RETHINKING FINE-TUNING WHEN SCALING TEST-TIME COMPUTE: LIMITING CONFIDENCE IMPROVES MATHEMATICAL REASONING

Feng Chen* **& Allan Raventós***
Stanford University
Stanford, CA 94305, USA
{fengc, aravento}@stanford.edu

Nan Cheng
University of Michigan
Ann Arbor, MI 48109, USA
nancheng@umich.edu

Surya Ganguli & Shaul Druckmann
Stanford University
Stanford, CA 94305, USA
{sganguli, shauld}@stanford.edu

ABSTRACT

Recent progress in large language models (LLMs) highlights the power of scaling test-time compute to achieve strong performance on complex tasks, such as mathematical reasoning and code generation. This raises a critical question: how should model training be modified to optimize performance under a subsequent test-time compute strategy and budget? To explore this, we focus on pass@N, a simple test-time strategy that searches for a correct answer in N independent samples. We show, surprisingly, that training with cross-entropy (CE) loss can be *misaligned* with pass@N in that pass@N accuracy *decreases* with longer training. We explain the origins of this misalignment in terms of model overconfidence induced by CE, and experimentally verify our prediction of overconfidence as an impediment to scaling test-time compute via pass@N. Furthermore we suggest a principled, modified training loss that is better aligned to pass@N by limiting model confidence and rescuing pass@N test performance. Our algorithm demonstrates improved mathematical reasoning on MATH and MiniF2F benchmarks under several scenarios: (1) providing answers to math questions; and (2) proving theorems by searching over proof trees. Overall our work underscores the importance of co-designing two traditionally separate phases of LLM development: training-time protocols and test-time search and reasoning strategies.

1 INTRODUCTION

Scaling test-time compute has been integral to unprecedented improvements in LLMs' reasoning skills for complex tasks such as math and coding. Thus, test-time compute has emerged as a new dimension for improving LLMs, leading to a key tradeoff between allocating additional compute to inference versus pretraining (Snell et al., 2024). Diverse test-time strategies include Chain-of-Thought (CoT) (Wei et al., 2022), tree-of-thought (Yao et al., 2023), self-consistency (Wang et al., 2023), self-reflection (Shinn et al., 2023), self-critique (Saunders et al., 2022), self-verification (Weng et al., 2023) and Monte-Carlo tree search (Zhao et al., 2023). These have shown great success in boosting model performance in the post-training phase or at inference time. More recently, OpenAI's O1 model (OpenAI, 2024) and DeepSeek's R1 model (DeepSeek-AI: Daya Guo et al., 2025) have combined some of these strategies with reinforcement learning to generate high-quality reasoning traces for problems of various difficulty levels, demonstrating clear performance improvements as more test-time compute is allocated.

These successes fit into a broader paradigm in which a frontier model is first fine-tuned on a reasoning task with supervised fine-tuning (SFT) (Wei et al., 2022; Ouyang et al., 2022; Chung et al.,

*Equal Contribution.

2022), and then a test-time algorithm is applied to further improve its performance (Yao et al., 2023; Wang et al., 2023; Chen et al., 2021). Many test-time algorithms are independent of the fine-tuning process. As a result, the fine-tuning is agnostic to and thus decoupled from the test-time algorithm (Chow et al., 2024). However, for a given choice of test-time strategy and compute budget, it is not *a priori* clear which fine-tuning approach, including the loss objective, would be best aligned with the test-time strategy so as to maximize the test accuracy under the overall strategy.

Our work studies the problem of aligning fine-tuning and test-time algorithms. We consider what is perhaps the simplest setting, supervised fine-tuning with CE loss under the pass@N test-time strategy. This setting reveals a case of misalignment: standard SFT is not the right choice for maximizing performance under pass@N. We believe that this kind of misalignment presents itself in several combinations of fine-tuning/test-time approaches, motivating our thorough study in this paper. Our main contributions are,

- We identify a misalignment between standard fine-tuning with CE loss and the pass@N coverage metric at test time. (Sec. 4.1)
- We develop and experimentally verify a framework that suggests this misalignment arises from overconfidence induced by training on CE loss. (Sec. 4.2 and 4.3)
- We propose a new loss function that directly optimizes the pass@N coverage metric, demonstrating consistent improvement over the CE loss objective, achieving superior accuracy frontiers in MATH and MiniF2F. (Sec. 5.1 to 5.3)
- We extend our algorithm to more complex test-time scenarios including searching over proof-trees of varying shapes to improve automated theorem proving, and answering math questions using Chain-of-Thought reasoning traces, demonstrating improved mathematical reasoning performance in both cases. (Sec. 5.3 and 5.4)

2 Related Works

Test-time compute and pass@N strategy. Multiple works have looked into the interaction between training and test-time strategies. Jones (2021) demonstrates a tradeoff between train- and test-time compute in the toy model of a board game. OpenAI's O1 model demonstrates remarkable performance gains when scaling test-time compute (OpenAI, 2024). Closely related to our work, Brown et al. (2024) observed an exponentiated power law between coverage and the number of samples for in-context-learning evaluation. Snell et al. (2024), in turn, have explored compute-optimal strategies for effectively scaling test-time compute. This paper focuses primarily on pass@N test-time strategy. Gui et al. (2024) showed that the best-of-N sample distribution is nearly optimal for alignment with a reward model. To mitigate the excessive test-time compute costs associated with best-of-N sampling, Gui et al. (2024); Sessa et al. (2024) proposed alignment algorithms to distill this best-of-N sample distribution. Li et al. (2024) found that fine-tuning with cross-entropy loss can limit output diversity and proposed a maximum entropy-based method to address this issue.

Post-training for mathematical reasoning. Multiple post-training techniques have been proposed to improve mathematical reasoning in LLMs. Instruction-tuning and reinforcement learning with human feedback have been shown to boost model performance on math (Yue et al., 2024; Lightman et al., 2024; Uesato et al., 2022), while continued training on math- or code-specific domain data enhances models' reasoning abilities for downstream mathematical tasks (Lewkowycz et al., 2022; Azerbayev et al., 2024; Yang et al., 2024; Shao et al., 2024; Ying et al., 2024). Rejection-sampling (Zelikman et al., 2022) and self-improvement techniques (Qi et al., 2024), in turn, are useful to augment the training data for SFT. More recent approaches (OpenAI, 2024; DeepSeek-AI: Daya Guo et al., 2025) have incorporated reinforcement learning and achieved exceptional reasoning capabilities in various domains, such as math and coding. Although our paper primarily focuses on supervised fine-tuning to enhance pretrained models' math capabilities, our loss function can be applied to other settings that train under CE loss, such as continual training, instruction-tuning, and data augmentation.

Data pruning and hard example mining. The interpretation of the loss function we derive below can be related to data pruning and hard example mining. Data selection is often applied to curate high-quality datasets for pretraining (Marion et al., 2023), where Sorscher et al. (2022) shows that pruning easy samples can improve pretraining loss scaling as a function of dataset size. Zhou et al.

(2023); Ye et al. (2025) demonstrate that supervised fine-tuning on even a very small dataset can yield remarkably strong performance in alignment and reasoning tasks. On the other hand, hard example mining focuses on identifying and emphasizing challenging samples to improve model performance (Shrivastava et al., 2016). In the domain of mathematical reasoning, Tong et al. (2024) found a difficulty imbalance in rejection-sampled datasets and showed that more extensive training on difficult samples improves model performance.

Our paper is closely related to a concurrent paper by Chow et al. (2024), which derives a similar training objective for RL to directly optimize for the best-of-N test-time strategy.

