Extended Abstract

Motivation Recent advancements have seen LLMs making large strides in performance against math and programming benchmarks via reasoning-oriented reinforcement learning algorithms like GRPO. Despite a clear overall effectiveness, the implications of some of the established algorithmic parameters are poorly understood and warrant further study. In this work we seek to profile different exploration strategies during the RL loop at the small language model scale in the hopes of improving performance or accelerating convergence, furthering the reasoning model frontier.

Method We apply a round of supervised fine-tuning to Qwen/Qwen2-0.5B-Instruct on a mathematical puzzle dataset. We use the resulting checkpoint as the starting point for various rounds of reinforcement fine-tuning. We train with LoRA and calculate two reward components, one for format adherence and another for puzzle correctness. The correctness reward is weighted 10:1 relative to the format reward. For the RL we train with GRPO, using a slightly modified loss formulation that mitigates length bias. Hyperparameters are set with the intention to reduce variance in our gradients and per-step rewards (more generations per instance, larger effective batch size). For exploration techniques we implement bottom_p sampling, forcing our model to sample less likely trajectories, as well as an element-wise temperature control for more flexible exploration.

Implementation We start with the base implementations from the trl and transformers open source packages. We integrate our custom exploration strategies, as well as any extra methods made in an effort to improve training stability, as extensions of these codebases. During evaluation we check correctness only, sampling five completions at low temperature per example and averaging the results.

Results As a baseline, we evaluate the untrained Qwen/Qwen2-0.5B-Instruct model against our holdout set and get 0% accuracy. After our supervised fine-tuning on a mix of public and synthesized data, we bring this up to 38%. Just by varying temperature during RFT we make it up to 42.2% accuracy. Sustained accuracy gains are difficult to achieve in the face of reward drops and training instability. Both bottom_p and adaptive temperature controls are unable to beat our best naive temperature configuration ($t = 1.5$)

Discussion The principal weakness of all our sampling procedures is they are purely random according to the distribution of the model. Bottom_p may more easily unlock lower probability trajectories, but there is no guarantee these trajectories are desirable according to our reward schemes. A better approach would be one that guides the sampling according to some reward heuristic. We do error analysis and stratify incorrect examples by the number of input integers and operation types, finding that the model struggles more with the examples that require increased exploration (four numbers, rather than three) – but a nontrivial number are incorrect with three numbers, and for easier operations like addition and multiplication. Training stability gets in the way of sustained performance improvements for both vanilla and custom RFT configurations; we implement several strategies to combat this such as pass thresholding, but nothing yields particular performance gains.

Conclusion While better exploration does help yield better accuracy as evidenced by our temperature sweep, surpassing this bar via more advanced exploration techniques is unsuccessful. Instability and model capacity are core issues, but so too is the fundamental flaw that temperature-based sampling and its variants do not account for rewards or the task parameters explicitly. Future work may benefit from training with more capable models and a more guided sampling approach.

Exploration Strategies for Reasoning Fine-tuning

Nick Mecklenburg
Department of Computer Science
Stanford University
nmecklen@stanford.edu

Abstract

In this work we study the application of different exploration strategies during reasoning fine-tuning at the small language model scale in the hopes of improving performance or accelerating convergence. Between element-wise adaptive temperature and constraining sampling to leverage bottom and middle distribution tokens, our experimental strategies do not yield significant gains. We are, however, partly dragged down by training instability, which seems to be pervasive at the small model scale for our nontrivial reasoning task with binary sparse rewards.

1 Introduction

Scaling the test-time budget for LLM inferencing to allow for long, rigorous chains-of-thought has created a new frontier of model performance on reasoning-intensive tasks. AI is making unprecedented advancements in previously unthinkable areas like competition math and professional-level coding; furthering these advancements afford a number of societal benefits, like the overall acceleration of research and human progress. At the heart of this movement lies RL-based reasoning fine-tuning (RFT) as first openly published by DeepSeek-AI et al. (2025).

Despite the field's momentum, as the RFT algorithm is recent, the implications of some of its algorithmic parameters are poorly understood. DeepSeek outlines a method that interweaves phases of supervised fine-tuning (SFT) and RFT, with RFT powered by group relative policy optimization (GRPO), its exploration relying on high temperatures to naively explore the action space. The criticality of GRPO and the optimality of the hyperparameters and precise loss formulation used therein are open questions; in this project we aim to explore one such topic, the sampling strategy that underlies the exploration of the RL procedure, with the goal of assessing whether different sampling strategies may unlock better model performance and/or faster convergence in RFT.

