Nested-ReFT: Efficient Reinforcement Learning for Large Language Model Fine-Tuning via Off-Policy Rollouts

Anonymous Author(s)
Affiliation
Address
email

RL-based fine-tuning with verifiable rewards corresponds to a specific fine-tuning problem called Reinforced Fine-tuning (ReFT) [Luong et al., 2024, Shao et al., 2024, Liu et al., 2025], which applies specifically to math and programming reasoning domains. As opposed to RL from human feedback (RLHF), a heuristic reward model can be used to verify and score the sampled completions instead of learning the reward model from human preferences [Rafailov et al., 2023].

Despite appealing performance gains, RL-based fine-tuning has a higher computational, and memory cost compared to supervised fine-tuning (SFT) Luong et al. [2024]. While ReFT circumvent the memory cost of storing a reward model, the computational cost of sampling multiple completions from a behavior model can be overwhelming [Kazemnejad et al., 2025, Shao et al., 2024]. This completion cost can add up significantly to the compute cost of updating the parameters of the target.

Practitioners currently sample 8 CoT completions per problem von Werra et al. [2020], and the base setup consists in using as behavior model the same version of the target LLM [Luong et al., 2024, Shao et al., 2024]. Although scaling up the number of CoT completions can lower the bias in the target model updates, it also adds a significant compute overhead. Therefore, there is a need to explore if the efficiency of the behavior model used for roll-outs can be improved. A broader impact would be to facilitate the generation of more completions to further improve the reasoning performance of LLMs.

Research question: *Is it possible to improve the computational efficiency of ReFT without compromising the performance of the fine-tuned target LLM?* We hypothesize that: i) given a target LLM to fine-tune, it is possible to perform roll-outs from a behavior model that has a lower computational cost than base formulations; ii) such low-cost roll-outs can be leveraged to update the target model with limited influence on performance.

1 Problem definition

Let $\mathcal{X}$ denote the space of possible prompts and $\mathcal{Y}$ denote the space of possible output sequences. Given a prompt $x_i \in \mathcal{X}$, an LLM encodes a generating policy π_θ, which defines a conditional probability distribution over output sequences $\hat{y}_i = (\hat{y}_{i,1}, \dots, \hat{y}_{i,L}) \in \mathcal{Y}$, where L is the number of tokens contained in the sequence. Let $\hat{y}_{i,<\ell}$ denote the tokens $(\hat{y}_{i,1}, \dots, \hat{y}_{i,\ell-1})$ in a sequence $\hat{y}_i$. The probability of sampling sequence $\hat{y}_i$ given a prompt x_i is defined in an auto-regressive manner: $\pi_\theta(\hat{y}_i|x_i) = \Pi_{\ell=1}^{L} \pi_\theta(\hat{y}_{i,\ell}|x_i, \hat{y}_{i,<\ell})$, where $\pi_\theta(\hat{y}_{i,\ell}|x_i, \hat{y}_{i,<\ell})$ is the probability of outputting token $\hat{y}_{i,\ell}$ given the prompt x_i and the previous tokens $\hat{y}_{i,<\ell}$.

Chain-of-thoughts and answers When applying LLMs to math reasoning, it is useful to distinguish chain-of-thought (CoT) sequences $\mathcal{Y}^{\text{cot}} \subset \mathcal{Y}$ from their value answers $\mathcal{Y}^{\text{val}} \subseteq \mathcal{Y}^{\text{cot}}$. The value is the exact solution to a math problem, while a CoT includes both the reasoning steps and the value. We assume access to a deterministic extraction function $v : \mathcal{Y}^{\text{cot}} \mapsto \mathcal{Y}^{\text{val}}$ that extracts values from CoTs.

Goal Consider a pretrained LLM, e.g., an open-sourced checkpoint from Hugging Face [Wolf et al., 2020]. The objective is to fine-tune this LLM such as to maximize the performance on

Submitted to NeurIPS 2025 Workshop on Efficient Reasoning. Do not distribute.

reasoning benchmarks. Consider benchmark k, denoted B_k. Let $(x_i, y_i^{k,\text{val}})$ denote the i-th example in benchmark B_k, where x_i is the prompt and $y_i^{k,\text{val}}$ is the target value answer. We compute the accuracy of the LLM answers $v(\hat{y}_i^k)$ on benchmark k as: $a_k = \frac{1}{|B_k|}\sum_{i=1}^{|B_k|} \mathbf{1}_{\left[v(\hat{y}_i^k)=y_i^{k,\text{val}}\right]}$. The goal is to maximize the overall performance, defined as $a = \frac{1}{K}\sum_{k=1}^{K} a_k$.

