Extended Abstract

Motivation The recent rise of interest in large language models (LLMs) has also led to developments in reinforcement learning, and applying ground paradigms to better fine-tune LLM learning. Two main paradigms have been: training on preference datasets, where ground truth rewards are ambiguous and instead responses are compared in relative quality, and training on verifier datasets, such as math and logical reasoning.

Method Two pipelines were implemented to enhance the Qwen 2.5 0.5B Base's reasoning capabilities. On the preference dataset of Ultrafeedback, we implemented a paradigm that comprises supervised fine-tuning (SFT), direct preference optimization (DPO) that incorporated the Bradley-Terry reward modeling objective, and an extension of weighted rewards using the OpenAssistant deberta large reward model. On the verifier dataset of Countdown, we partially implemented a paradigm that comprises SFT and Reinforce Leave One Out (RLOO).

Implementation First, we implemented SFT for a pre-trained language model. We then implemented the DPO objective using the SFT model as a reference policy, and then implemented the Bradley-Terry reward modeling objective. We separately implemented the RLOO online policy-gradient algorithm for the Countdown dataset (Pan et al. (2025))

For the Ultrafeedback preference dataset, we used the VLLM library to produce responses from our model. To evaluate the performance on Ultrafeedback, we used the parametric reward model (the Llama 3.1 Nemotron 70B Reward Model) to evaluate the performance of a given prompt and response. We constructed a win-rate binary label for each prompt and computed the win-rate as the mean of the binary label over all prompts.

For the verifier dataset, we first implemented SFT to warmstart the model using the cognitive behaviors dataset. We used a two-stage reward model to evaluate any answer from the Countdown dataset: 1) a format score to verify if a properly-formatted answer was provided, and 2) a verification score to ensure if the answer was correct or not

Results We see a considerable increase of performance in the SFT fine-tuned model, the SFT + DPO fine-tuned model, the DPO fine-tuned model, and the SFT + DPO + extension fine-tuned model.

Discussion A hybrid implementation of SFT + DPO + extension, as opposed to sole SFT fine-tuning, the base model training, or the SFT + DPO training, yields a better loss. While RLOO was not fully implemented,when compared to a naive Qwen model, the SFT-fine-tuned Warmstart dataset surpassed the baseline metric. This indicates that a paradigm of joined learning algorithms will perform both more effectively and more efficiently than any baseline model given.

A significant limitation we encountered throughout our work:for the weighted reward approach, we also encountered large variations in reward differences that destabilize the model. Ultimately, we would aim to improve on hyperparameter tuning for both DPO and RLOO, and integrate an SFT + RLOO + weighted-reward extension paradigm to measure the model's performance metric on mathematical and logical reasoning. We would also curate a more robust methodology for effective exploration such as retrospective replay.

Conclusion In sum, improving and implementing hybrid paradigms in fine-tuning reinforcement learning allows us to achieve higher performance and efficiency metrics amongst large language models. Difficulties arise in the implementation of weight-based reward models, which come at the risk of destabilizing the model and creating large variation in observed rewards. Further research would be to inspect how different parts of the hybrid paradigms can be connected together, such as an SFT + RLOO approach on a preference-based dataset, or an SFT + DPO approach on a verifier-based dataset. For our weight based reward model extension, we can also test different reward models to inspect similarities or differences in reward stabilization.

Fine-Tuning of Large Language Models using Retrospective Exploration with Critic-Augmented Progress

Caroline Van
Department of Computer Science
Stanford University
cmvan@stanford.edu

Natalie Wang
Department of Computer Science
Stanford University
ncwang19@stanford.edu

Abstract

The recent rise of interest in large language models (LLMs) has also led to developments in reinforcement learning, and applying ground paradigms to better fine-tune LLM learning. In this project, we introduce a hybrid paradigm called RECAP (Retrospective Exploration with Critic-Augmented Progress), which consists of performing SFT on a preference or verifier dataset, and subsequently applying DPO or RLOO to the fine-tuned model, as well as a weighted reward extension that would additionally permit us to explore more successful methodologies in efficient exploration or overall improved reasoning. Our results showed that our paradigm consistently outperformed baseline model metrics, even with the limitation of reward destabilization, opening up our efforts to replay or reasoning research endeavors.