In particular we constrain our study to work at the small language model scale (on the order of 0.5B parameters). In this regime of less powerful, more compact models, we are not able to rely on the pretrained knowledge corpus as extensively to power our RL gains, so the quality of our exploration becomes all the more critical as we must work harder to find quality, correct completions to push up in probability. Empowering small models is not contrived, either; small LLMs are being embedded locally on devices like phones and laptops, such as in the cases of Apple Intelligence (Apple, 2024) and Phi Silica (Microsoft, 2025). Pushing the performance frontier at this scale thus has large real world consumer implications.

2 Related Work

The principal work motivating this study is DeepSeek-AI et al. (2025). The authors take a powerful, vanilla "non-reasoning" LLM and apply:

1. supervised fine-tuning on cold-start data to encourage the generation of chains-of-thought

2. GRPO-based reinforcement-learning to improve the model's reasoning through self-evolution powered by function-based reward signals
3. rejection sampling and supervised fine-tuning to improve generalization
4. preference-focused RL for safety and alignment

This work focuses on the RL process executed in step (2). In particular the authors rely on the randomness inherent to high temperature sampling to explore the space of reasoning traces during training. Random sampling may break down for harder tasks that are more out-of-distribution for the model, and the model's ability to improve on these tasks may also be reliant on the reward design, where sparser reward schemes paired with insufficient exploration result in poor or no learning.

There have been a few studies that improved upon the DeepSeek R1 baseline, with Liu et al. (2025) discovering and removing a length bias in the loss, and Yu et al. (2025) applying a token-level policy-gradient loss in combination with higher clipping and dynamic sampling. Both saw nontrivial improvements in training performance. These efforts do not focus extensively on the exploration strategies, however.

In this vein, other relevant literature lies in the exploration strategies of general deep learning settings. In surveying this topic, Ladosz et al. (2022) breaks exploration into efficiency and safe-exploration subcategories; our sample efficiency motivation places us in the efficiency bucket, wherein exploration may be further divided into imitation-based (Vecerik et al., 2018; Hester et al., 2017), planning (Ecoffet et al., 2021; Gangwani et al., 2019), intrinsic reward (Schmidhuber, 2010), and random methods (Patrascu and Stacey, 1999; Rückstieß et al., 2010). We note per this framing that DeepSeek's cold-start SFT constitutes a type of imitation-based exploration to facilitate faster discovery of the chain-of-thought paradigm to improve reasoning (i.e., the methodology is not completely random end-to-end). We find intrinsic reward and random methods most promising for integration and further study.

Ladosz et al. (2022) cites parameterized exploration methods (such as ϵ-greedy) and network parameter perturbation methods as two main approaches within the random bucket. Purely random exploration is not suitable for reasoning training as our generations must both still be intelligible english and adhere to the chain-of-thought format we steered for in the cold-start SFT phase – making network parameter perturbation methods a poor fit for their penchant for uncontrolled ablation.

Within intrinsic reward methods, Ladosz et al. (2022) creates several other subcategories that broadly fit into either rewarding novel states or behaviors during exploration. Sampling during RFT with high temperature encourages novelty but does not reward for it; making algorithmic tweaks in this direction warrants further study.

3 Method

At the 0.5B model scale, we choose Qwen/Qwen2-0.5B-Instruct as our base model for all experiments (Yang et al., 2024). Other candidates were available, such as Qwen/Qwen2.5-0.5B-Instruct, which the authors note enjoys "significantly more knowledge and has greatly improved capabilities in coding and mathematics" (Qwen-Team, 2024, 2025), but for the sake of starting from a harder model and seeing to what extent exploration strategies can compensate for pretraining and base model capacity, we opt to use the earlier, less performant version.

Regarding training methodology, we apply one round of SFT on a dataset with ground truth completions for the task at hand, followed by a round of RFT. We leverage the open source trl (von Werra et al., 2020) and transformers (Wolf et al., 2020) packages for our base implementations and make surgical customizations in the code as necessary, rebuilding the (modified) libraries from source to run in our experiments. We leverage naive `model.generate()` syntax for sampling as we only have access to a single A100 40G GPU on Google Colab; there is not rich support for sharing a single GPU between a vllm server and the training runtime as of the time of writing (though a new colocate mode looks to be in the works).