1.1 Reinforced fine-tuning

Let θ_{ref} denote the parametrization of a pretrained LLM policy. Reinforced Fine-Tuning (ReFT) aims to further train $\pi_{\theta_{\text{ref}}}$ by leveraging reward feedback on a given dataset D, for S gradient steps contributing to E_{rft} epochs. The dataset $D = (x_i, y_i^{\text{cot}})_{i=1}^{|D|}$ contains prompts describing math problems $x_i \in \mathcal{X}$ and their associated CoT solutions $y_i^{\text{cot}} \in \mathcal{Y}^{\text{cot}}$. Note that none of these math problems should be contained in the evaluation benchmarks.

Warm-up with SFT Prior to performing ReFT, it is common practice to perform E_{sft} epochs of supervised fine-tuning (SFT) on dataset D as a warm-up [Luong et al., 2024]. Let θ_{sft} denote the parametrization of the resulting LLM policy, which serves as the initialization for the ReFT step.

Sample Generation via Behavior Policy Roll-outs Let π_θ denote the running target LLM policy that is being fine-tuned, and initialized with $\theta = \theta_{\text{sft}}$. During ReFT, a behavior LLM policy η_χ is used to perform roll-outs to generate solutions. For each prompt x in dataset D, the behavior model samples G solutions, $\{\hat{y}_g \sim \eta_\chi(\cdot|x)\}_{g=1}^{G}$. The samples are scored using a reward function $r : \mathcal{Y}^{\text{val}} \times \mathcal{Y}^{\text{val}} \to \mathbb{R}$ that compares the extracted value answer $v(\hat{y}_g)$ to the ground truth value associated to problem x, $r_g = r(v(\hat{y}_g), v(y^{\text{cot}})), g = \{1 \ldots G\}$. The scored samples are then used to update the target policy π_θ using a RL algorithm (e.g., GRPO, Shao et al. [2024]). The objective of the RL algorithm is to find parameters θ that maximize the expected reward.

Importance sampling Learning a target policy using rewards obtained from a separate behavior policy, is *off-policy RL*. Off-policy RL algorithms typically rely on *importance sampling* to account for the distribution difference between behavior and target policies. More specifically, rewards are reweighted using the importance sampling ratio: $h_{\texttt{base}}(\hat{y}, x; \pi_\theta, \eta_\chi) = \frac{\pi_\theta(\hat{y}|x)}{\eta_\chi(\hat{y}|x)} = \prod_{\ell=1}^{L} \frac{\pi_\theta(\hat{y}_\ell|x,\hat{y}_{<\ell})}{\eta_\chi(\hat{y}_\ell|x,\hat{y}_{<\ell})}$. The importance sampling ratio can suffer high variance, especially when the behavior and target policies diverge [Xie et al., 2019], which can negatively affect training quality. Variance reduction techniques are often employed to stabilize training [Munos et al., 2016, Metelli et al., 2020].

Current RL-fine-tuning configurations Most RL fine-tuning implementations [von Werra et al., 2020] and ReFT approaches [Luong et al., 2024] consider the behavior policy such that $\eta_\chi = \pi_{\theta_{\text{old}}}$, where θ_{old} corresponds to the target policy parametrization from the previous gradient step. Consequently, the importance sampling ratio is computed between the behavior policy $\pi_{\theta_{\text{old}}}$ and the target policy π_θ. However, there is no architectural difference between π_θ and $\pi_{\theta_{\text{old}}}$. This implies that the compute resources to generate samples from behavior policy is the same as the target policy.

In this work, we explore the possiblity of generating samples from a behavior policy that has a lower inference cost, thus accelerating the inference with an increase in the off-policyness. Further, we bound the importance sampling ratio to reduce the variance in the updates for smooth training.

2 The Nested-ReFT framework

The Algorithm 1 summarizes the framework, with purple highlighting main differences with current ReFT. Though we experiment the framework with the popular GRPO algorithm [Shao et al., 2024], `Nested-ReFT` is agnostic to the RL algorithm and can be combined with every sampling based RL post-training techniques [Ahmadian et al., 2024, Ziegler et al., 2020].

2.1 Rollouts based on nested models

Consider the target model π_θ to fine-tune using ReFT and a behavior model η. We instantiate nested models with layer skipping. Layer skipping consists in selecting a set of layers that are not used (skipped) during the forward pass, acting like a short-cut.

Definition 1 (Set of transformer layers indices)**.** The set of transformer layer indices in model η is noted $T_\eta = \{t_0, t_1, \ldots, t_N\}$, where $|T_\eta| = N + 1$ denotes the total number of transformer layers.

