Using $\mathcal{D}_{m-1}$, we approximate and solve the weighted log-likelihood objective

$$\theta_m = \arg\max_\theta \left\{ \sum_{i=1}^{n} \mathbb{E}_{(c,y)\sim\pi_{m-1}(\cdot|x_i)} \left[g_m(\tilde{r}_k(y, x_i, \theta_{m-1})) \log \pi_\theta(c, y|x_i) \right] \right\},$$

  Set $\pi_m \leftarrow \pi_{\theta_m}$

Proof. We prove this result by induction on the iteration number m.

Base Case ($m = 1$): At iteration $m = 1$, we solve:

$$\theta_1 = \arg\max_\theta \left\{ \sum_{i=1}^{n} \mathbb{E}_{(c,y)\sim\pi_0(\cdot|x_i)} \left[g_1(\tilde{r}_k(y, x_i, \theta_0)) \log \pi_\theta(c, y|x_i) \right] \right\}$$

As the optimization problem is decomposable, we can look at each x_i independently. Let x_i be a prompt from $\mathcal{P}$. By the previous Lemma A.1, the analytical solution is:

$$\pi_1(c, y|x_i) \propto g_1(\tilde{r}_k(y, x_i, \theta_0))\pi_0(c, y|x_i)$$

This matches our claimed form with $m = 1$:

$$\pi_1(c, y|x_i) \propto \left(\prod_{j=1}^{1} g_j(\tilde{r}_k(y, x_i, \theta_{j-1})) \right) \pi_0(c, y|x_i) = g_1(\tilde{r}_k(y, x_i, \theta_0))\pi_0(c, y|x_i)$$

Inductive Step: Assume the claim holds for some iteration $m - 1$, i.e.:

$$\pi_{m-1}(c, y|x_i) \propto \left(\prod_{j=1}^{m-1} g_j(\tilde{r}_k(y, x_i, \theta_{j-1})) \right) \pi_0(c, y|x_i)$$

At iteration m, we solve:

$$\theta_m = \arg\max_\theta \left\{ \sum_{i=1}^{n} \mathbb{E}_{(c,y)\sim\pi_{m-1}(\cdot|x_i)} \left[g_m(\tilde{r}_k(y, x_i, \theta_{m-1})) \log \pi_\theta(c, y|x_i) \right] \right\}$$

By the previous lemma, the analytical solution is:

$$\pi_m(c, y|x_i) \propto g_m(\tilde{r}_k(y, x_i, \theta_{m-1}))\pi_{m-1}(c, y|x_i)$$

Substituting the inductive hypothesis:

$$\begin{aligned}\pi_m(c, y|x_i) &\propto g_m(\tilde{r}_k(y, x_i, \theta_{m-1})) \left(\prod_{j=1}^{m-1} g_j(\tilde{r}_k(y, x_i, \theta_{j-1})) \right) \pi_0(c, y|x_i) \\ &\propto \left(\prod_{j=1}^{m} g_j(\tilde{r}_k(y, x_i, \theta_{j-1})) \right) \pi_0(c, y|x_i)\end{aligned}$$

This completes the induction and proves the claimed form. □

A.3 RoiRL solves the TTRL objective and beyond

The proposed RoiRL objective provides an offline, iterative alternative to the recently proposed TTRL algorithm. We recall that TTRL optimizes the KL regularized expected reward:

$$\max_{\pi} \left\{ \sum_{i=1}^{n} \mathbb{E}_{(c,y)\sim\pi(\cdot|x_i)} \left[\tilde{r}_k(y, x_i, \pi) \right] - \beta \, \mathrm{KL}(\pi, \pi_0 | x_i) \right\} .$$

This optimization problem differs from the classical regularized objective as the individual rewards depend themselves on the current policy we are optimizing. We can connect RoiRL and TTRL with the following proposition:

Proposition 3.1. *For any $\beta > 0$, there exists a choice of the reward transforms $(g_m)_{m\in\mathbb{N}}$ such that Equation (1) and Algorithm 1 admit the same solution.*

