Extended Abstract

Motivation The development of Large Language Models has revolutionized our ability to solve general tasks with machines. Recent developments in the field had shown great promise in leveraging chain of thought reasoning to improve model performance. The current state-of-the-art open source method relies on large scale reinforcement learning solely using final answer accuracy rewards. We seek to develop methods that leverage entropy in different ways to develop structured reasoning traces to improve beyond simple accuracy rewards.

Method We use the GRPO RL algorithm. It iteratively trains a model by sampling multiple outputs on the same prompt, then computing a relative reward-based advantage to reinforce responses with high reward compared to others. DeepSeek-R1, the current SOTA, uses simple binary accuracy rewards to compute advantage. We propose Answer-level entropy reward (ALER) and token-level entropy reward (TLER) to be used with GRPO. ALER works by calculating the entropy among the space of different answers given by the outputs and assigning higher rewards to those with the largest entropy contribution to the total. Answers that contributed the most to entropy, ie those who are the closet to other answers, are reinforced, to converge the model to a single answer. TLER encourages the model to have a high entropy at the token level at the start of its reasoning, then a gradually lower entropy at the end. It is meant to mimic exploration then exploitation to a unified final answer.

Implementation We use Qwen2.5-Math-1.5B and Qwen2.5-0.5B as our base models. We train on the GSM8k and DeepScaleR datasets and evaluate on the GSM8k and AIME 2024 datasets. Accuracy is evaluated based on simply prompting the model to answer within a box and comparing the value inside the box to the ground truth; preliminary experiments showed this was a simple, effective method.

Results We find that training our models on GSM8k was more effective than on DeepScaleR evaluated on both GSM8k and AIME. We find that the models struggler to learn effectively over training steps on DeepScaleR. On GSM8k, our entropy methods outperformed accuracy rewards. Notably, our TLER method trained on GSM8k outperformed Qwen2.5-Math-1.5B-Instruct by 1.25 points on AIME. On DeepScaleR, accuracy rewards performed better than entropy-based methods. Finally, we observe that our methods trained on TLER displayed higher entropy at the start of reasoning and lower entropy at the end of reasoning compared to other methods, the closest one to ideal.

Discussion Our entropy methods, trained on just the GSM8k dataset, were able to learn general problem solving strategies that extended to strong performance increased on AIME. However, the accuracy-only model learned the most from DeepScaleR. We theorize that accuracy rewards are more robust and entropy rewards require stronger baseline accuracy to prevent divergence due to exploration. Finally, our entropy curves after training show that we are effectively able to learn exploration then exploitation using the TLER method, possibly leading to the effective transfer accuracy.

Conclusion In order to leverage our entropy methods effectively, training first on an easy dataset like GSM8k then training on gradually harder problems leveraging a stronger baseline may be effective. Overall, TLER is a first attempt to improve LLM ability to explore then exploit and learn to reason in a more structured, guided way. We have evidence to show it takes a step towards achieving that, and explore the cases in which it succeeds and fails. The field of reasoning is moving fast, and there are many more explorations of reward functions beyond accuracy to be done in the coming future.

Entropy-Based Rewards for Chain of Thought Reasoning

Julien Darve
Department of Computer Science
Stanford University
jdarve@stanford.edu

Abstract

We introduce novel entropy-based rewards for chain-of-thought reasoning reinforcement learning. Past works have shown that prompting Large Language Models to think step by step in a process called Chain of Thought reasoning can improve their ability to think and solve problems. However, the current leading methods depend primarily on final answer accuracy rewards. Such rewards suffer from a lack of interpretability in the reasoning trace, in addition to being sparse and binary, which can make convergence more difficult. In this paper, we explore methods that go beyond accuracy rewards, focusing on the idea of entropy. We introduce Answer-level and Token-level Entropy Reward (ALER and TLER, respectively), methods that encourage the model to have a structured exploration and exploitation step in its reasoning. We experiment with post-training Qwen2.5-0.5B and Qwen2.5-Math-1.5B on base accuracy, answer-level and token-level entropy GRPO rewards on the GSM8k and DeepScaleR datasets. We find that training with TLER on GSM8k outperforms Qwen2.5-Math-1.5B-Instruct on AIME by 1.25 points and entropy-based methods learn general problem solving abilities that better optimize entropy across the thinking trace.