Definition 2 (Set of valid layers indices)**.** The *set of valid layers* indices $V_{\eta,b} = T_\eta \setminus \{t_0, \cdots t_b, t_{N-b}, \cdots, t_N\}$ contains transformer layers indices within a distance b from the borders. The set of invalid layer indices is noted $\bar{V}_{\eta,b}$

The set $\bar{V}_{\eta,b}$ points to layer indices that should not be skipped because they are too close to the inner and outer ends of the model [Elhoushi et al., 2024, Fan et al., 2019]. The inner and outer ends of deep learning models critically contribute to the generation of the model [van Aken et al., 2019]. The total number of valid layers is $|V_{b,\eta}| = |T_\eta| - 2b$.

Consider now a ratio of layers to skip $x\%$, which is set by the user. The total number of layers skipped, noted U_x is $U_x = \begin{cases} \min(1, \texttt{ceil}(|T_\eta| \cdot x)) & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$ Note that the number of layers to skip is a function of T_η and not $V_{\eta,b}$ to maintain the number of skipped layers proportional to the original model size independently of b.

Definition 3 (Layer skipping module)**.** Given a behavior model η, a skipping ratio $x\%$, and a border parameter b, a layer skipping module is a stochastic function $f_{x,b} : \eta \to [0,1]^{|T_\eta|}$ that outputs a binary vector $\sigma = f_{x,b}(\eta)$, such that: 1) The number of indices flagged to be skipped is equal to U_x i.e. $\sum_{i=0}^{T_\eta} \sigma_i = U_x$, 2) The invalid indices are never flagged i.e. $\sum_{i=0}^{T_\eta} \sigma_i \mathbf{1}_{\{i \in \bar{V}_{b,\eta}\}} = 0$ Conditions 1) and 2) ensure that only valid layers are sampled. The U_x skipped layers are sampled i.i.d. and uniformly with probability $1/U_x$.

Ensemble of nested models throughout training At a given gradient step s in the ReFT training of S steps, the behavior model is defined such that $\eta_s = \pi_{\theta_{s-1}}$. This is aligned with prior works, where the behavior model corresponds to the old target model (commonly referred to as $\pi_{\theta_{\text{old}}}$). The nested behavior model η'_s is instantiated using $f_{x,b}(\eta_s)$. Throughout `Nested-ReFT` training, we obtain a stochastic ensemble of nested models $\mathcal{Z} = \{\eta'_s\}_{s=1}^{S}$. Setting the ratio $x = 0$ and $b = 0$, `Nested-ReFT` reduces to classical ReFT: $\eta'_s = \eta = \pi_{\theta_{\text{old}}}$.

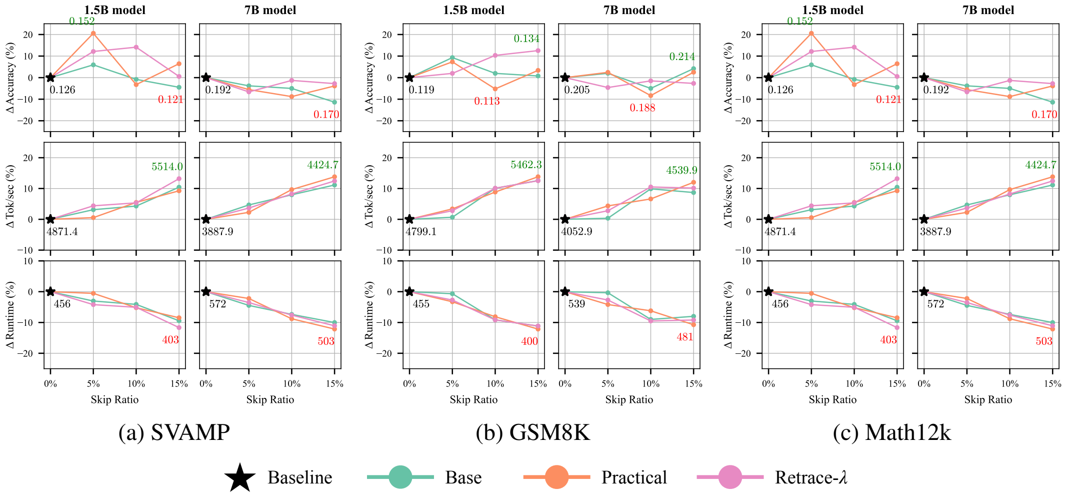
2.2 Mitigation of increased off-policyness

We explore techniques to mitigate the notable high variance on the importance sampling ratio caused by high off-policyness. Specifically, we use simpler variations of the base importance sampling ratio, summarized as $h_m(\cdot;\cdot)$, where $m \in \{\texttt{base}, 1, \lambda\}$: i) **Base approach**: The function is defined as the classical importance sampling ratio, corresponding to $h_{\texttt{base}}(\cdot,\cdot;\pi_{\theta_s},\eta'_s)$. This corresponds to the base importance sampling implementation [Shao et al., 2024]. ii) **Practical approach**: The function $h_1(\cdot,\cdot;\pi_{\theta_s},\eta'_s) = 1$. The motivation for this design choice is that the stochastic gradient descent is acknowledged as a powerful optimization protocol. iii) **Retrace-λ approach** The function $h_\lambda(\cdot,\cdot;\pi_\theta,\eta'_s) = \lambda \min(1, h_{\texttt{base}}(\cdot,\cdot;\pi_{\theta_s},\eta'_s))$. This approach is theoretically motivated following intuitions from Munos et al. [2016].