3 PROBLEM SETUP

Given a vocabulary set W, we consider a dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{M}$, where $x^{(i)} \in W^{n_i}$ is a prompt, $y^{(i)} \in W^{m_i}$ is its ground-truth completion, and n_i and m_i are the prompt and completion lengths. In the context of math, $x^{(i)}$ is the problem statement and $y^{(i)}$ is its solution. To model the conditional distribution $p(y^{(i)}|x^{(i)})$ we use an autoregressive transformer model (Vaswani et al., 2017), which is traditionally trained by minimizing the cross-entropy loss.

$$\mathcal{L}_{\text{CE}} = - \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \log \hat{p}(y|x) \tag{1}$$

where $\hat{p}$ denotes the model's distribution. To use and evaluate the model at test time, we assume the existence of an efficient oracle verifier V which takes as input an (x, y) pair and returns $V(x, y) = 1$ if y is a correct completion of x and otherwise returns $V(x, y) = 0$. Practical examples of verifiers include compilers or pre-defined unit tests for coding problems, or automatic proof checkers in mathematical theorem proving. In such applications, a simple method, known as pass@N, for trading test-time compute for accuracy involves sampling N completions from $\hat{p}$ given the test prompt x and applying the verifier V to all of them to search for a correct solution. The probability of a correct answer is then no longer the probability that 1 completion is correct, but rather the probability that *at least one* of N is correct. This probability, for a dataset $\mathcal{D}$, is given by the pass@N coverage metric

$$\mathcal{C}_{\mathcal{D}}^{N} = \mathop{\mathbb{E}}_{\substack{x\sim\mathcal{D} \\ \{y_i\}_{i\in[N]} \overset{\text{i.i.d.}}{\sim} \hat{p}(\cdot|x)}} \mathbb{P}(\exists j \in [N]\, s.t.\, V(x, y_j) = 1). \tag{2}$$

Minimizing the CE loss in Eq. (1) is equivalent to maximizing the pass@1 metric $\mathcal{C}_{\mathcal{D}}^{1}$ on a training set $\mathcal{D}$. But if we scale up test-time compute so that the pass@N metric $\mathcal{C}_{\mathcal{D}}^{N}$ on a test set $\mathcal{D}$ for $N \gg 1$ is the relevant performance metric, is $\mathcal{L}_{\text{CE}}$ still a good training loss, or can we do better?

4 MISALIGNMENT BETWEEN CE LOSS AND PASS@N

4.1 THE CE LOSS INDUCES OVERFITTING FOR PASS@N

To understand the impact of training with CE loss on pass@N test performance, we fine-tune Llama-3-8B-base (Grattafiori et al., 2024) on the MATH (Hendrycks et al., 2021) dataset. We start from the base model rather than LLama-3-8B-Instruct to avoid potential leakage of the MATH dataset into LLama-3-8B-Instruct through post-training. We follow Lightman et al. (2024) and use $12,000$

Table 1: Pass@N coverage metric on the MATH test set for a Llama-3-8B-base model fine-tuned with CE loss on direct answers from the MATH training set. Surprisingly, Pass@N test accuracy at large N decreases with number of training epochs.

	PASS@1	PASS@16	PASS@256	PASS@4K
EPOCH 1	4.4%	30.0%	**65.2%**	**82.5%**
EPOCH 2	5.3%	**31.4%**	64.5%	80.0%
EPOCH 3	6.5%	28.7%	54.5%	79.2%
EPOCH 4	**7.4%**	22.9%	44.5%	63.0%

problems for training and the remaining 500 for testing. Here we train the model to provide a direct answer. We will discuss training with CoT in Sec. 5.4 for MATH reasoning traces.

Table 1 reveals that the pass@N performance $\mathcal{C}_{\mathcal{D}}^{N}$ on a test set monotonically increases with the number of training epochs *only* for $N = 1$, when minimizing CE loss is equivalent to maximizing $\mathcal{C}_{\mathcal{D}}^{1}$. However, for $N \geq 16$, minimizing CE loss during training does *not* monotonically increase $\mathcal{C}_{\mathcal{D}}^{N}$ at test; indeed for $N \geq 256$, pass@N test performance, remarkably, monotonically *decreases* with the number of training epochs, despite the fact that pass@1 performance monotonically *increases*. This corresponds to a novel type of overfitting, in which test performance degrades over training, likely due to a mismatch between the test time (pass@N) and training time (pass@1) strategies.

4.2 Overfitting, confidence, and explore-exploit tradeoff

What are the origins of this overfitting? First, we show that in the simple case of a *single* problem x, overfitting cannot occur. Let $\hat{p}(x)$ denote the probability assigned by the model to all correct answers for problem x. Then the pass@1 coverage is $\mathcal{C}^1 = \hat{p}(x)$ while the pass@N coverage is $\mathcal{C}^N = 1 - (1 - \hat{p}(x))^N = 1 - (1 - \mathcal{C}^1)^N$. This formula for $\mathcal{C}^N$ in *the single problem* case obeys:

Lemma 1. *$\forall N,\ N' > 0$, $\mathcal{C}^N$ is monotonic in $\mathcal{C}^{N'}$.*

Thus increasing pass@N′ coverage implies increasing pass@N coverage for any N and N'. When $N' = 1$, this implies minimizing CE loss maximizes pass@N coverage.

However, lemma 1 can fail when there is more than one problem. To understand this in a simple setting, consider a test set with two problems x_1 and x_2 with corresponding unique correct answers y_1 and y_2. Consider two models. The first model assigns probabilities $\hat{p}_1(y_1|x_1) = 1$ and $\hat{p}_1(y_2|x_2) = 0$. If we think of the probability a model assigns to an answer as its *confidence* in that answer, this model is highly confident and correct for problem x_1, but highly confident and *wrong* for x_2. Its pass@1 coverage is thus 50%. Moreover, since it always gets x_1 right and x_2 wrong, its pass@N coverage remains at 50%. In contrast, consider a second model which assigns probabilities $\hat{p}_2(y_1|x_1) = \hat{p}_2(y_2|x_2) = 0.1$. This model has high confidence (0.9) but unfortunately on incorrect answers. Therefore its pass@1 accuracy is only 10%. However, it is willing to explore or hedge by placing some low confidence (0.1) on the other answer, which happens to be correct. Thus if this model samples N times, the pass@N coverage increases with N, eventually approaching 100% as $N \to \infty$. Thus the first model outperforms the second in terms of pass@1 but not pass@N for large N. This indicates a tradeoff amongst policies: those that do better on pass@1 may not do better at pass@N. Of course the best one can do is be confident *and* correct, corresponding to the optimal model with $p^*(y_1|x_1) = p^*(y_2|x_2) = 1$. However, this toy example reveals that if one cannot guarantee correctness, it can be beneficial to limit confidence and explore more solutions, especially if one can sample at large N.

To demonstrate more generally the existence of tradeoffs between confident exploitation of a few answers versus unconfident exploration of many answers, as a function of the number of passes N, we prove two lemmas. To set up these lemmas, given any problem, let $\hat{p}_i$ denote the model probability or confidence assigned to answer i, and assume answers are sorted from highest to lowest confidence so that $\hat{p}_i \geq \hat{p}_{i+1}, \forall\, i \geq 1$. Thus $\hat{p}_1$ is the model's *maximal confidence* across all answers. Moreover, let p_i be the probability that answer i is *actually* correct for the problem. Assume the model policy is approximately well calibrated so that higher confidence implies higher or equal probability of being correct, i.e. $p_i \geq p_{i+1}, \forall\, i \geq 1$. We empirically verify that the model trained with CE loss on the MATH dataset approximately satisfies this assumption (Fig. 6). Indeed, optimal policies maximizing pass@N coverage in Eq. (2) are approximately well calibrated (See lemma 4). Thus p_1 is the model's *maximal accuracy* across all answers. We prove:

Lemma 2 (Upper bound on max confidence)**.** *Assume a max accuracy p_1, and assume $\sum_{i=1}^{k} p_i \geq 1 - \epsilon$ for some $0 < \epsilon < 1$. Then optimal policies maximizing pass@N coverage in Eq. (2), subject to above accuracy and calibration constraints, must have max confidence upper bounded as*

$$\hat{p}_1 \leq 1 - \frac{k-1}{(k-1)^{\frac{N}{N-1}} p_1^{\frac{1}{N-1}} \left(1 - p_1 - \epsilon\right)^{\frac{1}{1-N}} + 1}.$$

This upper bound is monotonically decreasing in N, implying that at large N, an optimal policy for pass@N must limit its max confidence to be low, thereby favoring exploration and discouraging exploitation. Furthermore, we prove:

Lemma 3 (Lower bound on max confidence)**.** *Assume top two accuracies $p_1 > p_2$. Then optimal policies maximizing pass@N coverage in Eq.* (2)*, subject to above accuracy and calibration constraints, must have max confidence obeying*

$$\hat{p}_1 \geq 1 - \frac{(1 - p_1 + p_2) p_1^{\frac{1}{1-N}} p_2^{\frac{1}{N-1}}}{1 - p_1 + p_2 + p_1^{\frac{1}{1-N}} p_2^{\frac{N}{N-1}}}.$$

This lower bound is a monotonically decreasing function of N, implying that at small N, optimal policies for pass@N must have high max confidence, thereby favoring exploitation and discouraging exploration. Note in the limit $N \to 1^+$, the lower bound is always 1, recovering the intuitive result that the optimal policy for pass@1 is to place 100% confidence on the highest accuracy answer.