1 Introduction

In the past few years, Transformer-based Large Language Models (LLMs) have asserted themselves as powerful general task solvers. OpenAI was the first to publicly release results from web-scale pretraining of LLMs, and found that mere next-token prediction at scale can produce AI models good at many tasks Radford et al. (2019) Brown et al. (2020). Beyond pre-training, modern LLM research has focused on aligning language models to human preferences through the use of reinforcement learning from human feedback (RLHF) and instruction finetuning Ouyang et al. (2022). These methods have enabled LLMs to act as helpful assistants and solve general tasks very effectively for use in daily life.

The very latest progress in LLM research has been to improve reasoning ability through a process called Chain-of-Thought (CoT) reasoning. This was first introduced by Google, who showed that simply prompting a language model to "think step by step" before giving its response significantly increased its performance Wei et al. (2022). Many methods have been proposed since then to directly train models to "think" before answering, beyond prompting.

Closed source reasoning models like OpenAI's o1 have dominated benchmarks in terms of reasoning ability OpenAI (2024). However, in January 2025, DeepSeek released DeepSeek-R1, an open-sourced LLM that rivaled o1's performance on major benchmarks DeepSeek-AI (2025). It was trained with Group Relative Policy Optimization (GRPO) introduced in an earlier paper Shao et al. (2024). This reinforcement learning method proved highly effective by being data and memory efficient. However,

while it had impressive numerical performance, its reasoning traces lack structure, logical consistency, and many times, the reasoning did not even correlate to its final answer.

In this paper, we introduce Token-Level Entropy Reward (TLER). TLER rewards a language model to have high entropy at the start of its reasoning chain and gradually lower entropy until the end. Entropy has been shown to be a key metric in reducing LLM hallucination and correlating with final answer correctness. When solving a problem, humans will naturally brainstorm multiple ideas first, then think deeply about a few number before coming to the final answer. We propose this formulation from the hypothesis that high entropy will encourage the model to explore multiple different solutions at the start of its reasoning trace, then converge to a unified correct answer by the end of its reasoning. TLER reinforces reasoning traces with this format with dense, process-based rewards.

We explore accuracy rewards, TLER, as well as Answer-level Entropy Rewards (ALER), where we reward convergence in model answers. We train a language model using GRPO on all three of these rewards methods on the GSM8k and DeepScaleR datasets, for 1.5B and 0.5B models. Our TLER method trained on GSM8k is able to outperform Qwen2.5-Math-1.5B-Instruct on AIME by 1.25 percentage points on average and 2 questions out of 30 in pass@8. From our experiments, we find that entropy rewards improve most effectively when the base model has a strong grasp of the problem to start with, and learn general problem solving ability that generalizes even among datasets of varying difficulty. Furthermore, on such datasets, TLER trains the model for effective exploration and exploitation measured by a more optimal token entropy distribution.

2 Related Work

Past work in training LLMs for CoT reasoning has debated Process Reward Models (PRMs) vs Outcome Reward Models (ORMs). Early work in reasoning was in favor of PRMs. A 2023 work by OpenAI Lightman et al. (2023) designed a process reward model using 800k step-by-step human feedback annotations for solutions to mathematics problems. They found the PRM was significantly more correlated than an ORM trained on 100 times the data in correctness of its reasoning and solutions, and was more logical and aligned to human preference. However, this method requires the collection of human annotations, which is expensive and subject to bias and variation.

While DeepSeek-R1 gained high performance using its accuracy-based ORM with novel training techniques, it suffers from a lack of logical consistency and structured reasoning. The original paper pointed it out as a limitation: DeepSeek-R1-Zero, their version of the model trained purely with accuracy-based rewards, would sometimes reason in different languages even when prompted in english. While true of language models in general, its thinking process was not logical or mathematically precise. This evidence point to a lack of interpretability in the model's reasoning. Interpretable, mathematically logical reasoning steps are essential for validating the model's final answer and ensuring transparency and trust for the user.

Furthermore, research by Aviral Kumar and others has asserted that DeepSeek-R1's outcome-based rewards lead to suboptimal test-time compute and convergence to the correct answer Qu et al. (2025). DeepSeek-R1 was observed to go through different ideas in a seemingly random order before answering. Many times, its answer would not even correlate to its reasoning. They propose a PRM method that rewards a model for making the most amount of progress at each step in its reasoning. It does so by analyzing future reasoning paths at given steps, and rewarding those steps based on how many of those reasoning paths succeed.

However, their PRM approach was motivated primarily by optimizing test-time compute. The method we propose aims to introduce structure and interpretability in the language model's reasoning through regulating its token-level entropy at different stages in its reasoning. Furthermore, computing multiple future pathways at training time is expensive, and we sought to develop a method that only requires a linear reasoning path. Our GRPO-based method only requires linear reasoning traces to compute reward.

