Maximizing Confidence Alone Improves Reasoning

Mihir Prabhudesai[*]
Carnegie Mellon University

Lili Chen[*]
Carnegie Mellon University

Alex Ippoliti[*]
Carnegie Mellon University

Katerina Fragkiadaki
Carnegie Mellon University

Hao Liu
Carnegie Mellon University

Deepak Pathak
Carnegie Mellon University

Abstract

Reinforcement learning (RL) has enabled machine learning models to achieve significant advances in many fields. Most recently, RL has empowered frontier language models to solve challenging math, science, and coding problems. However, central to any RL algorithm is the reward function, and reward engineering is a notoriously difficult problem in any domain. In this paper, we propose **RENT: R**einforcement Learning via **Ent**ropy Minimization – a fully unsupervised RL method that requires no external reward or ground-truth answers, and instead uses the model's entropy of its underlying distribution as an intrinsic reward. We find that by reinforcing the chains of thought that yield high model confidence on its generated answers, the model improves its reasoning ability. In our experiments, we showcase these improvements on an extensive suite of commonly-used reasoning benchmarks, including GSM8K, MATH500, AMC, AIME, and GPQA, and models of varying sizes from the Qwen, Mistral, and Llama families. The generality of our unsupervised learning method lends itself to applicability in a wide range of domains where external supervision is unavailable. Website: `https://rent-rl.github.io/`.

1 Introduction

Imagine you're taking an exam. Once it begins, no new information is available and no external help can be sought. With only your own reasoning to rely on, how do you tackle a difficult problem? You might make an initial attempt, assess your confidence in the answer, and revise your reasoning until you feel sufficiently certain. Of course, confidence is not a guarantee of correctness – but in the absence of feedback, it is often the only intrinsic signal we have to guide further thought. In such settings, humans tend to optimize for confidence, or equivalently, to reduce uncertainty. In machine learning, uncertainty is commonly quantified via entropy – a measure of how peaked or diffuse a probability distribution is. Language models output distributions over tokens, and the entropy of these distributions reflects the model's confidence: lower entropy implies more confident predictions. Yet despite the growing use of language models in reasoning tasks, current approaches to improvement still rely heavily on external supervision, rewarding models based on correctness with respect to ground-truth labels [10, 39]. This dependence is often impractical, particularly in real-world or open-ended scenarios where supervision is scarce or unavailable.

To address this, we propose **RENT: R**einforcement Learning via **Ent**ropy Minimization – a fully unsupervised reinforcement learning method that improves reasoning performance by using the model's own confidence as a reward. Specifically, we define the reward as the negative entropy of the model's predicted token distributions. This signal is dense, general, and easy to compute, requiring no ground-truth answers. Importantly, not all parts of the response contribute equally to final

[*]Equal contribution.

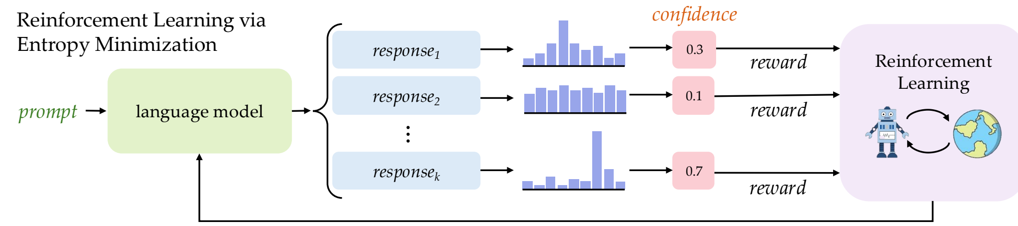


Figure 1: Overview of RENT: Reinforcement Learning via Entropy Minimization. For each response, we use the model's underlying confidence (negative entropy) as a reward for reinforcement learning. This enables the model to learn without any external reward or ground-truth answers.

performance. Through empirical analysis, we find that minimizing entropy over tokens near the end of the reasoning chain, especially those corresponding to the final answer, correlates most strongly with improved accuracy. In contrast, early tokens in the response show little correlation. This suggests that as the model approaches its final answer, it increasingly relies on its own confidence to guide reasoning, so encouraging confidence in these final steps is key to improving overall performance.

We demonstrate RENT's effectiveness across diverse reasoning benchmarks, including GSM8K [6], MATH500 [14, 25], AMC and AIME [23], and GPQA [33]. Our method scales across model families (Qwen, Mistral, and Llama) and sizes and consistently improves performance.