For the RL algorithm, we follow DeepSeek's example and run GRPO. The trl implementation uses a modified version of the GRPO loss that mitigates the original's length bias which "prefer[s] shorter completions with positive advantages and longer ones with negative advantages" (HuggingFace, 2025).

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{\sum_{i=1}^{G}|o_i|}\sum_{i=1}^{G}\sum_{t=1}^{|o_i|} l_{i,t} \tag{1}$$

$$l_{i,t} = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{[\pi_{\theta_{old}}(o_{i,t} \mid q, o_{i,<t})]_{\text{no grad}}}\hat{A}_{i,t} - \beta D_{KL}\left[\pi_\theta \,\|\, \pi_{\text{ref}}\right]$$

As we will discuss further below, we sweep over hyperparameters extensively, varying temperature, learning rate, batch size, gradient accumulation, peft vs non-peft, lora rank, and more.

With respect to algorithm tweaks, we mostly find ourselves in a battle between exploration and training stability. To improve exploration we consider constraining our sampled tokens to the bottom $p\%$ cumulative probability at some frequency to force steer trajectories away from what may be a suboptimal trajectory strongly encouraged from an overfitted SFT baseline. We also implement an adaptive exploration via element-wise temperature control. This encourages more exploration for examples the baseline SFT checkpoint got wrong and more exploitation for those it got right.

On the stability front, one lever we pull is vanilla curriculum learning, filtering out harder examples so that we train on easier ones first. We identity easy vs. hard examples based on whether the SFT checkpoint gets them right or wrong. This approach lowers the burden of our exploration, at least initially. We also mask completions in our loss that were truncated as these may have been good trajectories that we would otherwise want to drive up the probability of – they just happened to be too long to finish. A final stability improvement we tried is implementing reward thresholding – we only apply gradient updates for minibatches with mean grouped rewards above some pass threshold. This is a more rigorous form of our curriculum learning where we filter out hard examples dynamically during training, ensuring we only focus on higher quality samples and gradient updates during our training procedure.

4 Experimental Setup

For our datasets, first we start with the math reasoning task from PrimeIntellect's SYNTHETIC-1 (Mattern et al., 2025). This dataset consists of long-form complex math reasoning word problems and our 0.5B base model scores 3.8% on it in terms of accuracy. After rich hyperparameter exploration and tens of thousands of rows of SFT we bump to 4.1% accuracy, with RFT degrading to 3.4% (due to a length penalty causing more and more succinct generations as the only stable signal throughout training); ultimately we deem the dataset too hard and retreat to a simpler – but still nontrivial – reasoning task, specifically the countdown dataset.

The countdown dataset consists of a series of mathematical puzzles, where given three or four integers and some target integer, we must come up with an equation leveraging simple arithmetic operations (+, -, *, /) to reach the target, using the given numbers at most one time each. We show an example in figure 1 below.

Prompt: A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer.\nUser: Using the numbers [32, 4, 57, 19], create an equation that equals 24. You can use basic arithmetic operations (+, -, *, /) and each number can only be used once. Show your work in <think> </think> tags. And return the final answer in <answer> </answer> tags, for example <answer> (1 + 2) / 3 </answer>.\nAssistant: Let me solve this step by step.
Completion:<think>Let's analyze the numbers and the target:\n\nNumbers: 32, 4, 57, 19 \nTarget: 24\n\nStep 1: Consider simple operations with 4 first since 4 is a divisor of 32.\n\n32 / 4 = 8\n\nSo now, from the numbers, we get 8, still need to get to 24.\n\nStep 2: 8 * 3 = 24, so can we form 3 from 57 and 19 by some operation?\n\n57 - 19 = 38 (too big) \n57 / 19 ≈ 3 (specifically 3 since 19 * 3 = 57) \n\nWait, 57 / 19 = 3 exactly.\n\nSo:\n\n(32 / 4) * (57 / 19) = 8 * 3 = 24\n\nThis satisfies the target.\n\nFinal formula: (32 / 4) * (57 / 19)\n\n</think><answer>(32 / 4) * (57 / 19)</answer>

Figure 1: Example instance from our supervised fine-tuning countdown dataset.

We accumulate around 11,000 examples for the SFT set, with 1,000 coming from a HuggingFace dataset (Asap7772, 2025) and the other 10,000 being synthesized. For the synthesis, we generated random integer arithmetic expressions ranging from three to four operands that yielded some integer result, with the source numbers ranging between 1 and 100 and the target being less than 2000. We query o4-mini (OpenAI, 2025b) for our reference completions.