Entropy has been shown to correlate strongly with hallucinations in language models. One such paper introduced the concept of semantic entropy Farquhar et al. (2024) and found it correlated highly with hallucinations in LLM output. Semantic entropy is computed as the entropy across the space of different possible answers the model gives. Answers are grouped into semantic clusters, evaluated by bi-directional implication determined by a lightweight LLM (DeBERTa). This way,

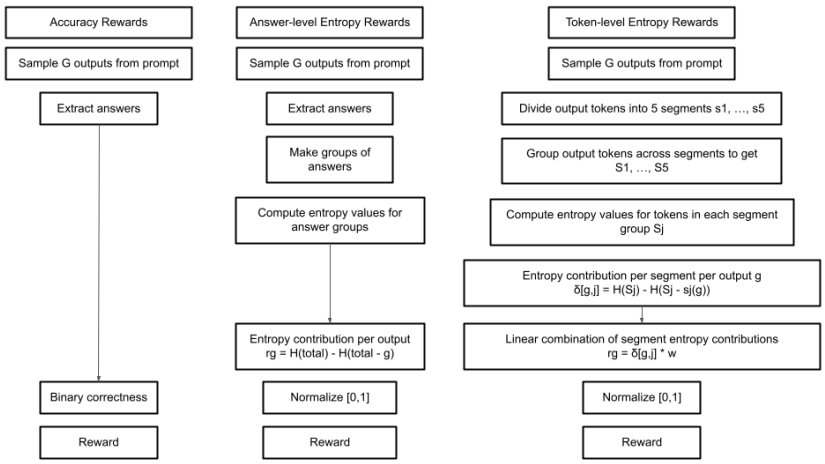

Figure 1: Method overview: comparison of accuracy, answer-level entropy and token-level entropy reward methods

even if two answers are different in terms of wording but represent fundamentally the same idea, they are considered the same for the entropy calculation. Answers with high entropy were most likely hallucinated, and answer with low entropy most likely were not.

Inspired by this result, we posed the question: could we train an LLM to produce solutions with high entropy? As the LLMs learns to give the correct answer, it also learns not to hallucinate in parallel. Beyond that, our method deviates from existing methods significantly. In order to avoid using an LLM as an evaluator to compute semantic equivalence, our method works at the token level instead of computing semantic clusters. Furthermore, we sought to consider the reasoning trace itself, not just the distribution of final solutions. We present the precise formulation of our PRM below.

3 Method

3.1 GRPO

All of our experiments were run using the GRPO algorithm proposed by DeepSeek. While traditional optimization algorithms like PPO require a separate critic and actor model, GRPO optimizes it away by considering the relative advantage between a group of G different outputs. It samples multiple outputs on the same prompt using policy $\pi_{\theta_{\text{old}}}$, and reinforces the log probabilities of all of tokens in the policy π_θ with the following formula:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}\big[\, q \sim P(Q),\ \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)\big]$$

$$\frac{1}{G}\sum_{i=1}^{G}\left[\min\Big(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}\,A_i,\ \text{clip}\Big(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)},\,1-\varepsilon,\,1+\varepsilon\Big)\,A_i\Big) - \beta\, D_{\text{KL}}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big)\right]$$

where D_{KL} is the familiar KL-divergence from PPO of the old and new policy to prevent drifting too fast from the old policy:

$$D_{\text{KL}}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big) = \sum_o \pi_\theta(o|q)\,\log\frac{\pi_\theta(o|q)}{\pi_{\text{ref}}(o|q)}$$

In DeepSeek-R1-Zero, we have the advantage function A_i based on the normalized reward

$$A_i = \frac{r_i - \text{mean}\{r_1,\ldots,r_G\}}{\text{std}\{r_1,\ldots,r_G\}}.$$

The advantage function can be used to optimize arbitrary rewards. In the original DeepSeek-R1 paper, they propose a simple accuracy reward, where $r_i = 1$ if the ith solution is correct, else $r_i = 0$. Thus correct answers are rewarded, and incorrect answers are punished, and their magnitude scales with

how rare their rewards were. They also used format rewards, where the models were rewarded to keep their thinking traces within special <think> tokens and answers within <answer> tokens. In this paper, we do not consider any format rewards. The model is free to think and answer in any order (in practice it always answers at the end) for simplicity.