2 Related Work

2.1 Reinforcement Learning for Reasoning

Initially, reinforcement learning (RL) for language models was mostly used for learning from human preferences [5] and, traditionally, the RL optimization was done with algorithms such as PPO [37]. With the capabilities of language models continuing to improve, researchers have begun to explore the possibility of using RL to improve the performance of language models on reasoning tasks such as math [6, 14, 23], science [13, 33], or coding [24, 4] problems. In these settings, the model is prompted to generate a chain-of-thought [51] and final answer, and receives a reward based on how closely its final answer matches the ground-truth answer. These efforts present RL as an alternative to search-based approaches to chain-of-thought reasoning such as Tree of Thoughts [56] and Graph of Thoughts [2]. Related lines of work include training a reward model to give feedback for every step in the chain of thought, and training RL models to encourage self-correcting behaviors in language models. Examples of RL methods in this space include Zelikman et al. [59], Singh et al. [42], Kumar et al. [21], Qu et al. [31], Uesato et al. [46], Lightman et al. [25], Wang et al. [50]. At scale, DeepSeek [10, 39] proposed an open-source model that showed OpenAI o1 [16]-level reasoning by performing RL in this manner, using a new algorithm GRPO [39].

2.2 Confidence and Calibration

Confidence measures quantify how certain a model is that its generated output is correct [58, 43]. In order to evaluate the confidence of machine learning models, it is necessary also to discuss *calibration* [19, 48] - i.e., how aligned those confidences are with actual correctness. As language models are increasingly trusted to make important decisions, providing users with a reliable confidence measure would be useful in many situations [8, 27, 47, 15, 17, 12, 43]. As such, researchers have developed various confidence metrics for modern deep learning models and studied the extent to which they are calibrated. These include both methods that assume access to the model's weights [11, 18, 54, 43] and methods that estimate confidence via prompting alone [52, 8, 45, 53, 55]. In our paper, we use the model's confidence to iteratively improve its own performance via reinforcement learning.

2.3 Test-Time Adaptation

Test-time adaptation is where a model is updated using data from the test distribution, without access to ground-truth labels. The goal is to improve performance in scenarios where there is a distribution shift between training and testing environments. Methods for adapting without labels include normalization techniques that recalibrate feature statistics at test time [32, 44, 28, 36, 29]. The most relevant work to ours is Tent [49], which performs entropy minimization on model predictions during test time. This approach assumes that predictions on test data should be low in entropy if the model is well-adapted to the new distribution. Tent builds on earlier work that uses entropy minimization as a regularization strategy in semi-supervised learning [9, 22, 1] and domain adaptation contexts [28, 41, 35], where encouraging confident predictions has proven effective for improving generalization. Recently, TTRL [61] proposed test-time reinforcement learning using majority voting as a reward. Compared to entropy, majority voting is a sparse reward and much less general; for example, it cannot be applied to long-form free-response questions.

2.4 Unsupervised Reinforcement Learning

Unsupervised RL trains agents using intrinsic rewards like novelty, entropy, or mutual information, enabling skill acquisition without extrinsic feedback. Prior methods include ICM and RND for prediction error [30, 3], APT [26] and ProtoRL [57] for entropy maximization, and DIAYN, APS, and SMM for skill discovery via mutual information [7, 26, 20]. An interesting observation is that while exploration methods primarily maximize entropy, we instead minimize it by reinforcing high-confidence outputs, and find that for language models, this leads to better reasoning performance without any external supervision.

3 Method

3.1 Reinforcement Learning for Language Models

The goal of reinforcement learning (RL) is to train a policy which generates actions that maximize the cumulative expected reward. In the context of modern language models, the policy π is a language model and the actions y_{pred} are sampled from the distribution over a discrete vocabulary. The task is formulated as a one-step RL problem in which the model generates $y_{\text{pred}} = \pi(x)$, where x is sampled from the dataset $\mathcal{D} = \{(x, y_{\text{target}})\}$, and receives some reward for the generation. Typically, the ground-truth answer y_{target} is used to give the model a reward $r = \mathcal{R}(y_{\text{target}}, y_{\text{pred}})$. One reward function which is currently used is simple string matching, where $\mathcal{R}(y_{\text{target}}, y_{\text{pred}}) = \mathbb{1}\{y_{\text{target}} = y_{\text{pred}}\}$. Our work focuses on instead doing *unsupervised* reinforcement learning, which does not require external supervision for the reward. Specifically, y_{target} is not used in the reward $r = \mathcal{R}(y_{\text{pred}})$ and we do not assume access to this at any point in training.

3.2 Group Relative Policy Optimization (GRPO)

To optimize the policy, we adopt GRPO [39], a reinforcement learning algorithm that emphasizes relative rather than absolute performance. Instead of directly maximizing the reward of the current policy, GRPO evaluates the policy in relation to a group of baseline policies. This comparison helps improve learning stability, especially in settings with noisy or unsupervised reward signals.