1 Introduction

The recent rise of interest in large language models (LLMs) has also led to developments in reinforcement learning, and applying ground paradigms to better fine-tune LLM learning. Two main paradigms have been: training on preference datasets, where ground truth rewards are ambiguous and instead responses are compared in relative quality, and training on verifier datasets, such as math and logical reasoning.

Research has been conducted on retrospective replay-based reinforcement learning (RRL), which enables a model to revisit earlier promising states and rewards Dou et al. (2025). This approach also allows the model to dynamically replay promising states when the LLM's exploration ability decreases, maintaining a high exploration efficiency through the training period.

A current limitation of RRL is that potential states may still contain errors, and thus, if the model generates trajectories from these erroneous states, it will not find the correct trajectories. By implementing a supervised fine-tuning (SFT) + direct preference optimization (DPO) or SFT + Reinforce Leave One Out (RLOO) design to our model, and thus enhancing the SFT-only trained model, we aim to show that the more precise approach will create a stronger value model to find more promising intermediate states and overall yield a higher performance.

.

2 Related Work

2.1 Exploration Efficiency

Similar research has been dedicated to enhancing exploration efficiency: for example, it has shown to be possible to incentivize LLM reasoning capability without using SFT, with the DeepSeek-R1-Zero pipeline utilizing self-reflection and long chains of thought (CoT) generations (DeepSeek-AI et al. (2025)). This paper also introduces the DeepSeek-R1 pipeline, which collects a small amount of CoT data to fine-tune the model as an initial RL actor. On exploration thoroughness, research has also been conducted on proximal policy optimization (PPO) and reinforcement learning from human feedback (RLHF) frameworks, which also included annotated datasets in both English and Chinese.

In terms of future incorporation, we would aim to further fine-tune our model and improve our paradigm on non-English annotated datasets. What distinguishes our approach from previous approaches on exploration efficiency is that we are combining the replay buffer and performance optimization on both SFT and DPO trained datasets.

2.2 Experience Replay

In terms of retrospective replay, a similar but contrasting concept is the idea of experience replay, which is storing states and actions in a buffer and replaying them to update policy and value functions; RRL, meanwhile, replays potential intermediate states for exploration. Ongoing research in experience replay suggests that both a small or a large size for a replay buffer hinders the learning rate, and that this must be fine-tuned for as a hyperparameter for memory buffer size (Zhang and Sutton (2018)). Researchers presented the paradigm of the combined experience replay (CER), a special case of prioritized experience replay (PER) where the latest transition is given the largest priority. CER's main advantage is its $O(1)$ extra computation compared to PER's $O(\log n)$ extra computation.

A current limitation of this approach is that experience replay is not a complete algorithm on its own and must be paired with other learning algorithms, so it is difficult to test how useful CER's implementation is without confound. Its tested learning algorithms are also not as robust as the SFT + DPO paradigm or the SFT + RLOO paradigm, allowing us to further explore performance metrics on top of a foundationally stable fine-tuning paradigm.

3 Method

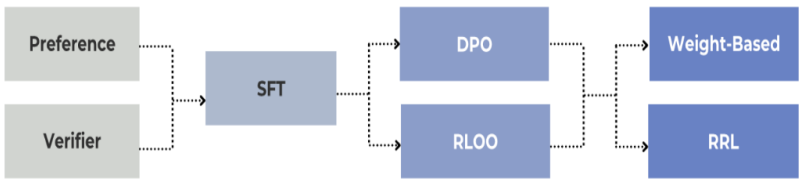


Figure 1: Method Overview.

Regardless of the dataset being classified as preference or verifier, a standard fine-tuning pipeline generally starts with supervised fine-tuning (SFT), which is the adaptation of a pre-trained LLM onto a labeled dataset, allowing the model to predict outputs for specific inputs. We modeled our SFT objective as:

$$\max_{\theta} E_{(x,y)\sim D} \sum_{t=1}^{|y|} \log \pi_\theta(y_t|x, y_{<t}) \tag{1}$$

The SFT objective was applied to a preference dataset (SmolTalk) and a verifier dataset (WarmStart) (Allal et al. (2025), Gandhi et al. (2025a)).

We present two pipelines on fine-tuning our model. We used Qwen 2.5 0.5B Base (Team (2024)) for all requirements in this project.

