Report: Collaborative Post-Training with RL Swarm

Gensyn AI Team

Abstract

The recent success of DeepSeek-R1 highlights the potential of reinforcement learning (RL) in post-training LLMs, particularly for reasoning-intensive tasks such as mathematics where correctness can be algorithmically verified. In particular, by leveraging the GRPO algorithm introduced by Shao et al. [2024], the DeepSeek-R1-zero model [DeepSeek-AI, 2025] highlighted that models can self-evolve and develop reasoning capabilities without supervised fine-tuning (SFT) or even a critic model. These capabilities are achieved by leveraging a strong pre-trained LLM as a "base" model that serves as the starting point for post-training, and then allowing the model to learn through a repeated self-assessment process in GRPO. Intuitively, each self-assessment involves a three step process where the model: i) generates multiple responses for a given prompt, ii) computes rewards individually for each of its generated responses using a rule-based system, and iii) compares the rewards for each individual response against the average reward across all responses it generated for said prompt.

A natural next step is to ask: what if we allow models to not just learn alone but *together* with multiple peer models on the same tasks? Can this new dimension allow each model to train more efficiently, and unlock new learning patterns for the models? This report presents `RL Swarm`, an open and collaborative network where models perform post-training together with an algorithm based on GRPO.

We compare models post-training on `RL Swarm` with the baseline of a model post-training by itself using GRPO. Our preliminary results show that models trained on the swarm produce better answers on unseen test data and generally provide more human-readable responses. These results open the path to many exciting research questions in the area of collaborative post-training using `RL Swarm`'s infrastructure.

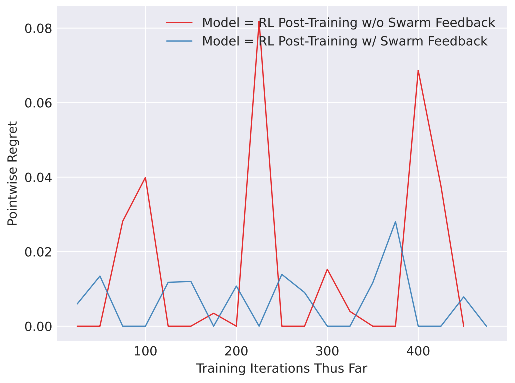
1 Methodology

This section presents an overview of our collaborative extension to the RL post-training algorithm used in the training of DeepSeek R1 DeepSeek-AI [2025], Shao et al. [2024]. We assume prior knowledge of these topics and refer the reader to the original paper for background. In a nutshell, the novelty of GRPO is avoiding the use of a critic model and instead having the models being trained generate multiple answers and self-assess what they produce. We take this to the next logical step: what if we had a *group of models* post-training together assess each other's solutions?

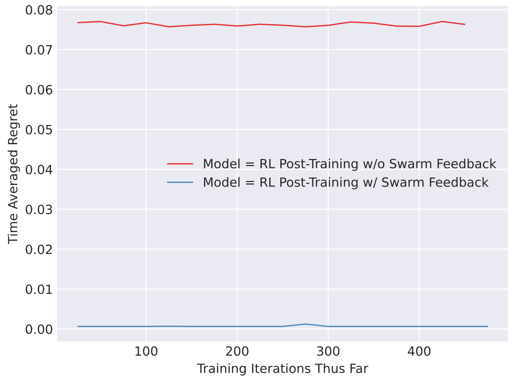
To test this, we design a three-stage procedure akin to a group of students studying together: models explore solutions on their own, share them with others, provide and receive feedback, and finally revise to form a final answer. Each training step proceeds as follows:

1. Stage 1: Answering. Each model performs a GRPO training step: it generates eight answers per question (there are about 7500 questions per step), calculates rewards, advantage, loss, and performs the gradient update. The model then shares its best answers for each question with the rest of the swarm.

2. Stage 2: Critiquing. Each model looks at the answers provided by each peer and provides its feedback.



(a) Pointwise Relative Regret



(b) Time Averaged Relative Regret

Figure 1: Comparison of "relative regret" between model trained solo with GRPO (red), and model trained in the swarm (blue) on test data. The pointwise relative regret graph shows that the swarm model produces the better-rewarded answers at most timesteps. Even when the solo model gains more rewards at a given step the difference is relatively low, making the time-averaged relative regret close to 0 for the swarm model.

3. Stage 3: Resolving. Each model has to vote for what the majority will think the best answer is for each question. Then, each model produces a final revised answer to the prompt.

The goal of this setup is to use peers as their own critics, without the need for a separate critic model. Viewed another way, by studying its peers' solutions, each model gets *exploration for free*. Similar benefits are also seen in traditional multi-agent reinforcement learning.

Reward structure. As done in DeepSeek-R1, we employ a rule-based model mechanism for models to calculate rewards at each stage (see Section 2.2.2 of DeepSeek-AI [2025]). In Stages 1 and 2, models are rewarded for formatting their answers and feedback correctly (i.e., in certain XML tags). In Stages 1 and 3, the model additionally gets a reward for getting the answer right. This is determined by parsing between `<answer>` and `</answer>` and then checking if that equals the ground truth answer. In Stage 3, models are first rewarded if they correctly predict which answer the majority will think is the best among those shared in the SWARM after Stage 1.