Let π denote the current policy, and let $\{\pi_1, \pi_2, \dots, \pi_K\}$ be a fixed or evolving set of reference policies. The GRPO objective is defined as:

$$\mathcal{L}(\pi) = \mathbb{E}_{y_{\text{pred}} \sim \pi(x)} \left[\mathcal{R}(y_{\text{pred}}) \right] - \frac{1}{K} \sum_{i=1}^{K} \mathbb{E}_{y_{\text{pred}} \sim \pi_i(x)} \left[\mathcal{R}(y_{\text{pred}}) \right]$$

The first term represents the expected reward under the current policy π, while the second term computes the average reward across the reference group. The learning signal is thus the improvement in reward relative to these baselines. For more details, we refer the reader to Shao et al. [39].

3.3 Entropy Reward

For a given prompt x, the model generates a response $y_{\text{pred}} = y_{\text{pred},1}, \cdots, y_{\text{pred},T} = \pi(x)$, where T is the number of tokens in the response. At each token $t \in \{1, \dots, T\}$, the model outputs a probability distribution p_t over the vocabulary $\mathcal{V}$, i.e., $p_t(v) = P(y_t = v \mid x, y_{<t})$. The entropy of

this distribution measures the model's uncertainty in predicting the next token and is given by:

$$H(p_t) = -\sum_{v \in \mathcal{V}} p_t(v) \log p_t(v)$$

To compute the total entropy of the response, we average the entropies across all tokens. The total entropy $H(\pi(x))$ provides a measure of the overall uncertainty in the model's response. Higher entropy indicates greater uncertainty or more diverse token predictions, while lower entropy suggests more confident and peaked distributions at each token. We use the negative entropy of the predicted token distribution as a reward signal:

$$\mathcal{R}(y_{\text{pred}}) = -H(\pi(x)) = \frac{1}{T} \sum_{t=1}^{T} \sum_{v \in \mathcal{V}} p_t(v) \log p_t(v)$$

This reward encourages the model to produce more confident and peaked distributions over the vocabulary, effectively promoting lower uncertainty in its predictions. Within the RL framework, the learning objective becomes maximizing the expected reward over the data distribution:

$$\max_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \left[\mathbb{E}_{y_{\text{pred}} \sim \pi(x)} \left[\mathcal{R}(y_{\text{pred}}) \right] \right]$$

By optimizing this objective, the model learns to generate responses with lower entropy without relying on external supervision or labeled target responses.

