

On the Effect of Negative Gradient in Group Relative Deep Reinforcement Optimization

Anonymous Authors¹

Abstract

Reinforcement learning (RL) has become popular in enhancing the reasoning capabilities of large language models (LLMs), with Group Relative Policy Optimization (GRPO) emerging as a widely used algorithm in recent systems. Despite GRPO’s widespread adoption, we identify a previously unrecognized phenomenon we term Lazy Likelihood Displacement (LLD), wherein the likelihood of correct responses marginally increases or even decreases during training. This behavior mirrors a recently discovered misalignment issue in Direct Preference Optimization (DPO), attributed to the influence of negative gradients. We provide a theoretical analysis of GRPO’s learning dynamic, identifying the source of LLD as the naive penalization of all tokens in incorrect responses with the same strength. To address this, we develop a method called NTHR, which downweights penalties on tokens contributing to the LLD. Unlike prior DPO-based approaches, NTHR takes advantage of GRPO’s group-based structure, using correct responses as anchors to identify influential tokens. Experiments on math reasoning benchmarks demonstrate that NTHR effectively mitigates LLD, yielding consistent performance gains across models ranging from 0.5B to 3B parameters.

1. Introduction

Reinforcement learning (RL) has become increasingly popular for improving reasoning capabilities of large language models (LLMs) (Guo et al., 2025; Jaech et al., 2024; Team et al., 2023). In particular, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) has emerged as a widely adopted algorithm in RL training pipelines for reasoning

tasks. Models such as DeepSeek-R1 (Guo et al., 2025), DeepSeek-Math (Shao et al., 2024), Med-R1 (Lai et al., 2025), and Search-R1 (Jin et al., 2025) have successfully employed GRPO to achieve notable performance gains in code generation, mathematical problem solving, medical reasoning, and retrieval-augmented generation. These successes highlight GRPO’s growing importance as a tool for aligning models to task-specific behaviors through rule-based or heuristic rewards.

Despite its empirical success, the optimization behavior of GRPO remains insufficiently studied, in part due to its recency. In this work, we focus on the impact of *negative gradients* in GRPO, which arise when the advantage function is negative—indicating that the selected action is worse than the average action at a given state (e.g., an incorrect or suboptimal response). In such cases, the learning algorithm decreases the probability of selecting this action. We draw a conceptual connection to Direct Preference Optimization (DPO) (Rafailov et al., 2023), which introduces negative gradients when penalizing dis-preferred responses. Recent studies (Razin et al., 2024; Pal et al., 2024; Yuan et al., 2024; Ren & Sutherland, 2024) have revealed that negative gradients can lead to an arguably counterintuitive behavior: a simultaneous decrease in the probabilities of preferred responses (y^+). Motivated by this insight, we examine whether similar gradient dynamics in GRPO might cause analogous issues and investigate their impact on policy learning. This effect, termed misalignment (Ren & Sutherland, 2024) or likelihood displacement (Razin et al., 2024), is undesirable as it often degrades model performance by diverting probability mass away from optimal responses. In this work, we empirically observe (see Fig. 1) that GRPO can suffer from what we call Lazy Likelihood Displacement (LLD): a failure to sufficiently increase, or even a decrease in, the likelihood of correct answers during training. To explain this phenomenon, we provide an analysis of GRPO’s update dynamics and identify that LLD stems from penalizing certain tokens in dispreferred responses. These updates inadvertently reduce the likelihood of correct answers due to shared structural or semantic features between positive and negative responses.

While several approaches have been proposed to mitigate

¹Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

the reduced probabilities of preferred responses (y^+) in DPO (Razin et al., 2024; Ren & Sutherland, 2024), they either compromise data efficiency or rely on training procedures that are not directly compatible with GRPO’s online learning paradigm. For instance, Razin et al. (2024) recommend discarding training examples where the likelihood of positive responses y^+ decreases, thereby avoiding harmful updates—but this strategy reduces data utilization and does not address cases where the increase in the likelihood of y^+ is insufficient. Similarly, Ren & Sutherland (2024) propose modifying DPO into a more online-compatible setup by training on both y^+ and y^- , which effectively mitigates the “squeezing effect” imposed by off-policy negative gradient. However, GRPO already employs online response sampling, and our empirical analysis reveals that it still suffers from the LLD effect in a subset of training examples (see Fig. 1).

To effectively address LLD in GRPO, we introduce a negative token hidden reward (NTHR) for *selective token penalization*, which reduces the penalty on tokens in incorrect responses that most strongly contribute to lowering the likelihood of correct ones. Interestingly, we find that these influential tokens often align closely with key reasoning steps in the correct responses (see Fig. 3). Our approach NTHR provides a fine-grained and targeted mitigation strategy that mitigates LLD during training. We evaluate GRPO+NTHR through per-sample training and compare it against baseline methods such as random token dropping and standard GRPO across multiple model sizes. We demonstrate that GRPO+NTHR successfully mitigates LLD and leads to improved generalization on held-out mathematical reasoning problems. In summary, our contributions are as follows:

- **GRPO exhibits LLD for correct responses.** We empirically show that GRPO suffers from Lazy Likelihood Displacement (LLD), an extended form of the misalignment phenomenon previously observed only in DPO.
- **Identifying the source of LLD.** We provide a theoretical explanation for LLD in GRPO, identifying its cause as a penalization of shared reasoning or correct tokens in dispreferred responses.
- **NTHR selective token penalization.** We introduce Negative Token Hidden Reward (NTHR) for selective *token* penalization that effectively mitigates LLD without sacrificing data efficiency.
- **Empirical validation on math reasoning tasks.** We demonstrate that GRPO+NTHR consistently outperforms GRPO on math benchmarks across model sizes ranging from 0.5B to 3B.

2. Preliminaries and Related Work

2.1. GRPO

GRPO loss, introduced in DeepSeek-Math (Guo et al., 2025) and DeepSeek-R1 (Shao et al., 2024), enhances fine-tuning by refining how reward and loss are calculated. Concretely, unlike traditional Proximal Policy Optimization (PPO) (Schulman et al., 2017), GRPO eliminates the need for value function estimation, employing group-relative rewards for a more nuanced optimization process.

For a query-answer pair (x, a) , the policy π_θ samples G responses $\{y_i\}_{i=1}^G$. Each y_i consists of a sequence of $|y_i|$ tokens, and we denote $y_{i,<k}$ the subsequence of the first k tokens. Let r_i denote the reward for response y_i . The advantage of the i -th response is computed by normalizing the group-level rewards $\{r_i\}_{i=1}^G$ and is the same for each token $k = 1, \dots, |y_i|$. Concretely, $\hat{A}_{i,k} := \frac{r_i - \mu}{\sigma}$, with $\mu = \mathbb{E}[\{r_i\}_{i=1}^G]$ and $\sigma = \sqrt{\widehat{\text{Var}}[\{r_i\}_{i=1}^G]}$ being the empirical average and standard deviation of the rewards. The GRPO objective $\mathcal{J}_{\text{GRPO}}(\theta)$ is then defined as:

$$\mathbb{E}_{\substack{(x,a) \sim \mathcal{D} \\ \{y_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|x)}} \left[\frac{1}{\sum_{i=1}^G |y_i|} \sum_{i=1}^G \sum_{k=1}^{|y_i|} \min(\gamma_{i,k}(\theta) \hat{A}_{i,k}, \hat{A}_{i,k} \cdot \text{clip}(\gamma_{i,k}(\theta), 1 - \varepsilon, 1 + \varepsilon)) \right] \quad (1)$$

where ε is a clipping hyperparameter, $\text{clip}(\cdot)$ is the clipping operation, and $\gamma_{i,k}(\theta) = \frac{\pi_\theta(y_{i,k}|x, y_{i,<k})}{\pi_{\theta_{\text{old}}}(y_{i,k}|x, y_{i,<k})}$ is the likelihood ratio between the current policy π_θ and the old policy $\pi_{\theta_{\text{old}}}$.

2.2. Positive and Negative Gradient

The GRPO loss includes several regularization mechanisms designed to stabilize training, such as the KL term and the clipping operation. However, it is common to treat the log-likelihood component of the generated responses as responsible for “gathering knowledge,” while viewing the regularization components as responsible for “stabilizing the training procedure.” Without loss of generality, we focus on the online training setting for GRPO, in which $\gamma_{i,t} \approx 1$ and $\min(\cdot, \text{clip}(\cdot))$ can be safely neglected. This simplification is justified by two observations. First, GRPO inherently operates in a near-online fashion, as it generates new samples on-the-fly and typically uses mini-batches that are comparable in scale to the full batch. Second, prior work (Chu et al., 2025) has shown that omitting the clipping operation does not degrade performance. Additionally, related studies (Chu et al., 2025; Hu et al., 2025) have demonstrated that the KL term can be omitted when other hyperparameters are carefully tuned. With this simplification, we can better understand the interactions between positive and negative responses in one roll-out. The subtle differences between GRPO and its variants, e.g., DAPO (Yu

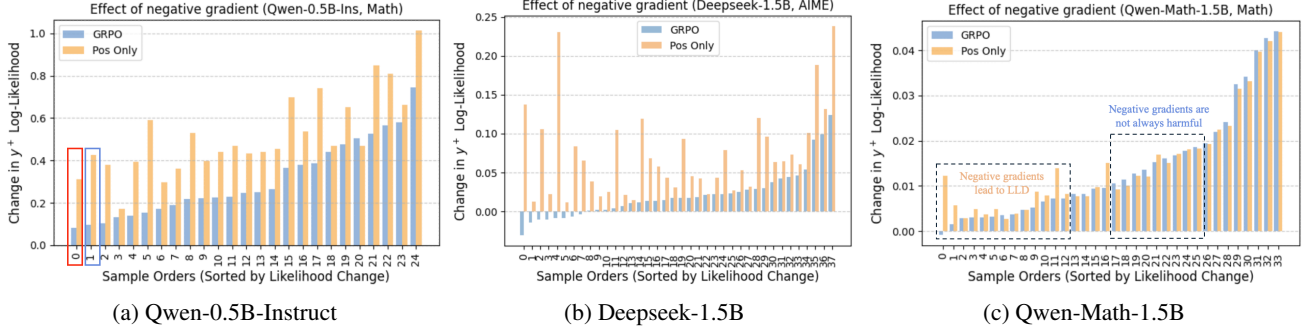


Figure 1: We show that negative gradients can lead to small or reduced likelihood change of positive samples in GRPO. The log-likelihood gains achieved by Pos Only training (orange) are significantly higher than those from GRPO (blue) for Qwen-0.5B-Instruct (a) and Deepseek-1.5B (b). In Qwen-Math-1.5B (c), samples with small or reduced $\Delta(\mathbf{x})$ (left) are primarily influenced by negative gradients, as evidenced by their larger $\Delta(\mathbf{x})$ in the Pos Only setup. However, some samples on the right show smaller $\Delta(\mathbf{x})$ than in GRPO, indicating that negative gradients are not always harmful.

et al., 2025), Dr.GRPO (Liu et al., 2025), GPG (Chu et al., 2025), etc., could also be well interpreted.