3 Experimental setup

We focus on the math reasoning task using five evaluation benchmarks, namely AIME2024 [Li et al., 2024], AMC [Li et al., 2024], MATH500 [Hendrycks et al., 2021], Minerva [Lewkowycz et al., 2022], and Olympiad [He et al., 2024]. We consider two large language models *Qwen2.5-Math-Instruct* models [Yang et al., 2024] of sizes 1.5B and 7B. We consider three datasets for fine-tuning, namely SVAMP [Patel et al., 2021], GSM8k [Cobbe et al., 2021], and Math12k [Hendrycks et al., 2021].

Instances of `Nested-ReFT` and baselines We consider instances of `Nested-ReFT` with a proportion of skipped layers $x \in \{5\%, 10\%, 15\%\}$. Since both 1.5B and 7B LLMs have the same number of layers, their number of skipped layers is identical (see Appendix E). We consider off-policyness mitigation strategies using variance mitigation strategies on the importance sampling ratio h_m, with $m \in \{\texttt{base}, 1, \lambda\}$. The case h_1 is referred to as "practical" and h_λ as "Retrace-λ"[Munos et al., 2016]. The border parameter is set to $b = 1$, implying that only the first and last layers of the models are never skipped. For a given model, a baseline to any instance of `Nested-ReFT` corresponds to the model fine-tuned with `Nested-ReFT` at ratio $x = 0\%$, border $b = 0$ and mitigation method $m = \texttt{base}$. This instance corresponds to the model fine-tuned with ReFT using the base off-policyness and importance sampling formulation from existing works [Shao et al., 2024, Luong et al., 2024]. To our knowledge, there is no prior work that could fit as a fair baseline in the proposed new framework.



(a) SVAMP (b) GSM8K (c) Math12k

Figure 1: Fine-tuning results across benchmarks. Red annotations indicate the smallest value; Green indicate the largest.

Performance metrics and delta to the baseline To characterize reasoning performance at test-time, we report the average accuracy on the 5 math benchmarks [Liu et al., 2025]. To characterize compute efficiency gains at train-time, we report the token speed (total number of tokens processed divided by total runtime), and the total run time (expressed in seconds). We characterize `Nested-ReFT` run instances using the relative *delta* (Δ) to the baseline, which is defined for any metric z as $\Delta(z) = 100 \cdot \frac{(z - z_{\texttt{baseline}})}{z_{\texttt{baseline}}}$, where the absolute delta is $\Delta_{\text{abs}}(z) = (z - z_{\texttt{baseline}})$.

3.1 Empirical Results

The setup comprises 12 distinct instances of `Nested-ReFT` per model, and 1 baseline. The experiment includes 2 models $\times$ 3 datasets $\times$ $(12 + 1)$ instances $= 78$ experimental configurations. The results are displayed in Figures 1a, 1b, and 1c. For Δ Accuracy $(\%)$ and Δ Tok/sec $(\%)$ the goal is to maximize the metric. For Δ Runtime $(\%)$, the goal is to minimize the metric. Table 2 references the absolute performance deltas.

Impact of off-policy roll-outs on performance On Table 2, for 1.5B checkpoints, the mean absolute performance delta for best case `Nested-ReFT` is higher than for the worst case, indicating that the magnitude of the performance gains achieved with `Nested-ReFT` is bigger than the magnitude of the performance drops. However, on 7B checkpoints, the mean absolute performance delta for gains is smaller than that of the drops. In both cases, the worst and best performances imply at most ± 2.6 points variation from the baseline performance. These results showcasing minor performance fluctuations corroborate the hypothesis that off-policy generations using `Nested-ReFT` have limited influence on the performance on reasoning benchmarks. Importantly, we highlight that some instances of `Nested-ReFT` yield performance improvements over the baseline while involving the generation of samples on a smaller model.