The first pipeline is for a preference dataset, where after applying SFT to the model, we use an implicit reward model to further fine-tune the model. Here, we use the direct preference optimization objective (DPO) (Rafailov et al. (2023)), which uses reward parameterization to perform supervised preference classification on human preference datasets. The loss for DPO is modeled as:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim D}[\log \sigma(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_l|x)} - \beta \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_l|x)})] \tag{2}$$

We also utilize the Bradley-Terry reward modeling objective, which can be represented as follows:

$$\mathcal{L}_{\mathbf{BT}} = \max_\phi \mathbb{E}_{(\mathbf{x},\mathbf{y}_w,\mathbf{y}_l)\sim D_{\text{pref}}}[\log \sigma(r_\phi(\mathbf{x}, \mathbf{y}_w) - r_\phi(\mathbf{x}, \mathbf{y}_l))] \tag{3}$$

The second pipeline is for a verifier dataset, where after applying SFT to the model, we use an explicit reward model to further fine-tune the model. Here, we use the Reinforce Leave One Out (RLOO), which estimates policy gradients to reduce variance of the averaged weighted rewards of the policy's other samples (Ahmadian et al. (2024)). The objective for RLOO is modeled as:

$$\frac{1}{k}\sum_{i=1}^{k}[R(y_{(i)}, x) - \frac{1}{k-1}\sum_{j\neq i} R(y_{(j)}, x)]\nabla \log \pi(y_{(i)}|x) \text{ for } y_{(1)}, ..., y_{(k)} \overset{i.i.d}{\sim} \pi_\theta(\cdot|x) \tag{4}$$

The model's policy gradient is fine-tuned on this trajectory, where q is defined as the programming problem and where $(y_1, y_2, ...y_n)$ represents the generated solution y with n tokens:

$$\max_\theta E_{(q,y)\sim D_{\pi\theta}}[\sum_i A^i \log P(a_i|s_i;\theta)] \tag{5}$$

Assuming that n^* is the correct solution's token number, the probability that a policy will generate the correct solution in order to obtain a positive reward to optimize is:

$$P = \prod_{i=1}^{n^*} p(a_i|s_i) \tag{6}$$

Addressing the possibility of errors in promising states, the optimization objective of the policy model is modeled as:

$$\text{Objective}(\theta) = E_{(s,\tau)\sim D_{\pi_\theta}}[r - \beta \log(\pi_\theta(\tau|s))/\pi^{ref}(\tau|s)] \tag{7}$$

where s and τ are the starting state and the policy model generated parts.

In addition to the RRL extension, we also implemented a weight-based reward model on the Ultrafeedback dataset by using the OpenAssistant/reward-model-deberta-v3-large-v2, which was used to score responses corresponding to the weights on DPO loss:

$$\Delta r = r_\theta(a^+) - r_\theta(a^-) \tag{8}$$

$$w = 1.0 + \text{zscore}(\Delta r) \tag{9}$$

$$\mathcal{L}_{\text{weighted}} = w \cdot \mathcal{L}_{\text{DPO}} \tag{10}$$

4 Experimental Setup

All training and evaluation pipelines were constructed in either AWS Instances or Lambda Cloud.

As an overview, our experimentation setup consists of the following: we devised a baseline for fine-tuning a LLM by implementing a set of popular fine-tuning algorithms for aligning parametric and rule based reward models with human preferences: supervised fine-tuning (SFT), direct preference optimization (DPO), REINFORCE Leave One-Out (RLOO), and the Bradley-Terry reward modeling objective. We then compared the performance of the response to a reference model (the Qwen 2.5 0.5B Instruct) Team (2024) by evaluating the learned policies and the baseline policies. We then extended our project with an Exploration extension by comparing the model's performance on our novel implementation, RECAP (Retrospective Exploration with Critic-Augmented Progress).