Specifically, the GRPO’s objective gradient $\nabla_{\theta} \mathcal{J}_{\text{GRPO}}(\theta)$ with respect to θ can be approximated as $\nabla_{\theta} \mathbb{E} \left[\frac{1}{\sum_{i=1}^G |\mathbf{y}_i|} \sum_{i=1}^G \sum_{k=1}^{|\mathbf{y}_i|} \hat{A}_{i,k} \gamma_{i,k}(\theta) \right]$ which using $\nabla_{\theta} \pi_{\theta} = \pi_{\theta} \nabla_{\theta} \log \pi_{\theta}$ further simplifies to

$$\mathbb{E} \left[\frac{1}{\sum_{i=1}^G |\mathbf{y}_i|} \sum_{i=1}^G \sum_{k=1}^{|\mathbf{y}_i|} \underbrace{\hat{A}_{i,k} \gamma_{i,k}(\theta)}_{\text{constant}} \nabla_{\theta} \log \pi_{\theta}(\mathbf{y}_{i,k} | \mathbf{x}, \mathbf{y}_{i,<k}) \right].$$

This expression allows us to interpret GRPO using a similar framework as proposed by Ren & Sutherland (2024), i.e., imposing a positive or a negative pressure on $\mathbf{y}_{i,k}$. Since $\gamma_{i,k}$ is a ratio of two probabilities (and hence it must be positive), the sign of the equivalent learning rate on $\mathbf{y}_{i,k}$ is determined by $\hat{A}_{i,k}$. Then, following a common practice that the reward is 1 for correct responses and 0 for incorrect ones, it is safe to conclude that all the tokens in the correct responses impose positive gradients while tokens in the wrong responses impose negative ones.

3. The Effect of Negative Gradients

We begin by empirically analyzing the impact of negative gradients in GRPO (Guo et al., 2025) on the likelihood of generating correct responses. Let each question \mathbf{x} be associated with G generated samples, of which $N^+ := N^+(\mathbf{x})$ give the correct answer so we call them *positive* and denote by $\mathbf{y}_i^+, i \in [N^+]$, and the rest $N^- = G - N^+$ are incorrect we call them *negative* and denote by $\mathbf{y}_j^-, j \in [N^-]$. Positive/negative samples are given rewards $r_i = 1$ and $r_j = 0$ respectively.

We conduct experiments using math-reasoning tasks to assess how negative gradient in GRPO training influences the log-likelihood of correct responses. Specifically, we examine a range of model-dataset combinations that differ

in terms of dataset difficulty and model scale: Qwen-2.5-0.5B (Yang et al., 2024a), Qwen-2.5Math-1.5B (Yang et al., 2024b) with the MATH dataset (Hendrycks et al., 2021), and Deepseek-1.5B (Guo et al., 2025) with the AIME dataset. For each question, we generate 8 response rollouts, filtering out samples where all responses are either entirely correct or incorrect. We retain only those examples containing a mix of both.

Our analysis focuses on the GRPO baseline (illustrated by the blue bars in Fig. 4). To precisely characterize the learning dynamics of the algorithm, we reinitialize the model parameters θ for each individual sample, perform a single GRPO update to obtain the updated parameters θ' , and evaluate the average change in the log-likelihood of the correct responses:

$$\Delta(\mathbf{x}) := \frac{1}{N^+} \sum_{i=1}^{N^+} [\ln \pi_{\theta'}(\mathbf{y}_i^+ | \mathbf{x}) - \ln \pi_{\theta}(\mathbf{y}_i^+ | \mathbf{x})], \quad (2)$$

for an input question \mathbf{x} with N^+ correct responses $\mathbf{y}_i^+, i \in [N^+]$. As illustrated in Fig. 1, some correct responses experience either a decline or only a marginal improvement in log-likelihood after training. For example, in Deepseek-1.5B (Fig. 1b), a significant portion of samples exhibit reduced likelihoods (negative values in the plot). Similarly, in Qwen-Math-1.5B and Qwen-0.5B, many samples show only minor changes in likelihood. A closer analysis shows the following.

Negative gradients lead to small or reduced likelihood change. In order to isolate the effect of negative gradients, we introduce a variant which we call Pos Only, where negative advantages are masked (i.e., values of $\hat{A}_{i,k} < 0$ are set to 0), effectively removing the influence of negative gradients during training. Fig. 1 shows the Pos Only variant in orange bars. The log-likelihood gains under Pos Only training are substantially higher than those from GRPO—particularly on the left side of the plots, where

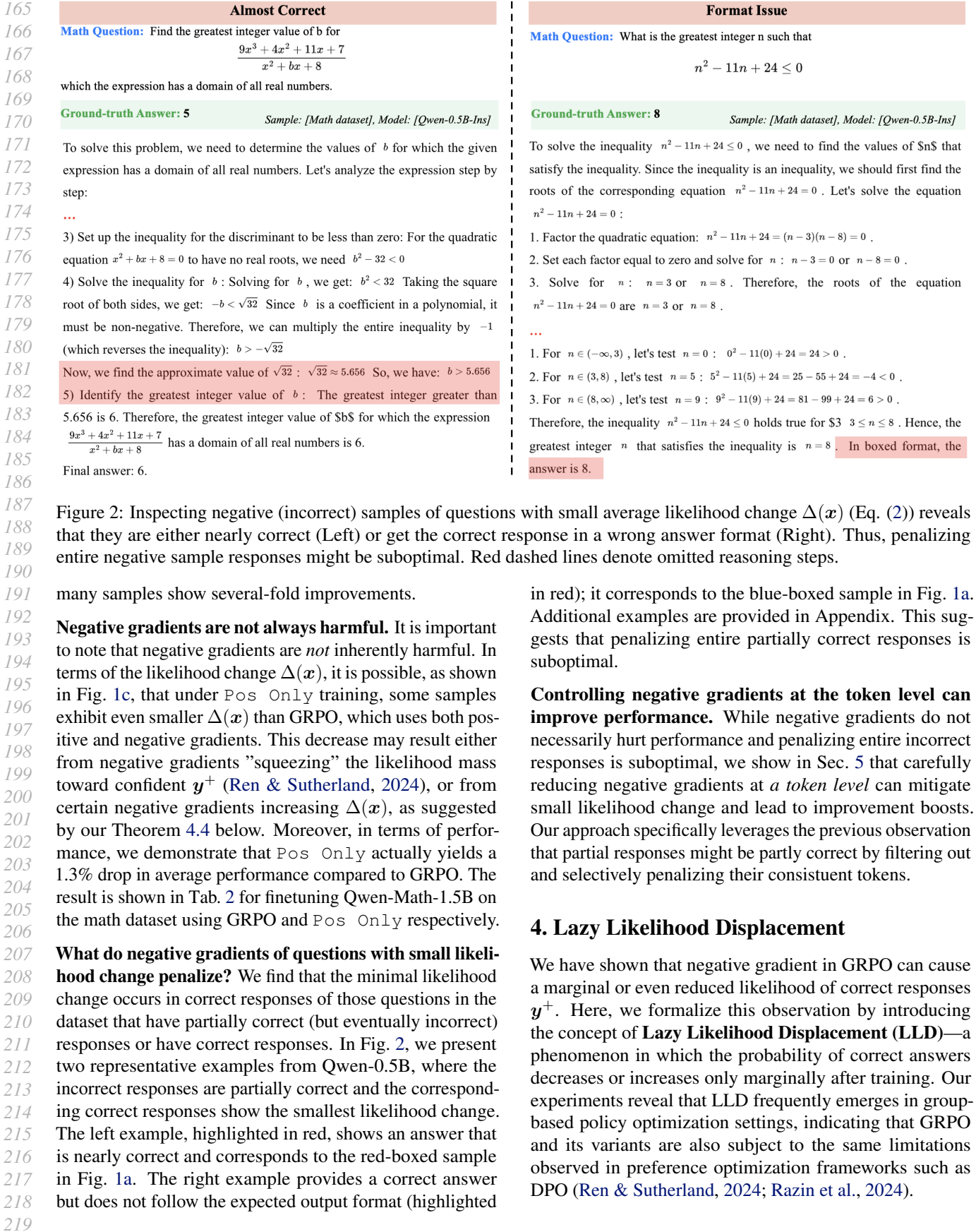


Figure 2: Inspecting negative (incorrect) samples of questions with small average likelihood change $\Delta(x)$ (Eq. (2)) reveals that they are either nearly correct (Left) or get the correct response in a wrong answer format (Right). Thus, penalizing entire negative sample responses might be suboptimal. Red dashed lines denote omitted reasoning steps.

many samples show several-fold improvements.

Negative gradients are not always harmful. It is important to note that negative gradients are *not* inherently harmful. In terms of the likelihood change $\Delta(x)$, it is possible, as shown in Fig. 1c, that under Pos Only training, some samples exhibit even smaller $\Delta(x)$ than GRPO, which uses both positive and negative gradients. This decrease may result either from negative gradients “squeezing” the likelihood mass toward confident y^+ (Ren & Sutherland, 2024), or from certain negative gradients increasing $\Delta(x)$, as suggested by our Theorem 4.4 below. Moreover, in terms of performance, we demonstrate that Pos Only actually yields a 1.3% drop in average performance compared to GRPO. The result is shown in Tab. 2 for finetuning Qwen-Math-1.5B on the math dataset using GRPO and Pos Only respectively.

What do negative gradients of questions with small likelihood change penalize? We find that the minimal likelihood change occurs in correct responses of those questions in the dataset that have partially correct (but eventually incorrect) responses or have correct responses. In Fig. 2, we present two representative examples from Qwen-0.5B, where the incorrect responses are partially correct and the corresponding correct responses show the smallest likelihood change. The left example, highlighted in red, shows an answer that is nearly correct and corresponds to the red-boxed sample in Fig. 1a. The right example provides a correct answer but does not follow the expected output format (highlighted

in red); it corresponds to the blue-boxed sample in Fig. 1a. Additional examples are provided in Appendix. This suggests that penalizing entire partially correct responses is suboptimal.

Controlling negative gradients at the token level can improve performance. While negative gradients do not necessarily hurt performance and penalizing entire incorrect responses is suboptimal, we show in Sec. 5 that carefully reducing negative gradients at a *token level* can mitigate small likelihood change and lead to improvement boosts. Our approach specifically leverages the previous observation that partial responses might be partly correct by filtering out and selectively penalizing their constituent tokens.