For our RFT dataset, we take a subset of 2,000 examples from Pan (2025) on HuggingFace. We preshuffle the dataset and set `shuffle_dataset=False` in our trainings for consistency between runs. For the validation dataset, we take the two hundred holdout examples from Asap7772 (2025) and concatenate this with another three hundred from our synthetic generation procedure, though here o4-mini need not be queried as we do not use reference completions during grading. We apply deduplication between this validation dataset and our SFT and RFT datasets to ensure no data incest will inflate our results. Deduplication is done based on the sorted numbers `(source num 1, source num 2, ... target num)`.

In terms of reward design and weighting, we stick to two binary reward components – one for task correctness and another for output formatting. The output formatting reward ensures we adhere to the <think></think><answer></answer> template with a weight of 0.1 in the final reward calculation. The task correctness is 1.0 if the model's expression uses only the provided numbers, each at most once, and evaluates to the target integer – else the reward is 0.0. Note that due to the commutative property of some of our arithmetic operators, there is more than one correct expression, which is why it makes sense only to grade the final result. The correctness reward has a weight of 1.0 (i.e., 10x the format weight).

For both training and evaluation, we limit the model to a maximum completion length of 1,024 tokens. During training we enforce a max prompt length of 256 tokens. When evaluating model performance on the holdout set, we sample five generations with `top_k=20`, `top_p=0.8`, `temperature=0.3`, and `repetition_penalty=1.1`; we then take the average correctness over the generations (i.e., we do not assess format adherence, since this is primarily added as a stability aid).

5 Results

5.1 A Quest for a Baseline

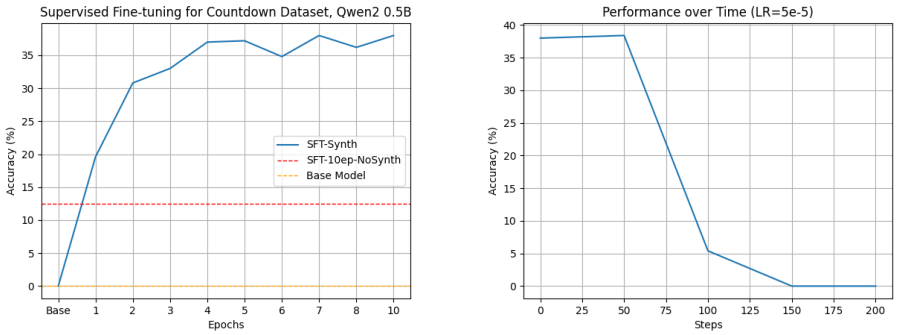
Our plan of action was to first demonstrate some nontrivial performance gain via naive GRPO then experiment with the exploration strategies to understand whether they led to performance gains, drops, or ties, perhaps with faster or slower convergence. The first part of this effort entailed applying supervised fine-tuning, since our base model scored a whopping 0% on our validation dataset.

As shown in figure 2a, the lower capacity Qwen/Qwen2-0.5B-Instruct model is only able to improve to 12.5% after 10 epochs of training on the HuggingFace SFT dataset from Asap7772 (2025). Augmenting this with our synthetic data and training with 10 epochs steadily brings us up to around 38% accuracy.

We use this 38% SFT checkpoint as our initial weights for reinforcement fine-tuning. Immediately we find that with too short a max context length or too high a learning rate our reward quickly collapses. This is because in the case of context length we truncate desirable trajectories, then push their probabilities down, and in the case of learning rate our oscillatory gradient updates quickly destabilize our policy and performance degrades to 0, as in the case of figure 2b. From these lessons we establish a minimum viable context length of 1024 and a learning rate cap of about 5e-6.

Using LoRA allows us to run with larger batch sizes and a higher number of generations given the same resources, so we apply it under the intuition that this will decrease the variance of our training via higher quality gradients. We sweep the LoRA rank but do not see extensive gains from increasing beyond rank 8, so we fix this for all future experimentation.