Our approach includes the following: we first generated a dataset for supervised fine-tuning (SFT) by using the SmolTalk dataset (Allal et al. (2025)). The SmolTalk dataset is formatted as follows in an example:

messages	**source**
`["content": "You are an AI chatbot playing the role of a charismatic rogue in a medieval role-playing game. You are quick-witted, charming, and skilled in stealth and thievery", "role": "system"]`	`smol-summarize-5k`

Table 1: Example of a Smoltalk dataset row

For the DPO algorithm, we also use the Ultrafeedback dataset (Cui et al. (2023). The Ultrafeedback dataset is formatted as follows in an example:

prompt	**prompt_id**	**chosen**
`Which animal has two hands, a hyrax or a dog?`	`d169f4610d69b...`	`["content": "Neither a hyrax nor a dog has hands. Hyraxes have four legs with feet that are adapted for...", "role": "system"]`

Table 2: Example of an Ultrafeedback dataset row

rejected	**messages**
`["content": "Thank you for your question. I'm happy to help you with that! However, I must point out...", "role": "assistant"]`	`["content": "Neither a hyrax nor a dog has hands. Hyraxes have four legs with feet that are adapted for...", "role": "system"]`

Table 3: Example of an Ultrafeedback dataset row (cont.)

First, we implemented SFT to finetune the Qwen/Qwen2.5-0.5B baseline model. We then implemented the DPO objective using the SFT model as a reference policy, incorporating Bradley-Terry reward modeling objective. We separately implemented the RLOO online policy-gradient algorithm for the Countdown dataset (Pan et al. (2025))

For the Ultrafeedback preference dataset, we used the VLLM Kwon et al. (2023) library to produce responses from our model. To evaluate the performance on Ultrafeedback, we used the parametric reward model (the Llama 3.1 Nemotron 70B Reward Model) to evaluate the performance of a given prompt and response Wang et al. (2024). We utilized the recommended evaluation approach by scoring the learned model's response with the reward model and compare the reward/score with a reference model to compute a win-rate. We did this by collecting prompts to evaluate, using VLLM to produce responses from our trained model and the Qwen/Qwen2.5-0.5B-Instruct reference model. After generating a reward score for both models, we constructed a win-rate binary label for each prompt and computed the win-rate as the mean of the binary label over all prompts.

For the verifier dataset, we first implemented SFT to warmstart the model using the cognitive behaviors dataset, of which an example is formatted as follows:

Query	**Completion**
`...User: Using the numbers [60, 37, 40], create an equation that equals 17. You can use basic arithmetic operations`	`<think> Let me try to find a path to 17 using these numbers. First, let me try working with the larger numbers: 60 - 40 = 20...`

Table 4: Example of a Warmstart dataset row

After implementing SFT on the cognitive behaviors dataset, we implement RLOO on the Countdown tasks, of which an example is formatted as follows:

target	**nums**
`98`	`[44, 19, 35]`

Table 5: Example of a Countdown dataset row

We used a two-stage reward model to evaluate any answer from the countdown task: 1) a format score to verify if a properly-formatted answer was provided, and 2) a verification score to ensure if the answer was correct or not (Gandhi et al. (2025b)).

For our RRL-based extension, we evaluated the performance of our model across SFT implementation, SFT + DPO implementation, DPO implementation, and DPO + extension implementation using the loss. Within the different normalization techniques in the weighed reward model, we compare the weighted and unweighted losses, the mean, min, max, and standard deviation of the reward difference, and the effective batch size.

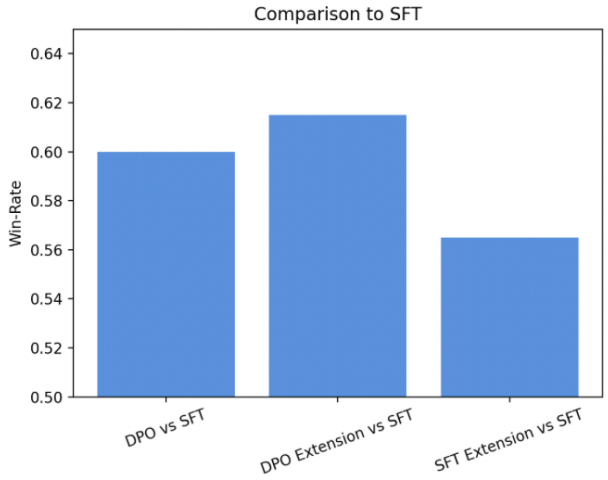
For the preference datasets, SFT was implemented on the entire dataset of SmolTalk, and both DPO and the extensions were subsequently implemented on the entire Ultrafeedback dataset. Each was completed with 1 epoch of training.

For the verifier datasets, SFT was implemented on the entire dataset of Warmstart, trained over two epochs. RLOO was subsequently implemented on 1500 data points from the Countdown dataset over 1 epoch of training.