4. Lazy Likelihood Displacement

We have shown that negative gradient in GRPO can cause a marginal or even reduced likelihood of correct responses y^+ . Here, we formalize this observation by introducing the concept of **Lazy Likelihood Displacement (LLD)**—a phenomenon in which the probability of correct answers decreases or increases only marginally after training. Our experiments reveal that LLD frequently emerges in group-based policy optimization settings, indicating that GRPO and its variants are also subject to the same limitations observed in preference optimization frameworks such as DPO (Ren & Sutherland, 2024; Razin et al., 2024).

Definition 4.1 Let $\pi_{\theta_{init}}$ and $\pi_{\theta_{fin}}$ denote the initial and final language models, before and after optimizing a preference learning objective \mathcal{J} (e.g., Eq. (1)) over a dataset \mathcal{D} , such that $\mathcal{J}(\theta_{fin}) < \mathcal{J}(\theta_{init})$. We say that **LLD** occurs for a tuple $(\mathbf{x}, \mathbf{y}^+) \in \mathcal{D}$ if, for small nonnegative constant $\epsilon \geq 0$,

$$\ln \pi_{\theta_{fin}}(\mathbf{y}^+ | \mathbf{x}) < \ln \pi_{\theta_{init}}(\mathbf{y}^+ | \mathbf{x}) + \epsilon. \quad (3)$$

4.1. Understanding LLD

We begin by showing that GRPO effectively performs a weighted group preference optimization between two groups of responses: correct and incorrect ones (detailed proof in the appendix).

Lemma 4.2 When reward is binary, GRPO performs preference optimization between two distinct groups: the group of correct responses ($r_i = 1$) and the group of incorrect responses ($r_i = 0$). Specifically, the optimization objective reduces to the following:

$$p^+ \sum_{i=1}^{N^+} \min \left(\frac{\pi_{\theta}(\mathbf{y}_i^+ | \mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}_i^+ | \mathbf{x})}, 1 + \epsilon \right) - p^- \sum_{j=1}^{N^-} \max \left(\frac{\pi_{\theta}(\mathbf{y}_j^- | \mathbf{x})}{\pi_{\theta_{old}}(\mathbf{y}_j^- | \mathbf{x})}, 1 - \epsilon \right), \quad (4)$$

where $p \triangleq p(\mathbf{x}) \triangleq \frac{1}{G} \sum_{i \in [G]} \mathbb{1}[r_i(\mathbf{x}) = 1]$ denotes the correctness rate for a given input, $N^+ = pG$ and $N^- = (1 - p)G$ are the sizes of the correct and incorrect response groups respectively, and the group-specific weights are defined as $p^+ = \frac{1-p}{\sqrt{p(1-p)}}$ and $p^- = \frac{p}{\sqrt{p(1-p)}}$.

Next, we introduce an assumption of unconstrained features, which allows us to focus our analysis on the final-layer hidden embedding $\mathbf{h}_{\mathbf{x}} \in \mathbb{R}^d$ and the token unembedding matrix $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}| \times d}$, where \mathcal{V} is the vocabulary of tokens.

Assumption 4.3 (Unconstrained Features) Expressive (enough) neural networks can produce unconstrained embeddings $\mathbf{h}_{\mathbf{x}} \in \mathbb{R}^d$ independent of the architecture’s specific complexities (Yang et al., 2017; Mixon et al., 2022; Zhao et al., 2024; Razin et al., 2024). These embeddings are subsequently transformed into logits by a token unembedding matrix $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}| \times d}$. The resulting logits are passed through a softmax function to yield a probability distribution over possible next tokens. To assign probabilities to sequences $\mathbf{y} \in \mathcal{V}^*$, the language model π_{θ} operates in an autoregressive manner, i.e., $\pi_{\theta}(\mathbf{y} | \mathbf{x}) = \prod_{k=1}^{|\mathbf{y}|} \text{Softmax}(\mathbf{W}\mathbf{h}_{\mathbf{x}, \mathbf{y}_{<k}})_{y_k}$.

Here, $\mathbf{h}_{\mathbf{x}, \mathbf{y}_{<k}}$ is the embedding of sequence $(\mathbf{x}, \mathbf{y}_{<k})$ and $\text{Softmax}(\cdot)_{y_k}$ is the y_k -th entry of the V -dimensional softmax map. We focus on the online training setting for

Top-K	Qwen-1.5B-deepseek		Qwen-1.5B-math	
	GWHEs	Random	GWHEs	Random
10	50%	17.5%	60%	21.3%
15	75%	26.3%	75%	31.9%

Table 1: Ranking questions by GWHEs results in a significantly higher Top-K overlap with the likelihood-change ranking compared to ranking randomly. This demonstrates the effectiveness of GWHEs in identifying LLD samples. GRPO, as state in Sec. 2.2, the effect of the clipping operation is effectively removed. We assume that the objective is optimized via gradient flow and theoretically analyze the dynamics of the log-likelihood of a positive response, $\frac{d}{dt} \ln \pi_{\theta(t)}(\mathbf{y}_i^+ | \mathbf{x})$, yielding the following characterization¹ (see the appendix for a proof).

Theorem 4.4 For any question \mathbf{x} , at any time $t \geq 0$ of training, and any correct response $\mathbf{y}_i^+, i \in [N^+]$, in addition to the dependence on token unembeddings, the likelihood change $\frac{d}{dt} \ln \pi_{\theta(t)}(\mathbf{y}_i^+ | \mathbf{x})$ exhibits increased laziness (that is, has smaller magnitude) as the following quantity increases:

$$p^- \sum_{k=1}^{|\mathbf{y}_i^+|} \sum_{j=1}^{N^-} \sum_{k'=1}^{|\mathbf{y}_j^-|} \underbrace{\alpha_{k,k'}^- \cdot \langle \mathbf{h}_{\mathbf{x}, \mathbf{y}_{i,<k}^+}, \mathbf{h}_{\mathbf{x}, \mathbf{y}_{j,<k'}^-} \rangle}_{\text{Negative Token Hidden Reward}} - p^+ \sum_{k=1}^{|\mathbf{y}_i^+|} \sum_{i'=1}^{N^+} \sum_{k''=1}^{|\mathbf{y}_{i'}^+|} \alpha_{k,k''}^+ \cdot \langle \mathbf{h}_{\mathbf{x}, \mathbf{y}_{i,<k}^+}, \mathbf{h}_{\mathbf{x}, \mathbf{y}_{i',<k''}^+} \rangle. \quad (5)$$

Here, $\alpha_{k,k'}^-$ and $\alpha_{k,k''}^+$ are token-level prediction error similarity weights, which quantify the similarity of token-level prediction error across responses (see Appendix for formal definitions.). We refer to the quantity in Eq. (5) as Group Weighted Hidden Embedding Score (GWHEs).

The first term in Eq. (5) captures the influence of negative gradients on the likelihood of the correct response. Specifically, this shows that a negative token with a large value of $\alpha_{k,k'}^- \cdot \langle \mathbf{h}_{\mathbf{x}, \mathbf{y}_{i,<k}^+}, \mathbf{h}_{\mathbf{x}, \mathbf{y}_{j,<k'}^-} \rangle$ is more likely to cause the LLD of the correct responses. This motivates the following corollary.

Corollary 4.5 For any question \mathbf{x} , the negative gradient associated with the hidden embedding of a token k' in any incorrect response $\mathbf{y}_j^-, j \in [N^-]$, will exert a greater adverse effect on the likelihood change $\frac{d}{dt} \ln \pi_{\theta(t)}(\mathbf{y}_i^+ | \mathbf{x})$

¹Theorem 4.4 can be seen as an extension of Thm.3 (Razin et al., 2024) for DPO to the more challenging GRPO setting with group-based structure. Unlike DPO, we will exploit this group-structure to selectively penalize tokens rather than responses.

as the following quantity gets larger:

$$\sum_{k=1}^{|y_i^+|} \alpha_{k,k'}^- \cdot \langle \mathbf{h}_{\mathbf{x}, y_{i,<k}^+}, \mathbf{h}_{\mathbf{x}, y_{j,<k'}^-} \rangle. \quad (6)$$

This can occur when the negative and positive tokens have high embedding similarity, theoretically supporting the observation in Fig. 2 that LLD samples are often nearly correct.

4.2. Identifying LLD Samples

Theorem 4.4 suggests using Δ_{GWHEs} as a metric to identify samples that cause LLD. Here, we verify the validity of this metric experimentally. For Qwen-1.5B-deepseek, we generate responses for the first 100 questions from the AIME dataset (1983–2023) and compute the GWHEs scores for those questions with non-extreme predicted probabilities ($0 < p < 1$), resulting in 57 valid questions. For Qwen-2.5Math-1.5B, we use questions from the Math dataset and retain 47 valid questions. We then calculate the Top- K overlap accuracy between two rankings of questions: one based on the ascending order of likelihood changes $\Delta(\mathbf{x})$ as defined in Eq. (2) (denoted Q_K^L), and the other based on the descending order of GWHEs scores as defined in Eq. (5) (denoted Q_K^S). The overlap accuracy is computed as $\frac{|Q_K^L \cap Q_K^S|}{K}$. The results in Tab. 1 indicate that the overlap achieved using GWHEs is significantly greater than that of the random ranking baseline, highlighting a strong alignment between high GWHEs scores and LLD samples.

5. Selective Token Penalization in Incorrect Responses

Theorem 4.4 identifies samples prone to causing LLD and Corollary 4.5 quantifies the influence of individual tokens’ negative gradients on the likelihood of correct responses. The goal of this section is to explore whether appropriately controlling negative gradients can mitigate LLD and further lead to performance improvements over vanilla GRPO. One approach could be using Δ_{GWHEs} to identify negative samples y_i^- that cause LLD and entirely discard them during training. However, this compromises data efficiency and fails to address cases where samples only marginally improve y^+ . Instead, we propose a more targeted approach: identify tokens that negatively impact $\Delta(\mathbf{x})$ and reduce the penalty applied to them.

5.1. Negative Token Hidden Reward (NTHR)

Motivated by Corollary 4.5, we define the impact that an individual token k' of incorrect response $y_j^-, j \in [N^-]$ has

on the likelihood of a group of correct responses as follows:

$$s_{j,<k'}^- := \sum_{i=1}^{N^+} \sum_{k=1}^{|y_i^+|} \alpha_{k,k'}^- \cdot \langle \mathbf{h}_{\mathbf{x}, y_{i,<k}^+}, \mathbf{h}_{\mathbf{x}, y_{j,<k'}^-} \rangle, \quad (7)$$

which defines a relative influence from imposing a negative gradient on y_j^- to all tokens in all positive responses. Specifically, a token from an incorrect response can negatively affect the likelihood of a group of correct responses when $s_{j,<k'}^- \geq 0$, with the magnitude of $s_{j,<k'}^-$ indicating the severity of this effect. We refer to this value as the **negative token hidden reward** (NTHR): it quantifies a token’s detrimental impact on correct responses’ likelihood. In Fig. 3, we visualize tokens with high $s_{j,<k'}^-$ values—highlighted in red. We find that many of these influential tokens are logically or stepwise correct terms (such as “odd,” “prime,” and “intersection” in the specific example). These terms exhibit strong semantic alignment with the correct responses.