2 Experiments

We build `RL Swarm` as a peer-to-peer network where nodes can join and collaboratively perform RL post-training.

Experimental setup. We collaboratively train 3 Qwen-2.5-1.5B models (Bai et al. [2023]) on the GMS8K mathematical reasoning dataset (Cobbe et al. [2021]) for 500 iterations. We refer to a model trained here as SWARM. The baseline we compare with is a model trained alone with the GRPO training algorithm for the same number of steps, which we call SOLO. Even on this simple setting and training for a relatively shorter number of steps, we find that models in the swarm learn quantifiably better.

2.1 Observation 1: Models trained on `RL Swarm` obtain higher rewards most of the time.

For each of the questions in a step, we compare the relative rewards obtained by the models in the two settings. We then measure a custom notion of dynamic regret that we call "relative regret". For a given step t, we let $\mathsf{Rew}(\mathsf{model})_t$ be the rewards of the model over all questions at that step. The pointwise "relative regret" of each model at t is: (where self is one of $\{\mathsf{SOLO}, \mathsf{SWARM}\}$)

$$\mathsf{PointwiseRelativeRegret}_t(\mathsf{self}) = \max\{\mathsf{Rew}(\mathsf{SOLO})_t, \mathsf{Rew}(\mathsf{SWARM})_t) - \mathsf{Rew}(\mathsf{self})\}_t$$

Figure 1 visualizes the calculation of regret for models trained over 500 iterations. To highlight how often the swarm model obtains higher rewards, we also compute the time-averaged regret: at each

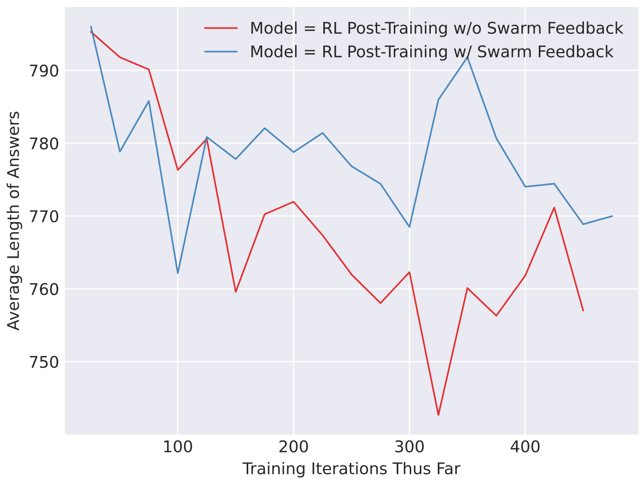


Figure 2: The average length of answers on test data produced by the swarm model versus the model trained alone, each for 500 steps. The swarm model almost always produces a slightly longer answer.

timestep T, this is the average difference in rewards between the better of the two answers versus the model's answer taken over each step from $t = 0$ to T. This is nearly flat at 0 because the swarm model obtains more rewards most of the time, and even when it doesn't, the reward difference is relatively low.

2.2 Observation 2: Models trained on `RL Swarm` produce more reasoned-out answers.

From reading the model's answers, we find that the structure and reasoning shown by the models trained in the swarm are more detailed. We believe this is because these models are trained to critique the responses of their peers and read the feedback they receive. Thus, models trained in the swarm are encouraged to make their responses more understandable to others.

A related observation is that the swarm model almost always produced longer answers, as shown in Figure 2. What is the swarm model spending these extra words on? We find its answers are almost always better formatted with bullet points, spacing, and even LaTeX for mathematics. Below is an example highlighting this for a relatively simple problem that both models answer correctly. Note that the newlines, tabs and LaTeX formatting are as specified by the models themselves.

```
// SOLO model output:
Mr. Gates bought 30 packages of hot dog buns, with each package containing
    8 buns. So, the total number of buns is 30 * 8 = 240 buns. He has four
    classes with 30 students in each class, so the total number of
    students is 4 * 30 = 120 students. To find out how many hot dog buns
    each student can get, we need to divide the total number of buns by
    the total number of students, i.e., 240 / 120 = 2 buns per
    student.\n</think>\n<answer>\n2\n</answer>",
```

```
// SWARM model output:
To determine how many hot dog buns each student can get, we need to follow
    these steps:
1. * * Calculate the total number of hot dog buns:* *
       - Mr. Gates bought 30 packages.
       - Each package contains 8 hot dog buns.
```

```
      \\[\n 30 \\text{ packages} \\times 8 \\text{ buns per package} =
          240 \\text{ buns}\n \\]

2. * * Calculate the total number of students:* *
      - Mr. Gates has four classes.
      - Each class has 30 students.
      \\[\n 4 \\text{ classes} \\times 30 \\text{ students per class} =
          120 \\text{ students}\n \\]
3. * * Determine how many hot dog buns each student can get:* *
      - We have 240 hot dog buns in total.
      - There are 120 students.
\\[\n \\frac{240 \\text{ buns}}{120 \\text{ students}} = 2 \\text{ buns
    per student}\n \\]
Thus, each of Mr. Gates' students can get <answer>\n2\n</answer> hot dog
    buns.
```

References

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. *arXiv preprint arXiv:2309.16609*, 2023.

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. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL `https://arxiv.org/abs/2501.12948`.

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. URL `https://arxiv.org/abs/2402.03300`.