GET HINT-AUGMENTED GRPO DONE RIGHT THROUGH ACTIVATION STEERING

Anonymous authors
Paper under double-blind review

ABSTRACT

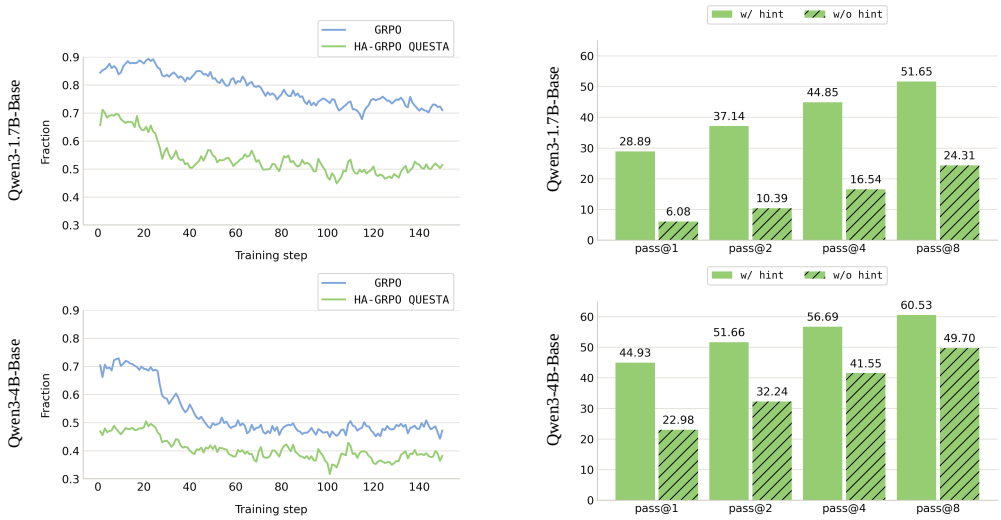
GRPO has become the dominant reinforcement learning method for eliciting reasoning capability in LLMs using verifiable rewards. However, it suffers from a fundamental failure mode: when a training question exceeds the base model's current capability, all generated responses fail and receive zero reward, yielding no gradient signal and stalling further capability expansion. An emerging class of GRPO extensions, termed Hint-Augmented GRPO (HA-GRPO), adds a "hint" to the context to reduce the difficulty of the question, thereby producing a usable gradient signal. Nevertheless, existing HA-GRPO methods share a key issue: no hint is available at inference time to append to the question, creating a training-inference context mismatch that leads to inconsistent performance gains. In this work, we reinterpret the function of hinting – not as reducing question difficulty, but as steering model behavior – and propose ASH (Activation Steering for Hinting), a novel HA-GRPO method built on this view. During training, the model is presented with both hinted and plain questions, while a trainable steering vector is optimized to widen the margins between the model's generation probabilities for responses to the hinted version relative to the plain version, thereby capturing the question-agnostic component of the behavioral shift induced by the hint. During inference, the trained steering vector is directly injected into the model's activation, reproducing the hint-induced behavior without requiring any hint in the context. Experiments with four LLMs (Qwen3-1.7B/4B-Base, Llama-3.2-1B/3B-Instruct) show that ASH consistently improves pass@k on competition-level mathematics benchmarks (AMC, OlympiadBench, AIME24, AIME25) and out-of-distribution benchmarks (Minerva, GPQA-Diamond). Notably, it improves average pass@8 of Qwen3-1.7B-Base and Qwen3-4B-Base by **2.60**% and **2.21**%.

1 INTRODUCTION

Recently, reinforcement learning with verifiable rewards (Lambert et al., 2024) has emerged as the key driver for eliciting reasoning capability in Large Language Models (LLMs), particularly in domains such as mathematics and coding, where correctness of final answers can be scored automatically (Jaech et al., 2024; Guo et al., 2025). Among various methods, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is widely adopted; it eliminates the critic model and estimates advantages for a group of generated responses through group-relative reward normalization. Although this design offers impressive computational efficiency, it fundamentally fails when a training question is **too difficult** for the base model: all generated responses fail and receive zero reward; their advantages consequently vanish, and the question yields no gradient signal, blocking any capability expansion beyond the base model (Yue et al., 2025; Akgül et al., 2026). This limitation motivates the need for methods that can produce gradient signals on such too-difficult questions.

One simple approach is to allocate more generation budget to too-difficult questions, in hopes of eventually reaching rare successful responses (Shrivastava et al., 2026; Yu et al., 2026); however, this multiplies computational cost. A second approach uses a proxy for successful responses, such as the model's confidence (Han et al., 2026) or certainty (Le et al., 2026), but recovers only indirect gradient signals. A third, more promising approach reduces the difficulty of the questions themselves by adding a "hint" to the context, giving rise to a class of methods we term Hint-Augmented GRPO (HA-GRPO) (Li et al., 2026; Liao et al., 2026; Xia et al., 2026). For example, given a training question, QUESTA (Li et al., 2026) uses the first 50% of tokens from the solution as a hint and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) The fraction of too-difficult questions per batch over the course of training.

(b) pass@k of QUESTA-trained models, with and without the hint in the questions.

Figure 1: The impact of the training-inference context mismatch. (a) QUESTA substantially lessens the fraction of too-difficult questions per batch over the course of training, relative to GRPO. (b) pass@k of QUESTA-trained models drops sharply without the hint at inference time.

augments the question to guide the model toward successful responses. Follow-up HA-GRPO methods employ more subtle hint-extraction mechanisms; however, they all share a critical, unresolved issue: a training-inference context mismatch, since no hint is available at inference time to append to the question. Intuitively, these methods allow the model to *cheat* by leveraging assistance during training, whereas no such assistance is permitted during inference (i.e., the real exams).

To characterize the impact of this training-inference context mismatch, we run preliminary experiments using GRPO and QUESTA to train Qwen3-1.7B-Base and Qwen3-4B-Base on a subset of the OpenR1-MATH-220k dataset (Hugging Face, 2025) (details in Section 6). As shown in Figure 1a, QUESTA substantially lessens the fraction of too-difficult questions per batch over the course of training, relative to GRPO, aligned with expectations. We then evaluate QUESTA-trained models on training data, with and without the hint in the questions. As shown in Figure 1b, pass@k drops sharply without the hint at inference time; for example, pass@8 falls by **27.34**% and **10.83**% for Qwen3-1.7B-Base and Qwen3-4B-Base, respectively. These huge performance gaps indicate that models trained with QUESTA rely heavily on the hint as a crutch during training: much of the capability boost that hinting unlocked is lost, and the models consequently generalize poorly to the hint-free setting they will actually face during inference.

To mitigate this issue, we reframe the function of hinting: rather than viewing it as reducing question difficulty, as existing HA-GRPO methods do, we view it as steering model behavior. This view situates hinting within a broader paradigm of context-based control, alongside few-shot in-context learning (Todd et al., 2024), which steers behavior through demonstrations, and instruction following (Stolfo et al., 2025), which steers behavior through explicit directives. Under this lens, the hint in QUESTA and other methods steers model behavior toward a higher-quality reasoning space rather than reducing question difficulty, suggesting that hint-reliance stems not from the hint itself, but from how the information is introduced during training yet withdrawn during inference.

Building on this view, we propose ASH (Activation Steering for Hinting), which leverages the hint to steer model behavior toward a higher-quality reasoning space not through the context, but through the model's activation. During training, both hinted and plain questions are presented, and a trainable steering vector is added to the model's activation at a particular layer and optimized to widen the margins between the generation probabilities the model assigns to responses under the hinted version relative to the plain version. This drives the steering vector to capture, within the activation space, the question-agnostic component of the behavioral shift that the hint induces, which is precisely the

component that transfers to unseen questions. During inference, the trained steering vector is directly injected into the model's activation, reproducing the hint-induced behavior without ever exposing the model to a hint. As the benefit of the hint is distilled into the steering vector that remains available at inference time, ASH eliminates the training-inference context mismatch, allowing the model to retain a substantial part of the capability boost hinting unlocked rather than losing it.

Our contributions are summarized as follows:

- **The training-inference context mismatch issue.** We identify a critical issue that existing HA-GRPO methods have overlooked and conduct preliminary experiments to characterize its impact.
- **ASH.** Our novel HA-GRPO method mitigates the identified issue through the lens of steering model behavior. ASH internalizes the hint's behavioral shift in a trainable steering vector that remains available and can be used at inference time.
- **Evaluation.** ASH consistently improves pass@k across four LLMs. On average, pass@8 gains reach **1.97**%/**2.25**% on competition-level mathematics and **3.84**%/**2.13**% on out-of-distribution benchmarks for Qwen3-1.7B/4B-Base, and reach **1.86**%/**2.15**% on competition-level mathematics and **2.80**%/**2.11**% on out-of-distribution benchmarks for Llama-3.2-1B/3B-Instruct.
- **In-depth analysis.** We showcase that the trained steering vector can transfer effectively to other reasoning domains such as logical question answering and coding.