5.2. NTHR selective token penalization.

Algorithm 1 NTHR selective token penalization

```

1: Input: Responses’ hidden embeddings  $\{\mathbf{h}_i^+\}_{i=1}^{N^+}$ ,
    $\{\mathbf{h}_j^-\}_{j=1}^{N^-}$ , coefficient  $\alpha$ , scale factor  $\eta$  and  $\beta$ .
2: for  $i' = 1$  to  $N^+$  do
3:    $\bar{s}_{i'}^+ \leftarrow \text{Eq. (7)}$ 
4: end for
5:  $\tau \leftarrow \beta \cdot \min_{i' \in [N^+]} \bar{s}_{i'}^+$ 
6: Select negative tokens:
7: for each negative response  $j$  do
8:    $\mathbf{V}_j^- \leftarrow \{\mathbf{y}_{j,k'}^- \mid s_{j,<k'}^- > \tau\}$ 
9: end for
10: Apply selective penalization:
11: for each retained token  $\mathbf{y}_{j,k'}^- \in \mathbf{V}_j^-$  do
12:    $\hat{A}_{j,k',\eta}^- \leftarrow \eta \cdot \hat{A}_{j,k'}^-$ 
13: end for

```

As shown in Corollary 4.5, tokens with high positive NTHR values tend to cause significant reductions in the likelihood of generated correct responses. To mitigate this adverse effect, we introduce a selective penalization strategy (detailed in Algorithm 1 and complexity discussion in Appendix) that attenuates the penalty on tokens from negative responses whose NTHR scores exceed a threshold τ . Concretely, we define the resulting set of selected tokens as:

$$\mathbf{V}_j^- = \{\mathbf{y}_{j,k'}^- \mid s_{j,<k'}^- > \tau\}. \quad (8)$$

To determine the threshold τ , an effective practical strategy is computing the minimum average token-level influence exerted by each correct response on all other correct responses.

Math Question: How many elements are in the intersection of the set of all the prime numbers less than 30 and the set of all the odd numbers greater than zero?

Ground-truth Answer: 9

Sample from [Math dataset]

Correct Response

To determine ... Step 1: Identify the set of all prime numbers less than 30. The prime numbers less than 30 are: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29. So, the set of all prime numbers less than 30 is {2, 3, 5, 7, 11, 13, 17, 19, 23, 29}.

Step 2: Identify the set of all odd numbers greater than zero. The set of all odd numbers greater than zero is: {1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, ...}. So, the set of all odd numbers greater than zero is {1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, ...}.

Step 3: Find the intersection of the two sets ... From the two sets, we see that the common elements are 3, 5, 7, 11, 13, 17, 19, 23, and 29. So, the intersection is {3, 5, 7, 11, 13, 17, 19, 23, 29}.

Step 4: Count the number of elements in the intersection. The number of elements in the intersection is 9. Therefore, the number of elements in the intersection of the set of all the prime numbers less than 30 and the set of all the odd numbers greater than zero is \boxed{9}.

Incorrect Response Contain Correct Steps

To **determine** ... **First**, let 's list **the prime** numbers less than 30 : 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 **So**, the set of all prime numbers less than 30 is : { 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 }

Next, let 's list the **set** of all odd numbers greater than zero . The **set** of all **odd** numbers **greater** than zero is : { 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27 }

Now, we **find** the intersection of these two sets . The **intersection consists** of numbers that are both in **the set** of **prime** numbers **less** than 30 and in the set of all odd numbers greater than zero . **From the** lists above , **we can** see that **the** only numbers that are both **prime** and odd are **the odd prime** numbers .

Therefore, the intersection is : { 3, 5, 7, 11, 13, 17, 19, 23 } To **find** the number of elements in **the** intersection , we count the **elements** in **the** list : { 3, 5, 7, 11, 13, 17, 19, 23 } There are 8 elements in **the** intersection . Thus , the number of elements in the intersection is \boxed{8} .

Figure 3: Key insight: Tokens of negative samples (incorrect responses) can be logically or step-correct. Tokens with high NTHR tend to strongly correlate with these types of tokens (highlighted in red). The bold dots represent omitted reasoning.

Formally, we set $\tau = \beta \cdot \min_{i' \in [N+]} \bar{s}_{i'}^+$, where β is a scale factor and $\bar{s}_{i'}^+$ measures the average impact of the i' -th correct response’s tokens on the likelihoods of other correct responses:

$$\bar{s}_{i'}^+ := \frac{1}{|y_{i'}^+|} \sum_{k''=1}^{|y_{i'}^+|} \sum_{i=1}^{N^+} \sum_{k=1}^{|y_i^+|} \alpha_{k,k''}^+(t) \cdot \langle \mathbf{h}_{\mathbf{x}, y_{i,k}^+}(t), \mathbf{h}_{\mathbf{x}, y_{i',k''}^+}(t) \rangle. \quad (9)$$

A larger $\bar{s}_{i'}^+$ indicates a stronger mutual influence among positive responses, which could be interpreted as an estimation of the “local elasticity” of a deep neural network (He & Su, 2020). Finally, to apply selective penalization, we define the advantage of each retained token using a scale factor $\eta < 1$, resulting in $\hat{A}_{j,k',\eta}^- := \eta \cdot \hat{A}_{j,k'}^-$. The scale factor will reduce the penalty on selected negative tokens.

5.3. NTHR Selective Token Penalization Mitigates LLD

To evaluate the effectiveness of our NTHR selective token penalization in mitigating LLD, we conduct experiments using the same setup described in Sec. 4. Specifically, we identify the selected token set \mathbf{V}_j^- in incorrect responses using a scale parameter $\beta = 1$, and apply no penalization to these tokens by setting $\eta = 0$. We refer to this variant as GRPO+NTHR (green bars). We then compare the following two baselines: (1) Standard GRPO (blue bars), and (2) GRPO+Random (orange bars), where we randomly select $|\mathbf{V}_j^-|$ tokens and set their advantage to 0 (thus avoid penalizing them in optimization).

The results in Fig. 4 show that GRPO+NTHR consistently improves likelihood change across all samples, as evidenced by the green bars surpassing the blue bars. Notably, while removing negative gradients entirely—as in the Pos Only in Fig. 1c—can sometimes reduce the likelihood change compared to GRPO in Qwen-Math-1.5B, our GRPO+NTHR approach (Fig. 4c) maintains consistent improvements. Furthermore, unlike GRPO+Random that provides only modest gains across all models, GRPO+NTHR delivers consistent and substantial improvements, highlighting the effectiveness of our proposed NTHR in identifying LLD tokens.

5.4. Performance of NTHR Selective Token Penalization

Having demonstrated the impact of negative gradients on likelihood change and the effectiveness of our method in mitigating the LLD issue in Sec. 5.3, we now study the effect of negative gradient on model performance. For this, we finetune models with different sizes using MATH dataset (levels 3–5) (Hendrycks et al., 2021) and use greedy decoding to evaluate finetuned models on five math benchmarks: AIME24, AMC, MATH500, Minerva, and Olympiad (detailed implementation details in Appendix).

We find that NTHR consistently boosts model performance across various sizes, with detailed results presented in Tab. 2. Notably, even for Qwen2.5-Math which shows a substantial improvement of over 20% after RL fine-tuning—outpacing gains seen in other models—NTHR still provides an average improvement of 0.8%. For Qwen2.5-0.5B-Ins and

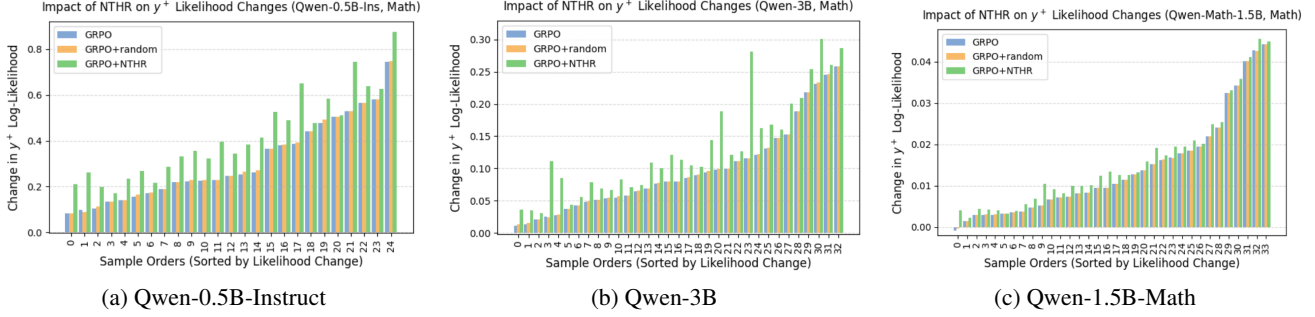


Figure 4: GRPO+NTHR consistently improves likelihood change of correct responses, as indicated by the green bars exceeding the blue bars. While GRPO+Random offers only modest improvements, GRPO+NTHR consistently outperforms it, highlighting the effectiveness of NTHR in identifying LLD tokens.

Model	Method	AIME24	AMC	MATH500	Minerva	Olympiad	Avg.
Qwen2.5-Math-1.5B	Base	3.3	20.0	39.6	7.7	24.9	19.10
	GRPO	13.3	57.5	71.8	29.0	34.1	41.14
	Pos Only	10.0	57.5	70.6	30.1	31.0	39.84
	NTHR	16.7	57.5	70.8	30.5	34.2	41.94
Qwen2.5-0.5B-Ins	Base	0.0	2.5	33.4	4.4	7.0	9.46
	GRPO	0.0	7.5	33.8	9.2	8.1	11.72
	NTHR	0.0	10.0	36.6	8.1	8.6	12.66
Qwen2.5-1.5B-Ins	Base	0.0	22.5	53.0	19.1	20.7	23.06
	GRPO	3.3	32.5	57.2	18.8	23.0	26.96
	NTHR	6.7	35.0	58.8	21.0	20.9	28.48
Qwen2.5-Math-1.5B (deepscale)	Base	3.3	20.0	39.6	7.7	24.9	19.10
	GRPO	10.0	42.5	72.4	32.4	31.9	37.80
	NTHR	16.7	47.5	73.2	29.4	31.4	39.60
Qwen2.5-3B	Base	10.0	37.5	58.6	26.1	24.6	31.36
	GRPO	6.7	35.0	66.6	31.2	29.9	33.88
	NTHR	10.0	47.5	65.6	31.6	26.8	36.30

Table 2: Results across selected math benchmarks for different Qwen2.5 models and methods. NTHR consistently provides average performance gains on various models.

