Efficient RL for Language Models

Justin Qiu

1 Problem Statement and Technical Approach

Deepseek R1 [DeepSeek-AI et al., 2025] recently introduced a novel way of training LLMs to reason better by using large-scale reinforcement learning on a pretrained model without prior supervised finetuning. They use GRPO (Group Relative Policy Optimization) to do RL on their language model. GRPO samples multiple outputs from the current policy at each step and computes the advantage of each output against the reward model compared to the other outputs. Rather than using a neural reward model, which is currently very common in current research, they use a rules-based reward function that takes into account things like mathematical accuracy, logical consistency, language consistency, and format. They optimize the language model with a loss function that takes into account both the advantage gained for each output from the updated policy and the KL divergence against a reference policy that prevents the model from deviating too far. GRPO might help improve performance with a sparse reward model over PPO.

The loss function from the paper is below for convenience:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)} \Bigg[\frac{1}{G}\sum_{i=1}^{G} \Bigg(\min \Bigg(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)} A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon \right) A_i \Bigg) \Bigg) - \beta D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}} f) \Bigg] \tag{1}$$

where D_{KL} is an approximation of the KL divergence and A_i is the advantage normalized over the group of outputs. For my project, I will explore using parameter-efficient ways to train language models using reinforcement learning. Generally speaking, current methods update all parameters of the model during the backpropagation in the RL loop. I would like to see whether we can avoid this and achieve similar performance. Some ideas I have now include:

1. Updating only the first few or last few layers of the model. This has been explored with supervised finetuning methods, and researchers have found that performance can be maintained or even improved with far fewer weight updates than full finetuning [Lee et al., 2023, 2019].
2. Selecting which layers to update during training. This has also been explored in the context of SFT [Ardakani et al., 2024, Liu et al., 2021].
3. Optimization tricks that can approximate weight updates effectively, inspired from LoRA [Hu et al., 2021] and its variants. This has been applied recently to RLHF by Sidahmed et al.

[2024], and they find that adapting LoRA to both training the reward model and updating the weights of the model being trained achieves comparable performance to regular RLHF.

As I experiment and do more research I'll definitely come across more ideas. I am currently planning to focus on math tasks, as it is relatively easy to work with, data is easy to find or synthetically generate, DeepSeek-R1 was meant for reasoning in the first place, and the reward model can be fairly straightforward, as math questions generally have a well defined answer.

This problem is important because the methods introduced by Deepseek-Math [Shao et al., 2024] and Deepseek-R1(-Zero) represent shifts in how we think about post-training, and their empirical results show that their methods can be very successful even with less compute. I will measure success by finding out whether a method that involves less weight updates using the pure RL approach in Deepseek-R1-Zero can work. The main constraint on my project is compute, as even finetuning a small pretrained model can be computationally expensive and I don't have access to very much compute. It seems right now that the main thing that can go wrong is working with all of the libraries that have recently come out and trying to replicate or build on top of their methods. I spent five hours today trying to get TinyZero to train on a GPU cluster but somehow couldn't get it to work. I probably need to meet with the Professor to ask for advice on debugging this.

In terms of data, one idea I had in mind is using some kind of curriculum learning method to train the model. This could entail training the model with progressively harder math data. I have found a lot of possible data online from sources like the ACL workshop on curriculum learning [Warstadt et al., 2023], recent reasoning datasets like OpenThoughts (https://huggingface.co/datasets/open-thoughts/OpenThoughts-114k), etc.

2 Initial Results and Next Steps

The first thing I wanted to do was to get TinyZero (https://github.com/Jiayi-Pan/TinyZero) working and try to freeze all but the first layer and see the results of training. For reference, TinyZero is a reproduction of DeepSeek R1 Zero for math tasks. Unfortunately, I couldn't get the training script running on my GPU cluster after spending 5+ hours trying to do it, and it was a nightmare of debugging issues with CUDA and wrestling with the lack of documentation. It would be extremely helpful to meet with the professor to talk about this.

I will try to produce tangible results by next milestone check-in. I tried to produce some working code for this milestone but it was really hard to get a minimal working example going quickly in just a jupyter notebook, and I spent hours reading through current open-source implementations like TinyZero, RAGEN (https://github.com/ZihanWang314/RAGEN/), and OpenR1 (https://github.com/huggingface/open-r1/) to understand what's going on. Also, since most of these implementations rely heavily on several frameworks like veRL (https://github.com/volcengine/verl) and TRL (https://github.com/huggingface/trl/), I also combed through those repositories, which took a long time. It seems that this project will involve heavy amounts of coding and modifying massive repositories.

For next steps, I need to:

1. Actually get TinyZero properly running and train a small model (like Qwen0.5B) with it.
2. Modify the RL algorithm (still need to look into DPO/PPO/GRPO/what to use) to only change the weights of one layer and run TinyZero again

3. **Most important thing is I probably need some help / advice on debugging and working with these massive ML codebases**

3 Self-Critique

The obvious improvement for the next draft is that I need to actually get my code working. Because this is such a glaring weakness I don't think there is much else to discuss in this section. I will try to sync with the Professor about the overall direction of my paper during the individual meetings next week as well. I think the direction is very exciting and timely, but it is also very ambitious and not super well defined right now. My next step is to continue trying to get a minimal implementation for my idea, which I have already elaborated on above.

References

Arash Ardakani, Altan Haan, Shangyin Tan, Doru Thom Popovici, Alvin Cheung, Costin Iancu, and Koushik Sen. SlimFit: Memory-efficient fine-tuning of transformer-based models using training dynamics. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6218–6236, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.345. URL https://aclanthology.org/2024.naacl-long.345/.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li,

Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Jaejun Lee, Raphael Tang, and Jimmy Lin. What would elsa do? freezing layers during transformer fine-tuning, 2019. URL https://arxiv.org/abs/1911.03090.

Yoonho Lee, Annie S. Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts, 2023. URL https://arxiv.org/abs/2210.11466.

Yuhan Liu, Saurabh Agarwal, and Shivaram Venkataraman. Autofreeze: Automatically freezing model blocks to accelerate fine-tuning, 2021. URL https://arxiv.org/abs/2102.01386.

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.

Hakim Sidahmed, Samrat Phatale, Alex Hutcheson, Zhuonan Lin, Zhang Chen, Zac Yu, Jarvis Jin, Simral Chaudhary, Roman Komarytsia, Christiane Ahlheim, Yonghao Zhu, Bowen Li, Saravanan Ganesh, Bill Byrne, Jessica Hoffmann, Hassan Mansoor, Wei Li, Abhinav Rastogi, and Lucas Dixon. Parameter efficient reinforcement learning from human feedback, 2024. URL https://arxiv.org/abs/2403.10704.

Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell, editors. *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, Singapore, December 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.conll-babylm.0/.