3.2 Answer-level Entropy Rewards

Before, we introduce TLER, we also experimented with an intermediary entropy-based reward method, closer to the result of Farquhar et al. (2024), which we will call Answer-Entropy reward. While their semantic entropy algorithm only measured the entropy across all answers, we want a different reward value for each answer. Thus, we converged on the idea of entropy contribution. On the same prompt, we sample G different outputs. We extract only the answers of those outputs. For each output, we then calculate (1) the total entropy across all answers, and (2) the total entropy across all answers without considering the answer from that output. In other words, for G_p being the set of outputs for a given prompt p, we count

$$c_a = \#\{\, i \in G_p : \text{answer}_i = a\}, \quad P(a) = \frac{c_a}{\sum_b c_b}$$

and its corresponding entropy

$$H_{\text{total}}(G_p) = -\sum_{a \in \text{answers}} P(a)\, \log P(a).$$

Then, for each generation $g \in G_p$, we do the same but instead calculate the entropy across all outputs minus the current generation g:

$$c'_a = \#\{\, i \in G_p/\{g\} : \text{answer}_i = a\}, \quad P(a)' = \frac{c'_a}{\sum_b c'_b}$$

and we get the remainder entropy

$$H_{\text{rest}}(G_p, g) = -\sum_a P'(a)\, \log P'(a).$$

We calculate the difference between the total and the remaining entropy to get a raw reward value

$$r_g = -\frac{H_{\text{total}}(G_p) - H_{\text{rest}}(G_p, g)}{H(G_p)}$$

We normalize by the total entropy to be consistent within batches. With our r_g value, we normalize it to be between $[0, 1]$ to get the final reward value for that generation

$$\hat{r_i} \;=\; \frac{r_i - \min_j(r_j)}{\max_j(r_j) - \min_j(r_j)}.$$

Thus, answers are rewarded based on how much they contributed to entropy. If an answer contributes positively to entropy, that means it was a common answer (it made the space of answers more ordered) and it is rewarded, and if it contributes negatively, it was an uncommon answer (it increased the disorder of the space and moved it closer to uniform) and it is negatively rewarded.

3.3 Token-level Entropy Reward

Our token-level entropy reward function is similar to the answer-entropy reward function, except that it measures entropy contribution based on the token distribution. Like before, we sample G outputs from a prompt p to get a set of outputs G_p. Then, each output is cut into n segments (for our experiments, $n = 5$) such that a given element $g \in G_p$ is represented as $g = s_1||s_2||s_3||s_4||s_5$. Each segment is of equal length for answers of varying length; thus the cutting is based on percentile. Define $S_j = \{s_j \in g : g \in G_p\}$ representing the set of the jth segment for all responses.

For each segment across all completions, we compute the entropy

$$H_{\text{total}}(S_j) = -\sum_{t \in \text{tokens}(S_j)} P(t) \log P(t)$$

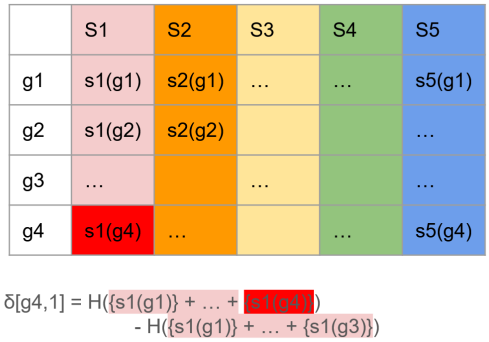

Figure 2: Visual Representation of the entropy contribution per segment calculation ($\delta_{g,j}$) in TLER. x-axis represents segments, y-axis represents different responses. Observe that the entropy contribution is computed as the total - rest entropy for a given segment.

for $P(t)$ being the number of times it appears divided by the total number of tokens. Then, similarly to the previous entropy method, for each generation $g = s_1||s_2||s_3||s_4||s_5$, for each of its segments s_j we compute the remainder entropy per token:

$$H_{\text{rest}}(S_j, s_j(g)) = - \sum_{t \in \text{tokens}(S_j / \{s_j(g)\})} P(t) \log P(t).$$

For each completion g this results in n values for the entropy contribution of each of its segments

$$\delta_{g,j} = - \frac{H_{\text{total}}(S_j) - H_{\text{rest}}(S_j, g)}{H_{\text{total}}(S_j)}.$$

Then, we take a linear combination of these segments entropy contributions to get our raw reward value. We chose $\mathbf{w} = \left[0.5,\ 0.3,\ 0.1,\ -0.3,\ -0.5\right]$ as our weights for the linear combination.

$$r_g = \sum_{j=1}^{n_{\text{seg}}} w_j\, \delta_{g,j}, \quad \mathbf{w} = \left[0.5,\ 0.3,\ 0.1,\ -0.3,\ -0.5\right]$$

Our choice of $\mathbf{w}$ is such that positive weights on the first three segments encourage exploration (high entropy) early, while negative weights on the last two segments encourage convergence (low entropy) later. After that, we put it through the same normalization function as before to get our final result

$$\hat{r}_i \ = \ \frac{r_i - \min_j(r_j)}{\max_j(r_j) - \min_j(r_j)}.$$

Thus, if an answer has high entropy at the start or low entropy at the end of its reasoning, that contributes positively to its reward, and vice versa. This token level heuristic allows us to approximate exploration and exploitation.