4 Experiments

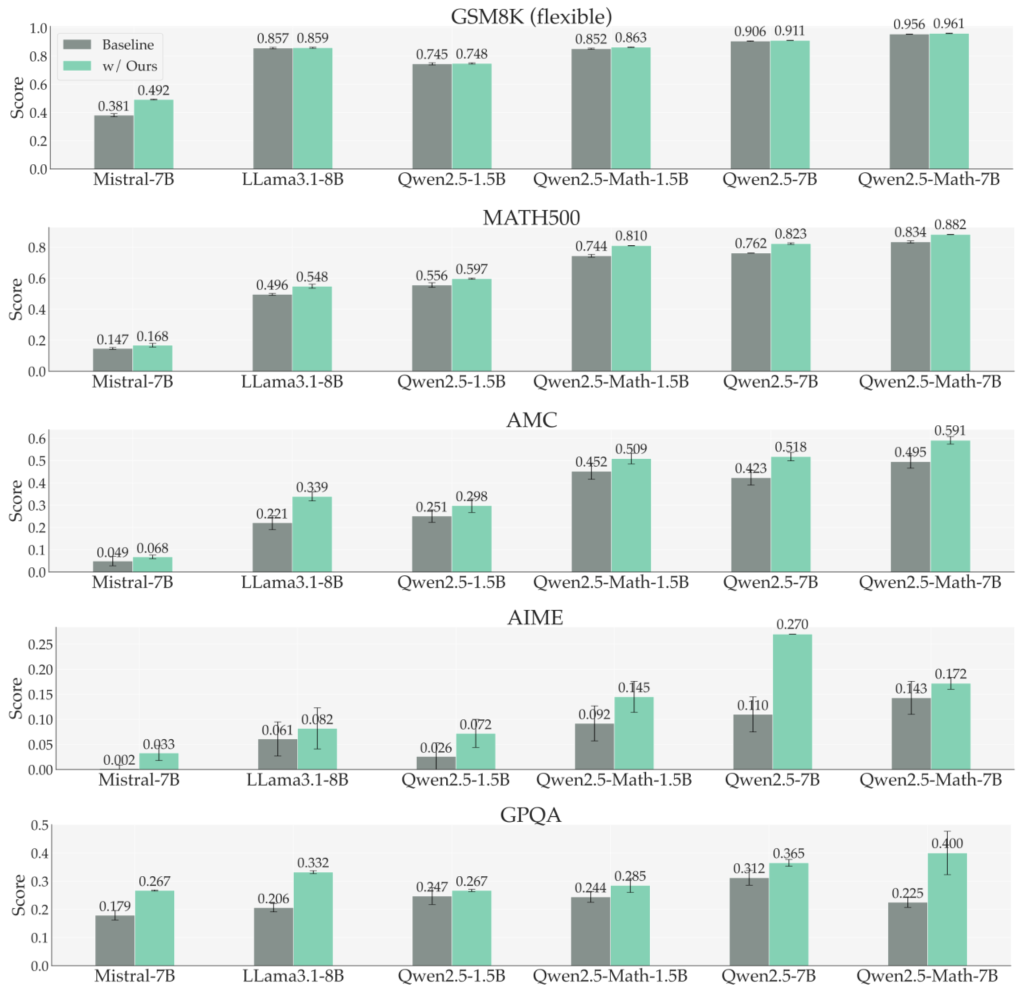
4.1 Experimental Setup

Benchmarks. We train a model with reinforcement learning on each dataset independently. We conduct our experiments on the following commonly-used benchmarks for evaluating the reasoning capabilities of large language models:

- GSM8K [6]: GSM8K contains 8792 grade-school math word problems. The train set contains roughly 7473 problems and the test set contains roughly 1319 problems.
- MATH500 [14, 25]: MATH [14] is a dataset containing competition math problems spanning seven categories. It contains 12500 problems, of which 7500 are used for training and 5000 are used for testing. MATH500 [25] is a subset of the MATH test set created by OpenAI by sampling uniformly at random from the test set.
- AMC [23]: The American Mathematics Competitions (AMCs) are competitions given to high school students. The specific dataset we use is comprised of 83 problems from the 2022 and 2023 AMC12 exams, which are given to 12th grade students. Although the original problems are in multiple-choice format, the dataset presents modified versions of the problem which expect an integer solution.
- AIME24 [23]: The American Invitational Mathematics Examination (AIME) is a prestigious high school mathematics competition. It consists of 15 questions meant to be completed in 3 hours and is given to top-scoring students on the AMC exam. Each year, there are two versions of the exam which consist of distinct questions. We train on the 30 problems from both versions of the 2024 exam.
- GPQA [33]: GPQA is a dataset of 448 multiple-choice problems in biology, physics, and chemistry at the PhD level. They are intended to be "Google-proof" in the sense that they require advanced reasoning skills.

Since we are interested in test-time adaptation, and we do not assume access to the ground-truth answer, we use the same dataset for both training and evaluation. Additionally, some of the benchmarks do not have standardized train sets. The exception is GSM8K, where we use the standard train and test sets; this shows that generalization does occur and RENT is not merely overfitting to the test set.

Models. To showcase the generality of our method, we conduct experiments on a wide range of models from different model families and of varying parameter counts. We test on Mistral-7B-Instruct-v0.3, Llama3.1-8B-Instruct, Qwen2.5-1.5B-Instruct, Qwen2.5-Math-1.5B-Instruct, Qwen2.5-7B-Instruct, and Qwen2.5-Math-7B-Instruct.



Note: Error bars may be large as they indicate the standard deviation over the samples, *not* training seeds.

Figure 2: Performance on GSM8K, MATH500, AMC, AIME, and GPQA. The standard deviations reported are over 5, 5, 32, 64, and 10 samples, respectively. Across benchmarks and models, we find that entropy minimization alone is an effective reward for improving the reasoning ability of language models. All models are Instruct models; the "Instruct" is omitted for brevity.

Implementation details. For the RL optimization we use GRPO [39] with a learning rate of 1×10^{-6} and the Adam optimizer. The batch sizes and sampling hyperparameters may vary among models and datasets. We provide a full list of hyperparameters in the Appendix.

4.2 Main Results

Figure 2 shows the performance of models before and after entropy minimization on GSM8K, MATH500, AMC, AIME24, and GPQA. We report standard deviations reported are over 5, 5, 32, 64, and 10 samples, respectively. Note that all models are Instruct models (e.g., Qwen2.5-1.5B refers to Qwen2.5-1.5B-Instruct). Across model families, model sizes, and benchmarks, entropy minimization allows large language models to improve their reasoning skills, without any external supervision. On the Math models such as Qwen2.5-Math-1.5B and Qwen2.5-Math-7B, the base model often struggles at following instructions and therefore the initial score is zero or near zero, and therefore the boost from entropy minimization is quite large. On models that are already proficient at instruction following, we can still see strong performance improvements from entropy minimization. Given the potential pitfall of overconfidence in language models, we performed extensive experimentation to

Figure 3: Accuracy and confidence over the course of training. The trends indicate that accuracy and confidence are indeed highly correlated and therefore it is natural to use confidence as a reward.