4.3 Overconfidence prevents improvements from scaling test-time compute

To summarize the consequences of our lemmas above, among the space of approximately well calibrated policies in which higher model confidence on an answer is correlated with higher accuracy on the answer, lemma 2 suggests that at large N it is beneficial to unconfidently explore by assigning low model confidence to many answers, while lemma 3 suggests that at small N it is beneficial to confidently exploit by assigning high model confidence to one or a few answers. These lemmas make a prediction that could explain the empirical observation in Table 1 that pass@N test performance for large N degrades over epochs when pass@1 is maximized at training time: namely, maximization of pass@1 makes the model overconfident, thereby preventing good performance on pass@N.

To test this theoretical prediction of model overconfidence, we estimated the max confidence of the model as $\hat{p}(y_{\text{greedy}}|x)$ where y_{greedy} is the greedy completion to x obtained by sampling autoregressively from the model $\hat{p}$, and at each step selecting the token with the highest probability. $\hat{p}(y_{\text{greedy}}|x)$ approximates the max confidence $\hat{p}_1$ in lemmas 2 and 3. For the model fine-tuned on MATH with CE loss, we plot the distribution of $\hat{p}(y_{\text{greedy}}|x)$ over the test set in Fig. 1 (a). This demonstrates the model becomes progressively more confident about its greedy completion over training, thereby confirming our theoretical insights. In Fig. 1 ((b)) we see why this is a problem: only some of the model's highly confident answers are correct. Thus the model becomes overconfident *and* largely wrong. Scaling test-time compute cannot easily rescue such a model, as over multiple samples, it is likely to confidently provide the same wrong answers, explaining the origins of poor pass@N test performance.

4.4 Easy data drives overconfidence

We have shown above that overconfidence limits performance gains from scaling test-time compute. Here we investigate whether all training data contribute equally to the drop in pass@N test performance. To quantify this effect, we define the test loss as the negative log-probability of coverage, $\mathcal{L}^N_{\text{test}} = -\sum_{(x,y)\in\mathcal{D}^{\text{test}}} \log \mathcal{C}^N_{(x,y)}$, where $\mathcal{C}^N_{(x,y)}$ denotes the coverage for a single test example (x, y). To measure how a training batch $\mathcal{B}$ influences the final test loss, we compute the directional derivative $\nabla_g \mathcal{L}^N_{\text{test}}$ for the gradient direction $g = -\sum_{(x_i,y_i)\in\mathcal{B}} \nabla \ell_{\text{CE}}(x_i, y_i)$, where ℓ_{CE} is the standard CE loss on a single example. A negative value, $\nabla_g \mathcal{L}^N_{\text{test}} < 0$, indicates that a gradient step in the direction of g decreases the test loss.

To analyze the impact of data difficulty, we group training data by difficulty level (as specified in the MATH dataset) and use the directional derivative to examine how a batch of **previously unseen** data from a **given difficulty level** would affect $\mathcal{L}^N_{\text{test}}$ at different stages of training. Early in training, we observe that $\nabla_g \mathcal{L}^N_{\text{test}}$ is negative across all difficulty levels, indicating that data from all difficulty levels contribute positively to reducing the test loss (Fig. 2, left). However, near the end of the first epoch, training on easier examples (difficulty levels 1 and 2) no longer improves performance, and the easiest examples (level 1) actively degrade pass@N test performance (Fig. 2, right). This shows how continued training on easy data can harm test performance when scaling test-time compute.

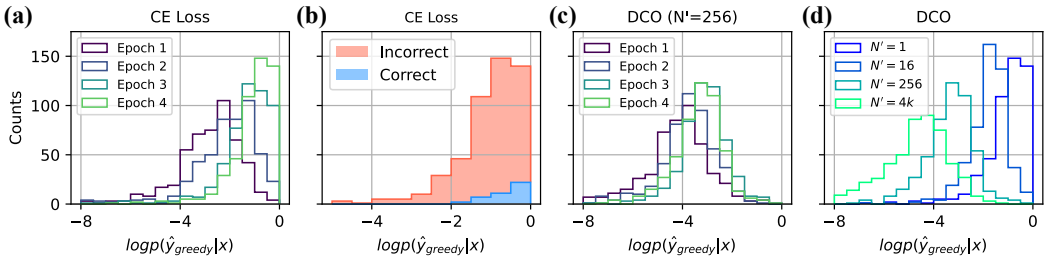

Figure 1: **A model trained with CE loss becomes overconfident in its greedy completions, which harms its pass@N coverage; our proposed DCO objective limits this overconfidence.** We finetune a Llama-3-8B base model on the MATH dataset to produce direct answers without CoT. $\hat{y}_{\text{greedy}}$ is the model's greedy completion. **(a)** The model trained with CE loss assigns progressively larger confidences $\hat{p}(\hat{y}_{\text{greedy}}|x)$ to its greedy completions over the course of training. **(b)** At the end of the training, only a small portion of the model's highly confident completions are correct. This will harm the model's pass@N performance when scaling up N. **(c)** Same as (a) but shown for the DCO loss with $N = 256$. Relative to the CE loss, the model trained on DCO shows a much milder overconfidence effect. **(d)** The confidence distribution of the greedy completions after 4 epochs with DCO for various choices of N. As N increases, the model's confidence on the greedy completion is more stringently limited, directly as a consequence of the overconfidence regularizer F.

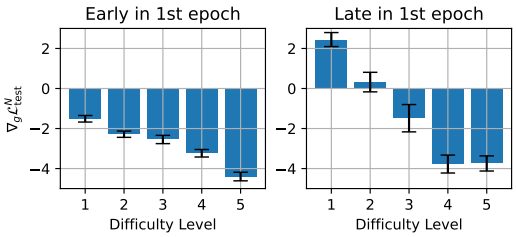

Figure 2: **Easy data drives overconfidence and degrades performance when scaling test-time compute**. For the experiments in Fig. 1, we compute the directional derivative $\nabla_g \mathcal{L}^N_{\text{test}}$ at two points during the first training epoch: 11% (early) and 86% (late). At each stage, the gradient direction $g = -\sum_{(x_i,y_i)\in\mathcal{B}} \nabla \ell_{\text{CE}}(x_i, y_i)$ is evaluated on batches of **previously unseen** training data of each difficulty level defined in the MATH dataset. Early in training, data from all difficulty levels contribute to decreasing the test loss (*left*). However, later in training, easier examples (difficulty level 2) provide no further benefit, while the easiest examples (difficulty level 1) actively degrade test performance (*right*). The plotted $\nabla_g \mathcal{L}^N_{\text{test}}$ is an average over batches of unseen data, and we use $N = 256$, corresponding to pass@256, for these plots.

5 Direct Coverage Optimization

5.1 A solution that naturally prevents overconfidence

The misalignment between CE loss and pass@N coverage suggests a simple solution: *directly optimize* pass@N coverage at training time when the pass@N strategy is to be used at test-time. We thus propose the Direct Coverage Optimization (DCO) objective, $\mathcal{L}^N_{\text{DCO}} = \mathbb{E}_{(x,y)\sim\mathcal{D}}\, \ell^N_{\text{DCO}}(x, y)$. Assuming each prompt x has a unique best completion y, the pass@N coverage is given by $\mathcal{C}^N = 1 - (1 - \hat{p}(y|x))^N$. From this, we define the loss for a single example (x, y) as,

$$\ell^N_{\text{DCO}}(x, y) = -\log\left(1 - (1 - \hat{p}(y|x))^N\right), \tag{3}$$

This loss naturally prevents model overconfidence, as can be seen via its gradient

$$\nabla_\theta \ell^N_{\text{DCO}}(x, y) = F\left(N, \hat{p}(y|x)\right) \nabla_\theta \ell_{\text{CE}}(x, y) \tag{4}$$

where $F(N, \hat{p}(y|x)) = \frac{N(1-\hat{p}(y|x))^{N-1}\hat{p}(y|x)}{1-(1-\hat{p}(y|x))^N}$ is an overconfidence regularization factor that multiplies the standard CE gradient. Note that $F(N, \hat{p}(y|x)) = 1$ for $N = 1$, so DCO reduces to CE for