4 Experimental Setup

We made initial experiments with prompting and final answer validation in preparation for training. After that, we explored a variety of datasets, models, and methods to evaluate different entropy approaches.

4.1 Base Models

We chose to use the Qwen2.5 family of models as our base models Team (2024) due to their wide use in existing literature. Our initial experiments were performed on Qwen2.5-0.5B. After that, we expanded to the Qwen2.5-Math-1.5B model Yang et al. (2024). Qwen2.5-Math-1.5B differs from Qwen2.5-1.5B in that it includes additional finetuning on a mathematical corpus. We also report

evaluation baselines for Qwen2.5-0.5B-Instruct and Qwen2.5-Math-1.5B-Instruct. These models were trained with instruction finetuning on top of the base model to align with user preferences and solve tasks directly. Furthermore, Qwen2.5-Math-1.5B-Instruct included post-training with an LLM-based 70B process reward model.

4.2 Datasets

We trained our initial models on the GSM8k dataset Cobbe et al. (2021). This is a high quality corpus of mathematical question answer pairs at the grade school level, spanning 7.4k training set and 1.3k testing set. The questions require only a few steps of reasoning and can be solved without equations, only using natural language.

We also experimented with the much harder DeepScaleR corpus Luo et al. (2025). This dataset is a combination of AIME problems from 1984–2023 and AMC problems before 2023, along with questions from the Omni-MATH and Still datasets. AIME, which stands for the American Invitational Math Exam, and AMC, the American Mathematics Competition, are nationwide math competitions, spanning a collection of very difficult questions. These problems require many steps and mathematical equations. The DeepScaleR corpus was curated to remove duplicates and answers that could not be easily graded, and all problems have numerical answers, culminating in 40k question answer pairs. We use the AIME 2024 questions as the testing set to evaluate our model on this dataset.

4.3 Answer Extraction

Our system prompt was "Please reason step by step, and put your final answer within `\boxed{}`." We found in our early experimentation that even the models without post-training were able to align to this format. Using this, we evaluated the models accuracy by searching for `\boxed{}` in the model's answer, and extracting the text inside to compare to the ground truth answer. We found that the model would always only answer with a single box and it would align with its final answer with high probability.

4.4 Reward Functions

We show results for training with accuracy-only reward, replicating DeepSeek-R1-Zero, as a baseline for both Qwen2.5-Math-1.5B and Qwen2.5-0.5B base models.

We further experiment with training both models on answer-level and token-level entropy methods. Our experiments with the two different entropy rewards also include accuracy rewards. For answer-level entropy rewards, both rewards have the same weight. For token-level entropy rewards, we report experiments with TLER weighted by 0.6 to keep the model focused on final answer accuracy. We do not perform exhaustive hyperparameter tuning on the reward weights; that is one limitations of this paper.

5 Results

We find that our token-level entropy reward method is able to outperform Qwen2.5-Math-1.5B-Instruct on the AIME'24 dataset when trained on GSM8k. Furthermore, answer-level entropy rewards performed best for the 0.5B base model. Models trained on the DeepScaleR dataset performed worse than those trained on GSM8k.

5.1 Quantitative Evaluation

We report full quantitative results of our main experiments in 5.1. Because AIME 2024 is a 30 question test, we compute an average score and 95% confidence interval over 8 runs. Furthermore, we report pass@8 accuracy and average response length (in terms of characters) for the same runs. Finally, we report percentage accuracy on the test set of GSM8k. We report results for 0.5B models trained on GSM8k, 1.5B models trained on GSM8k, and 1.5B models trained on DeepScaleR (noted as DSR).