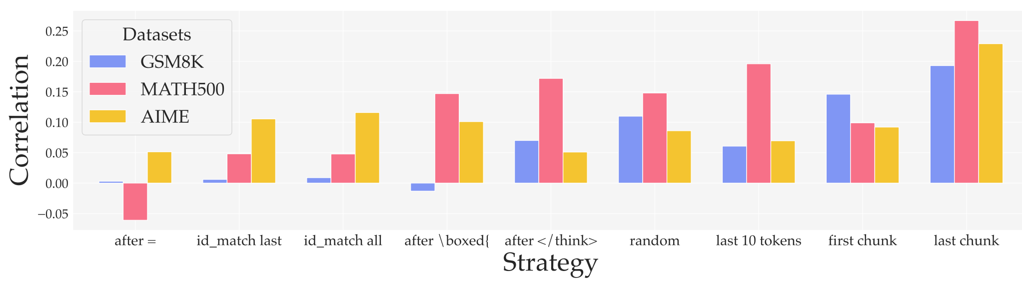


Figure 4: Evaluation (by computing correlation between accuracy and confidence) of various strategies for selecting which tokens to minimize the entropy over. We find the highest correlation between accuracy and confidence in the last few tokens of the response.

ensure empirically that entropy minimization is a robust and generalizable reward function across datasets and models.

4.3 Is It Just Formatting?

It is a well-known issue with reasoning benchmarks that language models can lose points simply because they do not know how to put their answers in the right format. For example, MATH500 expects final answers to be placed in "boxed". A nontrivial amount of engineering effort has gone into both designing prompts that encourage correct formatting and implementing parsers that effectively extract answers from language model responses, in attempts to mitigate this issue. Therefore, one might wonder if, instead of learning to perform complex reasoning, RENT merely encourages the model to put its answers in the right format. Table 1 shows that this is not the case. Models trained with the RENT reward outperform only using a format reward, which simply assigns a binary reward based on whether the correct format is followed in the response. In some cases, the performance of our method is similar to (or even slightly worse than) just using format reward, but of course it is expected that unsupervised RL methods might not always lead to significant improvements. For example, if the benchmark is extremely easy and the model only needs to learn the right format to achieve near-perfect scores, RENT would not outperform format reward. Or, if the benchmark is so hard that it is beyond the model's capabilities altogether, neither method would perform well. However, across datasets and model sizes, we find a consistent improvement over using the format reward and this assures us that the model is actually learning to think through difficult problems and improve its ability to reason.

Table 1: Comparison to RL with a format reward. The best result on each benchmark is indicated in bold. RENT generally outperforms only using a format reward.

	GSM8K	MATH500	AMC	AIME	GPQA
Mistral-7B-Instruct-v0.3					
Baseline	0.381	0.147	0.049	0.002	0.179
w/ Format reward only	0.393	0.150	0.051	0.015	0.240
w/ RENT (Ours)	**0.492**	**0.168**	**0.068**	**0.033**	**0.267**
LLama3.1-8B-Instruct					
Baseline	0.857	0.496	0.221	0.061	0.206
w/ Format reward only	**0.866**	0.533	0.265	**0.086**	0.282
w/ RENT (Ours)	0.859	**0.548**	**0.339**	0.082	**0.332**
Qwen2.5-1.5B-Instruct					
Baseline	0.745	0.548	0.251	0.026	0.247
w/ Format reward only	**0.754**	0.558	0.259	0.054	**0.271**
w/ RENT (Ours)	0.748	**0.597**	**0.298**	**0.072**	0.267
Qwen2.5-Math-1.5B-Instruct					
Baseline	0.852	0.744	0.452	0.092	0.244
w/ Format reward only	0.857	0.756	0.490	0.117	0.276
w/ RENT (Ours)	**0.863**	**0.810**	**0.504**	**0.145**	**0.285**
Qwen2.5-7B-Instruct					
Baseline	0.906	0.762	0.423	0.110	0.311
w/ Format reward only	**0.913**	0.774	0.458	0.156	0.338
w/ RENT (Ours)	0.911	**0.823**	**0.518**	**0.270**	**0.365**
Qwen2.5-Math-7B-Instruct					
Baseline	0.956	0.834	0.495	0.143	0.225
w/ Format reward only	0.957	0.873	0.560	0.154	0.340
w/ RENT (Ours)	**0.967**	**0.882**	**0.591**	**0.167**	**0.400**

4.4 Correlation Between Entropy and Accuracy

Figure 3 shows the accuracy and confidence throughout training Qwen2.5-Math-7B and Qwen2.5-7B-Instruct on the AMC and MATH500 datasets respectively. Critically, as the model improves its confidence via RENT, the accuracy of the model improves as well. This demonstrates the significant correlation between answer confidence and answer accuracy, supporting our initial hypothesis.