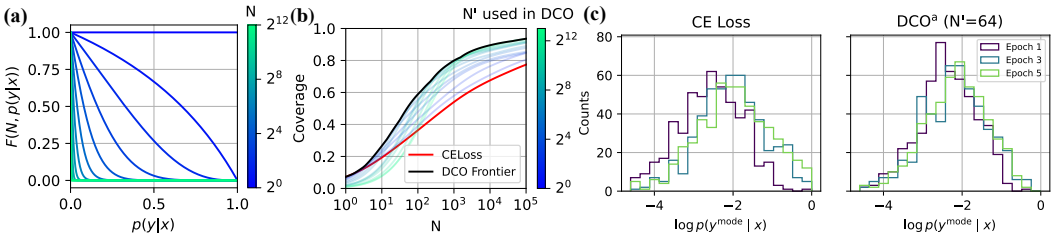

Figure 3: **(a) The DCO objective limits overconfidence by attenuating gradients for examples on which the model is highly confident.** We plot the confidence regularization factor $F(N, \hat{p}(y|x))$ in Eq. (4) which attenuates CE loss gradients to obtain DCO loss gradients. **(b) DCO improves on CE for pass@N test coverage over a broad range of N and traces a Pareto-optimal frontier.** We fine-tune Llama-3-8B base models on the MATH dataset to produce direct answers. We fine-tune one model for 4 epochs using CE loss, and several models under the $\mathcal{L}^{N'}_{\text{DCO}}$ objective, for choices of N' indicated by color. We plot pass@N test coverage as a function of N, with each curve (solid red or faint blue-green) corresponding to *one* fine-tuned model. The black curve is a Pareto-optimal frontier traced by the max of coverage curves for DCO over all N'. **(c) Overconfidence persists in CoT fine-tuning with CE loss, which DCO^a successfully limits.** We fine-tune a Llama-3-8B base model on the MATH dataset on CoT traces. We plot the distribution of the estimated model confidences $\hat{p}(y^{\text{mode}}|x)$ over samples in the test set at various points in training. For CE loss (*(b)*), the model becomes more confident in its most likely answers, as the confidence distribution shifts slightly to the right as training progresses. For DCO^a loss (*(c)*), the rightward shift in the confidence distribution of the most likely answer over training time is more limited less; DCO^a again limits overconfidence relative to CE loss.

pass@1. Furthermore, $F(N, \hat{p}(y|x))$ monotonically decreases in the model confidence $\hat{p}(y|x)$ (see Fig. 3 (a)). Thus, gradients for examples (x, y) on which the model is more confident are attenuated, and this attenuation is stronger for larger N. This justifies the interpretation of $F(N, \hat{p}(y|x))$ as a regularizer that prevents model overconfidence. Indeed, for large N, $F(N, \hat{p}(y|x)) \approx 0$ for confidence $\hat{p}(y|x) \gtrsim 1/N$. As soon as model confidence on an example exceeds $1/N$, its gradient becomes negligible. Thus interestingly, aligning training and test better through DCO naturally yields a simple emergent regularization of model overconfidence, which was itself identified as an impediment to performance gains through scaling test-time compute in Fig. 1 (a).

We note in practice, the introduction of F can lead to some samples in a batch contributing minimally to the current gradient step, thereby reducing the effective batch size. To maintain a stable effective batch size, we introduce a threshold ϵ, and if for a given example, (x, y), $F(N, \hat{p}(y|x)) < \epsilon$, we replace the example with a new one.

5.2 DCO CAN PREVENT OVERCONFIDENCE AND RESCUE TEST-TIME SCALING

We next test whether DCO can rescue test-time scaling by preventing model overconfidence. We first perform experiments on the MATH dataset in this section, and then in the next section we perform experiments on the LeanDojo automated theorem proving benchmark (Yang et al., 2023), a dataset extracted from the math library of LEAN4 (mathlib Community, 2020).

We fine-tune the LLama-3-8B-base model for 4 epochs on the MATH training set using DCO for $N = 256$ and confirm that the model is far less confident on its greedy completion than when trained with CE loss after multiple training epochs (compare Fig. 1 (c) and Fig. 1 (a)). Moreover, training with DCO at larger N yields lower model greedy confidences at the end of training (Fig. 1 (d)), consistent with lemma 2.

We next assess pass@N test performance as a function of N for a variety of models trained by minimizing $\mathcal{L}^{N'}_{\text{DCO}}$ for different values of N' (Fig. 3 (b)). For any given N in pass@N, there is an optimal N' for the training loss $\mathcal{L}^{N'}_{\text{DCO}}$ that maximizes pass@N test coverage, yielding a Pareto optimal performance frontier (black curve) that is achieved when N' is close to N. In particular the model trained with CE loss (equivalent to pass@1 maximization at training) performs poorly relative to the Pareto frontier at large N when the pass@N strategy is used at test time (red curve

below black at large N). Conversely, models trained with DCO at large N' perform poorly relative to the Pareto frontier when the pass@N strategy is used at test time with small N (green curves below black at small N).

Together these results indicate that the alignment of the training loss $\mathcal{L}_{\text{DCO}}^{N'}$ with the test time strategy pass@N, with N' close to N, is crucial for obtaining Pareto optimal performance. Moreover, these results once again confirm the tradeoff between exploration and exploitation, with good performance using pass@N test-time strategies requiring high (low) exploration with low (high) confidence at large (small) N. Indeed, this tradeoff prevents achieving Pareto optimality using pass@N at test-time for all N via training with DCO at any *single* value of N'.

5.3 Improved theorem proving via ensembled tree search through a modified step-wise DCO

To further test our method in realistic settings with a verifier, we conduct experiments in theorem proving using an interactive proof assistant on the LeanDojo benchmark (Yang et al., 2023) extracted from the math library of LEAN4 (mathlib Community, 2020). In this task, at each step i of the proof, the model is prompted with the current proof state $x[i]$, and it outputs a proof tactic $y[i]$ with a trainable model probability $\hat{p}(y[i]|x[i])$. Example tactics are: "simplify", "prove by contradiction", etc. with their associated arguments. The proof assistant then takes the sampled tactic $y[i]$, verifies whether it is a valid tactic, and if so, returns the next proof state $x[i+1]$ after the tactic $y[i]$ is applied. The model then samples the next tactic $y[i+1]$. At test time this interactive process between the model and the assistant continues until either: (1) it terminates in an invalid tactic; (2) hits a maximal allowed search depth set by computational constraints; or (3) terminates in a successful proof of the initial proof goal as verified by the proof assistant.

In our experiments, we use LEAN4 (Moura & Ullrich, 2021) as the formal proof language. We start from the model Qwen2.5-Math-1.5B (Yang et al., 2024), and fine-tune it on the LeanDojo benchmark (Yang et al., 2023). In contrast to solving math problems above where the model *first* autoregressively generates an entire answer y according to $\hat{p}(y|x)$ and *then* the entire y is verified for correctness, in theorem proving we *must* obtain all correct tactics $y[i]$, as verified by the proof assistant, at *every* proof step i. A straightforward application of DCO fine-tuning in this setting then involves replacing the model confidence $\hat{p}(y|x)$ in Eq. (3) on a single math answer y, with the model confidence on an *entire* successful, complete k step proof $\hat{p}(y[0], ..., y[k-1]|x[0]) = \prod_{i=0}^{k-1} \hat{p}(y[i]|x[i])$. Because of the chain rule, the DCO gradient on a single proof has the same form as in Eq. (4) but with $F(N, \hat{p}(y|x))$ replaced with $F(N, \prod_{i=0}^{k-1} \hat{p}\,(y[i]|x[i]))$.

We naively applied this DCO fine-tuning method with $N = 4$k and evaluated the resulting model, as well as a baseline model trained with CE loss, on MiniF2F. As a test strategy, we used pass@4k. The baseline model fine-tuned with CE loss achieves a proof success rate of 37.4%, while model fine-tuned with DCO at $N =$4k achieves a 38.7% success rate. Thus, a naive application of DCO to theorem proving by matching the parameter N in DCO to the parameter N in pass@N achieves only a modest improvement.