Proof. Let us focus on the TTRL objective. As the problem is decomposable over prompts, the optimal policy $\pi^\star$ can be recovered for each x_i. We then optimize:

$$J_i(\pi) = \sum_{c,y} \pi(c, y|x_i) \tilde{r}_k(y, x_i, \pi) - \beta \sum_{c,y} \pi(c, y|x_i) \log \frac{\pi(c, y|x_i)}{\pi_0(c, y|x_i)}$$

Using the method of Lagrange multipliers with the constraint $\sum_{c,y} \pi(c, y|x_i) = 1$:

$$\mathcal{L} = J_i(\pi) - \lambda_i \left(\sum_{c,y} \pi(c, y|x_i) - 1 \right)$$

Taking the functional derivative with respect to $\pi(c, y|x_i)$:

$$\frac{\partial \mathcal{L}}{\partial \pi(c, y|x_i)} = \tilde{r}_k(y, x_i, \pi) + \sum_{c',y'} \pi(c', y'|x_i) \frac{\partial \tilde{r}_k(y', x_i, \pi)}{\partial \pi(c, y|x_i)} - \beta \log \frac{\pi(c, y|x_i)}{\pi_0(c, y|x_i)} - \beta - \lambda_i = 0$$

The key challenge is the second term, which captures how changing $\pi(c, y|x_i)$ affects all other rewards $\tilde{r}_k(y', x_i, \pi)$ through the policy dependence.

For the optimal policy $\pi^\star$, rearranging:

$$\beta \log \frac{\pi^\star(c, y|x_i)}{\pi_0(c, y|x_i)} = \tilde{r}_k(y, x_i, \pi^\star) + \sum_{c',y'} \pi^\star(c', y'|x_i) \frac{\partial \tilde{r}_k(y', x_i, \pi^\star)}{\partial \pi^\star(c, y|x_i)} - \beta - \lambda_i$$

Taking the exponential:

$$\pi^\star(c, y|x_i) = \pi_0(c, y|x_i) \exp\left(\frac{1}{\beta} \left[\tilde{r}_k(y, x_i, \pi^\star) + \sum_{c',y'} \pi^\star(c', y'|x_i) \frac{\partial \tilde{r}_k(y', x_i, \pi^\star)}{\partial \pi^\star(c, y|x_i)} - \beta - \lambda_i \right] \right)$$

Using the normalization constraint to determine λ_i, the first-order conditions become:

$$\pi^\star(c, y|x_i) \propto \pi_0(c, y|x_i) \exp\left(\frac{1}{\beta} \left[\tilde{r}_k(y, x_i, \pi^\star) + \sum_{c',y'} \pi^\star(c', y'|x_i) \frac{\partial \tilde{r}_k(y', x_i, \pi^\star)}{\partial \pi^\star(c, y|x_i)} \right] \right) .$$

Finally, as the rewards are indicator functions, which is discontinuous, their derivative is null almost surely. This allows us to set all the rewards partial derivative to 0, obtaining a fixed point equation that $\pi^\star$ verifies:

$$\forall (c, y) \, , \pi^\star(c, y|x) \propto \exp\left(\frac{1}{\beta} \tilde{r}_k(y, x_i, \pi^\star) \right) \pi_0(c, y|x_i) \, . \tag{4}$$

TTRL targets this solution by solving Equation 1. However, obtaining this solution can also be conducted by solving the fixed point equation directly. The fixed point solution can be solved by iterating over $m \geq 1$ the following:

- Collect a dataset $\mathcal{D}_{m-1}$ with π_{m-1}.
- Update $\pi_m \propto \exp\left(\frac{1}{\beta}\tilde{r}_k(y, x_i, \pi_{m-1})\right)\pi_0(c, y|x_i)$

until you reach convergence of the policy π_m. The update step can be solved by optimization, and can be implemented by the following Algorithm:

Algorithm: Fixed Point Approach

Input: Policy π_θ, prompt dataset $\mathcal{P} = \{x_i\}_{i\in[n]}$, number of candidates k.
Initialize π_0.
for $m = 1, 2, \ldots$ **do**
  Construct offline dataset $\mathcal{D}_{m-1}$ with π_{m-1}
  Set $b_{m-1}(y, x_i) = \tilde{r}_k(y, x_i, \pi_{m-2})$ if $m > 2$ else $b_{m-1}(y, x_i) = 0$.
  Using $\mathcal{D}_{m-1}$, we approximate and solve the weighted log-likelihood objective

$$\pi_m = \arg\max_{\pi} \left\{ \sum_{i=1}^{n} \mathbb{E}_{\pi_{m-1}(\cdot|x_i)} \left[\exp\left(\frac{1}{\beta}(\tilde{r}_k(y, x_i, \theta_{m-1}) - b_{m-1}(y, x_i)) \right) \log \pi_\theta(c, y|x_i) \right] \right\}. \tag{5}$$

This algorithm is exactly Algorithm 1 with the particular choice of g_m to be:

$$g_m(y, x) = \exp\left(\frac{1}{\beta} \left(\tilde{r}_k(y, x, \pi_{m-1}) - b_m(x, y) \right) \right), \tag{6}$$

with $b_{m-1}(y, x) = \tilde{r}_k(y, x, \pi_{m-2})$ if $m > 2$ else $b_{m-1}(y, x) = 0$. `RoiRL` can exactly target the optimal solution of `TTRL` with the choice of g in Equation 6. This ends the proof.

□

B Extended Empirical Results

B.1 Experimental Details

B.1.1 Implementation Details

We compared three training methods using the same hyperparameters. In all experiments, we generate $k = 10$ candidates for each problem and then train on these candidates using the chosen reward function g. We used our custom `WeightedSFTTrainer` to implement learning with $g_\beta : x \to \exp(x/\beta)$ and used standard `SFTTrainer` to implement $g_I : x \to x$, as it is equivalent to using supervised finetuning on candidates with answers corresponding to majority vote. We will refer to the proccess of generating candidates and training on them as 1 round of `RoiRL`. In both methods, after generating candidates, we train for 3 epochs during every round. We trained all methods for 15 rounds, with early stopping if $\texttt{maj}_{10}$ accuracy on the train dataset did not improve for more than 5 rounds. We take the round with the best performance on train, when reporting the final accuracies.

All hyper-parameters were set to their default values in `SFTTrainer`, the only exception being the reduction of the learning rate from $2 \cdot 10^{-5}$ to 10^{-6} for Llama-3.2 with g_I because higher values result in overfitting. We implemented `TTRL` with Huggingface `GRPOTrainer` and used its default parameters.

In our experiments, we compare one round of `RoiRL` with one epoch of `TTRL` as both require the same number of generations, namely, kn. However, note that `TTRL` is computationally more demanding and thus requires more time per epoch. More details of the computational advantages of `RoiRL` over `TTRL` are discussed in B.1.3.

B.1.2 Evaluation

To obtain the majority vote, we generate k answers from the model using temperature sampling with a temperature equal to 1.0. The maximum number of new tokens is set to 1024. We then extract the

answers from the generated solution, find the one that has the most occurrences, and choose it as our majority vote answer. If two answers have equal number of occurrences, we pick one of them randomly.

To extract the answer, we find the first occurrences of `\boxed{}` in the generated solution and consider everything that was put inside the brackets as the final answer. Qwen2.5-Math automatically puts the final answer inside `\boxed{}` and to ensure the same behavior from the other two models, we add a phrase "`Put your answer in \boxed{}`" to the prompt.

Since the same answer can be written in multiple ways (for example, 0.5 or $\frac{1}{2}$) we used `sympy` to parse latex and then compared the answers as sympy objects. In case of parsing or comparison errors or timeouts, we fallback and compared the answers as strings.

B.1.3 Computational advantages of RoiRL

The proposed RoiRL method has several computational advantages over TTRL. Firstly, we strictly separate the generation and the training phases and this allows better batching during generation. More precisely, we can generate answers to multiple questions in a single batch, unlike online RL algorithms such as TTRL. Secondly, our reward method does not require storing logits, so we can use larger batch size during generation compared to TTRL. Unlike GRPO, our method does not use a reference model to compute the reward, further reducing its computational cost. And finally, using a sparse reward function (e.g., g_I) in RoiRL can significantly speed up the training phase, especially in the early stages, when the majority answers are sparse. With all these advantages combined, we achieve performance more than $\times 2.5$ times faster per 1 round of RoiRL compared to 1 epoch of TTRL. The exact time evaluations for TTRL and RoiRL with sparse (g_I) and dense (g_β) reward functions are presented in Table 2.