4.5 Comparison to Concurrent Work

In this section, we compare RENT to concurrent papers which use intrinsic rewards. We evaluate on GSM8K, MATH500, AMC, AIME, and GPQA and run all experiments with Qwen2.5-7B-Instruct as the baseline model. Table 2 shows comparisons to the following methods:

- Test-Time Reinforcement Learning (TTRL) [61] assigns a reward of 1 to the majority answer and 0 to all other answers. In our experiments, we reimplemented this majority voting reward in our codebase.
- Intuitor [60] uses the forward KL divergence between a uniform distribution and the model's distribution as the reward. In contrast, we use entropy, which is the reverse KL divergence from the uniform distribution. Intuitor is mode-seeking while RENT is mode-covering. We ran the publicly available Intuitor code (which is also implemented on top of verl framework [40]) with the same batch size, epochs and evaluation strategy as RENT for fair comparison.
- Shao et al. [38] suggested that even random or "spurious" rewards could be used to improve reasoning. To compare against spurious rewards, we modify our code to set the reward for every generation randomly to 0 or 1 with equal probability. Intuitively, we believe spurious rewards might work because gradients from correct examples contribute to learning, while gradients from incorrect examples might cancel each other out. Our hypothesis is supported

by work such as Rolnick et al. [34], which shows that learning can still happen even when diluting datasets with incorrect labels.

Empirically, we find that RENT is the best of the four methods on average. Compared to TTRL and Intuitor, performance is similar on most benchmarks except AIME, where RENT outperforms both by a large margin. This is especially interesting since AIME is the hardest benchmark in our evaluations (i.e., the initial accuracy of the model is the lowest). Spurious rewards are not competitive with the other three methods; we conclude that random reward values are not enough and it is indeed beneficial to use meaningful unsupervised rewards that explicitly encourage some measure of confidence.

Table 2: Comparison of RENT with three concurrent papers: TTRL [61], Intuitor [60], and Spurious Rewards [38]. The best result on each benchmark is indicated in bold. RENT is the best-performing method on MATH500, AIME, GPQA, and is the best on average.

	GSM8K	MATH500	AMC	AIME	GPQA	Average
Qwen2.5-7B-Instruct						
w/ TTRL	**0.933**	0.822	0.521	0.172	0.346	0.559
w/ Intuitor (forward KL)	0.929	0.783	**0.525**	0.200	0.337	0.555
w/ Spurious Rewards	0.910	0.774	0.459	0.156	0.342	0.528
w/ RENT (Ours)	0.911	**0.823**	0.518	**0.270**	**0.365**	**0.577**

4.6 Qualitative Samples

Table 3 shows a qualitative sample from GSM8K and Table 4 shows a qualitative sample from AIME. The qualitative samples verify that the model indeed learns meaningful reasoning skills via entropy minimization. It is not merely learning to format its answer correctly, or otherwise collapsing to some other reward-hacking behavior.

Table 3: The model splits the problem into its key steps and correctly calculates the target age

Prompt	system Please reason step by step, and put your final answer within \boxed{}. user Jaymee is 2 years older than twice the age of Shara. If Shara is 10 years old, how old is Jaymee? Let's think step by step and output the final answer within "\boxed{}". assistant
Response	To determine Jaymee's age, we need to follow these steps: 1. Identify Shara's age. 2. Calculate twice Shara's age. 3. Add 2 years to twice Shara's age. Step 1: Shara's age is given as 10 years old. Step 2: Twice Shara's age is calculated as: $$2 \times 10 = 20$$ Step 3: Jaymee is 2 years older than twice Shara's age, so we add 2 years to 20: $$20 + 2 = 22$$ Therefore, Jaymee is $\boxed{22}$ years old.

4.7 Which Tokens to Minimize Entropy Over

We investigated which response tokens are the best to minimize entropy over. To do this, we evaluated the following response token selection strategies:

- **last chunk** - split the response into k chunks, and select all tokens in the last chunk
- **first chunk** - split the response into k chunks, and select all tokens in the first chunk

Table 4: The model identifies the geometric series and succinctly soles the problem afterwards

Prompt	system Please reason step by step, and put your final answer within \boxed{}. user The infinite product $$\sqrt[3]{10} \cdot \sqrt[3]{\sqrt[3]{10}} \cdot \sqrt[3]{\sqrt[3]{\sqrt[3]{10}}} \cdots$$ evaluates to a real number $\sqrt{m}$. What is the value of m? Let's think step by step and output the final answer within \boxed{}. assistant
Response	To find the value of the infinite product $\sqrt[3]{10} \cdot \sqrt[3]{\sqrt[3]{10}} \cdot \sqrt[3]{\sqrt[3]{\sqrt[3]{10}}} \cdots$, we can express each term in the product as a power of 10. The first term is $\sqrt[3]{10} = 10^{1/3}$. The second term is $\sqrt[3]{\sqrt[3]{10}} = \left(10^{1/3}\right)^{1/3} = 10^{1/9}$. The third term is $\sqrt[3]{\sqrt[3]{\sqrt[3]{10}}} = \left(10^{1/9}\right)^{1/3} = 10^{1/27}$. And so on. The infinite product can be written as: $$10^{1/3} \cdot 10^{1/9} \cdot 10^{1/27} \cdots = 10^{1/3+1/9+1/27+\cdots}.$$ The exponent is an infinite geometric series with the first term $a = \frac{1}{3}$ and common ratio $r = \frac{1}{3}$. The sum S of an infinite geometric series is given by: $$S = \frac{a}{1-r} = \frac{\frac{1}{3}}{1-\frac{1}{3}} = \frac{\frac{1}{3}}{\frac{2}{3}} = \frac{1}{2}.$$ So the infinite product evaluates to: $$10^{1/2} = \sqrt{10}.$$ Thus, the value of m is 10. The final answer is: $$\boxed{10}$$