An observation we make is the model has a much easier time picking up on the formatting reward component than the accuracy reward component, as seen in figure 3. This begs the question of why, after 10 epochs of training and a (relatively) strong 38.0% correctness score, we see such low formatting rewards to begin with, as we start around 0.1 (scale is 0 to 1). We include an example completion from the model in figure 4; the model struggles with following the prompt's instructions, which requests a format of <think></think><answer></answer> and includes a reference. Note we



(a) (b)

Figure 2: **LEFT:** Performance of SFT at different epochs, with the orange dotted line denoting the Qwen2-0.5B-Instruct base model accuracy and the red dotted line denoting the accuracy of SFT after ten epochs of training on just the HuggingFace warm-start dataset with no synthetic examples from o4-mini. **RIGHT:** An example of one of the many runs that exhibited reward collapse over the duration of training, this one due to an overaggressive learning rate. Reported accuracy is on the validation dataset.

start the model's response out with an opening <think> tag to help guide it onto the distribution it learned from SFT. In particular we see multiple closing think tags, and while not in this instance, sometimes extraneous opening think tags are included as well. But it more consistently includes the <answer></answer> tag enclosing its candidate expression, perhaps because this is what the more heavily weighted correctness reward relies on.

Figure 3 leveraged a lower learning rate of 5e-7, a batch size of 16, and four generations per training instance. The lack of gains motivates a higher learning rate, and a few more rounds of iterations show learning rate 5e-6, batch size 16, and 16 generations per instance helps us sustain some improvements for the first hundred steps of training, which replicates across trials – this is shown in figure 5a. These improvements are matched with a corresponding bump in performance on the holdout set, where we peak at 42.2% across our checkpoints in the first 200 steps.

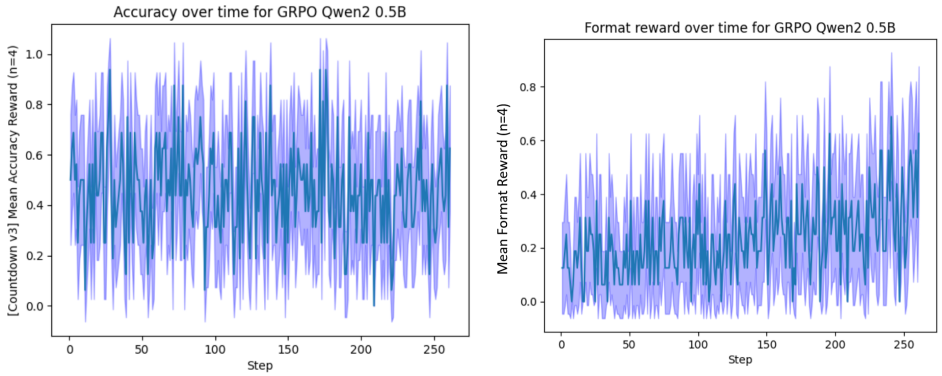
The reward drop in later iterations as shown in plot 5a inspires a number of further attempts, where we experiment with increasing the effective batch size via gradient accumulation as well as the number of generations per step to decrease variance in our gradients and rewards, to no avail. Switching to a more aggressively decreasing learning rate schedule is not fruitful either. Still, at this point we have reproducible gains via RFT on the countdown dataset, however small they may be at about 4%, so here we shift gears to exploration strategies.

5.2 Exploration

Temp	Accuracy
0.3	38.4
0.6	41.0
0.8	41.4
1.5	43.0

Table 1: Temperature vs. validation accuracy at 100 steps for our stable configuration. Default was 0.6 for previous experiments.

Before getting into custom strategies, as a quick sanity check, we vary the temperature during our RFT training and do observe a general trend that as we increase the temperature (which equates to increased exploration in the completion-trajectory space), we see an improvement in validation set performance at the 100 step mark. This is shown in table 1. Once we increase too much further beyond 1.5, however, the correctness reward during training plummets to 0. Interestingly our validation accuracy is undisturbed despite the poor training performance in these scenarios, still hovering around 38%; we attribute this to the relative nature of our advantages. If all of our generations get 0 reward,



(a) (b)

Figure 3: Trend of correctness and formatting rewards over the duration of a training run. Unlike formatting, the correctness reward component struggles to see consistent gains. Blue is score and periwinkle is for standard error out of the batch of generations. This model's best checkpoint after 250 steps only ekes out one percentage point of improvement, taking us up to 39% accuracy.