2 Related Work

In this section, we review prior work on GRPO advances before narrowing our focus to the Hint-Augmented GRPO (HA-GRPO) class. Then, we briefly discuss activation steering research.

GRPO Advances - A line of work aims to make GRPO better. For instance, given a group of generated responses per training question, SAGE-GRPO (Huang et al., 2026) seeks to control the lengths of these responses. Dr. GRPO (Liu et al., 2025) aims to overcome the question-level difficulty bias in GRPO, whereas GRPO-LEAD (Zhang & Zuo, 2025) embraces this bias by reweighting each question by its difficulty, estimated via the inverse success rate of the responses. Despite rapid progress (Surana et al., 2026), a fundamental failure mode of GRPO persists: when a training question is **too difficult** for the base model, all generated responses fail and receive zero reward; their advantages vanish, and the question then yields no gradient signal, blocking any capability gain beyond the base model (Yue et al., 2025; Akgül et al., 2026). To address this limitation, SA-GRPO (Han et al., 2026) and GRPO-ZVP (Le et al., 2026) leverage the model's confidence and certainty as a proxy to complement the response-level rewards. However, the proxy is decoupled from correctness, so the gradient signal recovered is not guaranteed to point in a direction of enhancing capability.

HA-GRPO - A class of GRPO extensions attempts to reduce the difficulty of such questions by adding a "hint" to the context, based on their reference solution. For example, given a training question, QUESTA (Li et al., 2026) uses the first 50% of tokens from the solution as a hint and augments the question to guide the model toward successful responses. BREAD (Zhang et al., 2026b) uses the first 50% of sentences, separated by "`\n`". Follow-up methods introduce more subtle mechanisms to extract the hint. Notably, Zhang et al. (2026a); Wang et al. (2026) draw on the scaffolding theory to construct a hierarchical, progressive hint rather than a solution prefix, though this strategy requires a powerful, proprietary teacher model (e.g., GPT). To relax this dependence, SAGE (Liao et al., 2026) extracts an adaptive hint from the model itself, conditioning not only on the solution but also on generated responses; HILL (Xia et al., 2026), meanwhile, concurrently creates a duplicated copy of the model to serve as the hint provider. Although these two methods eliminate the dependence on a proprietary teacher model, both nearly double the computational cost. Importantly, existing HA-GRPO methods focus on hint-extraction mechanisms while overlooking the issue that no hint is available at inference time to append to the question. In this work, we mitigate this training-inference context mismatch through the lens of steering model behavior.

Activation Steering - An activation-based paradigm for controlling the behavior of LLMs (Zou et al., 2023; Wehner et al., 2025). This is done by first searching for "steering vectors" within the activation space of particular layers in LLMs' architecture. During inference, these steering vectors are directly injected back to alter model generations. Turner et al. (2023) explicitly created steering vectors associated with contrastive concepts, e.g., "Love" and "Hate". Building on this idea, Todd et al. (2024) construct steering vectors that encode demonstrations to enhance few-shot in-context

learning, and Stolfo et al. (2025) construct steering vectors that encode explicit directives to enhance instruction following. In common, such techniques construct steering vectors offline from pairs of contrastive prompts, limiting their effectiveness on complex reasoning tasks. In this work, we build on this paradigm and construct a steering vector for hinting that is made trainable online by design.

3 PRELIMINARIES

We formalize GRPO and its failure on too-difficult questions, and introduce the Hint-Augmented GRPO (HA-GRPO) class built on top of it. Then, we describe how activation steering works.

GRPO. Recent efforts to strengthen the reasoning capability of LLMs have increasingly depended on GRPO, which estimates advantages from group-relative reward normalization. Let π_θ represent the model, which is parameterized by $\theta \in \mathbb{R}^d$. From the training data $\mathcal{D}$, sample the question q, the model generates G responses to it as a group $\{o_i \sim \pi_\theta(\cdot|q)\}_{i=1,...,G}$. Denote their rewards as $\{R_i\}_{i=1,...,G}$, the advantages of G responses are computed via:

$$\{A_i = \frac{R_i - \text{mean}(\{R_i\}_{i=1,...,G})}{\text{std}(\{R_i\}_{i=1,...,G}) + \varepsilon}\}_{i=1,...,G}.$$

Following that, the model is updated by minimizing a loss defined based on these advantages:

$$J(\theta; q) = -\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}(r_\theta(o_{i,t})A_i)\right],$$

$$\text{where} \quad r_\theta(o_{i,t}) = \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{[\pi_\theta(o_{i,t}|q, o_{i,<t})]_{\text{not_updated}}}. \tag{1}$$

Despite its strengths, this group-relative reward normalization design has a fragility that blocks capability expansion. When the question q is too difficult (all G generated responses fail), the reward satisfies $R_i = \text{mean}(\{R_i\}_{i=1,...,G})$ for all i, leading to $A_i = 0$ for all i, yielding $\nabla_\theta J(\theta; q) = 0$.

HA-GRPO. To address this limitation, a natural approach is to reduce the difficulty of the question q, which HA-GRPO methods achieve by adding a "hint" to the context. Let z_q denote the reference solution to q, and $\mathcal{F}$ denote a method-specific hint-extraction mechanism (e.g., in QUESTA, $\mathcal{F}$ returns the first 50% of tokens from z_q). The hinted question q^+, which replaces q, is devised as:

$$q^+ = \texttt{concat}[q, h_q], \qquad h_q := \mathcal{F}(q, z_q). \tag{2}$$

However, at inference time, the reference solution $z_{\overline{q}}$ for a given question $\overline{q}$ is naturally unavailable, so no hint $h_{\overline{q}}$ can be extracted and appended to $\overline{q}$ to form the hinted question $\overline{q}^+$. As a result, the model can no longer access the capability boost that hinting unlocked.

Activation Steering. Suppose the model π_θ consists of L layers, and let $\mathbf{h}_t^{(l)} \in \mathbb{R}^{d'}$ denote the model's activation at layer $l \in \{1, ..., L\}$ (output of the decoder block) and token t. With a steering vector $\mathbf{v} \in \mathbb{R}^{d'}$ associated with a chosen layer l, $\mathbf{v}$ is added to the model's activation at that layer for every token t:

$$\tilde{\mathbf{h}}_t^{(l)} = \mathbf{h}_t^{(l)} + \mathbf{v}, \quad \forall t. \tag{3}$$

Then, the intervened $\tilde{\mathbf{h}}_t^{(l)}$ is propagated through the remaining layers $l+1, ..., L$ in place of $\mathbf{h}_t^{(l)}$. We denote the model with activation steering as $\pi_\theta^{\mathbf{v}}$ to distinguish it from π_θ.

A central concern in activation steering is obtaining $\mathbf{v}$ that reliably captures the behavioral shift. Normally, $\mathbf{v}$ is computed offline and kept static – suitable when the target behavior is stable over time. In complex reasoning tasks, the target behavior (hint-induced one) is entangled with a reference solution that varies across questions and must track the model as it evolves as training progresses.

4 ACTIVATION STEERING FOR HINTING

We present ASH, a novel HA-GRPO method that operationalizes the view of hinting as steering model behavior. The design goal is to internalize the hint's behavioral shift in a trainable steering vector that remains available and can be used at inference time. Figure 2 gives a conceptual illustration of ASH.

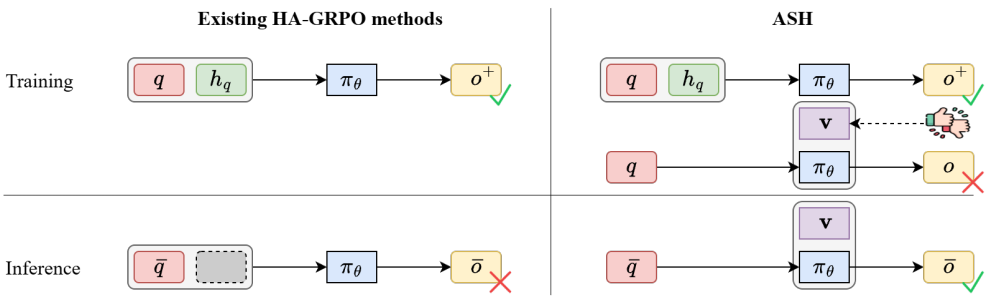


Figure 2: A conceptual illustration of ASH, in comparison with existing HA-GRPO methods.