Qwen2.5-1.5B-Ins, NTHR consistently outperforms GRPO by 1.1% and 1.5% on average respectively. Similar improvements are observed with a larger model Qwen2.5-3B, where NTHR reaches 36.30 % compared to GRPO’s 33.88 % and the base model’s 31.36 %. We further fine-tune Qwen2.5-Math on the more challenging DeepScaler (Luo et al., 2025) dataset and observe that NTHR outperforms GRPO by 1.8%.

6. Conclusion

In this work, we study how negative gradients affect the likelihood of correct responses in GRPO. We identify a phenomenon we term Lazy Likelihood Displacement (LLD), where penalization of incorrect responses inadvertently reduces or lead to small likelihood change of correct ones. To

address this, we propose the negative token hidden reward (NTHR) along with a selective token-level penalization strategy. NTHR reduces the penalty on tokens in incorrect responses that contribute most to lowering the likelihood of correct responses, thus successfully mitigating the LLD issue. Through experiments on math reasoning tasks across models ranging from 0.5B to 3B, we demonstrate that NTHR improves GRPO, validating the effectiveness of our approach in addressing LLD and enhancing performance.

Limitation: Due to limited resources, we were unable to run experiments on DeepSeek-1.5B using a sufficiently large context window (Luo et al., 2025). As a result, we trained the model with a 4k context window, leading to significant length constraints—causing the response length to progressively decrease. The corresponding results are reported in the Appendix.

References

- Chu, X., Huang, H., Zhang, X., Wei, F., and Wang, Y. Gpg: A simple and strong reinforcement learning baseline for model reasoning. *arXiv preprint arXiv:2504.02546*, 2025.
- Deng, W., Zhao, Y., Vakilian, V., Chen, M., Li, X., and Thrampoulidis, C. Dare the extreme: Revisiting delta-parameter pruning for fine-tuned models. *arXiv preprint arXiv:2410.09344*, 2024.
- Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- He, C., Luo, R., Bai, Y., Hu, S., Thai, Z. L., Shen, J., Hu, J., Han, X., Huang, Y., Zhang, Y., et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.
- He, H. and Su, W. The local elasticity of neural networks. In *International Conference on Learning Representations*, 2020.
- Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.
- Hu, J., Zhang, Y., Han, Q., Jiang, D., Zhang, X., and Shum, H.-Y. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.
- Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.
- Jin, B., Zeng, H., Yue, Z., Yoon, J., Arik, S., Wang, D., Zamani, H., and Han, J. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.
- Lai, Y., Zhong, J., Li, M., Zhao, S., and Yang, X. Med-r1: Reinforcement learning for generalizable medical reasoning in vision-language models. *arXiv preprint arXiv:2503.13939*, 2025.
- Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022.
- Liu, Z., Chen, C., Li, W., Qi, P., Pang, T., Du, C., Lee, W. S., and Lin, M. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.
- Luo, M., Tan, S., Wong, J., Shi, X., Tang, W., Roongta, M., Cai, C., Luo, J., Zhang, T., Li, E., Popa, R. A., and Stoica, I. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025. Notion Blog.
- Mixon, D. G., Parshall, H., and Pi, J. Neural collapse with unconstrained features. *Sampling Theory, Signal Processing, and Data Analysis*, 20(2):11, 2022.
- Pal, A., Karkhanis, D., Dooley, S., Roberts, M., Naidu, S., and White, C. Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint arXiv:2402.13228*, 2024.
- Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36: 53728–53741, 2023.
- Razin, N., Malladi, S., Bhaskar, A., Chen, D., Arora, S., and Hanin, B. Unintentional unalignment: Likelihood displacement in direct preference optimization. *arXiv preprint arXiv:2410.08847*, 2024.
- Ren, Y. and Sutherland, D. J. Learning dynamics of llm finetuning. *arXiv preprint arXiv:2407.10490*, 2024.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Team, G., Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- Veeraboina, H. Aime problem set 1983-2024, 2023. URL <https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024>.
- Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024a.
- Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., Lu, K., Xue, M., Lin, R., Liu, T.,

Ren, X., and Zhang, Z. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024b.

Yang, Z., Dai, Z., Salakhutdinov, R., and Cohen, W. W. Breaking the softmax bottleneck: A high-rank rnn language model. *arXiv preprint arXiv:1711.03953*, 2017.

Yu, L., Jiang, W., Shi, H., Yu, J., Liu, Z., Zhang, Y., Kwok, J. T., Li, Z., Weller, A., and Liu, W. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Fan, T., Liu, G., Liu, L., Liu, X., et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yuan, L., Cui, G., Wang, H., Ding, N., Wang, X., Deng, J., Shan, B., Chen, H., Xie, R., Lin, Y., et al. Advancing llm reasoning generalists with preference trees. *arXiv preprint arXiv:2404.02078*, 2024.

Zeng, W., Huang, Y., Liu, Q., Liu, W., He, K., Ma, Z., and He, J. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025.

Zhao, Y., Behnia, T., Vakilian, V., and Thrampoulidis, C. Implicit geometry of next-token prediction: From language sparsity patterns to model representations. *arXiv preprint arXiv:2408.15417*, 2024.

A. Appendix

Notation. For any time $t \geq 0$, we use $\mathbf{W}(t)$, $\mathbf{w}_z(t)$, and $\mathbf{h}_z(t)$ to denote the token unembedding matrix, unembedding of a token $z \in \mathcal{V}$, and hidden embedding of $z \in \mathcal{V}^*$, respectively. We let z_k be the k -th token in z and $z_{<k}$ be the first $k-1$ tokens in z . For a question \mathbf{x} , the old policy $\pi_{\theta_{\text{old}}}$ generates a group of G samples and resulting $(\mathbf{x}, \{\mathbf{y}_i^+\}_{N^+}, \{\mathbf{y}_j^-\}_{N^-})$, where $N^+ + N^- = G$. Lastly, we denote by $\mathbf{e}_z \in \mathbb{R}^{|\mathcal{V}|}$ the standard basis vector corresponding to $z \in \mathcal{V}$.

A.1. Proof of Lemma 4.2: GRPO as Group Preference Optimization

In this section, we demonstrate that training with GRPO constitutes preference optimization. We adopt a binary reward system, assigning $r = 1$ to correct responses and $r = 0$ to incorrect ones, consistent with recent works (Liu et al., 2025; Yu et al., 2025). It is notable that this also applies to its variants, e.g., DAPO (Yu et al., 2025), Dr.GRPO (Liu et al., 2025), GPG (Chu et al., 2025), etc.

For a single question \mathbf{x} , we simplify the expected loss $\mathbb{E}_{\{\mathbf{y}_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|\mathbf{x})}[\mathcal{J}_{\mathbf{x}}]$ in Eq. (1) by omitting the token normalization term $\frac{1}{\sum_{i=1}^G |\mathbf{y}_i|}$, yielding:

$$\mathbb{E}_{\{\mathbf{y}_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|\mathbf{x})} \left[\sum_{i=1}^G \sum_{k=1}^{|\mathbf{y}_i|} \min \left(\gamma_{i,k}(\theta) \hat{A}_{i,k}, \hat{A}_{i,k} \cdot \text{clip}(\gamma_{i,k}(\theta), 1 - \varepsilon, 1 + \varepsilon) \right) \right]. \quad (10)$$

The success probability for question \mathbf{x} is $p = \hat{\mathbb{P}}_i(r_{i,\mathbf{x}} = 1) \approx \mathbb{P}(r_{\mathbf{x}} = 1)$ where we use a population approximation for large enough G . In this case, we take $\mu = p$ and $\sigma = \sqrt{p(1-p)}$. Consequently, the advantage $\hat{A}_{i,k}$ becomes:

$$\hat{A}_{i,k} = \begin{cases} \frac{1-p}{\sqrt{p(1-p)}} & \text{if } r_i = 1, \\ -\frac{p}{\sqrt{p(1-p)}} & \text{if } r_i = 0. \end{cases} \quad (11)$$

Since

$$\min \left(\gamma_{i,k}(\theta) \hat{A}_{i,k}, \hat{A}_{i,k} \cdot \text{clip}(\gamma_{i,k}(\theta), 1 - \varepsilon, 1 + \varepsilon) \right)$$

is equivalent to

$$\begin{cases} \hat{A}_{i,k} \cdot \min(\gamma_{i,k}(\theta), 1 + \varepsilon), & \text{if } \hat{A}_{i,k} > 0 \Leftrightarrow r_i = 1, \\ \hat{A}_{i,k} \cdot \max(\gamma_{i,k}(\theta), 1 - \varepsilon), & \text{if } \hat{A}_{i,k} < 0 \Leftrightarrow r_i = 0, \end{cases}$$

the expected loss then becomes:

$$\begin{aligned} \mathbb{E}_{\{\mathbf{y}_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|\mathbf{x})}[\mathcal{J}_{\mathbf{x}}] &= \frac{1-p}{\sqrt{p(1-p)}} \mathbb{E}_{\{\mathbf{y}_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|\mathbf{x})} \left[\min \left(\frac{\pi_{\theta}(\mathbf{y}_i|\mathbf{x})}{\pi_{\theta_{\text{old}}}(\mathbf{y}_i|\mathbf{x})}, 1 + \varepsilon \right) \mathbf{1}_{r_i=1} \right] \\ &\quad - \frac{p}{\sqrt{p(1-p)}} \mathbb{E}_{\{\mathbf{y}_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|\mathbf{x})} \left[\max \left(\frac{\pi_{\theta}(\mathbf{y}_i|\mathbf{x})}{\pi_{\theta_{\text{old}}}(\mathbf{y}_i|\mathbf{x})}, 1 - \varepsilon \right) \mathbf{1}_{r_i=0} \right]. \end{aligned} \quad (12)$$

When generating G sampled responses, the loss becomes:

$$p^+ \sum_{i=1}^{N^+} \min \left(\frac{\pi_{\theta}(\mathbf{y}_i^+|\mathbf{x})}{\pi_{\theta_{\text{old}}}(\mathbf{y}_i^+|\mathbf{x})}, 1 + \varepsilon \right) - p^- \sum_{j=1}^{N^-} \max \left(\frac{\pi_{\theta}(\mathbf{y}_j^-|\mathbf{x})}{\pi_{\theta_{\text{old}}}(\mathbf{y}_j^-|\mathbf{x})}, 1 - \varepsilon \right), \quad (13)$$

where $N^+ = pG$, $N^- = (1-p)G$, $p^+ = \frac{1-p}{\sqrt{p(1-p)}}$ and $p^- = \frac{p}{\sqrt{p(1-p)}}$. This represents group preference optimization, i.e., increasing the likelihood of correct responses while penalizing incorrect ones.