However, in theorem proving, since *every single* intermediate proof step tactic $y[i]$ must be valid, it is natural to consider a *step-wise generalization* of DCO. For example, at each step i, the model chooses a tactic $y[i]$ with confidence $\hat{p}(y[i]|x[i])$. Our proposed step-wise DCO optimizes this single-step confidence according to Eq. (4) as before, except now the *step-wise* confidence regularizer is given by $F(N_{\text{eff}}, \hat{p}(y[i]|x[i]))$. Here one can think of N_{eff} as a step-wise DCO hyperparameter that controls the exploration width at every proof step. In essence, the confidence regularizer prevents the confidence of any chosen tactic from becoming much higher than $1/N_{\text{eff}}$. Since the sum of the confidences over all tactics at each step must be 1, this means that at test time, model exploration at each step corresponds to searching on a search tree in which each proof state $x[i]$ allows the exploration of approximately only N_{eff} tactics. Thus using N_{eff} in step-wise DCO during fine-tuning selects the approximate branching factor N_{eff} of the model's proof search tree at test time.

In particular, a valid (partial) proof of length k will have probability of order N_{eff}^{-k}. Thus small N_{eff} limits the exploration at test time to a tree with small branching factor, but allows longer proofs with larger numbers of steps k to have higher sampling probability. This limited width search strategy should work well at test time using pass@N as long as two conditions hold: (1) a successful proof

Table 2: Proof success rate testing with pass@4k on Mathlib and MiniF2F, training with DCO$^{\text{step}}$ for various choices of N_{eff}.

N_{eff}	1 (CE loss)	4	8	16	32	Ensemble of all N_{eff}'s	$N_{\text{eff}} = 1$ (5x test compute)
Mathlib	55.6%	56.4%	56.1%	**56.5%**	55.8%	**62.2%**	57.0%
MiniF2F	37.4%	39.0%	**39.5%**	37.0%	37.4%	**43.6%**	39.5%

of length k is present in the search tree of branching factor N_{eff}; and (2) the N in pass@N at test time is large enough that this proof of probability $O(N_{\text{eff}}^{-k})$ is selected with high probability in N passes (which starts to occur as soon as $N \gtrsim N_{\text{eff}}^k$).

Conversely, larger N_{eff} allows for more exploration at test time using a wider search tree of larger branching factor, but it makes finding longer proofs with large k harder, since the approximate success condition for pass@N of $N > N_{\text{eff}}^k$ is harder to satisfy at fixed N and large N_{eff} and k. However, this wide exploration strategy can work well for short proofs with small k if a successful short proof does not lie in the limited width search tree obtained at small N_{eff}, but does lie in a wider search tree at larger N_{eff}.

We note that the mean and median lengths of the proofs in the training set is 4.2 and 2 respectively. Thus we explore fine-tuning on LeanDojo with the step-wise DCO algorithm DCO$^{\text{step}}$ with N_{eff} ranging from 1 (corresponding to standard CE-loss training) to $N_{\text{eff}} = 32$. We evaluated the pass@N test performance for $N = 4$k of each of the resulting models on Mathlib and MiniF2F test sets, finding improved performance at larger N_{eff} compared to the baseline $N_{\text{eff}} = 1$ CE loss training (Table 2). Interestingly, we find that the optimal N_{eff} increases with more passes at test time (Table 4), similar to the Pareto frontier in Fig. 3 (b).

Since different choices of N_{eff} correspond to very different search strategies at test time, specializing in finding short proofs on wide search tress for large N_{eff} and long proofs on narrow search trees for small N_{eff}, we hypothesized that ensembling these methods could significantly boost performance. We found this was indeed the case (ensemble entry in Table 2). Of course the Ensemble strategy samples 5×4k times, so a proper baseline is CE loss with the same test-time compute of pass@N with $N = 20$k. We obtained the proof success rate for this baseline to be 57.0% on Mathlib and 39.5% on MiniF2F. The ensemble strategy outperforms this stringent baseline with significant excesses of 5.2% on Mathlib and 4.1% on MiniF2F.

Overall, these results indicate that varying N_{eff} in step-wise DCO at training time allows a diversity of tree search strategies trading depth and breadth that can be effectively exploited by simply scaling test time compute via pass@N.

5.4 Approximate DCO also improves MATH answering with chain-of-thought reasoning

In Sec. 5.2 on solving problems in MATH, the model is trained to directly give an answer y to a problem x. In this case, one can implement DCO at training time by explicitly computing the model confidence $\hat{p}(y|x)$ and using it in Eq. (4). However, in an alternate powerful Chain-of-Thought (CoT) training paradigm, the training data consists of triplets (x, c, y) where x and y are the problem and answer as before, but now c is a CoT reasoning trace that explains how to derive y from x. The model is trained on the triplet (x, c, y), and at test time, when given a new problem x', it generates a reasoning trace c' and then gives answer y'. Importantly, the model is evaluated on whether its answer y' is correct *independent* of whatever reasoning trace c' it emits.

Thus, to estimate the probability $\hat{p}(y|x)$ that the model assigns to any answer y, one can no longer compute it directly as in the direct answer case in Sec. 5.2. Instead, one must marginalize over all possible reasoning traces c to obtain $\hat{p}(y|x) = \sum_c \hat{p}(y, c|x)$. Because exact marginalization is intractable, to employ DCO in Eq. (4) in the CoT setting, we replace the marginalization with a Monte Carlo estimate of $\hat{p}(y|x)$, and insert this estimate into the overconfidence regularization factor F in Eq. (4). We call this algorithm approximate DCO, denoted by DCO$^{\text{a}}$. We also compute Monte Carlo estimates of the probability the model assigns to its most likely final answer, $\hat{p}(y^{\text{mode}}|x)$, as a

Table 3: Pass@N test coverage on MATH obtained from fine-tuning Llama-3-8B-base on CoT traces using CE or DCO$^{\text{a}}$ loss with $N = 64$. $\pm$ error is standard error of the mean.

	EPOCH 1		EPOCH 3		EPOCH 5	
PASS@N	CE	DCO$^{\text{a}}$	CE	DCO$^{\text{a}}$	CE	DCO$^{\text{a}}$
PASS@1	4.3±0.1%	**5.0**±0.1%	**9.0**±0.1%	8.3±0.1%	**9.9**±0.1%	7.8±0.1%
PASS@16	30.2±0.7%	**34.2**±0.6%	41.6±0.4%	**42.6**±0.6%	40.9±0.4%	**42.8**±0.5%
PASS@64	51.2±1.2%	**55.3**±1.0%	61.7±0.4%	**63.1**±0.8%	60.9±0.6%	**64.3**±0.6%

proxy for the greedy completion probability $\hat{p}(y^{\text{greedy}}|x)$ used to study overconfidence in the direct answer setting in Sec. 5.2. Note that in the CoT setting the greedy answer would correspond to a single *most* likely reasoning trace, but we want the probability of the most likely *final* answer obtained by marginalizing over *all* reasoning traces.

We conduct experiments on the MATH dataset. For the baseline experiment, we first fine-tune a Llama-3-8B-base model with CE loss on the golden CoT solutions. We find in Table 3 that pass@1 coverage robustly improves over the course of CE loss training. Intriguingly, in the CoT setting, overfitting induced by the misalignment of CE loss at training time with the pass@N strategy at test time, is not as severe as in the direct answer setting in Table 1. If overfitting indeed arises from the model overconfidence induced by CE loss, we would expect that minimizing CE loss in the CoT setting does not make the model as overconfident as it does in the direct answer case. This is indeed confirmed in Fig. 3 (b)=, where the confidence of the modal answer shifts slightly to the right over epochs, whereas it shifts much further right over epochs in the direct answer case in Fig. 3 (a).

We next explore the properties of DCO$^{\text{a}}$ with $N = 64$. We find that training with DCO$^{\text{a}}$ ($N' = 64$) underperforms relative to training with CE loss for pass@1 at test time, but outperforms when the test strategy is pass@N for larger N (Table 3). This is similar to what happens in Fig. 3 (b) in the direct answer case. The largest improvement occurs when training with DCO$^{\text{a}}$ at $N' = 64$ is aligned with pass@N at $N = 64$. Furthermore, with our understanding developed in Sec. 4.2, we expect that DCO$^{\text{a}}$ training should lower model confidence relative to CE loss training, thereby explaining the improved pass@64 performance. We confirm this in Fig. 3 (c), which reveals that DCO$^{\text{a}}$ successfully prevents the model from becoming overconfident in its most likely answer y^{mode}.

Overall these results once again validate our algorithmic approaches and understanding, now in a CoT reasoning setting. In essence, improved performance by scaling test time compute via a pass@N strategy at large N can be best obtained by aligning the training strategy via choosing DCO$^{\text{a}}$ with the same N, and the origin of the performance improvement comes from limiting model overconfidence.