5 Results

We see a considerable increase of performance in the SFT fine-tuned model, the SFT + DPO fine-tuned model, the DPO fine-tuned model, and the SFT + DPO + extension fine-tuned model.

5.1 Quantitative Evaluation

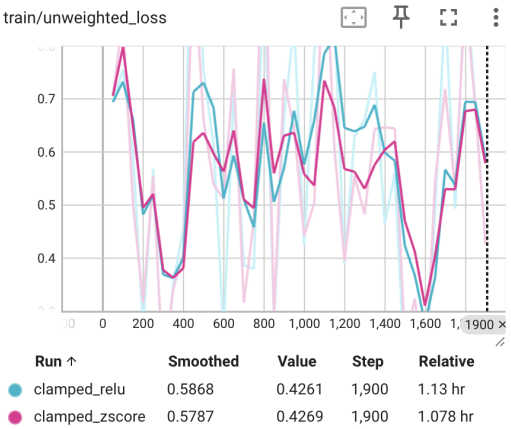


On the collective project leaderboard, our SFT implementation of Smoltalk and Cognitive Behaviors significantly passed the baseline metric. On the hybrid paradigm implementation of SFT + DPO + weighted reward extension, the SFT + DPO outperformed against the SFT implementation and the SFT evaluative model on the leaderboard submissions, and the SFT + DPO + weighted reward outperformed both against the SFT + DPO and against the extended evaluative model conducted on leaderboard submissions.

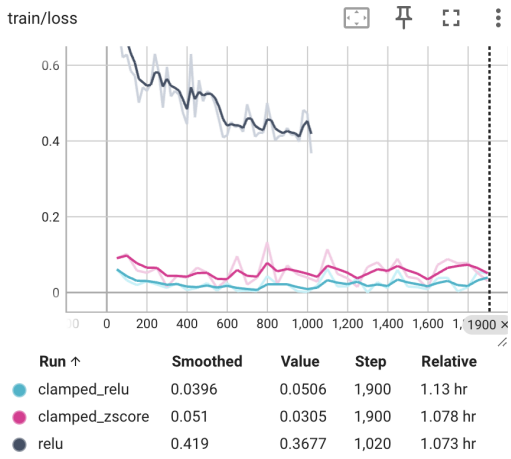
We note that, regarding the model's performance on SFT, both the performances of the DPO implementation and the DPO + extension was higher than the SFT + extension, and the SFT + extension outperformed the SFT-trained-only model.

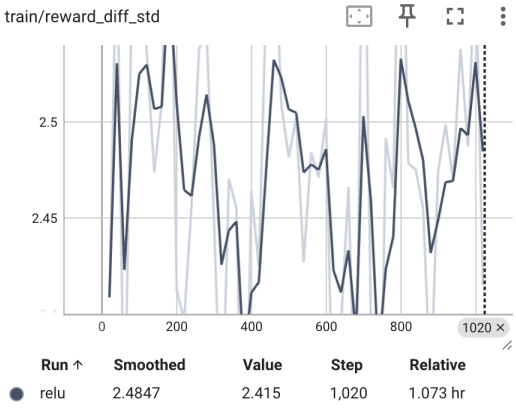
5.2 Qualitative Analysis

In our SFT + DPO model, we can observe a decrease in loss and an increase in reward.



In our SFT + DPO + weighted reward extension model, we can observe a decrease in loss and increases and decreases in reward through steps.



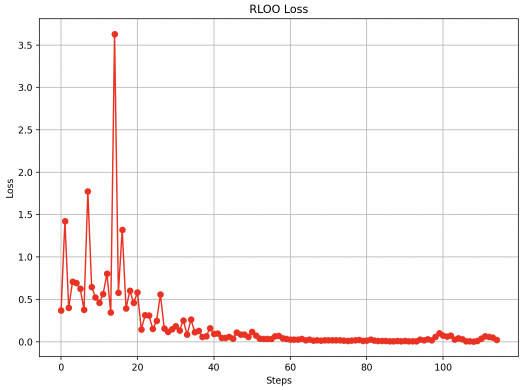


We also include a loss decrease in the RLOO training for the Countdown dataset:

While not visualized, the RLOO reward increased from -17 to -10 over 125 batches.