Effectiveness of the off-policyness mitigation strategy We consider 3 off-polyciness mitigation strategies, namely `Base`, `Practical` and `Retrace-`λ. We observe that `Retrace-`λ displays the most stable performance across all models, fine-tuning datasets, and skipping ratios. Over the three datasets and two models (i.e. 6 configurations), the `Base` strategy achieves $1/6$ best case count, $3/6$ worst case count and $2/6$ neutral count. The `Practical` strategy achieves $3/6$ best case performance, $3/6$ worst case, and $1/6$ neutral count. This indicates that although `Practical` achieves peak performance, it is also unstable across configurations. The `Retrace-`λ strategy achieves $2/6$ best case, $0/6$ worst case and $4/6$ neutral. These results indicate that `Retrace-`λ offers overall more stable performance compared to the `Base` and `Practical` mitigation strategies.

Compute efficiency gains from off-policy roll-outs Following the theoretical analysis on the complexity, the efficiency gains translate into linear trends on the total runtime, and on the token generation speed. This trend is observed in all the settings covered (see Figures 1b, 1c and 1a). Specifically, the efficiency gain on both metrics increases linearly with the ratio $x\%$ of skipped layers.

References

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

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

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.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature machine intelligence*, 5(3):220–235, 2023.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layerskip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019.

Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI conference on artificial intelligence*, 2023.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*, 2024.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems, 2024. URL `https://arxiv.org/abs/2402.14008`.

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, 2021. URL `https://arxiv.org/abs/2103.03874`.

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. *arXiv preprint arXiv:2106.09685*, 2021.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. Vineppo: Refining credit assignment in rl training of llms, 2025. URL `https://arxiv.org/abs/2410.01679`.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! *arXiv*, 2019.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

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):9, 2024.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*, 2024.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

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*, 3:2, 2024.

Alberto Maria Metelli, Matteo Papini, Nico Montali, and Marcello Restelli. Importance sampling techniques for policy optimization. *Journal of Machine Learning Research*, 21(141):1–75, 2020.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning, 2016. URL `https://arxiv.org/abs/1606.02647`.

Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Seungyeon Kim, Neha Gupta, Aditya Krishna Menon, and Sanjiv Kumar. Faster cascades via speculative decoding. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=vo9t20wsmd`.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems?, 2021. URL `https://arxiv.org/abs/2103.07191`.

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.

Nicolas Le Roux, Marc G Bellemare, Jonathan Lebensold, Arnaud Bergeron, Joshua Greaves, Alex Fréchette, Carolyne Pelletier, Eric Thibodeau-Laufer, Sándor Toth, and Sam Work. Tapered off-policy reinforce: Stable and efficient reinforcement learning for llms. *arXiv preprint arXiv:2503.14286*, 2025.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Taiwei Shi, Yiyang Wu, Linxin Song, Tianyi Zhou, and Jieyu Zhao. Efficient reinforcement finetuning via adaptive curriculum learning, 2025. URL `https://arxiv.org/abs/2504.05520`.

Akiyoshi Tomihari and Issei Sato. Understanding linear probing then fine-tuning language models from ntk perspective, 2024. URL `https://arxiv.org/abs/2405.16747`.

Betty van Aken, Benjamin Winter, Alexander Löser, and Felix A. Gers. How does bert answer questions?: A layer-wise analysis of transformer representations. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 1823–1832. ACM, November 2019. doi: 10.1145/3357384.3358028. URL `http://dx.doi.org/10.1145/3357384.3358028`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *In Proc. NeurIPS*, 30, 2017.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`, 2020.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45. Association for Computational Linguistics, 2020. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *arXiv preprint arXiv:2401.07851*, 2024.

Heming Xia, Yongqi Li, Jun Zhang, Cunxiao Du, and Wenjie Li. Swift: On-the-fly self-speculative decoding for llm inference acceleration, 2025. URL `https://arxiv.org/abs/2410.06916`.

Tengyang Xie, Yifei Ma, and Yu-Xiang Wang. Towards optimal off-policy evaluation for reinforcement learning with marginalized importance sampling. *Advances in neural information processing systems*, 32, 2019.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft& verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 11263–11282. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.607. URL `http://dx.doi.org/10.18653/v1/2024.acl-long.607`.

Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data, 2025. URL `https://arxiv.org/abs/2412.07762`.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences, 2020. URL `https://arxiv.org/abs/1909.08593`.

A Related Works

B Related works

`Nested-ReFT` is connected to the literature on *speculative decoding* [Xia et al., 2024], as the idea of using smaller nested models exists in that literature [Elhoushi et al., 2024, Zhang et al., 2024, Xia et al., 2025]. The main difference is that nesting was explored to accelerate inference at test time, while we instantiate nesting at train time. We will see in the theoretical analysis and empirical evaluation that transposing the nesting idea at train time brings a set of new unique challenges. Note that the proposed technique employs a depth wise nesting approach (layer skipping), but other width-wise nesting techniques of transformer layers exist as well Narasimhan et al. [2025]. This work also differs from Roux et al. [2025] where off-policyness is formulated as the collection of positive and negative samples to improve learning performance. In contrast, we aim to find a cost effective behavior policy while maintaining stability affected by the degree of off-policyness.