TRAINING

First, as in other HA-GRPO methods, the hinted question q^+ is devised from the plain question q. Here, we adopt the simple mechanism $\mathcal{F}$ of QUESTA (Li et al., 2026), which returns the first 50% of tokens from z_q (see examples in Appendix A). Subsequently, the model is presented with both the hinted question q^+ and the plain question q, and generates $\hat{G}$ responses to each version: *hinted* $\{o_i^+ \sim \pi_\theta(\cdot|q^+)\}_{i=1,\dots,\hat{G}}$ for q^+ and *plain* $\{o_i \sim \pi_\theta(\cdot|q)\}_{i=1,\dots,\hat{G}}$ for q. Then, we add the trainable steering vector $\mathbf{v}$ to the model's activation at a chosen layer l (Equation 3) and optimize it to increase the differences in generation probabilities between *hinted* responses relative to *plain* responses, conditioned on q. Drawing on the analogy between our goal and that of Direct Preference Optimization (DPO) (Rafailov et al., 2023; Xiao et al., 2024), we formulate a loss adapted from it:

$$\mathcal{L}(\mathbf{v};q) = -\left[\frac{1}{\hat{G}}\sum_{i=1}^{\hat{G}}\log\sigma\left(\beta\sum_{t=1}^{|o_i^+|}\log\frac{\pi_\theta^{\mathbf{v}}(o_{i,t}^+|q,o_{i,<t}^+)}{\pi_\theta(o_{i,t}^+|q,o_{i,<t}^+)} - \beta\sum_{t=1}^{|o_i|}\log\frac{\pi_\theta^{\mathbf{v}}(o_{i,t}|q,o_{i,<t})}{\pi_\theta(o_{i,t}|q,o_{i,<t})}\right)\right], \quad (4)$$

where β is a scaling factor controlling how strongly the model with activation steering $\pi_\theta^{\mathbf{v}}$ is penalized for drifting too far from π_θ. By minimizing this loss, the direction of $\mathbf{v}$ is accurately aligned with the question-agnostic component of the behavioral shift the hint induces. Note that we do not filter $\hat{G}$ responses to the hinted question q^+ and $\hat{G}$ responses to the plain question q based on their correctness at this stage, as we strive to isolate the behavioral shift in the model, independent of whether the final answers are correct.

Once the steering vector $\mathbf{v}$ has been trained, the model itself is then trained with GRPO as in other HA-GRPO methods, but on both the hinted question q^+ and the plain question q, and what it has generated, i.e., *hinted* $\{o_i^+ \sim \pi_\theta(\cdot|q^+)\}_{i=1,\dots,\hat{G}}$ for q^+ and *plain* $\{o_i \sim \pi_\theta(\cdot|q)\}_{i=1,\dots,\hat{G}}$ for q. Denote their rewards as $\{R_i^+\}_{i=1,\dots,\hat{G}}$ and $\{R_i\}_{i=1,\dots,\hat{G}}$, respectively, the advantages of each $\hat{G}$ responses are computed via:

$$\{A_i^+ = \frac{R_i^+ - \text{mean}(\{R_i^+\}_{i=1,\dots,\hat{G}})}{\text{std}(\{R_i^+\}_{i=1,\dots,\hat{G}}) + \varepsilon}\}_{i=1,\dots,\hat{G}} \text{ and } \{A_i = \frac{R_i - \text{mean}(\{R_i\}_{i=1,\dots,\hat{G}})}{\text{std}(\{R_i\}_{i=1,\dots,\hat{G}}) + \varepsilon}\}_{i=1,\dots,\hat{G}}.$$

Following that, the model is updated by minimizing the sum of $J(\theta;q^+)$ and $J(\theta;q)$:

$$J(\theta;q^+) + J(\theta;q) = -\left[\frac{1}{\hat{G}}\sum_{i=1}^{\hat{G}}\frac{1}{|o_i^+|}\sum_{t=1}^{|o_i^+|}\left(r_\theta(o_{i,t}^+)A_i^+\right)\right] - \left[\frac{1}{\hat{G}}\sum_{i=1}^{\hat{G}}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left(r_\theta(o_{i,t})A_i\right)\right],$$

$$\text{where} \quad r_\theta(o_{i,t}^+) = \frac{\pi_\theta(o_{i,t}^+|q^+,o_{i,<t}^+)}{[\pi_\theta(o_{i,t}^+|q^+,o_{i,<t}^+)]_{\text{not_updated}}} \text{ and } r_\theta(o_{i,t}) = \frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{[\pi_\theta(o_{i,t}|q,o_{i,<t})]_{\text{not_updated}}}. \quad (5)$$

Note that the model is trained on both q^+ and q jointly, rather than on q^+ alone. This is possible without interference, since $J(\theta;q^+)$ and $J(\theta;q)$ are computed from independently estimated advantages and simply summed, so they do not compete with each other. Doing so makes fuller use of the already-generated responses for both q^+ and q, and additionally acts as a form of **regularization** that reduces the model's reliance on the hint.

Algorithm 1 ASH's training procedure.

Input: A model π_θ; training data $\mathcal{D}$; number of training steps T and the step size η
Output: The trained steering vector $\mathbf{v}^*$; the trained parameters θ^*

$\mathbf{v} \leftarrow \mathbf{0}$ ▷ initialize the steering vector
for training step $= 1$ to T **do**
  sample the question q and the reference solution z_q from $\mathcal{D}$
  $q^+ = \texttt{concat}[q, h_q], \qquad h_q := \mathcal{F}(q, z_q)$ ▷ Equation 2
  obtain *hinted* $\{o_i^+ \sim \pi_\theta(\cdot|q^+)\}_{i=1,\ldots,\hat{G}}$ and *plain* $\{o_i \sim \pi_\theta(\cdot|q)\}_{i=1,\ldots,\hat{G}}$
  compute $\mathcal{L}(\mathbf{v}; q)$ using $\{o_i^+\}_{i=1,\ldots,\hat{G}}$ and $\{o_i\}_{i=1,\ldots,\hat{G}}$ ▷ Equation 4
  $\mathbf{v} \leftarrow \mathbf{v} - \eta \nabla_{\mathbf{v}} \mathcal{L}(\mathbf{v}; q)$
  if training step $< T$ **then**
    compute $\{A_i^+\}_{i=1,\ldots,\hat{G}}$ and $\{A_i\}_{i=1,\ldots,\hat{G}}$
    compute $J(\theta; q^+) + J(\theta; q)$ ▷ Equation 5
    $\theta \leftarrow \theta - \eta \nabla_\theta \left[J(\theta; q^+) + J(\theta; q)\right]$
  end if
end for
return $\mathbf{v}^*$, θ^*

Co-Training Ordering. These two stages together form a single training step, and their order is crucial. The steering vector $\mathbf{v}$ must be optimized before the model is updated since optimizing $\mathbf{v}$ afterward would leave it fit to a stale model, misaligned with the one that actually generated the *hinted* $\{o_i^+ \sim \pi_\theta(\cdot|q^+)\}_{i=1,\ldots,\hat{G}}$ and *plain* $\{o_i \sim \pi_\theta(\cdot|q)\}_{i=1,\ldots,\hat{G}}$ in Equation 4. Such misalignment causes $\mathbf{v}$ to steer model behavior in a direction that no longer corresponds to the hint's true behavioral shift. This order incurs no interference, as optimizing $\mathbf{v}$ leaves the model untouched. Moreover, because $\mathbf{v}$ is re-optimized at every training step, it tracks the model's growth as training progresses, keeping activation steering faithful to the current model rather than to an outdated one. To preserve this correspondence at termination, the model is not updated at the final training step. Algorithm 1 depicts the training procedure. Empirical training dynamics are shown in Appendix D.

Inference

After all, given a question $\overline{q}$, the trained steering vector $\mathbf{v}^*$ is directly injected into the model's activation at the same layer l (Equation 3) without any accompanying hint $h_{\overline{q}}$. This facilitates the model with activation steering $\pi_{\theta^*}^{\mathbf{v}^*}$ to reproduce the hint-induced behavior on $\overline{q}$, thus precisely satisfying the design goal of ASH.

5 Theoretical Analysis

We provide a theoretical analysis establishing how ASH bounds and shrinks the performance gap caused by the training-inference context mismatch. The complete proof is deferred to Appendix B.