- **last 10 tokens** - select the last k response tokens
- **random** - select k random tokens from the response
- **after </think>** - select all tokens after the thought process (after the "</think>" tag)
- **after \boxed{** - select all tokens after the boxed answer
- **id_match_last** - select the tokens in the model's final answer
- **id_match_all** - select all tokens tokens in the model's final answer and any previous occurrences
- **after =** - minimize all tokens after the equality symbol ("=")

Figure 4 shows the initial correlation between negative entropy and accuracy for each of these strategies on three datasets. While most token selection strategies do result in a positive correlation between confidence and accuracy, we note that the "last chunk" strategy has a significantly higher correlation compared to the "first chunk" strategy. This suggests that the most important tokens to minimize entropy over are tokens that occur later in the response. Furthermore, based on the low correlation results from the "id_match_last" and "id_match_all" strategies, we find that it is not sufficient to simply minimize the entropy of the final answer tokens; this suggests that, counterintuitively, the token-level confidence of the final answer tokens is not well-calibrated to its true response confidence/accuracy.

5 Limitations

Fundamentally, unsupervised learning alone is relatively limited compared to methods which are able to use external supervision for learning. Therefore, it is not surprising that our method is not able to match the performance of methods that have access to the ground-truth answers. It is, of course, a possibility for the model to be confidently wrong. Overconfidence is a well-known issue with language models and these calibration errors can cause RENT to fail catastrophically. It could be dangerous to deploy such an unsupervised learning method in the real world without any safeguards. However, we generally find empirically that confidence does correlate with accuracy and the performance does improve by using confidence alone. This indicates that even if the model is overconfident on some answers, it is well-calibrated overall.

6 Conclusion

We presented RENT, an unsupervised reinforcement learning method which uses entropy as a reward. In our experiments, we showed that by simply minimizing entropy, we can improve the reasoning performance of language models on GSM8K, MATH500, AMC, AIME, and GPQA. Our reward function is general and can be applied on a wide range of domains. We are excited about the possibility of using entropy minimization and, more broadly, unsupervised reinforcement learning to improve the capabilities of machine learning models in regimes where external supervision is unavailable.

7 Acknowledgements

We would like to acknowledge Nikhil Chandak, Shashwat Goel, and Ameya Prabhu for insights on evaluation that helped us to improve our paper.

References

[1] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. *Advances in neural information processing systems*, 32, 2019.

[2] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690, 2024.

[3] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[4] Sahil Chaudhary. Code alpaca: An instruction-following llama model for code generation. `https://github.com/sahil280114/codealpaca`, 2023.

[5] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[7] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[8] Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koeppl, Preslav Nakov, and Iryna Gurevych. A survey of confidence estimation and calibration in large language models. *arXiv preprint arXiv:2311.08298*, 2023.

[9] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, 17, 2004.

[10] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[11] Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Language model cascades: Token-level uncertainty and beyond. *arXiv preprint arXiv:2404.10136*, 2024.

[12] Jiuzhou Han, Wray Buntine, and Ehsan Shareghi. Towards uncertainty-aware language agent. *arXiv preprint arXiv:2401.14016*, 2024.

[13] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

[14] 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. *arXiv preprint arXiv:2103.03874*, 2021.

[15] Bairu Hou, Yujian Liu, Kaizhi Qian, Jacob Andreas, Shiyu Chang, and Yang Zhang. Decomposing uncertainty for large language models through input clarification ensembling. *arXiv preprint arXiv:2311.08718*, 2023.

[16] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[17] Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, 2023.

[18] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

[19] Adam Tauman Kalai and Santosh S Vempala. Calibrated language models must hallucinate. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 160–171, 2024.

[20] Seongun Kim, Kyowoon Lee, and Jaesik Choi. Variational curriculum reinforcement learning for unsupervised discovery of skills. *arXiv preprint arXiv:2310.19424*, 2023.

[21] Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

[22] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896. Atlanta, 2013.

[23] 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, 2024.

[24] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *arXiv preprint arXiv:2203.07814*, 2022.