Parameter efficient fine-tuning (PEFT) Parameter efficient fine-tuning [Fu et al., 2023, Ding et al., 2023] consists in adapting only a subset of parameters. Low-rank (LoRA) adaptation [Hu et al., 2021] and its variants [Liu et al., 2024, Hayou et al., 2024] optimize training efficiency through the number of flops. Similarly, linear probing consists in optimizing the fine-tuning efficiency by restricting the parameter updates to the last layer of the LLM Tomihari and Sato [2024]. The proposed work is orthogonal to PEFT because we improve compute efficiency while all the parameters of the target LLM are updated.

Reinforcement learning for LLMs Luong et al. [2024], Kool et al. [2019], Schulman et al. [2017] propose frameworks but the limitation of the proposed frameworks is compute and memory cost. Dropping the reward model as in DPO Rafailov et al. [2023] and KTO Ethayarajh et al. [2024] can mitigate the memory overhead. However, in the specific problem of ReFT the reward model is a heuristic function, and the computational overhead is due to the behavior policy generating completions. To our knowledge, there is no work that aims to improve this completion generation cost in ReFT. More broadly in reinforcement fine-tuning (e.g. RLHF Bai et al. [2022]) efficiency is addressed from a data selection perspective Zhou et al. [2025], Shi et al. [2025] while the proposed framework addresses algorithmic variations for improved completion generation efficiency.

C Algorithm

Algorithm 1 `Nested-ReFT`

Require: Target model π_θ (LLM), Dataset $\mathcal{D}$, reward function $R(x, y)$, skip ratio $x \in (0, 1)$, SFT epochs E_{sft}, RFT gradient steps S, border parameter b, choice of stabilization method $h_m(\cdot, \cdot), m \in \{\texttt{base}, 1, \lambda\}$

Step 1: Supervised Fine-Tuning (SFT)
for $e = 1$ **to** E_{sft} **do**
  Train π_θ on $(x, y^{cot}) \sim \mathcal{D}$ using cross-entropy loss
end for
Step 2: Reinforced Fine-Tuning (ReFT)
for $s = 1$ **to** S **do**
  Sample batch of prompts
  Set $\eta_s = \pi_{\theta_{s-1}}$
  Sample skip set with $f_{b,x}(\eta_s)$
  Deactivate layers in η_s using $f_{b,x}(\eta_s)$ to get η'_s
  Generate G samples for each x in batch using η'_s
  Score samples using reward model
  Compute stabilization $h_m(\eta', \pi_\theta)$
  Update π_{θ_s} with rewards and $h_m(\eta'_s, \pi_{\theta_s})$ using RL objective
  Set $\pi_{\theta_{s-1}} = \pi_{\theta_s}$
end for
return π_{θ_S}

D Analysis of Nested-ReFT

D.1 Theoretical speed-up

Consider a behavior model η that contains T_η identical transformer layers. Each layer has a computational complexity that can be expressed as:

$$\mathcal{C}_{layer} = O\left(L^2 d + L d^2\right),$$

where L is the generated sequence length and d is the hidden dimension (i.e., width) of the layer [Vaswani et al., 2017].

Property 1 (Complexity with Layer Skipping)**.** *Given a model η with T_η transformer layers, if we skip U_x layers, the computational complexity of a nested model η' is:*

$$\mathcal{C}_{\eta'} = O\left((T_\eta - U_x) \times \mathcal{C}_{layer}\right).$$

The complexity of the inference or the forward-pass of η' is reduced proportionally to the number of skipped layers U_x through the skip ratio $x\%$. Assuming fixed generation length L and hidden dimension d, the layer skipping achieves a linear complexity improvement.

D.2 Unbiased Convergence on the bounded off-policy

In reinforcement learning, we seek to optimize a policy π_θ using gradient-based updates. However, direct sampling from π_θ is not always feasible, so we employ a behavior model η for an off-policy update.

Unlike game RL environments [Brockman et al., 2016] where the states are independent from the policies, tasks like language generation involve generating the next token conditioned on a prompt and all the previous tokens. Therefore, the action is the prediction of the next token y_ℓ and the state is $s_\ell = (x, \hat{y}_{<\ell})$. Inducing exploration is non-trivial as it requires preserving the structural consistency of the state.