Let's analyze the problem step-by-step:</think>\n\nWe have the numbers: 57, 76, 76 \nTarget result: 57\n\nSince we have two 76s, let's try to combine them using addition or subtraction:\n\n- 76 - 76 = 0\n- 76 / 76 = 1\n- 76 * 76 = very large (too big)\n- 57 + 0 = 57 (this works!)\n\nSo the expression is simply:\n\n57 + 0 = 57\n\nThis uses all three numbers exactly once.\n\n</think><answer>57 + 0</answer>

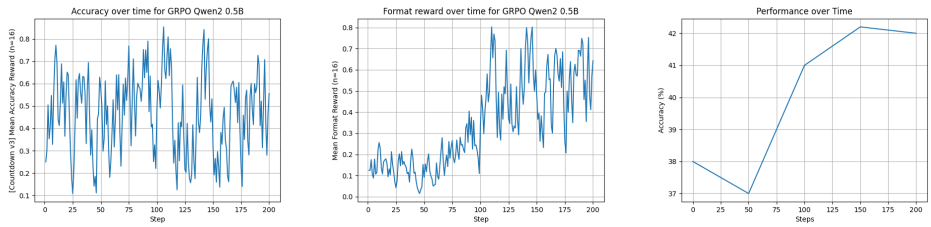
Figure 4: Example completion from model after RFT.

$r_i = \frac{1}{G}\sum_i r_i = 0$, and our advantages become 0 in turn, leading to gradient updates that only discourage deviating from our reference policy by the way of our KL penalty from equation 1 (which is lightly weighted with $\beta = 0.001$).

From here we implement bottom_p sampling. It is similar to the already popular top_p sampling, except instead of truncating the bottom (1-p) probability tokens, we do this to the top (1-p) with some frequency instead. The motivation behind this is we may have strongly weighted but undesirable trajectories that our naive temperature-based sampling may struggle to get past, even with temperature 1.0 or above. This may be due to the biases of the pretrained base model, overfitting during our SFT phase, or perhaps distributional differences in the datasets between RFT and other phases. We note this is composable with top_p sampling so as to enforce some flexible interval of probabilities (e.g., middle 80% of probabilities). In the not uncommon case of a heavily skewed distribution, say where there is no one token precisely in the middle range, we constrain each half of the truncation to leave at least one token to be sampled, with top_p truncation taking place first. Note also we apply these logit warpers after temperature is applied, so with higher temperatures the distributions are flatter, and the truncations have more of an effect.

In line with this, with experimentation we immediately notice that if the truncation is too aggressive or the temperature is too high, the model has trouble sampling completions with nonzero training reward, meaning we again only weakly optimize for KL and holdout performance does not change. We include two samples of this in the first and third rows of table 2. There is a middle range with bottom_p and temperature that samples good completions and scores nonzero rewards during training, shown in the middle row, though ultimately this suffers from the same reward drop and variance in later steps as our baseline and only drops in terms of holdout performance to 40.6% from our previous best of 42.2% from figure 5c.

A fundamental issue with bottom_p sampling, as with naive temperature sampling, is the exploration is purely random according to the distribution of the model. While bottom_p may more easily unlock lower probability trajectories that pure temperature-based sampling would otherwise struggle to uncover, there is no guarantee these trajectories are within the subset of ones applicable for our



(a) (b) (c)

Figure 5: Mean accuracy and format rewards per step with most stable configuration found after all hyperparameter sweeping. We include validation performance as evaluated every fifty steps on the rightmost graph.

Bottom P	Temperature	Train Reward @ 100
0.9	5.0	0.0
0.9	1.0	40.6
0.5	1.0	0.0

Table 2: Some sample experiments using the bottom_p strategy.

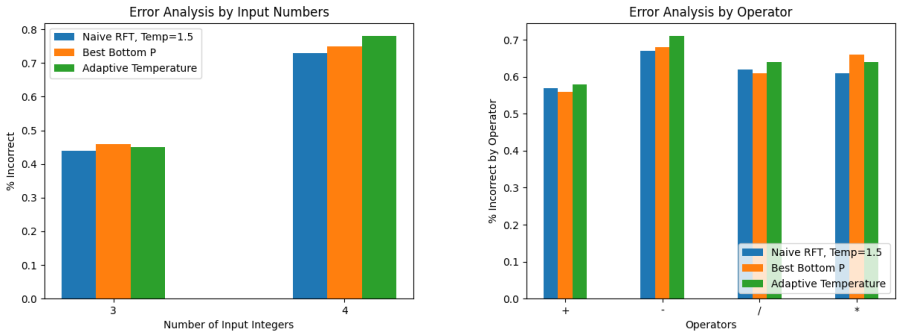
task that will uncover high reward. Discovering and pushing up the probability of a very unlikely trajectory may also require more than one gradient step of sampling and reinforcing that regime. bottom_p does not necessarily solve this core problem of task-blind search.