All experiments were conducted on a Google Cloud Platform (GCP) instance with a single NVIDIA A100 (80Gb VRAM), 12 vCPUs and 170 GB RAM running on Debian GNU/Linux 11 (bullseye). RoiRL and TTRL methods were implemented using the Hugging Face Transformers library v4.52.4, TRL v0.18.2 and PyTorch v2.7.1 with CUDA 12.4.

Table 2: Time per round comparison

Method	Time per round
RoiRL, g_I (sparse reward)	6552.5s
RoiRL, g_β (dense reward)	8883.5s
TTRL	17019.25s

B.2 Detailed Results and Discussions

Figures 1, 2 and 3 illustrate the training curves for Qwen2.5-Math-1.5B [3], Phi-4-mini-reasoning-4B [1], and Llama-3.2-3B-Instruct [5] trained with RoiRL and TTRL. The greedy decoding baseline and the $\texttt{maj}_{128}$ baseline are represented by horizontal lines.

RoiRL consistently improves the accuracy on the MATH500-train dataset and successfully generalizes on a holdout MATH500-test dataset, AMC and AIME datasets. Note that unlike supervised finetuning, RoiRL does not require ground-truth labels even on the train dataset, so this is a self-improvement process. Compared to TTRL, RoiRL demonstrates faster convergence with the same number of training rounds and less computations.

RoiRL improves not only the sampling accuracy (dotted lines), but also the $\texttt{maj}_{10}$ accuracy (solid lines). Moreover, for Qwen-2.5 and Phi-4, after several epochs the sampling accuracy exceeds the initial $\texttt{maj}_{10}$ accuracy. This demonstrates how RoiRL does not just distill the majority vote performance into the base model, but improves the general ability to solve mathematical problems.

Finally, Figure 4 illustrates the evolution of the average entropy during training for Qwen2.5 on MATH500-Train and Test. When using the RoiRL, entropy rapidly decreases to almost zero. However, for TTRL, the entropy stays relatively high during the whole training process. This can explain faster convergence of RoiRL during our experiments. In addition, the fast reduction of the entropy to zero with RoiRL raises a natural question of applying more regularization, implicitly by reducing the learning rate or using alternative reward functions, which may be the subject of further research.