Method (Dataset)	GSM8k (%)	AIME'24 (95% CI)	Pass@8 (%)	Avg Length
Qwen2.5-Math-1.5B	60.58	7.08 ± 3.21	23.33	2984.43
Qwen2.5-Math-1.5B-Instruct	83.78	10.00 ± 2.04	23.33	2844.71
Accuracy Rewards (DSR)	**79.30**	$\mathbf{7.92 \pm 3.32}$	26.66	2974.30
Answer-level Entropy (DSR)	64.37	6.25 ± 2.48	26.66	3074.93
Token-level Entropy (DSR)	78.70	7.50 ± 2.56	26.66	2819.33
Accuracy Rewards (GSM8k)	80.67	8.33 ± 3.54	26.66	3158.50
Answer-level Entropy (GSM8k)	**82.56**	9.58 ± 2.48	26.66	3083.33
Token-level Entropy (GSM8k)	76.95	$\mathbf{11.25 \pm 2.34}$	**30.00**	3151.24
Qwen2.5-0.5B	40.56	0.00 ± 0.00	0.00	3100.13
Qwen2.5-0.5B-Instruct	44.20	0.00 ± 0.00	0.00	2406.85
Accuracy Rewards (0.5B)	**51.86**	0.83 ± 1.02	3.33	2833.29
Answer-level Entropy (0.5B)	49.73	$\mathbf{1.25 \pm 1.14}$	**10.00**	2895.42
Token-level Entropy (0.5B)	50.27	0.83 ± 1.02	6.67	2309.58

Table 1: Test set accuracy comparison across all base models, reward methods, and datasets. We report GSM8k test set and AIME 2024 % accuracy. We additionally report a 95% confidence interval for AIME 2024 based on its standard deviation over 8 runs, and its Pass@8 accuracy. We also report average response length in characters on the AIME questions

We began by training Qwen2.5-Math-1.5B and Qwen2.5-0.5B base models with simple accuracy-based GRPO. Our GRPO baseline is able to increase the accuracy on GSM8k by 20 points for the 1.5B model and 11 points for the 0.5B model when training on GSM8k.

Next, we implemented answer-level and token-level entropy rewards. Training on GSM8k, we found that answer-level entropy increased performance on GSM8k by 2 points from the accuracy rewards, almost reaching the Qwen2.5-Math-1.5B-Instruct accuracy. On 0.5B, our answer-level entropy rewards outperformed all other methods on AIME results, solving 3/30 questions in pass@8. We also see that our entropy-based methods have significantly lower standard deviation in their accuracy compared to the accuracy rewards, yet their pass@8 accuracy is equal to or better.

Furthermore, we found that, even though the 1.5B models were only trained on GSM8k, their performance on AIME'24 increased significantly. The answer-level entropy method increased AIME accuracy by 3 points from the base model, as well as 1.25 points above the accuracy-only rewards. Token-level entropy increased it by 5 points, even surpassing Qwen2.5-Math-1.5B-Instruct by 1.25 points. Furthermore, its pass@8 accuracy evaluation showed that it was able to solve one additional question (out of 30 questions on the AIME exam) compared to other post-training methods, and 2 questions compared to the base models. While the model does not outperform Qwen2.5-Math-1.5B-Instruct by a statistically significant margin based on our confidence interval computation, the results are striking nonetheless.

We then decided to train with our three different methods on the DeepScaleR dataset. Surprisingly, training on this dataset had significantly worse final answer accuracy than when trained on GSM8k, even for AIME'24 evaluations. The answer-level entropy reward model's accuracy sank to below the baseline on AIME. GSM8k accuracy increased, and AIME accuracy increased by less than a point for the other two methods. The models learned significantly less on the DeepScaler dataset than GSM8k, which we discuss further below.

We report training curves in 3. The top two curves show accuracy rewards (proportion of correct answers) for the GSM8k and DeepScaleR experiments, respectively, across all three models. The entropy_rewards curve on the bottom left represent the TLER reward value, and the bottom right curve represents Answer-level entropy reward (ALER in the labels) for both GSM8k and DeepScaleR (since the corpora are differently sized, the DeepScaleR curves go on for longer).

We first observe that, across experiments, the rewards rise fast initially, then largely stabilize for the rest of training. The effect is less pronounced for the DeepScaleR models, which was indicated by their reduced improvement on the testing sets compared to the base models. Indeed, accuracy rewards

Figure 3: Reward curves per training step for 1.5B models (Model | Dataset). Top left is GSM8k accuracy, right is DeepScaleR accuracy, bottom left is TLER, right is ALER across relevant 1.5B models.

on the DeepScaleR dataset for all models rises very slowly across training, while GSM8k accuracy rewards show notable improvement across training steps.

The two entropy methods in the GSM8k experiment showed significantly higher performance over time than the base model. On the DeepScaleR experiment, there was little difference between each method, and the rewards had a large amount of noise across steps. On the token-level entropy method, we see the model is able to learn the reward significantly more effectively on the former dataset, although it exhibits more noise. The same is not as clear for the answer-level entropy rewards. On average, its answer-level entropy values are higher, but not significantly so, exhibiting significant noise.