[25] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

[26] Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. *Advances in Neural Information Processing Systems*, 34:18459–18473, 2021.

[27] Potsawee Manakul, Adian Liusie, and Mark JF Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*, 2023.

[28] Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulo. Autodial: Automatic domain alignment layers. In *Proceedings of the IEEE international conference on computer vision*, pages 5067–5075, 2017.

[29] Zachary Nado, Shreyas Padhy, D Sculley, Alexander D'Amour, Balaji Lakshminarayanan, and Jasper Snoek. Evaluating prediction-time batch normalization for robustness under covariate shift. *arXiv preprint arXiv:2006.10963*, 2020.

[30] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[31] Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. *Advances in Neural Information Processing Systems*, 37:55249–55285, 2024.

[32] Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and N Lawrence. Covariate shift and local learning by distribution matching. *Dataset Shift in Machine Learning*, pages 131–160, 2008.

[33] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

[34] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

[35] Kuniaki Saito, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Semi-supervised domain adaptation via minimax entropy. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8050–8058, 2019.

[36] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in neural information processing systems*, 33:11539–11551, 2020.

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

[38] Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, et al. Spurious rewards: Rethinking training signals in rlvr. *arXiv preprint arXiv:2506.10947*, 2025.

[39] 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.

[40] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

[41] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. *arXiv preprint arXiv:1802.08735*, 2018.

[42] Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, et al. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*, 2023.

[43] Claudio Spiess, David Gros, Kunal Suresh Pai, Michael Pradel, Md Rafiqul Islam Rabin, Amin Alipour, Susmit Jha, Prem Devanbu, and Toufique Ahmed. Calibration and correctness of language models for code. *arXiv preprint arXiv:2402.02047*, 2024.

[44] Baochen Sun, Jiashi Feng, and Kate Saenko. Correlation alignment for unsupervised domain adaptation. *Domain adaptation in computer vision applications*, pages 153–171, 2017.

[45] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. *arXiv preprint arXiv:2305.14975*, 2023.

[46] Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

[47] Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation. *arXiv preprint arXiv:2307.03987*, 2023.

[48] Yuvraj Virk, Premkumar Devanbu, and Toufique Ahmed. Enhancing trust in llm-generated code summaries with calibrated confidence scores. *arXiv preprint arXiv:2404.19318*, 2024.

[49] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

[50] Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Y Wu, and Zhifang Sui. Math-shepherd: A label-free step-by-step verifier for llms in mathematical reasoning. *arXiv preprint arXiv:2312.08935*, 2023.

[51] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[52] Liangru Xie, Hui Liu, Jingying Zeng, Xianfeng Tang, Yan Han, Chen Luo, Jing Huang, Zhen Li, Suhang Wang, and Qi He. A survey of calibration process for black-box llms. *arXiv preprint arXiv:2412.12767*, 2024.

[53] Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *arXiv preprint arXiv:2306.13063*, 2023.

[54] Tianyang Xu, Shujin Wu, Shizhe Diao, Xiaoze Liu, Xingyao Wang, Yangyi Chen, and Jing Gao. Sayself: Teaching llms to express confidence with self-reflective rationales. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5985–5998, 2024.

[55] Daniel Yang, Yao-Hung Hubert Tsai, and Makoto Yamada. On verbalized confidence scores for llms. *arXiv preprint arXiv:2412.14737*, 2024.

[56] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[57] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pages 11920–11931. PMLR, 2021.

[58] Dongkeun Yoon, Seungone Kim, Sohee Yang, Sunkyoung Kim, Soyeon Kim, Yongil Kim, Eunbi Choi, Yireun Kim, and Minjoon Seo. Reasoning models better express their confidence. *arXiv preprint arXiv:2505.14489*, 2025.

[59] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

[60] Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*, 2025.

[61] Yuxin Zuo, Kaiyan Zhang, Shang Qu, Li Sheng, Xuekai Zhu, Biqing Qi, Youbang Sun, Ganqu Cui, Ning Ding, and Bowen Zhou. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

A Hyperparameters

A full list of hyperparameters can be found in Table 5.

Table 5: Hyperparameters.

Hyperparameter	Value
Max prompt length	1024
Max response length	3072
Batch size	64 GSM8K 500 MATH500 80 AMC 30 AIME 64 Countdown 196 GPQA
Policy mini batch size	32 GSM8K 32 MATH500 80 AMC 30 AIME 32 Countdown 32 GPQA
Policy micro batch size per GPU	8
Learning rate	1×10^{-6}
Weight decay	0.01
Learning rate warmup	Constant
Optimizer	Adam
Temperature	1.0 for train 0.8 for validation
Top k	-1
Top p	1
Number of samples per example n	5
Remove padding	True
Use KL loss	True
KL loss coefficient	0.001
Clip ratio	0.2
Grad clip	1.0