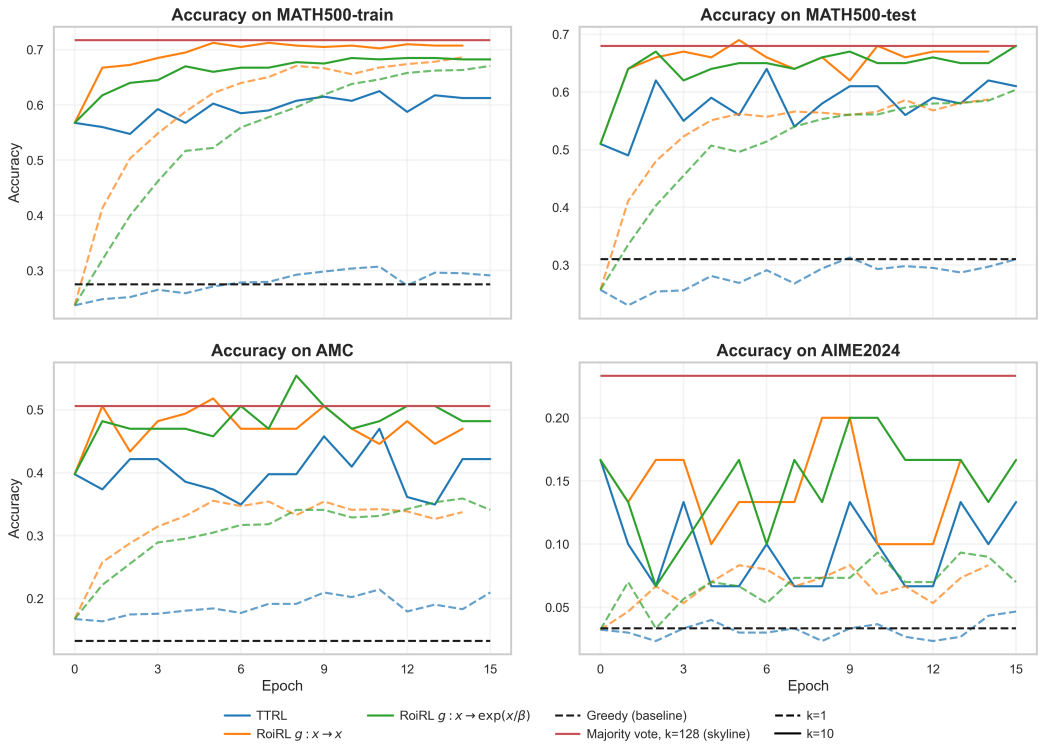

Figure 1: Training curves for Qwen-2.5-Math

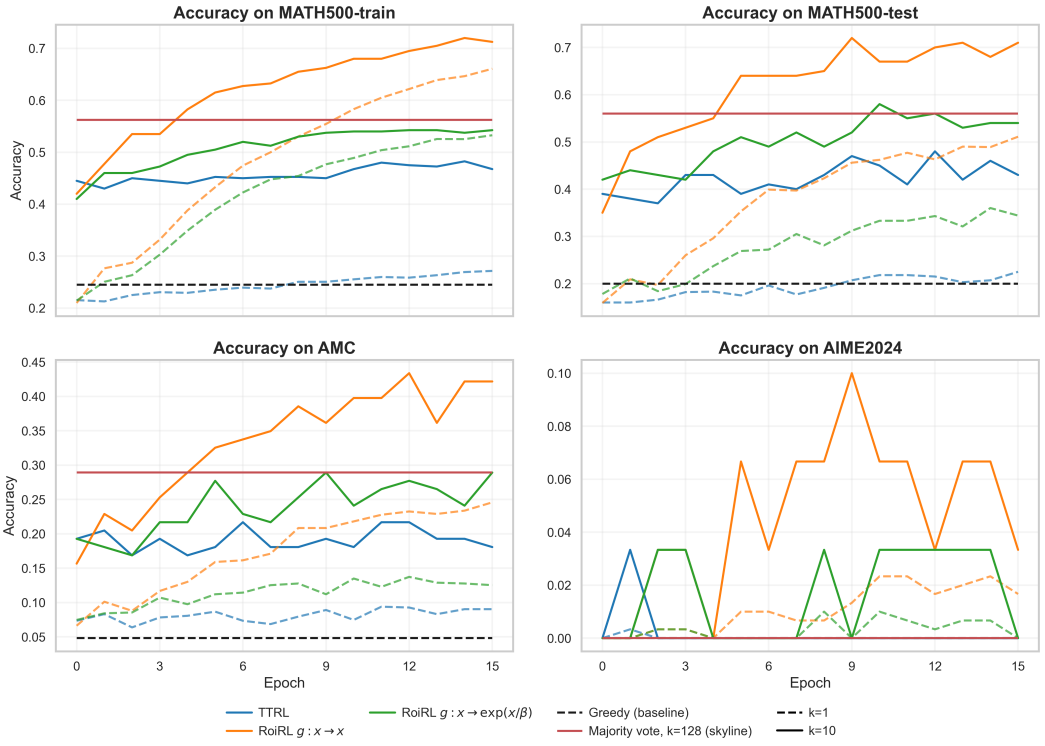

Figure 2: Training curves for Phi-4