5.2 Qualitative Analysis

To more deeply investigate the difference in token-level entropy rewards and the model's ability to explore and exploit, we graph the average entropy contribution per segment for each model after training on DeepScaleR in 4. We see that, across the board, GRPO reduces token-level entropy; the base model has the highest entropy in segments 1-5. Accuracy rewards created the lowest entropy generations in segments 0-4, representing a collapse in diversity that may have hindered its accuracy across datasets. Notably, TLER successfully has the highest first segment entropy and the lower last segment entropy, representing its strongest reward coefficients. It aligns closest to the ideal curve compared to others, although remains far off. Answer-level entropy, which performed poorly trained on DeepScaleR, is shown here as having high final-segment entropy, potentially predicting its lower eval scores.

Our experiments reveal significant differences in how different reward models impact model behavior across datasets of different difficulty and diversity. Our results indicate that TLER is able, to some degree, increase model exploration and convergence to a solution, shown by its higher entropy value for the first segment and lower entropy in its last segment and answer standard deviation. When trained on GSM8k, whose solution paths are relatively homogeneous, TLER enabled the model to learn general problem solving abilities; its performance transferred effectively to AIME. DeepScaleR is a harder problem space to optimize over, where only a small set of solution paths are able to find the correct answer. There, the TLER method's exploration bonus sometimes led the model astray, causing early saturation and lower final accuracy.

Answer-level entropy rewards echo some of these conclusions. ALER is able to converge effectively around solutions, if common solutions exist with a reasonable frequency. It performed the best on GSM8k when trained on GSM8k, but performed the worst on AIME when trained on DeepScaleR. If

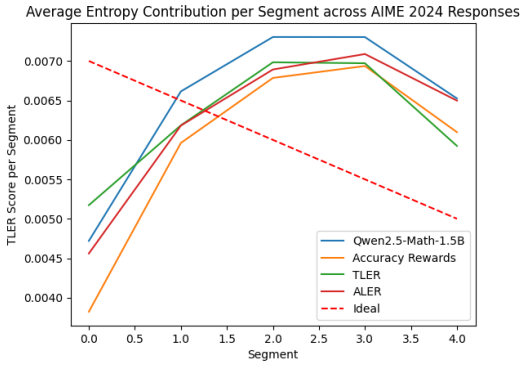

Figure 4: Entropy contribution per segment compared to the hypothesized ideal based on our TLER method. We present our three methods and our base models trained on DeepScaleR, evaluated and averaged over 8 responses to all AIME 2024 problems

the number of answers that are correct are low, the algorithm will reward answers that are not correct, which might cause divergence in training. The token-level entropy of its final segment is higher, reflective of its inability to converge to a solution on DeepScaleR. On the other hand, on an easier dataset where the model is more frequently correct, it makes the model converge more effectively to those answers, causing the high performance on GSM8k. Its also transfers the most effectively from GSM8k to AIME on the 0.5B model, a medium difficulty task in between 1.5B on AIME and GSM8k. Perhaps at that size, there are enough correct answers that the answer-level entropy reward reinforces training with a dense reward signal.

Our training curves show that models trained on GSM8k are able to better learn our entropy rewards compared to on DeepScaleR. By contrast, simple accuracy rewards proved more robust on the heterogeneous DeepScaleR dataset, being able to continuously improve however slowly, reaching the best accuracy on that dataset compared to the entropy-based methods. A clear correctness signal is essential to guide the model toward the narrow set of valid approaches on this dataset. For GSM8k, accuracy rewards performed consistently on the GSM8k test set but generalized worse to AIME. In addition, from this model's entropy curves per segment, we see that its overall entropy decreased, collapsing to a worse level of exploration and exploitation that entropy-rewards mitigated.

6 Discussion

Our results were limited in multiple ways. We were limited in terms of compute, so we were only able to experiment with 1.5B models and 0.5B models on a 2000 token context window. Larger scales may change convergence behavior and emergent properties. We also did not implement format-based restrictions or rewards, instead extracting the boxed answer and comparing it to the ground truth, requiring the model to learn format rewards through accuracy, which can be slower.

Futhermore, our entropy-based methods run the risk of reward hacking due to evaluating only on token-level entropy. For example, the model could output many rare tokens to artificially increase its entropy in the first stages. However, it nevertheless teaches the model to have more new tokens at the start, which can increase the amount of possible branches and therefore opportunities for successful branches, if exploited upon successfully later in reasoning.