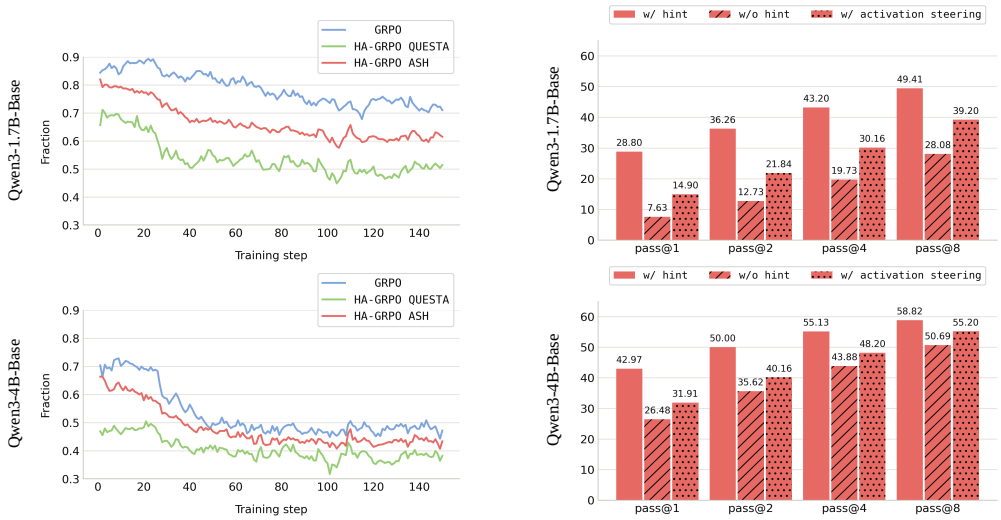
Theorem 1 (Performance Gap Bound with Activation Steering)**.** *Given a training question q and the hinted question q^+ devised from it, and a bounded reward function $\mathcal{R} : \mathcal{O} \to [0, 1]$ over generated responses $o \in \mathcal{O}$, the expected performance gap is upper-bounded by:*

$$|\mathbb{E}_{o\sim\pi_\theta(\cdot|q^+)}[\mathcal{R}(o)] - \mathbb{E}_{o\sim\pi_\theta^{\mathbf{v}}(\cdot|q)}[\mathcal{R}(o)]| \leq \sqrt{2D_{KL}(\pi_\theta(\cdot|q^+) \parallel \pi_\theta^{\mathbf{v}}(\cdot|q))}. \tag{6}$$

Intuitively, minimizing $\mathcal{L}(\mathbf{v}; q)$ increases the likelihood of *hinted* responses relative to *plain* responses, under q, which we expect to move $\pi_\theta^{\mathbf{v}}(\cdot|q)$ toward $\pi_\theta(\cdot|q^+)$ and thus reduce the divergence on the right-hand side and tighten the upper bound on the performance gap.

6 Experiments

In this section, we systematically conduct a series of experiments to empirically evaluate our proposed ASH in expanding the reasoning capability of LLMs. Our evaluation is structured around the following key research questions (RQs):



(a) The fraction of too-difficult questions per batch over the course of training.

(b) pass@k of ASH-trained models, with and without the hint in the questions, and with activation steering.

Figure 3: The impact of the training-inference context mismatch. (a) ASH substantially lessens the fraction of too-difficult questions per batch over the course of training, relative to GRPO. (b) pass@k of ASH-trained models drops sharply without the hint at inference time. However, activation steering significantly restores it.

- **RQ1 - Mechanistic Validity:** Does the regularization of joint training on both hinted questions and plain questions help, and to what extent? Importantly, to what extent does activation steering recover the capability boost that hinting unlocked?
- **RQ2 - Effectiveness:** Does the design of ASH translate into tangible improvements in challenging reasoning benchmarks over standard GRPO and other HA-GRPO methods?
- **RQ3 - Cross-Domain Transferability:** Does the trained steering vector, which was fit on mathematics, transfer effectively to other reasoning domains?

6.1 Setup

Training Data. We employ the challenging OpenR1-MATH-220k dataset (Hugging Face, 2025), comprising 220k mathematics questions collected from diverse sources, each paired with a reference solution generated by DeepSeek-R1 (Guo et al., 2025). Following prior work (Liao et al., 2026; Xia et al., 2026), we curate a subset of 15k questions and utilize it as training data (Appendix C).

Base LLMs. We evaluate four LLMs of varying scales: Qwen3-1.7B-Base and Qwen3-4B-Base (Yang et al., 2025), and Llama-3.2-1B-Instruct and Llama-3.2-3B-Instruct (Grattafiori et al., 2024).

Baselines. We compare ASH against standard GRPO and representative HA-GRPO methods, including QUESTA, SAGE, and HILL.

Implementation Details. For baselines, each model generates 8 responses (i.e., $G = 8$) per training question, with an instruction prepended to form the input prompt: "`Please reason step by step, and put your final answer within \boxed{}`". For ASH, each model instead generates 4 responses to the hinted question and 4 responses to the plain question (i.e., $\hat{G} = 4$), matching the generation budget of baselines, preserving an apples-to-apples comparison. Other key training hyperparameters are provided in Appendix C.

Testing Benchmarks. We evaluate on six benchmarks across two categories: competition-level mathematics with AMC (Li et al., 2024), OlympiadBench (He et al., 2024), AIME24, AIME25, and out-of-distribution with Minerva (Lewkowycz et al., 2022), GPQA-Diamond (Rein et al., 2024).

Metric. We report pass@k (Chen et al., 2021) with k up to 8, which meaningfully reflects the expansion in models' reasoning capability. We provide a discussion of this metric in Appendix E.

Table 1: pass@8 of models on six testing benchmarks. Best results are in **bold**.

Model	Method		AMC	OlympiadBench	AIME24	AIME25	Average	Minerva	GPQA-Diamond	Average
Qwen3-1.7B-Base	Base	-	55.77	46.48	10.24	13.84	31.58	39.10	39.79	39.45
	GRPO	-	56.06	50.14	21.44	14.99	35.66	48.99	61.10	55.05
	HA-GRPO	QUESTA	56.30	51.04	17.30	14.72	34.84	47.69	69.78	58.73
		SAGE	58.65	51.56	20.24	14.88	36.33	48.77	63.54	56.16
		HILL	59.13	52.75	21.72	16.51	37.53	49.23	64.76	57.00
		ASH	**60.73**	**54.59**	**23.79**	**18.91**	**39.50**	**51.52**	**70.15**	**60.84**
Qwen3-4B-Base	Base	-	69.11	59.35	29.27	25.75	45.87	53.00	52.36	52.68
	GRPO	-	71.67	63.83	34.24	32.43	50.54	60.37	73.36	66.86
	HA-GRPO	QUESTA	73.75	64.49	32.00	29.32	49.89	58.61	75.72	67.16
		SAGE	76.25	64.75	33.61	32.15	51.69	60.28	75.75	68.02
		HILL	77.10	65.85	36.40	34.95	53.58	60.49	76.19	68.34
		ASH	**79.21**	**67.71**	**39.11**	**37.31**	**55.83**	**62.65**	**78.28**	**70.47**
Llama-3.2-1B-Instruct	Base	-	30.77	16.80	07.25	00.00	13.71	12.27	10.80	11.54
	GRPO	-	34.75	21.15	11.97	00.00	16.97	20.07	13.85	16.96
	HA-GRPO	QUESTA	38.06	24.00	10.86	00.00	18.23	18.20	16.10	17.15
		SAGE	40.66	24.56	11.56	02.24	19.75	19.21	14.85	17.03
		HILL	41.02	25.14	13.10	03.83	20.77	20.91	15.52	18.21
		ASH	**43.57**	**27.18**	**14.33**	**05.45**	**22.63**	**21.05**	**20.97**	**21.01**
Llama-3.2-3B-Instruct	Base	-	49.23	38.09	19.72	00.83	26.97	36.87	26.34	31.60
	GRPO	-	51.58	38.95	22.46	04.17	29.29	39.14	26.34	32.74
	HA-GRPO	QUESTA	54.40	40.38	19.35	04.81	29.73	38.12	28.11	33.11
		SAGE	55.60	43.87	21.16	05.58	31.55	38.37	28.29	33.33
		HILL	57.06	44.80	24.24	06.06	33.04	39.88	28.31	34.09
		ASH	**61.19**	**46.25**	**25.99**	**07.34**	**35.19**	**40.91**	**31.50**	**36.20**

6.2 RQ1 - Mechanistic Validity

With the described setup, we first report the impact of the training-inference context mismatch. As shown in Figure 3a, ASH substantially lessens the fraction of too-difficult questions per batch over the course of training, relative to GRPO, aligned with expectations. We then evaluate ASH-trained models on training data, with and without the hint in the questions. As shown in Figure 3b, pass@k drops sharply without the hint at inference time; for example, pass@8 falls by **21.33**% and **08.13**% for Qwen3-1.7B-Base and Qwen3-4B-Base, respectively.