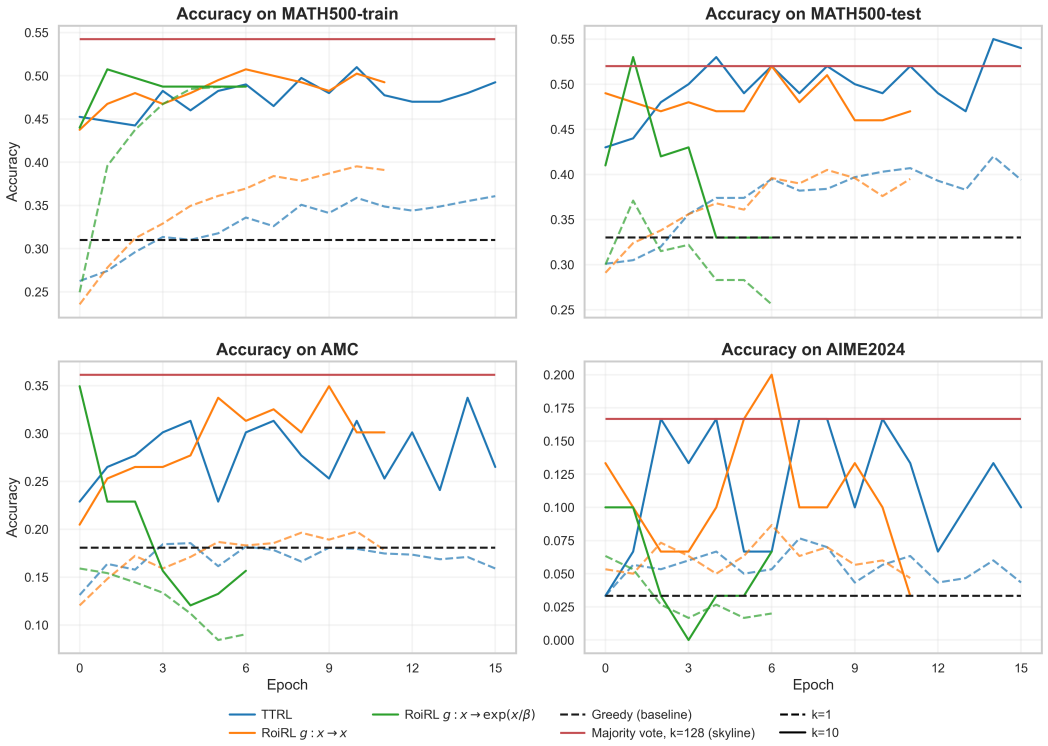

Figure 3: Training curves for Llama-3.2

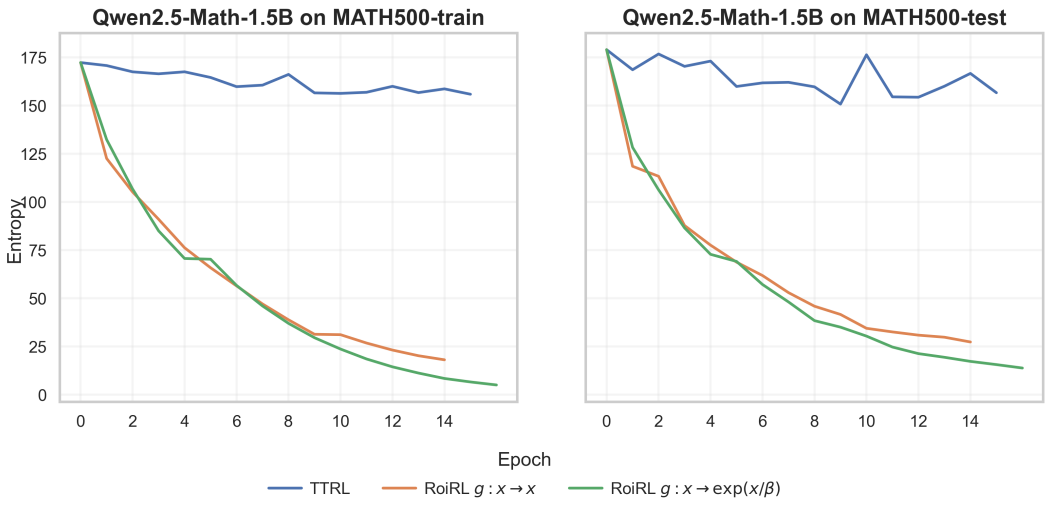

Figure 4: Entropies for Qwen2.5 on MATH500