A.2. Proof of Theorem 4.4

Assume that all responses are sequences whose first tokens are distinct from each other. Then we analyze the likelihood change for a correct response \mathbf{y}_i^+ by measuring $\frac{d}{dt} \ln \pi_{\theta(t)}(\mathbf{y}_i^+|\mathbf{x})$. According to the chain rule, using the loss in Eq. (13) and denoting $\pi_{<k} = \pi_{\theta}(\mathbf{y}_{i,k}^+|\mathbf{x}, \mathbf{y}_{i,<k}^+)$ and $\pi_{\text{old},<k} = \pi_{\theta_{\text{old}}}(\mathbf{y}_{i,k}^+|\mathbf{x}, \mathbf{y}_{i,<k}^+)$ for brevity, we can obtain:

$$\frac{d}{dt} \ln \pi_{\theta(t)}(\mathbf{y}_i^+ | \mathbf{x}) = \left\langle \nabla \ln \pi_{\theta(t)}(\mathbf{y}_i^+ | \mathbf{x}), \frac{d\theta(t)}{dt} \right\rangle \quad (14)$$

$$\begin{aligned} &= \left\langle \nabla \ln \pi_{\theta(t)}(\mathbf{y}_i^+ | \mathbf{x}), \right. \\ &\quad p^+ \sum_{i'=1}^{N^+} \sum_{k=1}^{|\mathbf{y}_{i'}^+|} \frac{\pi_{<k}}{\pi_{\text{old}, <k}} \delta \left(1 + \epsilon - \frac{\pi_{<k}}{\pi_{\text{old}, <k}} \right) \nabla \ln \pi_{\theta(t)}(\mathbf{y}_{i',k}^+ | \mathbf{x}, \mathbf{y}_{i',<k}^+) \\ &\quad \left. - p^- \sum_{j=1}^{N^-} \sum_{k'=1}^{|\mathbf{y}_j^-|} \frac{\pi_{<k'}}{\pi_{\text{old}, <k'}} \delta \left(\frac{\pi_{<k'}}{\pi_{\text{old}, <k'}} - 1 + \epsilon \right) \nabla \ln \pi_{\theta(t)}(\mathbf{y}_{j,k'}^- | \mathbf{x}, \mathbf{y}_{j,<k'}^-) \right\rangle \end{aligned} \quad (15)$$

where $\delta(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$ is the gradient of the clip function. We assume $\pi_{\theta(t)} = \pi_{\text{old}}$ since GRPO is online or nearly online due to sampling new responses at each iteration. Thus we obtain:

$$\frac{d}{dt} \ln \pi_{\theta(t)}(\mathbf{y}_i^+ | \mathbf{x}) = \left\langle \nabla \ln \pi_{\theta(t)}(\mathbf{y}_i^+ | \mathbf{x}), p^+ \sum_{i'=1}^{N^+} \nabla \ln \pi_{\theta(t)}(\mathbf{y}_{i'}^+ | \mathbf{x}) - p^- \sum_{j=1}^{N^-} \nabla \ln \pi_{\theta(t)}(\mathbf{y}_j^- | \mathbf{x}) \right\rangle.$$

As per the unconstrained features Assumption 4.3, the model's trainable parameters are

$$\theta = \left(\mathbf{W}, \mathbf{h}_x, \{ \mathbf{h}_{\mathbf{x}, \mathbf{y}_{i',<k}^+} \}_{i' \in [N^+], k \in \{2, \dots, |\mathbf{y}_{i'}^+|\}}, \{ \mathbf{h}_{\mathbf{x}, \mathbf{y}_{j,<k'}^-} \}_{j \in [N^-], k' \in \{2, \dots, |\mathbf{y}_j^-|\}} \right).$$

Here, we also used the (mild) assumption that all responses differ in their first token. Unfolding the gradients with respect to these parameters yields:

$$\begin{aligned} \frac{d}{dt} \ln \pi_{\theta(t)}(\mathbf{y}_i^+ | \mathbf{x}) &= \left\langle \nabla_{\mathbf{W}} \ln \pi_{\theta(t)}(\mathbf{y}_i^+ | \mathbf{x}), p^+ \sum_{i'=1}^{N^+} \nabla_{\mathbf{W}} \ln \pi_{\theta(t)}(\mathbf{y}_{i'}^+ | \mathbf{x}) - p^- \sum_{j=1}^{N^-} \nabla_{\mathbf{W}} \ln \pi_{\theta(t)}(\mathbf{y}_j^- | \mathbf{x}) \right\rangle \\ &\quad + \left\langle \nabla_{\mathbf{h}_x} \ln \pi_{\theta(t)}(\mathbf{y}_i^+ | \mathbf{x}), p^+ \sum_{i'=1}^{N^+} \nabla_{\mathbf{h}_x} \ln \pi_{\theta(t)}(\mathbf{y}_{i',1}^+ | \mathbf{x}) - p^- \sum_{j=1}^{N^-} \nabla_{\mathbf{h}_x} \ln \pi_{\theta(t)}(\mathbf{y}_{j,1}^- | \mathbf{x}) \right\rangle \\ &\quad + \sum_{k=2}^{|\mathbf{y}_i^+|} \|\nabla_{\mathbf{h}_{\mathbf{x}, \mathbf{y}_{i,<k}^+}} \ln \pi_{\theta(t)}(\mathbf{y}_{i,k}^+ | \mathbf{x}, \mathbf{y}_{i,<k}^+)\|^2. \end{aligned} \quad (16)$$

For softmax model output the gradients can be easily computed as follows:

$$\nabla_{\mathbf{W}} \ln \pi_{\theta(t)}(z | \mathbf{x}) = \sum_{k=1}^{|z|} (\mathbf{e}_{z_k} - \pi_{\theta(t)}(\cdot | \mathbf{x}, \mathbf{z}_{<k})) \mathbf{h}_{\mathbf{z}_{<k}}^\top(t)$$

In addition, the gradient with respect to the hidden representation at each position k is:

$$\nabla_{\mathbf{h}_{\mathbf{x}, \mathbf{z}_{<k}}} \ln \pi_{\theta(t)}(z | \mathbf{x}) = \mathbf{w}_{z_k}(t) - \sum_{z \in \mathcal{V}} \pi_{\theta(t)}(z | \mathbf{x}, \mathbf{z}_{<k}) \cdot \mathbf{w}_z(t), \quad k \in \{1, \dots, |z|\}$$

Putting this back in (16) together with a few algebra steps, yields

$$\frac{d}{dt} \ln \pi_{\theta(t)}(\mathbf{y}_i^+ | \mathbf{x}) = \text{(I)} - \text{(II)} + \text{(III)} + \text{(IV)} \quad (17)$$

where:

$$(I) = p^+ \sum_{k=1}^{|y_i^+|} \sum_{i'=1}^{N^+} \sum_{k''=1}^{|y_{i'}^+|} \alpha_{k,k''}^+(t) \cdot \langle \mathbf{h}_{\mathbf{x}, y_{i', < k}^+}(t), \mathbf{h}_{\mathbf{x}, y_{i', < k''}^+}(t) \rangle \quad (18)$$

$$(II) = p^- \sum_{k=1}^{|y_i^-|} \sum_{j=1}^{N^-} \sum_{k'=1}^{|y_j^-|} \alpha_{k,k'}^-(t) \cdot \langle \mathbf{h}_{\mathbf{x}, y_{i, < k}^-}(t), \mathbf{h}_{\mathbf{x}, y_{j, < k'}^-}(t) \rangle \quad (19)$$

$$(III) = \left\langle \mathbf{w}_{y_{i,1}^+}(t) - \sum_{z \in \mathcal{V}} \pi_{\theta(t)}(z | \mathbf{x}) \cdot \mathbf{w}_z(t), \sum_{i'=1}^{N^+} p^+ \mathbf{w}_{y_{i',1}^+} - \sum_{j=1}^{N^-} p^- \mathbf{w}_{y_{j,1}^-} \right\rangle \quad (20)$$

$$(IV) = \sum_{k=2}^{|y_i^+|} \left\| \mathbf{w}_{y_{i,k}^+}(t) - \sum_{z \in \mathcal{V}} \pi_{\theta(t)}(z | \mathbf{x}, y_{i, < k}^+) \cdot \mathbf{w}_z(t) \right\|^2 \quad (21)$$

where $\alpha_{k,k''}^+(t) = \langle \mathbf{e}_{y_{i,k}^+} - \pi_{\theta(t)}(\cdot | \mathbf{x}, y_{i, < k}^+), \mathbf{e}_{y_{i',k''}^+} - \pi_{\theta(t)}(\cdot | \mathbf{x}, y_{i', < k''}^+) \rangle$ and $\alpha_{k,k'}^-(t) = \langle \mathbf{e}_{y_{i,k}^-} - \pi_{\theta(t)}(\cdot | \mathbf{x}, y_{i, < k}^-), \mathbf{e}_{y_{j,k'}^-} - \pi_{\theta(t)}(\cdot | \mathbf{x}, y_{j, < k'}^-) \rangle$.

Specifically, (I) and (II) capture how token embeddings influence changes in likelihood, while (III) and (IV) reflect how the geometry of token unembeddings governs such changes. Our focus is on (I) and (II) because:

- Token embeddings encapsulate the contribution of all network parameters excluding the token unembedding layer.
- Token embeddings are influenced by the words in the sample, which span a broader space than token unembeddings.

Our results in Tab. 1 further validate the effectiveness of these terms. Thus we arrive at Theorem 4.4: as (II) − (I) increases, the likelihood change decreases.

A.3. Implementation Details

Dataset. For training, we use the MATH dataset (levels 3–5) to train the model. Additionally, we include a subset of the DeepScaler dataset (Luo et al., 2025), which contains more challenging problems. For evaluation, we assess the reasoning capabilities of the fine-tuned models on five standard math benchmarks: AIME 2024 (Veeraboina, 2023), AMC, MATH500, Minerva Math (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024).

Models. We select models range from 0.5B-3B (Yang et al., 2024a). We use 0.5B-ins and 1.5B-ins models as (Zeng et al., 2025) found small base model may not follow the format prompt well. We also use Qwen2.5-Math-1.5B (Yang et al., 2024b) to show the influence of math knowledge. For 3B model, we use Qwen-2.5-3B (Yang et al., 2024a) base model. For all models, we conduct reinforcement fine-tuning using GRPO and NTHR using the same hyperparameters.

Hyperparameters. For the 0.5B model, we use two A6000 GPUs with a batch size of 32, maximum rollout length of 2500 tokens, learning rate $5e^{-7}$ and a mini-batch size of 16, resulting in two iteration updates per training step. We use math dataset to train the model for 105 steps.