We do a brief error analysis comparing our best naive temperature sweep to bottom_p sampling as shown in figure 6. We plot relative error by number of input integers and unsurprisingly see the model struggles with four-integer-three-operator problems compared to three-integer-two-operator ones. We also notice the model finds addition easier and subtraction harder out of the operators. Our bottom_p exploration is not able to steer the model to discover any strategies that shift the relative error distributions away from the patterns of the temperature RFT baseline.

One observation we make across trials, with and without bottom_p, is that we consistently see drops in performance at around the same spots during training (steps 25, 45, and 95 for example); due to our consistent shuffle, these drops should correspond to the same examples from the training dataset each time. Seeing this, we try filtering out the examples our SFT checkpoint got wrong, including near the tricky regimes mentioned above, as a form of curriculum learning. This is done with the intent of removing the rough patches from our optimization and seeing more consistent gains. However, while our optimization trend does change in shape with drops no longer in the same place, the eventual reward decreases and destabilization remains, with our holdout performance doing no better than before.

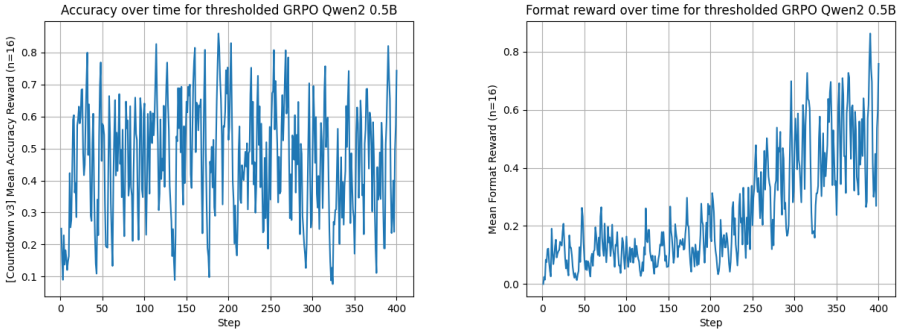
To aid optimization, we instead consider rather than naively sampling the same amount for all examples, we can adaptively explore more for hard problems and exploit more for easier ones, as proxied again by SFT performance per instance. For the easier examples (perhaps, our three number addition-rich examples) we use temperature 0.6 and for harder ones (for example, our four number subtraction-rich examples) we try 1.5. This yields no improvement, however, either in terms of stability or holdout performance (which drops), and we add a series to figure 6 to illustrate this.

It is unclear to us whether our performance degradations are wholly representative of deficiencies in our strategies, or whether unilateral instability with sustained training is getting in the way; we take figure 6 as only a weak negative signal. We try one last mitigation by implementing a pass_threshold in our training procedure. For all batches with mean rewards less than the pass threshold, we mask out that batch such that no gradient updates are made. This is inspired by the graders from OpenAI's reinforcement fine-tuning interface (OpenAI, 2025a). Conceptually we are forcing the model to find higher quality completions, which our exploration strategies may help uncover. Per figure 7, our training accuracy over time does come out much cleaner, still noisy but missing the sudden sustained drops we saw before. Yet, just because we threshold out bad completions does not mean the model stops producing them: our holdout performance is no better even after training on the entire 2k row



(a) (b)

Figure 6: Error analysis on our stable hyperparameter configuration. The model clearly struggles with more input integers. In terms of operators, it finds addition slightly easier and subtraction slightly harder, with multiplication and division about the same. Results for both plots are averaged over five trials.



(a) (b)

Figure 7: Mean accuracy and format rewards per step with pass threshold=0.3 and temperature 0.6.

dataset (with our same batch size 16, num generations 16, etc. parameters that saw small but sustained gains from before).

6 Conclusion

In this study we experimented with a number of approaches to improve exploration and training stability on Qwen/Qwen2-0.5B-Instruct. Increasing the sampling temperature during RFT does show signs of improved performance, with $t = 1.5$ outperforming $t = 0.3$ by five percentage points; this validates our intuition that better exploration facilitates better policy improvement. However, surpassing this performance bar via advanced exploration techniques is unsuccessful in the face of training stability issues and limited model capacity. Future work may consider replicating these results at higher parameter scales and/or with more powerful models to see if the burden of temperature-based exploration (and novel variants of it) is too great to bear when the representational capacity of the model is too low. Future work may also consider leveraging more guided exploration techniques that more readily emphasize areas of high predicted reward.