The AIME 2024 benchmark is also a very small evaluation set, prone to a lot of variation in performance. While our results, especially for TLER, are higher than baseline, some results are not statistically significant at the level of 95%, so they cannot be accepted outright. More experimentation would have to follow to fully verify our results. Nevertheless, the improvements on entropy and across datasets remain significant.

Some difficulties we faced during the project were mainly getting models running on the computing cluster. It required tuning the batch size, gradient accumulation, and the number of generations per step in order to fit everything efficiently in the context window. Training also took over 24hrs in some

cases, requiring constant attention for bugs and inaccuracies in implementation so that results are accuracy and quality controlled.

Beyond that, we hope that our work can be used to inspire methods to train reasoning models that can think in human-interpretable ways. We hope to gain a broader interpretability of LLM thinking processes, and how they compare with human's, so that we can understand the answers they give us, and in which directions we can improve future language models.

7 Conclusion

We conclude that our entropy-based methods are able to reinforce the accuracy of tasks the model already has some grasp over. Furthermore, training on easier datasets allow the models to learn general problem solving skills that transfer to more difficult datasets. On these tasks, the models learn to explore and exploit effectively. In difficult datasets without a warm start, entropy rewards confuse and hinder learning from the accuracy rewards.

We propose that, for training models with entropy-based rewards in the future, a strong strategy might be to incorporate a staged training procedure. First, we can train on easy datasets the base model already has a strong grasp of. Then, we can gradually extend to more difficult datasets, where the model can now continue to improve its problem solving ability. This methods enables the model to only train on tasks where it starts with a strong base accuracy, where it has been proven to work effectively.

Furthermore, we can scale these experiments to larger models to measure scaling laws from our entropy rewards. With more compute, we can increase the context length of our models to allow for longer, complex reasoning traces. TLER may perform better if it has more room to explore in the context window. We can also explore varying the number of segments from 5 to, say, 2 or 10, to measure if a higher level of granularity can improve TLER. We can also place a higher reward coefficient on the TLER rewards, and see if we can continue the positive trend we observed in our entropy per segment curves to get closer to our hypothesized idea.

The field of open source methods for training LLMs for mathematical reasoning has just begun. After DeepSeek-R1 was introduced 6 months ago, countless hours of work across the globe have been spent to recreate their results and explore improvements. We present one such example in this work, but there is still much more work to do. There are many signs that chain-of-thought reasoning is the future of LLMs, and large scale reinforcement learning is the most effective method of training we have. It is a path research will continue to explore, and we will continue to emphasize interpretability in reasoning and thinking.

8 Team Contributions

- **Julien Darve:** Prompt engineering. Answer extraction pipeline. Base model and dataset curation. Reward function formulation and design. GRPO trainer implementation and deployment. Running and monitoring all experiments. Model evaluation on testing sets. Entropy experiments and analysis.

Changes from Proposal We originally proposed to implement a method closer to Farquhar et al. (2024), but landed on token-level entropy for the reasons described in our methods (not LLM-based so less hackable, faster to compute, clearer to implement/evaluate). ALER essentially covers the exploration of that technique. We also planned for our entropy reward feedback to be at the advantage level, ie computing $\hat{A}_i = A_i - \alpha H(i)$ to reduce the entropy in the answers. However, we decided to use the idea of entropy contribution to have a different reward per output, in order to use GRPO as it was intended with individual rewards per output to compute relative advantage. We originally planned to train the model on NuminaMath, but the dataset was less standardized than DeepScaleR, so we went with the latter.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel

Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 159, 25 pages.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168* (2021).

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

Sebastian Farquhar, Joshua Kossen, Leonard Kuhn, et al. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature* 630 (2024), 625–630.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's Verify Step by Step. arXiv:2305.20050 [cs.LG] https://arxiv.org/abs/2305.20050

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. DeepScaleR: Surpassing O1-Preview with a 1.5B Model by Scaling RL. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a Notion Blog.

OpenAI. 2024. Introducing OpenAI *o1*-preview. https://openai.com/index/introducing-openai-o1-preview/. OpenAI Blog, accessed 6 Jun 2025.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs.CL]

Yuxiao Qu, Matthew Y. R. Yang, Amrith Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan Salakhutdinov, and Aviral Kumar. 2025. Optimizing Test-Time Compute via Meta Reinforcement Fine-Tuning. arXiv:2503.07572 [cs.LG] https://arxiv.org/abs/2503.07572

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI* (2019). https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300 [cs.CL] https://arxiv.org/abs/2402.03300

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, Article 1800, 14 pages.

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. 2024. Qwen2.5-Math Technical Report: Toward Mathematical Expert Model via Self-Improvement. *arXiv preprint arXiv:2409.12122* (2024).