For the larger models, we utilize four A100 GPUs with a batch size of 256, learning rate $1e^{-6}$ and a mini-batch size of 64, leading to four iteration updates per step, for the math dataset, we train for 40 steps, which approximates one epoch. For the DeepScaler dataset, which contains more data, we train for 45 steps.

Across all models, we generate 8 rollouts per prompt. We use a default sampling temperature of 1.0, a clipping ratio of 0.2, and set the KL loss coefficient to 1×10^{-4} . The Qwen-math model (Yang et al., 2024b) uses its full context length of 3072 tokens for rollouts, while all other models use a maximum rollout length of 4000 tokens. To accelerate training, we employ **dynamic sampling** (Yu et al., 2025), which filters out samples with zero advantage.

Lastly, we use $\beta = 1$ and define the weighting factor as $\eta = 2 \times |0.5 - p|$, where p is the success rate. This formulation penalizes questions with success rates near 0.5 less, avoiding excessive reward for easy samples while not over-relying on difficult questions, whose scarce correct responses may not yield effective token selection. Finally, we set the temperature to 0 to perform greedy decoding, following standard practice in math evaluation tasks (Yu et al., 2023; Deng et al., 2024; Liu

et al., 2025). This enables the assessment of the model’s most confident output, which is essential for math tasks requiring exact correctness and providing users with consistent answers to repeated queries.

A.4. Complexity Discussion

In this section, we introduce several techniques to enhance the computational efficiency of NTHR, which is essential for practical implementation.

Output last-layer embeddings with old policy probabilities. Since GRPO relies on the old policy π_{old} to compute old probabilities for the $\text{clip}(\cdot)$ operation, we only need to output the last-layer embeddings during the old policy’s forward pass. This does not require any extra forward passes, thereby keeping the computational overhead minimal.

Calculate summations first. We more conveniently rewrite (7) as a matrix inner product.

$$\left\langle \sum_{i=1}^{N^+} \sum_{k=1}^{|\mathbf{y}_i^+|} \left(\mathbf{e}_{y_{i,k}^+} - \pi_{\theta(t)}(\cdot | \mathbf{x}, \mathbf{y}_{i,<k}^+) \right) \mathbf{h}_{\mathbf{x}, \mathbf{y}_{i,<k}^+}^T, \left(\mathbf{e}_{y_{j,k'}^-} - \pi_{\theta(t)}(\cdot | \mathbf{x}, \mathbf{y}_{j,<k'}^-) \right) \mathbf{h}_{\mathbf{x}, \mathbf{y}_{j,<k'}^-}^T \right\rangle$$

Importantly, our reformulation involves calculating the summations over i, k first before taking the inner product. This reduction reduces the overall complexity for compute (7) by 3.

Focus on vocabulary of the responses. As the formulation involves computing the outer product between the prediction error vector (e.g., $\mathbf{e}_{y_{i,k}^+} - \pi_{\theta(t)}(\cdot | \mathbf{x}, \mathbf{y}_{i,<k}^+)$) and the hidden embedding, which incurs a computational complexity of $O(|\mathcal{V}|d)$. Since the probability mass is primarily concentrated on the output words, for each question \mathbf{x} , we restrict the computation to the vocabulary $\mathcal{V}_{\mathbf{x}}^*$ associated with its generated responses. Since $|\mathcal{V}_{\mathbf{x}}^*| \ll |\mathcal{V}|$, this significantly lowers the overall cost to $O(|\mathcal{V}_{\mathbf{x}}^*|d)$.

Running time of each module. We also track the average time cost of each module during training, as reported in Tab. 3. Notably, the data generation (Data Gen) module that using dynamic sampling accounts for the majority of the total training time. In contrast, the overhead introduced by NTHR is minimal, contributing only a small fraction to the overall cost. Notably, although deepseek-1.5B has a longer average output length of approximately 3,400 tokens, our NTHR still maintains a low time overhead.

Model+dataset	Data Gen	Model Upd	NTHR	Ref	Old Prob	Total (Sec)
Qwen2.5-Math-1.5B (Math)	250	140	22	55	55	522
Qwen2.5-Math-1.5B (Deepscaler)	262	144	25	60	60	551
deepseek-1.5B (Deepscaler)	700	260	44	86	86	1176

Table 3: Average running time (per step, in seconds) of each module for different models and tasks.

As a result, our method does not introduce additional inference overhead, and the overall complexity remains small.

A.5. Performance across training iterations

In this section, we present the performance of various models across training iterations. As illustrated in Fig. 5, although performance fluctuates throughout training, NTHR consistently outperforms GRPO. Notably, for the Qwen-2.5-1.5B-Ins model, we report results at update step 100 (corresponding to 25 training steps), as performance begins to decline beyond this point.

A.6. Ablation study

In this section, we conduct ablation studies to shed more light on the role played by different modules.

Ablation on β . We conduct ablation study on β , i.e., the scaling factor for the threshold τ in Eq. (8), which is then used to control how bad the negative tokens should be masked out. We set β with $\{-\infty, 0, 0.1, 1, \}$ and train with Qwen2.5-Math-1.5B using Math dataset. Choosing $\beta = -\infty$ corresponds to apply η to all incorrect tokens. As shown in Tab. 4, this removal results in worse performance compared to GRPO. As further shown in Tab. 4, incorporating NTHR consistently improves performance over GRPO. While different β values result in varying degrees of improvement, all settings outperform GRPO, indicating that reducing the influence of identified negative gradients is beneficial.

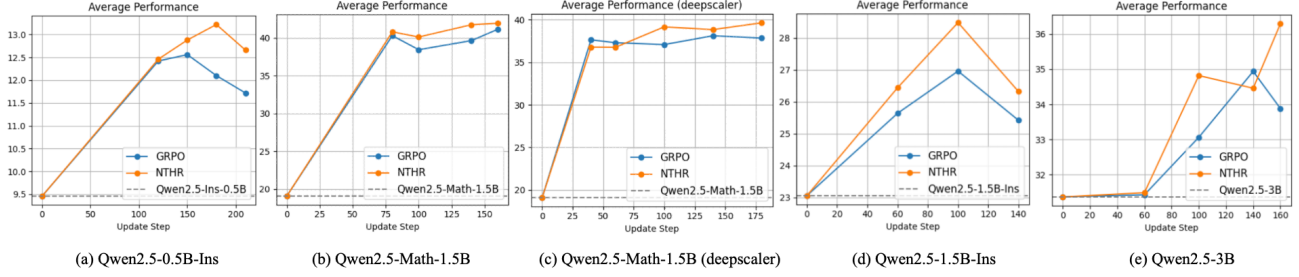


Figure 5: Performance across training iterations for various models, NTHR consistently outperforms GRPO for most of the training process.

Base model + Method	AIME24	AMC	MATH500	Minerva	Olympiad	Avg.
GRPO	13.3	57.5	71.8	29.0	34.1	41.14
NTHR ($\beta = -\infty$)	13.3	50.0	71.8	30.5	34.4	40.00
NTHR ($\beta = 0$)	20.0	55.0	70.0	29.8	32.9	41.54
NTHR ($\beta = 0.1$)	13.3	57.5	<u>71.4</u>	30.9	34.4	41.50
NTHR ($\beta = 1.0$)	<u>16.7</u>	57.5	70.8	<u>30.5</u>	<u>34.2</u>	41.94

Table 4: Ablation study on β . Although changes in β (except ∞) lead to variations in performance, they consistently improve GRPO, indicating that reduce influence of identified negative gradients is beneficial.

Ablation on η . We conduct an ablation study on η using three variants: $\eta = 1 - p$, $\eta = p$, and $\eta = 2 \cdot |0.5 - p|$. As shown in Tab. 5, while the optimal choice of η varies across different models, the variant $\eta = 2 \cdot |0.5 - p|$ consistently yields strong performance across all models. Therefore, we adopt $\eta = 2 \cdot |0.5 - p|$ in our training.

Base model + Method	AIME24	AMC	MATH500	Minerva	Olympiad	Avg.
Qwen2.5Math-1.5B						
GRPO	13.3	57.5	71.8	29.0	34.1	41.14
NTHR ($\eta = p$)	13.3	55.0	72.4	29.0	33.3	40.60
NTHR ($\eta = 1 - p$)	13.3	62.5	72.8	30.5	34.2	42.66
NTHR ($\eta = 2 \cdot 0.5 - p $)	16.7	57.5	70.8	30.5	34.2	41.94
Qwen2.5-3B						
GRPO	6.7	35.0	66.6	31.2	29.9	33.88
NTHR ($\eta = p$)	10.0	47.5	64.6	33.8	26.8	36.54
NTHR ($\eta = 1 - p$)	6.7	42.5	63.8	32.4	28.0	34.68
NTHR ($\eta = 2 \cdot 0.5 - p $)	10.0	47.5	65.6	31.6	26.8	36.30

Table 5: Ablation study on η .

A.7. Results on Deepseek-1.5B

We utilize four A100 GPUs with a batch size of 256 and a mini-batch size of 64, resulting in four update iterations per training step. Each prompt generates 8 rollouts. We adopt a default sampling temperature of 0.6, set the clipping ratio to 0.2, and use a KL loss coefficient of 1×10^{-4} . The maximum rollout length is capped at 4000 tokens. We set $\beta = 0$ and define the weighting factor as $\eta = 2 \cdot |0.5 - p|$. We limit training to 35 steps, as we observed a consistent decline in response length with continued training.

Since the context window length is limited, the results are very unstable, as a results, we calculate the last two checkpoints' average performance and report the results in the table below. In this setting, we observed that our NTHR outperforms GRPO on average.

Base model + Method	AIME24	AMC	MATH500	Minerva	Olympiad	Avg.
Deepseek-1.5B						
Base	20.0	50.0	68.8	20.6	29.6	37.8
GRPO	25.0	65.0	81.10	34.6	40.0	49.1
NTHR	25.0	70.0	80.60	33.7	40.7	50.0

Table 6: Results across selected math benchmarks for deepseek-1.5B.

A.8. Examples of LLD Incorrect Responses

We present the top-ranked incorrect response based on LLD for Qwen-2.5Math-1.5B in Fig. 1c. As shown in Tab. 7, the answer is nearly correct, with the only issue occurring in the final step.

Similarly, Fig. 1b shows the top-ranked incorrect response by LLD for Deepseek-1.5B. As detailed in Tab. 8, the response includes the correct answer multiple times; however, due to continued self-reflection, it becomes truncated by the context window limit, resulting in a final answer that is wrong.

A.9. Words with high NTHR

We present additional examples of words with high NTHR scores. As shown in Tab. 9, using the same response from Tab. 7, we observe that the correct parts are associated with high NTHR values, while the incorrect portion in step 5 receives low NTHR values (in black). We also show response with incorrect output format in Tab. 10, using the same response from Fig. 2, we observe that core steps and correct answer 8 are with high NTHR.