The regularization helps. ASH retains the core properties of HA-GRPO methods like QUESTA, while the regularization introduced by joint training on both hinted questions and plain questions positions it in the *middle* of GRPO and QUESTA. This simple regularization reduces the extent to which models trained with ASH rely on the hint as a crutch during training and generalizes better to the hint-free setting during inference, as reflected in narrowed gaps: **21.33**% and **08.13**%, compared to **27.34**% and **10.83**%.

Moreover, activation steering recovers about half of the capability boost that hinting unlocked. Figure 3b also shows that activation steering significantly restores pass@k of ASH-trained models, raising pass@8 back by **11.11**% and **04.51**% for Qwen3-1.7B-Base and Qwen3-4B-Base, i.e., about half of the gaps. This supports the idea that the steering vector captured the question-agnostic component of the behavioral shift the hint induces.

6.3 RQ2 - Effectiveness

Having achieved mechanistic validity, we next test whether ASH's design produces concrete improvements in pass@k, evaluated on both competition-level mathematics benchmarks and out-of-distribution benchmarks. Table 1 reports pass@8 of models on six testing benchmarks. The remaining pass@k are provided in Appendix F.

ASH improves pass@8 across four models and all six benchmarks. Compared to GRPO, ASH achieves comprehensive and significant performance gains across all base models. On average, it improves pass@8 by **3.84**%/**5.29**% on competition-level mathematics and **5.79**%/**3.61**% on out-of-distribution benchmarks for Qwen3-1.7B/4B-Base, and by **5.66**%/**5.90**% on competition-level mathematics and **4.05**%/**3.46**% on out-of-distribution benchmarks for Llama-3.2-1B/3B-Instruct.

ASH surpasses existing HA-GRPO methods. Relative to GRPO, QUESTA yields inconsistent performance gains across benchmarks, falling below GRPO on the challenging AIME24/AIME25 and Minerva. HILL, which employs a more advanced hint-extraction mechanism, achieves consistent gains, but at nearly double the computational cost (Appendix C). ASH, in contrast, achieves this consistency with only manageable computational overhead. Over these methods, ASH delivers average gains of **2.60**%/**2.21**% for Qwen3-1.7B/4B-Base, and **2.17**%/**2.14**% for Llama-3.2-1B/3B-Instruct. These gains split into **1.97**%/**2.25**% on competition-level mathematics and **3.84**%/**2.13**% on out-of-distribution benchmarks for Qwen3-1.7B/4B-Base, and **1.86**%/**2.15**% on competition-level mathematics and **2.80**%/**2.11**% on out-of-distribution benchmarks for Llama-3.2-1B/3B-Instruct.

These results demonstrate that the training-inference context mismatch limits performance gains achievable by existing HA-GRPO methods. By mitigating it, ASH substantially expands the reasoning capability of LLMs, with gains that generalize consistently across model families and scales.

Besides, we show an ablation study in Appendix G that isolates the contribution of activation steering to ASH's performance gains by evaluating ASH-trained models with and without applying it.

6.4 RQ3 - Cross-Domain Transferability

We investigate whether the trained steering vector fit on mathematics can transfer to other reasoning domains. Thus, we examine ASH-trained models with and without applying activation steering on two domains beyond mathematics: logical question answering and coding, using LogiQA benchmark and LiveCodeBench-v5 benchmark (Liu et al., 2020; Jain et al., 2025), respectively. Table 2 reports pass@8 on these two benchmarks.

Table 2: pass@8 of ASH-trained models on the LogiQA and LiveCodeBench-v5 benchmarks.

Model		LogiQA	LiveCodeBench-v5
Qwen3-1.7B-Base	w/o activation steering	79.57	16.13
	w/ activation steering	88.69	18.77
Qwen3-4B-Base	w/o activation steering	81.87	30.47
	w/ activation steering	89.10	33.47
Llama-3.2-1B-Instruct	w/o activation steering	54.84	05.74
	w/ activation steering	59.10	08.20
Llama-3.2-3B-Instruct	w/o activation steering	68.20	19.35
	w/ activation steering	71.94	20.89

The trained steering vector can transfer effectively to other reasoning domains. On top of ASH-trained models, activation steering brings salient gains in both logical question answering and coding, achieving transferability without any further training. On LogiQA and LiveCodeBench-v5 benchmarks, it improves pass@8 for Qwen3-1.7B/4B-Base by **9.12**%/**7.23**% and **2.64**%/**3.00**%, and for Llama-3.2-1B/3B-Instruct by **4.26**%/**3.74**% and **2.46**%/**1.54**%.

These results indicate that the trained steering vector from our proposed ASH genuinely steers model behavior toward a higher-quality reasoning space and is transferable across reasoning domains.

7 Limitations

One potential limitation of ASH is the computational overhead of training the steering vector. This overhead is modest in practice for two reasons. First, the implicit rewards in Equation 4 require no additional generation: they are computed by parallel prefilling over the *hinted* and *plain* responses that have already been generated. Second, we compute $\mathcal{L}(\mathbf{v}; q)$ only over index-wise pairs of responses rather than all possible combinations, which is sufficient for training the steering vector (Appendix D). As a result, ASH requires about 1.5× the training time of GRPO, substantially less than the 2× required by SAGE and HILL (Appendix C), while achieving higher effectiveness.

8 Conclusion

We introduced ASH, a novel HA-GRPO method that mitigates the training-inference context mismatch by reinterpreting hinting as a means of steering model behavior. Across four LLMs, ASH consistently improves pass@k on all six benchmarks and establishes a steering vector that can transfer effectively across reasoning domains. Moreover, ASH is orthogonal to, and compatible with, other HA-GRPO methods such as SAGE and HILL. Finally, ASH employs a single global steering vector; enriching its capacity with task- or question-specific steering vectors is promising for future work.

AI USE STATEMENT

The authors used a generative AI tool to improve the grammar and readability of portions of the manuscript and to assist with coding. All AI-generated suggestions and code were reviewed, verified, and edited by the authors. The authors take full responsibility for the final content of the manuscript, including the code and analyses.

REFERENCES

Ömer Faruk Akgül, Rajgopal Kannan, Willie Neiswanger, and Viktor Prasanna. Rethinking rl for llm reasoning: It's sparse policy selection, not capability learning. *arXiv preprint arXiv:2605.06241*, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Peixuan Han, Adit Krishnan, Gerald Friedland, Jiaxuan You, and Luyang Kong. Self-aligned reward: Towards effective and efficient reasoners. In *International Conference on Learning Representations*, volume 2026, pp. 155846–155871, 2026.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

Zixuan Huang, Xin Xia, Yuxi Ren, Jianbin Zheng, Xuanda Wang, Zhixia Zhang, Hongyan Xie, Songshi Liang, Zehao Chen, Xuefeng Xiao, et al. Does your reasoning model implicitly know when to stop thinking? *arXiv preprint arXiv:2602.08354*, 2026.

Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL `https://github.com/huggingface/open-r1`.

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

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Thanh-Long V Le, Myeongho Jeon, Kim Vu, Viet Lai, and Eunho Yang. No prompt left behind: Exploiting zero-variance prompts in llm reinforcement learning via entropy-guided advantage shaping. In *International Conference on Learning Representations*, volume 2026, pp. 121956–121982, 2026.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

Jiazheng Li, Hongzhou Lin, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Yi Wu, and Jingzhao Zhang. Questa: Expanding reasoning capacity in llms via question augmentation. In *International Conference on Learning Representations*, volume 2026, pp. 144191–144215, 2026.

Baohao Liao, Hanze Dong, Xinxing Xu, Christof Monz, and Jiang Bian. Self-hinting language models enhance reinforcement learning. *arXiv preprint arXiv:2602.03143*, 2026.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*, 2020.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Vaishnavi Shrivastava, Ahmed H Awadallah, Vidhisha Balachandran, Shivam Garg, Harkirat Behl, and Dimitris Papailiopoulos. Sample more to think less: Group filtered policy optimization for concise reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 98439–98462, 2026.

Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. Improving instruction-following in language models through activation steering. In *International Conference on Learning Representations*, volume 2025, pp. 55790–55823, 2025.

Rohan Surana, Gagan Mundada, Xunyi Jiang, Chuhan Wang, Zhenwei Tang, Difan Jiao, Zihan Huang, Yuxin Xiong, Junda Wu, Sheldon Yu, et al. Generate, filter, control, replay: A comprehensive survey of rollout strategies for llm reinforcement learning. *arXiv preprint arXiv:2605.02913*, 2026.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron Wallace, and David Bau. Function vectors in large language models. In *International conference on learning representations*, volume 2024, pp. 17282–17333, 2024.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Xinyi Wang, Jinyi Han, Zishang Jiang, Jiaqing Liang, Sihang Jiang, Zhaoqian Dai, Ma Shuguang, Fei Yu, Yanghua Xiao, et al. Don't tell the answer, truly guide the reasoning during rl rollouts. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 3437–3455, 2026.