6 DISCUSSION

In summary, all of our results suggest the need for a tight co-design of two traditionally separate phases of LLM development: (1) model training or fine-tuning and (2) test-time search/reasoning strategies and budget. If the former is misaligned with the latter, then more training can actually *impair* performance gains from scaling test-time compute. On the other hand, if they are properly co-designed, end-to-end training and test time performance can be far more effective. We have shown how to modify standard cross-entropy loss for training to be better aligned to a pass@N strategy for large N at testing. Moreover, we have suggested and empirically confirmed why this co-design of training loss and pass@N strategy is essential, because optimal policies for pass@N at large N should unconfidently explore while the optimal policies for pass@N at small N should confidently exploit.

This notion of co-design opens up many more theoretical and empirical questions associated with different test-time strategies such as self-verification and MCTS. Furthermore, one can go beyond test-time search to recursive self-improvement, where new solutions found by search are filtered and then used to retrain the model. A theoretical understanding of the capabilities achievable by recursive self-improvement through co-design of training, search, and filtering, remains an outstanding research problem.

ACKNOWLEDGMENTS

We would like to thank Zhinan Cheng, Clémentine Dominé, Marco Fumero, David Klindt, Daniel Kunin, Ben Sorscher and Atsushi Yamamura for helpful discussions. S.G. thanks the James S. McDonnell Foundation, Simons Foundation, NTT Research, Schmidt Foundation and an NSF CAREER Award for support. S.D. and F.C. were partially supported by the McKnight Foundation, the Simons Foundation and an NIH CRCNS grant R01DC020874.

REFERENCES

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024. doi: 10.1145/3620665.3640366. URL https://pytorch.org/assets/pytorch2-2.pdf.

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/d8e1344e27a5b08cdfd5d027d9b8d6de-Paper.pdf.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4WnqRR915j.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024. URL https://arxiv.org/abs/2407.21787.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Yinlam Chow, Guy Tennenholtz, Izzeddin Gur, Vincent Zhuang, Bo Dai, Sridhar Thiagarajan, Craig Boutilier, Rishabh Agarwal, Aviral Kumar, and Aleksandra Faust. Inference-aware fine-tuning for best-of-n sampling in large language models. *arXiv preprint arXiv:2412.15287*, 2024. URL https://arxiv.org/pdf/2412.15287.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin,

Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022. URL https://arxiv.org/abs/2210.11416.

Haowei Zhang DeepSeek-AI: Daya Guo, Dejian Yang et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate, 2022.

Lin Gui, Cristina Gârbacea, and Victor Veitch. Bonbon alignment for large language models and the sweetness of best-of-n sampling, 2024. URL https://arxiv.org/abs/2406.00832.

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. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Andy L. Jones. Scaling scaling laws with board games, 2021. URL https://arxiv.org/abs/2104.03113.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 3843–3857. Curran Associates, Inc., 2022.

Ziniu Li, Congliang Chen, Tian Xu, Zeyu Qin, Jiancong Xiao, Ruoyu Sun, and Zhi-Quan Luo. Entropic distribution matching in supervised fine-tuning of llms: Less overfitting and better diversity, 2024. URL https://arxiv.org/abs/2408.16673.

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*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. When less is more: Investigating data pruning for pretraining llms at scale, 2023. URL https://arxiv.org/abs/2309.04564.

The mathlib Community. The lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, POPL '20. ACM, January 2020. doi: 10.1145/3372885.3373824. URL http://dx.doi.org/10.1145/3372885.3373824.

Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In André Platzer and Geoff Sutcliffe (eds.), *Automated Deduction – CADE 28*, pp. 625–635, Cham, 2021. Springer International Publishing. ISBN 978-3-030-79876-5.

OpenAI. Learning to reason with llms, 2024. URL https://openai.com/index/learning-to-reason-with-llms/.

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. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=-P7G-8dmSh4.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. Mutual reasoning makes smaller llms stronger problem-solvers, 2024. URL https://arxiv.org/abs/2408.06195.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators, 2022. URL https://arxiv.org/abs/2206.05802.

Pier Giuseppe Sessa, Robert Dadashi, Léonard Hussenot, Johan Ferret, Nino Vieillard, Alexandre Ramé, Bobak Shariari, Sarah Perrin, Abe Friesen, Geoffrey Cideron, Sertan Girgin, Piotr Stanczyk, Andrea Michi, Danila Sinopalnikov, Sabela Ramos, Amélie Héliou, Aliaksei Severyn, Matt Hoffman, Nikola Momchev, and Olivier Bachem. Bond: Aligning llms with best-of-n distillation, 2024. URL https://arxiv.org/abs/2407.14622.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023.

Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 19523–19536. Curran Associates, Inc., 2022.

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. DART-math: Difficulty-aware rejection tuning for mathematical problem-solving. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=zLU21oQjD5.

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, 2022. URL https://arxiv.org/abs/2211.14275.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=s4xIeYimGQ.

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

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J Prenger, and Animashree Anandkumar. Leandojo: Theorem proving with retrieval-augmented language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 21573–21612. Curran Associates, Inc., 2023.

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. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning, 2025. URL https://arxiv.org/abs/2502.03387.

Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, Yudong Wang, Zijian Wu, Shuaibin Li, Fengzhe Zhou, Hongwei Liu, Songyang Zhang, Wenwei Zhang, Hang Yan, Xipeng Qiu, Jiayu Wang, Kai Chen, and Dahua Lin. Internlm-math: Open math large language models toward verifiable reasoning, 2024. URL https://arxiv.org/abs/2402.06332.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MAmmoTH: Building math generalist models through hybrid instruction tuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=yLClGs770I.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 15476–15488. Curran Associates, Inc., 2022.

Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for large-scale task planning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 31967–31987. Curran Associates, Inc., 2023.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. Lima: Less is more for alignment. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 55006–55021. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/ac662d74829e4407ce1d126477f4a03a-Paper-Conference.pdf.

A PROOFS

A.1 PROOF OF APPROXIMATELY WELL CALIBRATION UNDER OPTIMAL POLICY

Lemma 4. *Given any problem, let $\hat{p}_i$ denote the model probability or confidence assigned to answer i, and assume answers are sorted from highest to lowest confidence so that $\hat{p}_i \geq \hat{p}_{i+1}, \forall\, i \geq 1$. Let p_i be the probability that answer i is actually correct for the problem. Then optimal policies maximizing pass@N coverage in Eq.* (2) *is approximately well calibrated. Higher confidence implies higher or equal probability of being correct, i.e. $p_i \geq p_{i+1}, \forall\, i \geq 1$.*

Proof. For a given problem, let M be the total number of answers the model can generate. Let $\hat{p}_i$ denote the model probability or confidence assigned to answer i under optimal policy maximizing pass@N coverage in Eq. (2), and assume answers are sorted from highest to lowest confidence so that $\hat{p}_i \geq \hat{p}_{i+1}, \forall\, i \geq 1$. Let p_i be the probability that answer i is *actually* correct for the problem. Let A_i be the event that the i^{th} ranked answer is not chosen in the N tries and B_i be the event that i^{th} ranked answer is chosen in the N tries. Let 1_{B_i} be the indicator function of B_i. Then, given an event ω, the probability of getting the correct answer is

$$\sum_{i=1}^{M} p_i 1_{B_i}(\omega). \tag{5}$$

Hence, the expected probability of getting the correct answer is

$$\mathbb{E}(\sum_{i=1}^{M} p_i 1_{B_i}) = \sum_{i=1}^{M} p_i \mathbb{P}(B_i) = \sum_{i=1}^{M} p_i (1 - \mathbb{P}(A_i)) = 1 - \sum_{i=1}^{M} p_i \mathbb{P}(A_i) = 1 - \sum_{i=1}^{M} p_i (1 - \hat{p}_i)^N, \tag{6}$$