6 Discussion

We can see through the performance metrics that a hybrid implementation of SFT + DPO + extension, as opposed to sole SFT fine-tuning, the base model training, or the SFT + DPO training, yields a



better loss. While RLOO was not fully implemented, we also noted that when compared to a naive Qwen model, the SFT-fine-tuned Warmstart dataset surpassed the baseline metric. This indicates that a paradigm of joined learning algorithms will perform both more effectively and more efficiently than any baseline model given.

We encountered several limitations throughout our work: in the implementation of RLOO, there were tokenization errors in the evaluation pipeline, as well as the model having difficulty performing arithmetic on numbers into the thousands. For the weighted reward approach, we also encountered large variations in reward differences that destabilize the model. Ultimately, we would aim to improve on hyperparameter tuning for both DPO and RLOO. If we were to expand our work, we would also integrate an SFT + RLOO + extension paradigm to measure the model's performance metric on mathematical and logical reasoning. We would also curate a more robust methodology for retrospective replay.

7 Conclusion

Improving and implementing hybrid paradigms in fine-tuning reinforcement learning allows us to achieve higher performance and efficiency metrics amongst large language models. Difficulties arose in the memory usage, runtime, and parametrization of such paradigms; additionally, implementing weight-based reward models and retrospective replay comes at the risk of destabilizing the model, or creating large variation in observed rewards. Further research would be to inspect how different parts of the hybrid paradigms can be connected together, such as an SFT + RLOO approach on a preference-based dataset, or an SFT + DPO approach on a verifier-based dataset. For our weight based reward model extension, we can also test different reward models to inspect similarities or differences in reward stabilization.

8 Team Contributions

- **Caroline Van:** We implemented SFT together. I implemented DPO and the extension. We debugged the code and wrote the final report together.
- **Natalie Wang:** We implemented SFT together. I implemented RLOO. We debugged the code and wrote the final report together.

Changes from Proposal We focused our extension on weighted rewards, and also conducting research on how retroactive replay would be incorporated. We removed meta reinforcement fine-tuning from our pipeline due to computational and time costs. We also split the pipeline into two smaller approaches on separate datasets: SFT + DPO + extension on the Ultrafeedback preference dataset, and SFT + RLOO on the Countdown verifier dataset.

References

Arash Ahmadian, Chris Cremer, Matthias Galle, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Ustun, , and Sara Hooker. 2024. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. https://arxiv.org/abs/2402.14740

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2025. SmolLM2: When Smol Goes Big – Data-Centric Training of a Small Language Model. arXiv:2502.02737 [cs.CL] https://arxiv.org/abs/2502.02737

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. UltraFeedback: Boosting Language Models with High-quality Feedback. arXiv:2310.01377 [cs.CL]

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. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 [cs.CL] https://arxiv.org/abs/2501.12948

Shihan Dou, Muling Wu, Jingwen Xu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. 2025. Improving RL Exploration for LLM Reasoning through Retrospective Replay. arXiv:2504.14363 [cs.LG] https://arxiv.org/abs/2504.14363

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. 2025a. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. https://arxiv.org/abs/2503.01307

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. 2025b. Cognitive Behaviors that Enable Self-Improving Reasoners, or, Four Habits of Highly Effective STaRs. arXiv:2503.01307 [cs.CL] https://arxiv.org/abs/2503.01307

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language

Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. 2025. TinyZero. https://github.com/Jiayi-Pan/TinyZero. Accessed: 2025-01-24.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model.

Qwen Team. 2024. Qwen2.5: A Party of Foundation Models. `https://qwenlm.github.io/blog/qwen2.5/`

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J. Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. 2024. HelpSteer2: Open-source dataset for training top-performing reward models. arXiv:2406.08673 [id='cs.CL' full_name='Computation and Language' is_active=True alt_name='cmp-lg' in_archive='cs' is_general=False description='Covers natural language processing. Roughly includes material in ACM Subject Class I.2.7. Note that work on artificial languages (programming languages, logics, formal systems) that does not explicitly address natural-language issues broadly construed (natural-language processing, computational linguistics, speech, text retrieval, etc.) is not appropriate for this area.']

Shangtong Zhang and Richard S. Sutton. 2018. A Deeper Look at Experience Replay. arXiv:1712.01275 [cs.LG] `https://arxiv.org/abs/1712.01275`