Hence, we opt for an ensemble of nested behavior policies $\mathcal{Z} = \{\eta'_i\}_{i=1}^{|Z|}$ to generate off-policy updates of π_θ, where $|\mathcal{Z}| = S$ in `Nested-ReFT`. To estimate the policy gradient, we use importance sampling with the weight $h^i_{\texttt{base}}(y_\ell, s_\ell; \pi, \eta'_i)$. The behavior policy is selected uniformly from $\mathcal{Z}$, leading to an ensemble-weighted objective. Defining the mean behavior policy over the ensemble $\mathcal{Z}$ as:

$$\bar{\eta}_{\mathcal{Z}}(\hat{y}_\ell|s_\ell) = \frac{1}{|\mathcal{Z}|}\sum_{i=1}^{|\mathcal{Z}|} \eta'_i(\hat{y}_\ell|s_\ell), \tag{1}$$

the expected objective can be rewritten in terms of $\bar{\eta}_{\mathcal{Z}}$, ensuring stable updates as long as the importance weights remain bounded.

Suppose, we are interested in the policy gradient updates using advantage ($A_\ell^{\lambda,\gamma,\pi}$) (referred to A^π for brevity) estimation based on discounted λ-returns. Then, the advantage estimation for the target policy (π) is estimated as:

$$A^\pi(s, y) = Q^\pi(s, y) - V^\pi(s)$$

To estimate the policy gradient update we compute the derivative of the expected advantage objective ($\mathcal{J}$):

$$\mathcal{J} = \max\ \mathbb{E}_\pi\left[A^\pi\right] \tag{2}$$

$$\nabla\mathcal{J} = \nabla\mathbb{E}_\pi[A^\pi] \tag{3}$$

$$= \nabla \sum_{\ell=0}^{L} A^\pi(s_\ell, \hat{y}_\ell) \cdot \pi(\hat{y}_\ell|s_\ell) \tag{4}$$

$$= \nabla \sum_{\ell=0}^{L} A^\pi(s_\ell, \hat{y}_\ell) \cdot \pi(\hat{y}_\ell|s_\ell) \cdot \frac{\eta'_i(\hat{y}_\ell|s_\ell)}{\eta'_i(\hat{y}_\ell|s_\ell)} \tag{5}$$

$$= \nabla \sum_{\ell=0}^{L} A^\pi(s_\ell, \hat{y}_\ell) \cdot \eta'_i(\hat{y}_\ell|s_\ell) \cdot \frac{\pi(\hat{y}_\ell|s_\ell)}{\eta'_i(\hat{y}_\ell|s_\ell)} \tag{6}$$

$$= \nabla \sum_{\ell=0}^{L} A^\pi(s_\ell, \hat{y}_\ell) \cdot \eta'_i(\hat{y}_\ell|s_\ell) \cdot h^i_{\texttt{base}}(\hat{y}_\ell|s_\ell; \pi, \eta'_i) \tag{7}$$

Then, the objective for a $\eta'_i \in \mathcal{Z}$ can be re-written as,

$$\mathcal{J} = \max\ \mathbb{E}_{\eta'_i}[h^i_{\texttt{base}} \cdot A^{\eta'_i}]$$

Theorem 1. *(Convergence of Policy Gradient with Ensemble Behavior Policies) Let,*

1. *The importance weights $h^i_{\mathtt{base}}(\hat{y}_\ell|s_\ell)$ are bounded,*
2. *The learning rate sequence over gradient steps s, α_s satisfies $\sum_s \alpha_s = \infty$ and $\sum_s \alpha_s^2 < \infty$, and*
3. *The behavior policy ensemble $\mathcal{Z}$ ensures sufficient exploration.*

With these assumptions, the policy gradient update using an ensemble of behavior policies converges to an optimum off-policy update from the expected advantage function weighted by the mean behavior policy $\bar{\eta}_{\mathcal{Z}}$.

Proof. The goal is to show that the using an ensemble of behavior policies for off-policy updates converges to the mean policy of the set, and the variance of the updates is controlled by a bounded measure on the importance ratio.

Step 1: Unbiasedness of the Gradient Estimate We start by rewriting the objective:

$$\mathcal{J}^{\mathcal{Z}} = \max \mathbb{E}_{\eta_i' \sim \mathcal{Z}}[h^i_{\mathtt{base}} \cdot A^{\eta_i}]. \tag{8}$$

Since the behavior policy is selected uniformly at random, we can express this expectation as:

$$\mathbb{E}_{\eta_i' \sim \mathcal{Z}}[h^i_{\mathtt{base}} \cdot A^{\eta_i'}] = \sum_{i=1}^{|Z|} p^i \cdot \mathbb{E}_{\eta_i'}[h^i_{\mathtt{base}} \cdot A^{\eta_i'}], \tag{9}$$

where $p^i = \frac{1}{|Z|}$. Substituting the definition of w^i, we get:

$$\sum_{i=1}^{|Z|} \mathbb{E}_{\eta_i'} \left[\frac{h^i_{\mathtt{base}}}{|Z|} A^{\eta_i'} \right]. \tag{10}$$