If the current strategy is already optimal, any small change of $(\hat{p}_1, ..., \hat{p}_M)$ would not increase the expected probability of getting the correct answer. Now suppose we don't always have $p_i \geq p_{i+1}$, in other words, $p_j < p_{j+1}$ for some $j \geq 1$. If $\hat{p}_j < \hat{p}_{j+1}$, we have

$$\mathbb{E}(\sum_{i=1}^{M} p_i 1_{B_i}) < 1 - \sum_{i=1}^{j-1} p_i (1 - \hat{p}_i)^N - \sum_{i=1}^{j+1} p_i (1 - \hat{p}_i)^N - p_{j+1}(1 - \hat{p}_j)^N - p_j (1 - \hat{p}_{j+1})^N, \tag{7}$$

where the last inequality says that we can increase the expected probability of getting the correct answer by swapping the confidence on answer originally labeled as j and $j+1$, which is a contradiction. If $\hat{p}_j = \hat{p}_{j+1}$, then for all $\delta > 0$ small enough we always have

$$\mathbb{E}(\sum_{i=1}^{M} p_i 1_{B_i}) < 1 - \sum_{i=1}^{j-1} p_i (1 - \hat{p}_i)^N - \sum_{i=1}^{j+1} p_i (1 - \hat{p}_i)^N - p_j (1 - \hat{p}_j - \delta)^N - p_{j+1}(1 - \hat{p}_{j+1} + \delta)^N, \tag{8}$$

which also contradict with the current policy being optimal. □

A.2 PROOF OF LEMMA 4.2

Proof. Let M be the total number of answers the model can generate, p_i be the probability that the i^{th} ranked response is correct and N be the number of tries the model can make. We assume that $p_i \geq p_{i+1}, \forall i \geq 1$. Let A_i be the event that the i^{th} ranked answer is not chosen in the N tries and B_i be the event that i^{th} ranked answer is chosen in the N tries. Let 1_{B_i} be the indicator function of B_i. Then, given an event ω, the probability of getting the correct answer is

$$\sum_{i=1}^{M} p_i 1_{B_i}(\omega). \tag{9}$$

Hence, the expected probability of getting the correct answer is

$$\mathbb{E}(\sum_{i=1}^{M} p_i 1_{B_i}) = \sum_{i=1}^{M} p_i \mathbb{P}(B_i) = \sum_{i=1}^{M} p_i (1 - \mathbb{P}(A_i)) = 1 - \sum_{i=1}^{M} p_i \mathbb{P}(A_i) \tag{10}$$

Now, let a_i be the probability of choosing the i^{th} ranked response in a single try. Then

$$\mathbb{P}(A_i) = (1 - a_i)^N. \tag{11}$$

Hence the expected probability of choosing the correct answer is

$$\mathbb{E}(\sum_{i=1}^{M} p_i 1_{B_i}) = 1 - \sum_{i=1}^{M} p_i (1 - a_i)^N. \tag{12}$$

Optimal strategy maximizes Eq. (12). Let

$$\pi_1(\hat{p}_1, ..., \hat{p}_m) = \hat{p}_1, \tag{13}$$

be the projection onto the first entry, then

$$\hat{p}_1 = \pi_1(\text{argmin}\{\sum_{i=1}^{M} p_i (1 - a_i)^N\}). \tag{14}$$

It is worth noting that first

$$\pi_1(\text{argmin}\{\sum_{i=1}^{M} p_i (1 - a_i)^N\}) \leq \pi_1(\text{argmin}\{\sum_{i=1}^{k} p_i (1 - a_i)^N + \sum_{i=k+1}^{M} p_i\}). \tag{15}$$

Second, given p_1 and $\sum_{i=1}^{k} p_i \geq 1 - \epsilon$, $\pi_1(\text{argmin}\{\sum_{i=1}^{k} p_i (1 - a_i)^N + \sum_{i=k+1}^{M} p_i\})$ is bounded from above by

$$\pi_1(\text{argmin}\{p_1 (1 - a_1)^N + \sum_{i=2}^{k} \frac{1 - p_1 - \epsilon}{k - 1} (1 - a_i)^N + \sum_{i=k+1}^{M} p_i\}). \tag{16}$$

Third, fix a_1, Jensen's inequality tells us that

$$p_1 (1 - a_1)^N + \sum_{i=2}^{k} \frac{1 - p_1 - \epsilon}{k - 1} (1 - a_i)^N \geq p_1 (1 - a_1)^N + \sum_{i=2}^{k} \frac{1 - p_1 - \epsilon}{k - 1} (1 - \frac{1 - a_1}{k - 1})^N. \tag{17}$$

Combining all three inequalities above we have

$$\hat{p}_1 = \pi_1(\text{argmin}\{\sum_{i=1}^{M} p_i (1 - a_i)^N\}) \leq \text{argmin}\{p_1 (1 - a_1)^N + \sum_{i=2}^{k} \frac{1 - p_1 - \epsilon}{k - 1} (1 - \frac{1 - a_1}{k - 1})^N\}. \tag{18}$$

The right hand side of Eq. (18) can be easily computed, and $\text{argmin}\{p_1 (1 - a_1)^N + \sum_{i=2}^{k} \frac{1 - p_1 - \epsilon}{k - 1} (1 - \frac{1 - a_1}{k - 1})^N\}$ equals

$$1 - \frac{k - 1}{(k - 1)^{\frac{N}{N-1}} p_1^{\frac{1}{N-1}} (1 - p_1 - \epsilon)^{\frac{1}{1-N}} + 1}. \tag{19}$$

This concludes the proof of Lemma 4.2. □

A.3 Proof of Lemma 4.3

Proof. Let s be the smallest integer such that $p_1 + s p_2 \geq 1$. We have first,

$$\hat{p}_1 = \pi_1(\text{argmin}\{\sum_{i=1}^{M} p_i (1 - a_i)^N\}) \geq \pi_1(\text{argmin}\{p_1 (1 - a_1)^N + \sum_{i=2}^{s+1} p_2 (1 - a_i)^N\}). \tag{20}$$

Second, for a fixed a_1, Jensen's inequality tells us that

$$p_1 (1 - a_1)^N + \sum_{i=2}^{s+1} p_2 (1 - a_i)^N \geq p_1 (1 - a_1)^N + \sum_{i=2}^{s+1} p_2 (1 - \frac{1 - a_1}{s})^N \tag{21}$$

Therefore, we have

$$\pi_1(\text{argmin}\{p_1(1-a_1)^N+\sum_{i=2}^{s+1}p_2(1-a_i)^N\}) = \text{argmin}\{p_1(1-a_1)^N+\sum_{i=2}^{s+1}p_2(1-\frac{1-a_1}{s})^N\} \quad (22)$$

The right hand side of Eq. (22) is

$$\text{argmin}\{p_1(1-a_1)^N+\sum_{i=2}^{s+1}p_2(1-\frac{1-a_1}{s})^N\} = 1-\frac{sp_1^{\frac{1}{1-N}}p_2^{\frac{1}{N-1}}}{s+p_1^{\frac{1}{1-N}}p_2^{\frac{1}{N-1}}} \quad (23)$$

Hence we have

$$\hat{p}_1 \geq 1-\frac{sp_1^{\frac{1}{1-N}}p_2^{\frac{1}{N-1}}}{s+p_1^{\frac{1}{1-N}}p_2^{\frac{1}{N-1}}} \quad (24)$$

Since s is the smallest integer such that $p_1+sp_2 \geq 1$, we have

$$1+p_2 > p_1+sp_2, \quad (25)$$

Combining Eq. (25) and Eq. (24) we arrive at

$$\hat{p}_1 \geq 1-\frac{(1-p_1+p_2)p_1^{\frac{1}{1-N}}p_2^{\frac{1}{N-1}}}{1-p_1+p_2+p_1^{\frac{1}{1-N}}p_2^{\frac{N}{N-1}}}. \quad (26)$$

□

B EXPERIMENTAL DETAILS

All experiments are performed on machines with 8 NVIDIA H100 GPUs or 8 NVIDIA A100 GPUs. Our codebase uses PyTorch (Ansel et al., 2024), Accelerate (Gugger et al., 2022), and deepspeed (`https://github.com/microsoft/DeepSpeed`) to enable efficient training with memory constraints, and vllm (Kwon et al., 2023) for efficient inference. Code to reproduce the results in the paper are available at https://github.com/allanraventos/refine.

MATH For experiments with MATH dataset, we fine-tune the LLama-3-8B-base (Grattafiori et al., 2024) on the MATH (Hendrycks et al., 2021) dataset. We start from the base model rather than LLama-3-8B-Instruct to avoid potential leakage of the MATH dataset into LLama-3-8B-Instruct through post-training process. We follow Lightman et al. (2024) and use $12,000$ problems for training and the remaining 500 for testing. In Sec. 4 and 5.2 and Figs. 1 and 3, we fine-tune the model for 4 epochs with a learning rate of 2e-5 and batch size 64. We adopt a linear learning rate warmup in the first 20 steps. For experiments with DCO, some of the data may have an extreme factor value, if the model is already quite confident in the problem. To maintain an approximately fixed batch size, we set a threshold of 0.3 on the factor. Training data with a factor lower than the threshold will be replaced. For the CoT experiments in Sec. 5.4, we use learning rate 2e-5 and batch size 128, with the same learning rate warmup.