Jan Wehner, Sahar Abdelnabi, Daniel Tan, David Krueger, and Mario Fritz. Taxonomy, opportunities, and challenges of representation engineering for large language models. *arXiv preprint arXiv:2502.19649*, 2025.

Yu Xia, Canwen Xu, Zhewei Yao, Julian McAuley, and Yuxiong He. Learning to hint for reinforcement learning. *arXiv preprint arXiv:2604.00698*, 2026.

Wenyi Xiao, Zechuan Wang, Leilei Gan, Shuai Zhao, Zongrui Li, Ruirui Lei, Wanggui He, Luu Anh Tuan, Long Chen, Hao Jiang, et al. A comprehensive survey of direct preference optimization: Datasets, theories, variants, and applications. *arXiv preprint arXiv:2410.15595*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Jixiao Zhang and Chunsheng Zuo. Grpo-lead: A difficulty-aware reinforcement learning approach for concise mathematical reasoning in language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5642–5665, 2025.

Xichen Zhang, Sitong Wu, Yinghao Zhu, Haoru Tan, Shaozuo Yu, Ziyi He, and Jiaya Jia. Scaf-grpo: Scaffolded group relative policy optimization for enhancing llm reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 131946–131974, 2026a.

Xuechen Zhang, Zijian Huang, Yingcong Li, Chenshun Ni, Jiasi Chen, and Samet Oymak. Bread: Branched rollouts from expert anchors bridge sft & rl for reasoning. *Advances in Neural Information Processing Systems*, 38:96726–96752, 2026b.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A CREATING HINTED QUESTIONS

We provide an example of how hinted questions are devised from plain questions. Note that we adopt the simple hint-extraction mechanism $\mathcal{F}$ of QUESTA (Li et al., 2026).

An example:

Plain Question

Concatenate the natural number N to the right of every natural number (for example: appending 2 to 35 results in 352). If the new number can always be divided by N, then N is called a magic number. Among the natural numbers less than 130, the number of magic numbers is _____.

Reference Solution

To determine the number of magic numbers less than 130, we need to identify natural numbers N such that appending N to any natural number results in a number divisible by N. This requires N to divide 10^d, where d is the number of digits in N.

1. **For 1-digit numbers** ($d = 1$)**:**
   - Divisors of $10^1 = 10$ that are 1-digit numbers: $1, 2, 5$.
2. **For 2-digit numbers** ($d = 2$)**:**
   - Divisors of $10^2 = 100$ that are 2-digit numbers: $10, 20, 25, 50$.
3. **For 3-digit numbers** ($d = 3$)**:**
   - Divisors of $10^3 = 1000$ that are 3-digit numbers less than 130: $100, 125$.

Listing all magic numbers less than 130:

- 1-digit: $1, 2, 5$ (3 numbers)
- 2-digit: $10, 20, 25, 50$ (4 numbers)
- 3-digit: $100, 125$ (2 numbers)

Summing these, we get a total of $3 + 4 + 2 = 9$ magic numbers.
Thus, the number of magic numbers less than 130 is $\boxed{9}$.

Hinted Question

Concatenate the natural number N to the right of every natural number (for example: appending 2 to 35 results in 352). If the new number can always be divided by N, then N is called a magic number. Among the natural numbers less than 130, the number of magic numbers is _____.

The following text is the beginning part of the solution, which you can refer to for solving the problem:
To determine the number of magic numbers less than 130, we need to identify natural numbers N such that appending N to any natural number results in a number divisible by N. This requires N to divide 10^d, where d is the number of digits in N.

1. **For 1-digit numbers** ($d = 1$)**:**
   - Divisors of $10^1 = 10$ that are 1-digit numbers: $1, 2, 5$.
2. **For 2-digit numbers** ($d = 2$)**:**
   - Divisors of $10^2 = 100$ that are 2-digit numbers: $10, 20, 25, 50$.
3. **For 3-digit numbers** ($d = 3$)**:**
   - Divisors of $10^3 = 1000$ that are 3-digit numbers less than 130: $100, 125$.

B Proof of Theorem 1

Proof.

The absolute difference in expected rewards (expected performance gap) is written as:

$$|\mathbb{E}_{o\sim\pi_\theta(\cdot|q^+)}[\mathcal{R}(o)] - \mathbb{E}_{o\sim\pi_\theta^{\mathbf{v}}(\cdot|q)}[\mathcal{R}(o)]| = \left|\sum_{o\in\mathcal{O}} \mathcal{R}(o)(\pi_\theta(o|q^+) - \pi_\theta^{\mathbf{v}}(o|q))\right|$$

Since the reward function $\mathcal{R}$ is bounded such that $0 \le \mathcal{R}(o) \le 1$ for all $o \in \mathcal{O}$:

$$\left|\sum_{o\in\mathcal{O}} \mathcal{R}(o)(\pi_\theta(o|q^+) - \pi_\theta^{\mathbf{v}}(o|q))\right| \le \sum_{o\in\mathcal{O}} \mathcal{R}(o)|\pi_\theta(o|q^+) - \pi_\theta^{\mathbf{v}}(o|q)| \le \sum_{o\in\mathcal{O}} |\pi_\theta(o|q^+) - \pi_\theta^{\mathbf{v}}(o|q)|$$

By definition, the Total Variation (TV) distance between two distributions is:

$$D_{\text{TV}}(\pi_\theta(\cdot|q^+), \pi_\theta^{\mathbf{v}}(\cdot|q)) = \frac{1}{2}\sum_{o\in\mathcal{O}} |\pi_\theta(o|q^+) - \pi_\theta^{\mathbf{v}}(o|q)|$$

Substituting this into the inequality gives:

$$|\mathbb{E}_{o\sim\pi_\theta(\cdot|q^+)}[\mathcal{R}(o)] - \mathbb{E}_{o\sim\pi_\theta^{\mathbf{v}}(\cdot|q)}[\mathcal{R}(o)]| \le 2D_{\text{TV}}(\pi_\theta(\cdot|q^+), \pi_\theta^{\mathbf{v}}(\cdot|q))$$

Applying Pinsker's Inequality, which relates the Total Variation distance to the KL divergence:

$$D_{\text{TV}}(\pi_\theta(\cdot|q^+), \pi_\theta^{\mathbf{v}}(\cdot|q)) \le \sqrt{\frac{1}{2} D_{\text{KL}}(\pi_\theta(\cdot|q^+) \parallel \pi_\theta^{\mathbf{v}}(\cdot|q))}$$

Substituting this back into the bound yields:

$$|\mathbb{E}_{o\sim\pi_\theta(\cdot|q^+)}[\mathcal{R}(o)] - \mathbb{E}_{o\sim\pi_\theta^{\mathbf{v}}(\cdot|q)}[\mathcal{R}(o)]| \le \sqrt{2D_{\text{KL}}(\pi_\theta(\cdot|q^+) \parallel \pi_\theta^{\mathbf{v}}(\cdot|q))}$$

Recall and simplify our DPO-style $\mathcal{L}(\mathbf{v}; q)$:

$$\mathcal{L}(\mathbf{v}; q) = -\left[\frac{1}{\hat{G}}\sum_{i=1}^{\hat{G}} \log\sigma\left(\beta\log\frac{\pi_\theta^{\mathbf{v}}(o_i^+|q)}{\pi_\theta(o_i^+|q)} - \beta\log\frac{\pi_\theta^{\mathbf{v}}(o_i|q)}{\pi_\theta(o_i|q)}\right)\right]$$

□

Connecting $\mathcal{L}(\mathbf{v}; q)$ to To the Bound. $\mathcal{L}(\mathbf{v}; q)$ maps directly onto DPO (Rafailov et al., 2023) by establishing preference between the *hinted* responses $o^+ \sim \pi_\theta(\cdot|q^+)$ relative to the *plain* responses $o \sim \pi_\theta(\cdot|q)$, evaluated under q with the reference model π_θ. Within DPO, the implicit reward function parameterized by the model with activation steering $\pi_\theta^{\mathbf{v}}$ relative to the reference model π_θ is defined as:

$$r^{\mathbf{v}}(q, o) = \beta\log\frac{\pi_\theta^{\mathbf{v}}(o|q)}{\pi_\theta(o|q)}.$$

Importantly, minimizing $\mathcal{L}(\mathbf{v}; q)$ performs maximum likelihood estimation of this implicit reward under the Bradley-Terry preference framework. As demonstrated by Rafailov et al. (2023), this estimation procedure aligns $\pi_\theta^{\mathbf{v}}(\cdot|q) \propto \pi_\theta(\cdot|q)\exp\left(\frac{1}{\beta}r^*(q,\cdot)\right)$ with the target (optimal) $\pi^*(\cdot|q)$ – here, defined as $\pi_\theta(\cdot|q^+)$ – that is, it pushes $\pi_\theta^{\mathbf{v}}(\cdot|q)$ to track $\pi_\theta(\cdot|q^+)$. Therefore, minimizing $\mathcal{L}(\mathbf{v}; q)$ is a surrogate for minimizing $D_{\text{KL}}(\pi_\theta(\cdot|q^+) \parallel \pi_\theta^{\mathbf{v}}(\cdot|q))$.