Rewriting as a sum over the sequence steps:

$$\sum_{i=1}^{|Z|} \sum_{\ell=0}^{L} h^i_{\mathtt{base}} \cdot A^{\eta_i'}(s_\ell, \hat{y}_\ell) \cdot \frac{\eta_i'(\hat{y}_\ell|s_\ell)}{|Z|}. \tag{11}$$

By using the definition of the mean behavior policy, the equation simplifies to

$$\sum_{t=0}^{T} c \cdot A^{\bar{\eta}_{\mathcal{Z}}}(s_t, \hat{y}_t) \cdot \bar{\eta}_{\mathcal{Z}}(\hat{y}_t|s_t). \tag{12}$$

Since $\bar{\eta}_{\mathcal{Z}}$ is the expectation over the behavior policies, the modified objective to update the target policy (π) with an ensemble of behavior policies $\mathcal{Z}$ is unbiased.

Step 2: Bounded Variance The importance sampling ratio influences the policy gradient update:

$$\text{Var}\left(h^i_{\mathtt{base}} \cdot A^{\eta_i'}\right). \tag{13}$$

Since $h^i_{\mathtt{base}}$ is the ratio between the log-probabilities of both policies, the variance depends on how different η_i' is from π. Approaches like TB(λ), Retrace(λ), Off-policy Q(λ) have explored the variance minimization through bounding the off-policyness of the behavior policy [Munos et al., 2016]. The scale c can be bounded with h^i_1, or h^i_λ. The assumption that h^i_{base} is bounded ensures that:

$$\text{Var}(h^i_{\mathtt{base}}) \leq c < \infty. \tag{14}$$

This ensures learning stability.

□

E Architecture detail

Model	N	W	Skipped layers at ratio x		
			5%	10%	15%
Qwen2.5-1.5B	28	1536	1	3	4
Qwen2.5-7B	28	3584	1	3	4

Table 1: Skipped layers for various ratios x on Qwen2.5-Math-Instruct. L = # of layers, W = hidden layer width.

F Experiment details

Training generation details We consider $E_{\text{sft}} = 2$ epochs for the SFT warm-up stage, similarly to [Luong et al., 2024]. The β parameter of GRPO is set to 0, implying no KL penalty is used, following emerging evidence that the extra compute brought by the reference model is optional [Roux et al., 2025]. The batch size is set to 16 for all models sizes, using gradient accumulation. We consider $S = 99$ gradient steps for ReFT, this corresponds to $E_{\text{rft}} = 1$, $E_{\text{rft}} = 0.11$, and $E_{\text{rft}} = 0.07$ for SVAMP, GSM8k and Math12k datasets, respectively. Fractions of epoch imply that a proportional subset of the shuffled dataset D is used for fine-tuning. This allows for fair cross-dataset and model comparisons. The prompts are formatted using a Qwen-chat template commonly used by practitioners [Liu et al., 2025]. For the behavior model, we set the minimum and maximum length of the generated completions to to 256 tokens. This implies that all the completions have equal length. For training, all the other parameters follow the default from GRPO TRL library [von Werra et al., 2020].

Evaluation generation details For evaluation, we use a math reasoning benchmark composed of 5 datasets [Liu et al., 2025].The temperature is set to 0.6, the top-p to 0.95 and the maximum number of tokens to 32k. We perform pass@K with $K = 1$, implying the model generates 1 response per problem. This corresponds to a strict setup as the model is only given one single chance to answer correctly.

G Additional results

H Discussion

Our controlled experiments show that it is possible to train smoothly, even when the degree of off-policyness increases. The influence on performance of `Nested-ReFT` has limited impact on performance. These results are achieved for fixed size generation for the behavior model. However, an increasing number of LLMs can produce adaptive responses, either short for simple problems or long for complex problems. The interaction between generating completion off-policy through layer skipping and its influence on the completion length is on open research problem. Furthermore, the nesting strategy (e.g. layer skipping) may have non uniform interaction effects on the generation length depending on the dataset and model scale. This suggests that increasing off-policyness with a learned strategy rather than a heuristic based approach based on layer skipping may handle the interactions more effectively.

Model	Instance	SVAMP	GSM8k	Math12k	Δ_{abs}
1.5B	Best	+0.008	+0.015	+0.026	0.016
	Worst	−0.016	−0.006	−0.005	0.009
7B	Best	+0.022	+0.010	−0.003	0.009
	Worst	−0.070	−0.017	−0.022	0.036

Table 2: Best and worst observed deltas per dataset and model size. Δ_{abs} is the mean absolute change to baseline.