Theorem proving We adopt the random train and test split as introduced in Yang et al. (2023). The random test set includes 2,000 theorems. We fine-tune the model Qwen2.5-Math-1.5B (Yang et al., 2024) on the training set for 3 epochs with learning rate 1e-5 and batch size 64. We adopt a linear learning rate warmup in the first 20 steps. To evaluate the model, we use LeanDojo (Yang et al., 2023) to interact with the proof assistant. We impose a maximum clock-time budget for each theorem, in addition to limiting the number of passes per problem. For experiments with 4k passes, the clock-time budge is fixed at 5,000 seconds. In order to avoid model going infinitely deep in the searching tree, we limit the maximum proof steps to be under 50 steps.

DCO$^{\text{a}}$ objective. The DCO$^{\text{a}}$ introduced in Sec. 5.4 is an approximation for the DCO.To construct a batch of size B with DCO$^{\text{a}}$, we process batches of samples sequentially; for each batch, we run online inference on each of the samples and discard all samples with probability of success rate larger than the p^{thresh} depending on N'. We choose to discard the samples which has a factor lower than 0.01, corresponding to $p^{\text{thresh}} = 0.1$ for $N' = 64$. This process continues until we have enough training data for a single batch. In Fig. 4, we plot the number of discarded samples as a function of training step for our DCO$^{\text{a}}$ experiments (same ones as in Table 3).

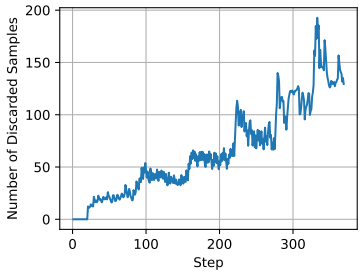

Figure 4: Number of discarded samples as a function of training step for the DCO$^{\text{a}}$ experiments. The step structure reflects the model revisiting examples it has seen previously in training, where each step closely matches the start of a new epoch.

Implementing online inference. Throughout our experiments and analysis, we extensively use the open-source vllm package (Kwon et al., 2023) for efficient inference. Integrating inference into the training loop to enable training under the DCO$^{\text{a}}$ objective, as in Sec. 5.4, poses a challenging implementation problem. We solve this problem by placing vllm worker processes, that together perform inference on all GPUs, in a separate process group and use Ray (`https://github.com/ray-project/ray`) to isolate them from the training loop. This enables running concurrent training and inference on the same set of GPUs. We believe this inference in the loop setup will be useful to the community, as it enables straightforward implementation of all sorts of online data filtering approaches for LLM training.

C ADDTIONAL RESULTS

C.1 THEOREM PROVING

In Table 4, we show additional results for model performance under the DCO$^{\text{STEP}}$ objective. We find that the optimal N_{eff} grows with increasing passes, agreeing with results in Sec. 5.2 and 5.4. We also conduct expert iteration (Anthony et al., 2017; Polu et al., 2023) on Mathlib with theorems that do not have proof traces in the training set. We use pass@1k to prove those theorems. We find that our algorithm achieves a stronger improvement over the baseline for pass@4k after the 1$^{\text{st}}$ iteration. This improvement might result from the fact that the models can prove more easy theorems where the model has a higher confidence. As a result, we believe our method will perform better with expert iteration.

Table 4: Success rate on lean-dojo benchmark random test set trained with DCO$^{\text{step}}$.

	EXPERT ITERATION 0					EXPERT ITERATION 1		
DCO$^{\text{STEP}}$	PASS@16	PASS@64	PASS@256	PASS@1K	PASS@4K	PASS@16	PASS@256	PASS@4K
$N_{\text{eff}} = 1$ (CE)	**30.0%**	38.75%	46.05	50.75%	55.55%	40.3%	52.65%	58.55%
$N_{\text{eff}} = 4$	**30.15%**	**39.5%**	**47.2%**	**52.95%**	56.35%	**40.8%**	**53.05%**	59.45%
$N_{\text{eff}} = 8$	**30.2%**	38.9%	**47.15%**	**52.7%**	56.1%	40.1%	**53.25%**	59.5%
$N_{\text{eff}} = 16$	28.65%	46.7%	46.45%	**52.9%**	**56.5%**	39.05%	52.8%	**60.05%**
$N_{\text{eff}} = 32$	26.05%	46.7%	45.6%	51.5%	55.8%	37.05%	52.2%	59.15%
ENSEMBLE	40.6%	49.15%	54.6%	59.0%	62.15%	49.05%	59.3%	64.8%

C.2 PLOT OF THE UPPER BOUND AND THE LOWER BOUND

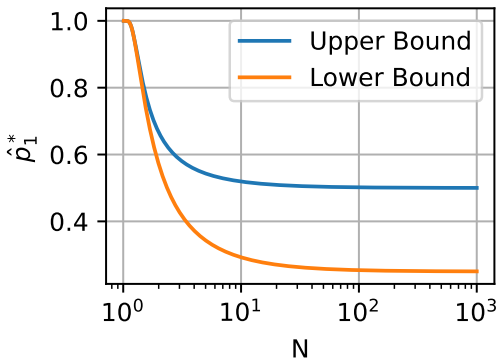

Figure 5: **Plot of the upper bound (Eq. (18)) and the lower bound (Eq. (24)).** We plot the upper bound (blue) and lower bound (orange) for $p_1 = \frac{1}{2}$, $p_2 = \frac{1}{4}$, $\epsilon = \frac{1}{4}$ and $k = 2$. Both the upper bound and the lower decrease monotonically in N and they both tends to 1 as $N \to 1^+$.

C.3 Empirical Evaluation of the approximately well calibrated assumption

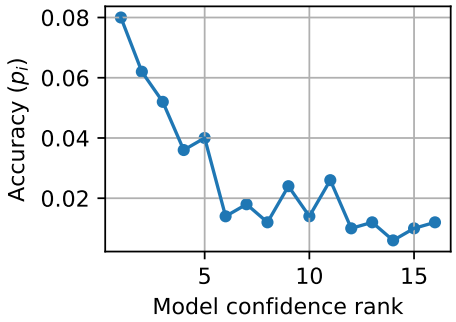

Figure 6: We empirically verify the assumption that models are approximately well-calibrated. To do this, we fine-tune a Llama-3-8B-base model with CE loss on the MATH dataset and perform beam search with a width of 256 to obtain the top 16 most probable completions. We then measure the accuracy of these completions at each confidence rank. The results demonstrate that test accuracy decreases approximately monotonically with model confidence rank, supporting the assumption.

C.4 The data dependency of confidence and factor

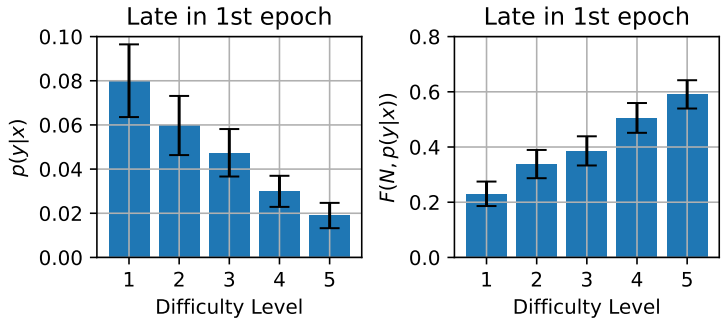

Figure 7: **Model is more confident on easy problems; DCO improves test-time scaling by regularizing the easy examples.** We fine-tune a Llama-3-8B-base model on the MATH dataset with CE loss and plot the model confidence $p(y|x)$ and the factor $F(N, p(y|x))$ at 86% of the first training epoch. Both model confidence and factor are evaluated on the **unseen** data grouped by difficulty level from the MATH dataset. Model confidence decreases with increasing difficulty, whereas the regularization factor increases with problem difficulty. As a result, DCO effectively regularizes contribution from easy examples. This regularization mitigates the potential detrimental effects of overconfidence from easy examples as discussed in Sec. 4.4. We use $N = 256$ corresponding to pass@256 for the plots.