C Training Data and Hyperparameters

Due to limited computational resources, we follow prior work (Liao et al., 2026; Xia et al., 2026) to curate a condensed subset of the OpenR1-MATH-220k dataset (Hugging Face, 2025) as follows. Starting from the default 94k subset, we apply Math-Verify to filter out questions whose reference solution cannot be validly parsed, yielding a 64k subset. From this, 15k questions whose reference solution is under 4096 tokens are randomly selected and utilized as training data.

We implement ASH using `verl` (Sheng et al., 2025), and for baselines, we adopt their publicly available implementations. Key training hyperparameters for reproducibility are listed in Table 3. We run all experiments on 2 NVIDIA A100 80GB GPUs, and report the corresponding training times in Table 4. A max prompt length of 1088 ensures that no questions are filtered. Empirically, our training logs indicate that the response truncation rate was less than 5% when capping the maximum response length at 4096 tokens. This indicates that a max response length of 4096 does not affect our observations and conclusions while reducing computational expense. Following prior research (Zou et al., 2023), we choose the **middlemost** layer of each model for activation steering, avoiding the extensive layer-wise sweeps that would otherwise demand high computational expense. We fix the scaling factor β to 0.1, as in the original DPO.

Table 3: Key Training Hyperparameters.

max prompt length	1088
max response length	4096
temperature	1.0
step size	1e-6
warmup steps ratio	0.1
warmup style	cosine
KL loss coefficient	0.0
clip ratio (lower)	0.2
clip ratio (upper)	0.2
batch size	128
mini batch size	64
micro batch size per GPU	8
total epochs	1

Table 4: Training Times of ASH and Baselines.

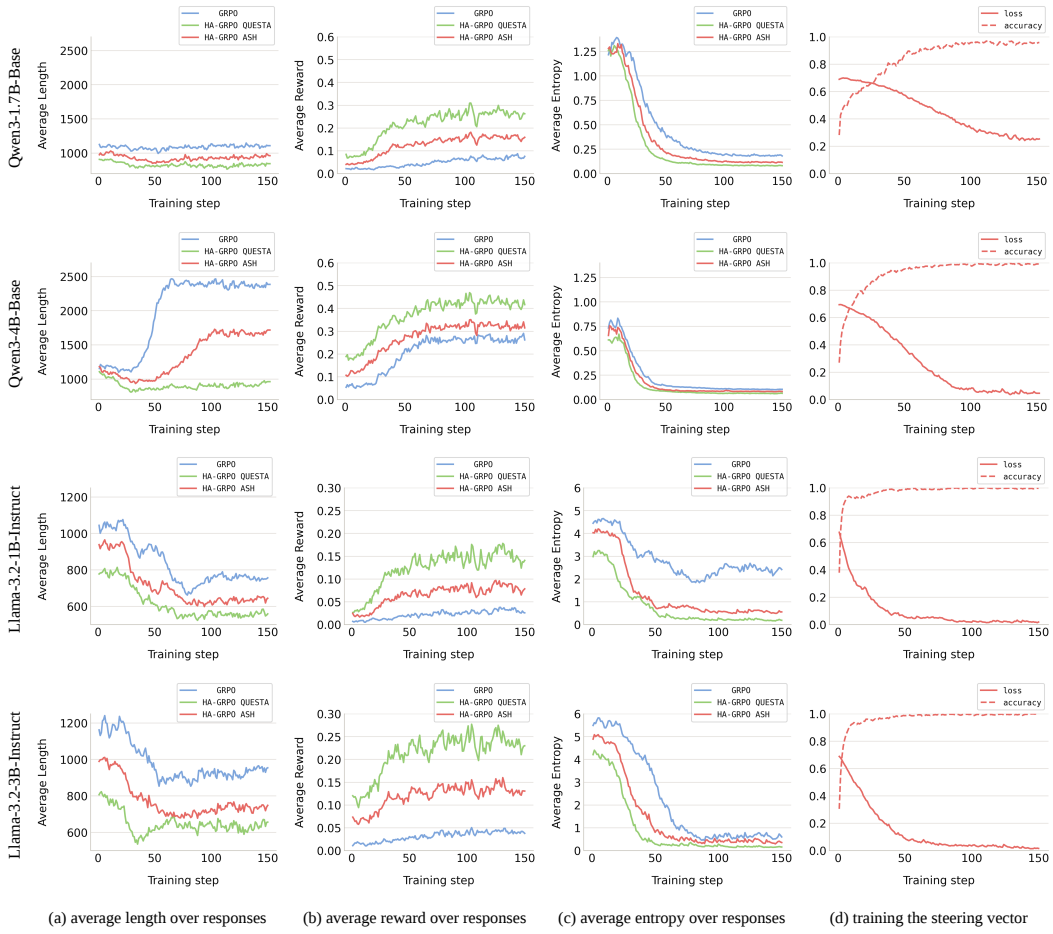
Model	Method		Training Times
Qwen3-1.7B-Base	GRPO	-	∼10 hours
	HA-GRPO	QUESTA	∼12 hours
		SAGE	∼18 hours
		HILL	∼20 hours
		ASH	∼15 hours
Qwen3-4B-Base	GRPO	-	∼20 hours
	HA-GRPO	QUESTA	∼24 hours
		SAGE	∼36 hours
		HILL	∼40 hours
		ASH	∼30 hours

D Training Dynamics

In this section, we report supplementary training dynamics that offer deeper insight into the training patterns of ASH and the baselines, and further support the claims made in the paper.

Figures 4a, b, and c display, respectively, the average response length, reward, and entropy of the generation probabilities per batch over the course of training. These metrics saturate during the final third of training, indicating that models have converged. Moreover, across three metrics, ASH consistently falls *between* GRPO and QUESTA, reflecting the influence of the regularization introduced by joint training on both hinted questions and plain questions. Notably, the average response reward remains below 0.5 for Qwen3-1.7B-Base and Qwen3-4B-Base and 0.3 for Llama-3.2-1B-Instruct and Llama-3.2-3B-Instruct, highlighting the difficulty of questions in the training data.

Meanwhile, Figure 4d shows the loss $\mathcal{L}(\mathbf{v}; q)$ (Equation 4) and accuracy for training the steering vector, which decrease and increase as expected, respectively, indicating that the steering vector is trained properly, where accuracy is defined as the frequency of positive margins.



(a) average length over responses (b) average reward over responses (c) average entropy over responses (d) training the steering vector

Figure 4: Empirical training dynamics.

E On the pass@k Metric

Here, we provide a thorough discussion of the pass@k metric used in the paper.

Definition. pass@k (Chen et al., 2021) measures the probability that at least one of k independently generated responses from the model succeeds (i.e., produces a correct final answer) for a given question. Formally, given the question and n responses to it, of which c succeed, an unbiased estimator of pass@k is:

$$\text{pass@}k = \mathbb{E}_{\text{questions}}\left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right].$$

When $k = 1$, the metric reduces to the average success rate under temperature-controlled generation; when $k > 1$, it captures the model's ability to produce a correct final answer at least once across multiple attempts. We compute the metric using a temperature of 0.6 and a top-p of 0.95.

A model might fail to solve a difficult question on a single attempt yet still possess the latent ability to solve it given more tries; conventional metrics miss this distinction entirely. Meanwhile, pass@k with $k > 1$ addresses this by considering k generated responses and counting the question as solved if any one of those responses is correct, directly testing whether the correct reasoning path exists anywhere within the model's output distribution rather than just whether it's the most likely output.

Therefore, we use pass@k to measure the expansion of a model's reasoning capability, and, to balance evaluation coverage against computational cost, we limit our evaluation to $k = 8$.