Model & Data	Problem & Answer	Incorrect Response
Qwen2.5math-1.5B Math (order 1)	<p>Question: A portion of the graph of a quadratic function $f(x)$ is shown.</p> <p>Let $g(x) = -f(x)$, $h(x) = f(-x)$. If a is the number of points where the graphs of $y = f(x)$ and $y = g(x)$ intersect, and b is the number of points where the graphs of $y = f(x)$ and $y = h(x)$ intersect, what is $10a + b$?</p> <p>Answer: 21</p>	<p>To solve the problem, we need to analyze the given functions $f(x)$, $g(x)$, and $h(x)$ and determine the points of intersection between the graphs of these functions.</p> <p>Graph of $f(x)$: The graph of $f(x)$ is a quadratic function. From the given graph, we can observe the following:</p> <ul style="list-style-type: none"> The roots of $f(x)$ are at $x = 1$ and $x = 3$. The vertex of the parabola is at $x = 2$. <p>Graph of $g(x) = -f(x)$: The graph of $g(x)$ is the reflection of the graph of $f(x)$ across the x-axis. Therefore:</p> <ul style="list-style-type: none"> The roots of $g(x)$ are also at $x = 1$ and $x = 3$. The vertex of the parabola for $g(x)$ is at $x = 2$. <p>Graph of $h(x) = f(-x)$: The graph of $h(x)$ is the reflection of the graph of $f(x)$ across the y-axis. Therefore:</p> <ul style="list-style-type: none"> The roots of $h(x)$ are at $x = -1$ and $x = -3$. The vertex of the parabola for $h(x)$ is at $x = -2$. <p>Next, we determine the points of intersection:</p> <p>Intersection of $f(x)$ and $g(x)$: Since $g(x) = -f(x)$, the graphs of $f(x)$ and $g(x)$ intersect at the points where $f(x) = -f(x)$. This simplifies to:</p> $f(x) = 0$ <p>The roots of $f(x)$ are at $x = 1$ and $x = 3$. Therefore, the graphs of $f(x)$ and $g(x)$ intersect at two points:</p> $a = 2$ <p>Intersection of $f(x)$ and $h(x)$: Since $h(x) = f(-x)$, the graphs of $f(x)$ and $h(x)$ intersect at the points where $f(x) = f(-x)$. This implies that $f(x)$ is an even function, and its graph is symmetric about the y-axis. The roots of $f(x)$ are at $x = 1$ and $x = 3$, but these are not symmetric about the y-axis. Therefore, the graphs of $f(x)$ and $h(x)$ do not intersect. Thus: $b = 0$</p> <p>Finally, we calculate $10a + b$: $10a + b = 10 \cdot 2 + 0 = 20$</p> <p>Thus, the final answer is:</p> <div style="border: 1px solid red; padding: 2px; display: inline-block;">20</div>

Table 7: Incorrect response example from qwen2.5math-1.5B. The incorrect response is almost correct.

Model & Data	Problem & Answer	Incorrect Response
Deepseek-1.5B AIME (order 1)	<p>Question: One commercially available ten-button lock may be opened by pressing – in any order – the correct five buttons. The sample shown below has $\{1, 2, 3, 6, 9\}$ as its combination. Suppose that these locks are re-designed so that sets of as many as nine buttons or as few as one button could serve as combinations. How many additional combinations would this allow?</p> <p>Answer: 770</p>	<p>To solve the problem, explain your reasoning step by step, and put your final answer within boxed. Alright, so I have this problem about ten-button locks and figuring out how many additional combinations a redesign would allow. Let me try to break it down step by step. First, the original lock. It seems like a ten-button lock where you need to press five buttons in any order to open it. The combination given is 1,2,3,6,9, which is a set of five buttons. So, the question is asking about how many additional combinations would be possible if we redesign the locks to allow sets of as many as nine buttons or as few as one button. I think I need to figure out the total number of possible combinations before and after the redesign and then find the difference. That should give me the number of additional combinations. Starting with the original lock</p> <p>...</p> <p>Therefore, the number of additional combinations is $1022 - 252 = 770$. Wait, that's the same answer as before. So, perhaps 770 is the correct answer. But wait, let me make sure. Is the original number of combinations one, or 252?</p> <p>...</p> <p>Then, the redesigned lock allows for any number of buttons from one to nine, so $2^{10} - 2 = 1022$ possible combinations, but the original combination was just one. So, the number of additional combinations would be $1022 - 1 = 1021$. But that seems like a lot more. Wait, which is it? Is the original combination one specific combination, or is it 252 unique combinations? In the problem, it says, "one commercially available ten-button lock may be opened by pressing – in any order – the correct five buttons."</p> <p>...</p> <p>After redesign, the number of possible combinations is 1022. So, the number of additional combinations is $1022 - 1 = 1021$. But wait, the redesign allows for any combination from one to nine buttons. So, the number of possible combinations is 1022, which includes the original 252 combinations. Therefore, the number of additional combinations is $1022 - 252 = 770$.</p> <p>...</p> <p>The redesigned allows for more combinations, but if you consider that each combination is a set of buttons, then the number of additional combinations is $1022 - 252 = 770$. But the problem says, "the sample shown below has 1,2,3,6,9 as its combination." So, the sample is a specific combination, so that's one combination. So, originally, the number of possible combinations was 252, including the correct one. So, if you redesign it, the number of possible combinations is 1022, which includes all the original 252 combinations plus $1022 - 252 = 770$ additional combinations. Therefore, the number of additional combinations is 770. But to make sure, let me think of another approach. If the original lock had only one combination, the number of possible combinations is 252, each being a unique set of five buttons. Then, if you redesign it so that you can have any number of buttons from one to nine, the number of possible combinations is</p> <p>Stop due to limited context window length.</p>

Table 8: Incorrect response example from deepseek-1.5B on AIME: Although the response includes the correct answers, excessive self-reflection causes the output to exceed the context window, preventing it from being formatted correctly. Green dashes means omitted reasoning.

Model & Data	Problem & Answer	Incorrect Response
Qwen2.5math-1.5B Math	<p>Question: A portion of the graph of a quadratic function $f(x)$ is shown.</p> <p>Let $g(x) = -f(x)$, $h(x) = f(-x)$. If a is the number of points where the graphs of $y = f(x)$ and $y = g(x)$ intersect, and b is the number of points where the graphs of $y = f(x)$ and $y = h(x)$ intersect, what is $10a + b$?</p> <p>Answer: 21</p>	<p>To solve the problem, we need to analyze the given functions $f(x)$, $g(x)$, and $h(x)$ and determine the points of intersection between the graphs of these functions.</p> <p>1. Graph of $f(x)$: The graph of $f(x)$ is a quadratic function. From the given graph, we can observe the following: - The roots of $f(x)$ are at $x = 1$ and $x = 3$. - The vertex of the parabola is at $x = 2$.</p> <p>2. Graph of $g(x) = -f(x)$: The graph of $g(x)$ is the reflection of the graph of $f(x)$ across the x-axis. Therefore: - The roots of $g(x)$ are also at $x = 1$ and $x = 3$. - The vertex of the parabola for $g(x)$ is at $x = 2$.</p> <p>3. Graph of $h(x) = f(-x)$: The graph of $h(x)$ is the reflection of the graph of $f(x)$ across the y-axis. Therefore: - The roots of $h(x)$ are at $x = -1$ and $x = -3$. - The vertex of the parabola for $h(x)$ is at $x = -2$.</p> <p>Next, we determine the points of intersection:</p> <p>4. Intersection of $f(x)$ and $g(x)$: Since $g(x) = -f(x)$, the graphs of $f(x)$ and $g(x)$ intersect at the points where $f(x) = -f(x)$. This simplifies to:</p> $f(x) = 0$ <p>The roots of $f(x)$ are at $x = 1$ and $x = 3$. Therefore, the graphs of $f(x)$ and $g(x)$ intersect at two points:</p> $a = 2$ <p>5. Intersection of $f(x)$ and $h(x)$: Since $h(x) = f(-x)$, the graphs of $f(x)$ and $h(x)$ intersect at the points where $f(x) = f(-x)$. This implies that $f(x)$ is an even function, and its graph is symmetric about the y-axis. The roots of $f(x)$ are at $x = 1$ and $x = 3$, but these are not symmetric about the y-axis. Therefore, the graphs of $f(x)$ and $h(x)$ do not intersect.</p> <p>Thus:</p> $b = 0$ <p>Finally, we calculate $10a + b$:</p> $10a + b = 10 \cdot 2 + 0 = 20$ <p>Thus, the final answer is:</p> <div style="border: 1px solid black; display: inline-block; padding: 2px 5px;">20</div>

Table 9: High NTHR (highlight in red) example from qwen2.5math-1.5B. Core steps in correct parts are with high NTHR. The erroneous words of step 5 are with low NTHR values.

Model & Data	Problem & Answer	Incorrect Response
Qwen2.5-0.5B-Ins Math	Question: What is the greatest integer n such that $n^2 - 11n + 24 \leq 0$? Answer: 8	<p>To solve the inequality $n^2 - 11n + 24 \leq 0$, we need to find the values of n that satisfy the inequality. Since the inequality is an inequality, we should first find the roots of the corresponding equation $n^2 - 11n + 24 = 0$. Let's solve the equation $n^2 - 11n + 24 = 0$: 1. Factor the quadratic equation: $n^2 - 11n + 24 = (n - 3)(n - 8) = 0$. 2. Set each factor equal to zero and solve for n: $n - 3 = 0$ or $n - 8 = 0$. 3. Solve for n: $n = 3$ or $n = 8$. Therefore, the roots of the equation $n^2 - 11n + 24 = 0$ are $n = 3$ and $n = 8$. Now, we need to find the values of n that satisfy the inequality $n^2 - 11n + 24 \leq 0$. Since the inequality is an inequality, we should check the intervals between the roots. The roots $n = 3$ and $n = 8$ divide the number line into three intervals: $(-\infty, 3]$, $[3, 8]$, and $[8, \infty)$. Let's test a point in each interval to determine where the inequality holds true. 1. For n in $(-\infty, 3]$, let's test $n = 0$: $0^2 - 11(0) + 24 = 24 > 0$. 2. For n in $(3, 8]$, let's test $n = 5$: $5^2 - 11(5) + 24 = 25 - 55 + 24 = -4 < 0$. 3. For n in $(8, \infty)$, let's test $n = 9$: $9^2 - 11(9) + 24 = 81 - 99 + 24 = 6 > 0$. Therefore, the inequality $n^2 - 11n + 24 \leq 0$ holds true for $3 \leq n \leq 8$. Hence, the greatest integer n that satisfies the inequality is $n = 8$. In boxed format, the answer is 8.</p>

Table 10: High NTHR (highlight in red) example from qwen2.5-0.5B-Ins. This example have correct answer but in wrong format. Core steps and correct answer 8 are with high NTHR.