7 Team Contributions

- **Nick Mecklenburg** Solo project.

Changes from Proposal The main change is that we focused on Qwen/Qwen2-0.5B-Instruct exclusively. After other tasks proved challenging by the milestone, we were motivated to use the

dataset and model from the default project as it had been pre-vetted and we wanted to spend less time on building out a valid RL use case and more time on playing with exploration; unfortunately we confused Qwen**2**-0.5B-Instruct and Qwen**2.5**-0.5B-Instruct, and by the time we realized we chose the earlier, less performant Qwen2, we had too much data to consider abandoning it.

8 Code

`https://github.com/nmecklenburg/transformers/commits/nmeck/proj-changes`
`https://github.com/nmecklenburg/trl/commits/nmeck/proj-changes/`
`https://github.com/nmecklenburg/cs224r-proj-dump`

References

Apple. 2024. Introducing Apple's On-Device and Server Foundation Models. https://machinelearning.apple.com/research/introducing-apple-foundation-models Accessed: 2025-06-07.

Asap7772. 2025. cog_behav_all_strategies. https://huggingface.co/datasets/Asap7772/cog_behav_all_strategies Accessed: 2025-06-07.

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

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. 2021. Go-Explore: a New Approach for Hard-Exploration Problems. arXiv:1901.10995 [cs.LG] https://arxiv.org/abs/1901.10995

Tanmay Gangwani, Qiang Liu, and Jian Peng. 2019. Learning Self-Imitating Diverse Policies. arXiv:1805.10309 [stat.ML] https://arxiv.org/abs/1805.10309

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. 2017. Deep Q-learning from Demonstrations. arXiv:1704.03732 [cs.AI] https://arxiv.org/abs/1704.03732

HuggingFace. 2025. GRPO Trainer. https://huggingface.co/docs/trl/main/en/grpo_trainer Accessed: 2025-06-07.

Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. 2022. Exploration in deep reinforcement learning: A survey. *Information Fusion* 85 (Sept. 2022), 1–22. https://doi.org/10.1016/j.inffus.2022.03.003

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding R1-Zero-Like Training: A Critical Perspective. arXiv:2503.20783 [cs.LG] https://arxiv.org/abs/2503.20783

Justus Mattern, Sami Jaghouar, Manveer Basra, Jannik Straube, Matthew Di Ferrante, Felix Gabriel, Jack Min Ong, Vincent Weisser, and Johannes Hagemann. 2025. SYNTHETIC-1: Two Million Collaboratively Generated Reasoning Traces from Deepseek-R1. https://www.primeintellect.ai/blog/synthetic-1-release

Microsoft. 2025. Get started with Phi Silica in the Windows App SDK. https://learn.microsoft.com/en-us/windows/ai/apis/phi-silica Accessed: 2025-06-07.

OpenAI. 2025a. Graders. https://platform.openai.com/docs/guides/graders Accessed: 2025-06-09.

OpenAI. 2025b. Introducing OpenAI o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/ Accessed: 2025-06-07.

Jiayi Pan. 2025. Countdown-Tasks-3to4. https://huggingface.co/datasets/Jiayi-Pan/Countdown-Tasks-3to4 Accessed: 2025-06-07.

R. Patrascu and D. Stacey. 1999. Adaptive exploration in reinforcement learning. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339)*, Vol. 4. 2276–2281 vol.4. https://doi.org/10.1109/IJCNN.1999.833417

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

Qwen-Team. 2025. Qwen2.5-0.5B-Instruct. https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct Accessed: 2025-06-07.

Thomas Rückstieß, Frank Sehnke, Tom Schaul, Daan Wierstra, Yi Sun, and Jürgen Schmidhuber. 2010. Exploring Parameter Space in Reinforcement Learning. *Paladyn, Journal of Behavioral Robotics* 1 (03 2010), 14–24. https://doi.org/10.2478/s13230-010-0002-4

Jürgen Schmidhuber. 2010. Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development* 2, 3 (2010), 230–247. https://doi.org/10.1109/TAMD.2010.2056368

Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. 2018. Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards. arXiv:1707.08817 [cs.AI] https://arxiv.org/abs/1707.08817

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

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. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://www.aclweb.org/anthology/2020.emnlp-demos.6

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 Technical Report. arXiv:2407.10671 [cs.CL] https://arxiv.org/abs/2407.10671

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. 2025. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. arXiv:2503.14476 [cs.LG] https://arxiv.org/abs/2503.14476