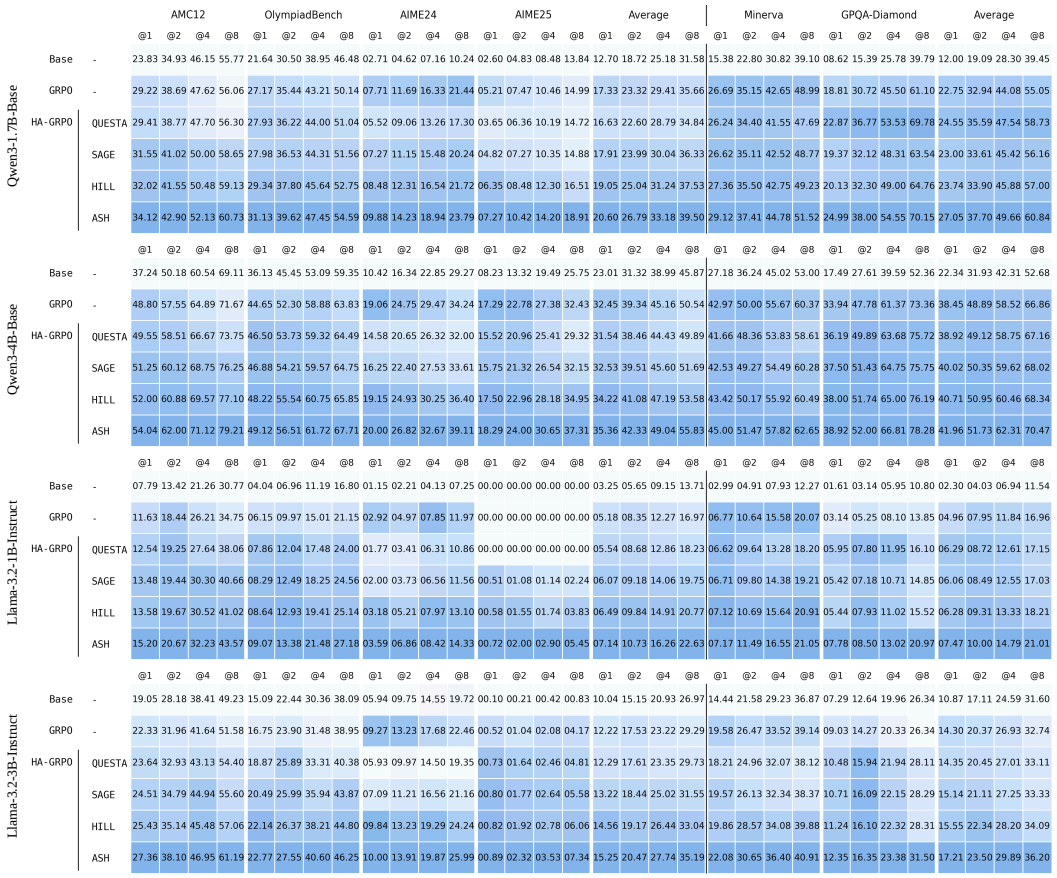


Figure 5: pass@k of models on six testing benchmarks. Shading ranks results within each column.

F Full pass@k

We report full pass@k (pass@1/pass@2/pass@4/pass@8) in Figure 5, complementing the pass@8 alone presented in Table 1. Remarkably, observations are consistent across all k.

G Ablation Study

To quantify how much of ASH's performance gains come from activation steering, we evaluate ASH-trained models with and without applying it. The remaining improvements over GRPO can be attributed to regularization from joint training. Table 5 reports pass@8 on six testing benchmarks.

Applying activation steering plays a pivotal role in ASH's performance gains. In pass@8, applying it accounts for **3.78**%/**4.55**% on competition-level mathematics and **1.72**%/**2.66**% on out-of-distribution benchmarks for Qwen3-1.7B/4B-Base, and for **3.70**%/**4.66**% on competition-level mathematics and **3.13**%/**2.92**% on out-of-distribution benchmarks for Llama-3.2-1B/3B-Instruct. Compared to GRPO, on average, applying activation steering contributes to approximately 68.9%/83.1% for Qwen3-1.7B/4B-Base, and 68.6%/80.2% for Llama-3.2-1B/3B-Instruct.

These results imply the importance of the trained steering vector, which indeed genuinely steers model behavior toward a higher-quality reasoning space.

H Failure Case Study

To intuitively understand ASH's drawbacks, we explore cases where it fails to rectify model behavior. Table 6 shows a representative case on Qwen3-1.7B-Base and Qwen3-4B-Base.

Table 5: pass@8 of ASH-trained models on six testing benchmarks.

Model		AMC	OlympiadBench	AIME24	AIME25	Average	Minerva	GPQA-Diamond	Average
Qwen3-1.7B-Base	w/o activation steering	57.35	51.51	19.28	14.74	35.72	48.31	69.92	59.12
	w/ activation steering	60.73	54.59	23.79	18.91	39.50	51.52	70.15	60.84
Qwen3-4B-Base	w/o activation steering	75.15	64.68	33.45	31.82	51.28	59.88	75.74	67.81
	w/ activation steering	79.21	67.71	39.11	37.31	55.83	62.65	78.28	70.47
Llama-3.2-1B-Instruct	w/o activation steering	39.31	24.38	11.22	00.80	18.93	18.81	16.94	17.88
	w/ activation steering	43.57	27.18	14.33	05.45	22.63	21.05	20.97	21.01
Llama-3.2-3B-Instruct	w/o activation steering	55.21	41.18	20.47	05.27	30.53	38.32	28.25	33.28
	w/ activation steering	61.19	46.25	25.99	07.34	35.19	40.91	31.50	36.20

We present a representative case that illustrates a failure mode of ASH, in which activation steering causes Qwen3-1.7B-Base and Qwen3-4B-Base to drift slightly from the input question (highlighted in red), leading to an incorrect reasoning path and final answer. We hypothesize that this phenomenon stems from the limited capacity of a single steering vector. Enriching this capacity with task- or question-specific steering vectors is a promising direction for future work.

Table 6: A representative case that illustrates a failure mode of ASH.

Model	Question: N is an integer, its representation in base b is 777. Find the smallest positive integer b such that N is a fourth power of an integer in decimal notation. Ground Truth: 18
Qwen3-1.7B-Base	To solve this problem, we need to find the smallest positive integer b such that the number N represented in base b as 777 is a perfect power of an integer in decimal notation. Step 1: Convert 777 from base b to decimal notation $$N = 7b^2 + 7b + 7 = 7(b^2 + b + 1)$$ Step 2: Set N equal to a perfect power We need N to be a perfect square, so let $N = k^2$ for some integer k: $$7(b^2 + b + 1) = k^2$$ Step 3: Divisibility by 7 Since 7 is prime and $7 \mid k^2$, we have $7 \mid k$. Let $k = 7m$. Then: $$7(b^2 + b + 1) = 49m^2$$ $$b^2 + b + 1 = 7m^2$$ Step 4: Test small values of m ($m = 1$): $$b^2 + b + 1 = 7$$ $$b^2 + b - 6 = 0$$ $$b = \frac{-1 \pm \sqrt{1+24}}{2} = \frac{-1 \pm 5}{2}$$ $b = 2$ (since b must be positive) Since $b = 2$ is a positive integer, it is the smallest possible base. Step 5: Verify the solution $$N = 7 \cdot 2^2 + 7 \cdot 2 + 7 = 28 + 14 + 7 = 49 = 7^2$$ Since $49 = 7^2$ is a perfect square, $b = 2$ satisfies the condition. Final Answer: $\boxed{2}$ *Why it's wrong*: The problem asks for a fourth power, not a square. Base 2 also cannot contain the digit 7.
Qwen3-4B-Base	To solve this problem, we need to follow these steps: Step 1: Convert the base b representation to decimal $$N = 7b^2 + 7b + 7$$ Step 2: Set up the fourth-power condition $$7b^2 + 7b + 7 = k^4$$ Dividing both sides by 7: $$b^2 + b + 1 = \frac{k^4}{7}$$ Since $b^2 + b + 1$ is an integer, $7 \mid k^4$, so $7 \mid k$. The smallest such k is $k = 7$: $$b^2 + b + 1 = \frac{2401}{7} = 343$$ $$b^2 + b - 342 = 0$$ Step 3: Solve the quadratic equation $$b = \frac{-1 \pm \sqrt{1+1368}}{2} = \frac{-1 \pm \sqrt{1369}}{2} = \frac{-1 \pm 37}{2}$$ $b = 18$ or $b = -19$ We discard the negative root, so $b = 18$, which is greater than 7 and therefore a valid base. Step 4: Compute N $$N = 7(18)^2 + 7(18) + 7 = 2268 + 126 + 7 = 2401 = 7^4$$ Since $k = 7$ is the smallest possible value of k, $N = 2401$ is the smallest fourth power of the form 777_b. Therefore, the answer to the problem is N. Final Answer: $\boxed{2401}$ *Why it's wrong*: The question asks for the smallest base b, not the value of N. The model finds $b = 18$ along the way but reports $N = 2401$ instead.