Goodhart-Aware Test-Time Reasoning

Anonymous authors
Paper under double-blind review

Abstract

Recent work has shown that injecting learnable latent tokens into frozen large language models (LLMs) and optimizing them at test time by maximizing a confidence-based proxy reward improves reasoning performance. However, we show that optimizing such rewards without bound can lead to Goodhart overoptimization, where accuracy peaks at a problem-dependent threshold and degrades beyond it. We propose **GoATT-R** (Goodhart-Aware Test-Time Reasoning), which treats the frozen LLM as an uncertain system and steers its output using an Ensemble Kalman Filter (EnKF) that tracks a prescribed proxy target reward rather than maximizing the proxy indefinitely. To set this target adaptively, GoATT-R employs a lightweight online policy network, trained via SGD during evaluation, that predicts a suitable optimization target from prompt features and initial ensemble statistics using majority-vote agreement as a label-free reward signal. GoATT-R consistently outperforms state of the art baselines across multiple language models and reasoning benchmarks.

1 Introduction

Test-time compute scaling has emerged as a promising paradigm for improving LLM reasoning without additional training (Snell et al., 2024). Rather than relying solely on the knowledge encoded during pretraining, these methods allocate additional computation at inference to search for better answers. The key insight is that a frozen model can produce dramatically different outputs depending on the input representation, and optimizing that representation at test time can unlock emergent capabilities. Liu et al. (2025) further demonstrated that even a 1B-parameter model can outperform a 405B model when test-time compute is allocated optimally.

Latent Thought Optimization (LTPO; Ye et al. 2026) operationalizes this idea by injecting special "thought tokens" into the input and optimizing their embeddings using a REINFORCE-style gradient-free update. A confidence-based reward signal measures how peaked the model's output distribution is after conditioning on the thought tokens, and the embeddings are iteratively adjusted to maximize this reward. While elegant, we find that this approach suffers from an overoptimization problem observed both in the original LTPO work (see Figure 2 in Ye et al. 2026) and in our experiments. The confidence-based reward serves as a proxy for answer correctness, but its continued optimization beyond a critical point degrades the very accuracy it aims to improve. We identify this phenomenon as Goodhart's Law in latent space and, unlike prior work, address it directly.

Goodhart's Law in latent space. We observe empirically that the confidence-based reward is only locally informative in the sense that a *small* increase in reward (relative to the initial operating point) correlates with improved accuracy, but *large* increases lead to systematic degradation (Figure 1). This is a manifestation of Goodhart's Law (Goodhart, 1984; Karwowski et al., 2024), when a proxy metric is optimized beyond its region of validity, it ceases to be a good measure. As Gao et al. (2023) demonstrate in the context of RLHF, optimizing a proxy reward too aggressively hinders ground truth performance. Prior work has formalized the degree of optimization as *optimization pressure*,parameterized variously as KL divergence from an initial policy(Gao et al., 2023) or entropy regularization strength (Karwowski et al., 2024); in both cases, increasing pressure beyond a critical point degrades true performance (see Appendix C for a fuller treatment). We observe the same

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.

phenomenon in latent thought optimization where pushing the confidence-based reward too high degrades accuracy. To quantify how far the optimization is pushed, we define the *optimization pressure* as the desired increase in the confidence-based reward relative to the unoptimized baseline. The latent embeddings are then optimized toward this prescribed reward level rather than toward unbounded reward maximization. LTPO has no mechanism to control this pressure. It maximizes the confidence-based reward(proxy) without bound and selects the answer from the step with the highest proxy reward, which under Goodhart dynamics risks preferentially choosing over-optimized steps.

The optimal pressure is unobservable at test time. Even if one knew that a critical optimization pressure existed, identifying it would require sweeping over all possible pressures and evaluating accuracy at each which requires ground-truth labels and is infeasible at inference. A fixed optimization pressure chosen on one dataset may be far from optimal on another, and even within a dataset, different problems may tolerate different levels of pressure. This motivates learning the optimization pressure online during evaluation, which we address through a lightweight policy network that adapts the optimization pressure using only a label-free signal.

These observations motivate **GoATT-R** (Goodhart-Aware Test-Time Reasoning). We replace LTPO's unbounded REINFORCE update with an Ensemble Kalman Filter that drives the confidence-based reward toward a prescribed target rather than maximizing it without bound. As the reward approaches this target, the filter's error vanishes and updates become negligible, providing a built-in safeguard against over-optimization. To set the optimization pressure, a lightweight policy network is trained online during evaluation using majority-vote agreement strength as a label-free reward signal, requiring no ground-truth labels and no updates to the frozen model. We aggregate answers across the optimization trajectory via majority voting, which consistently outperforms LTPO's best-reward selection. Our experiments show that GoATT-R outperforms both LTPO and standard chain-of-thought baselines across four language models and five reasoning benchmarks. To our knowledge, we are the first to propose a principled mitigation of Goodharting in test-time optimization. Rather than halting at a predetermined point or requiring a labeled validation sweep, our online policy network learns a suitable optimization pressure from label-free reward signals.

2 Background: Latent Thought Optimization

LTPO (Ye et al., 2026) augments the input to a frozen LLM M_θ with K special thought tokens whose embeddings $H \in \mathbb{R}^{K\times d}$ are optimized at test time. The optimization uses a REINFORCE-style zeroth-order update with perturbation $\xi \sim \mathcal{N}(0, \sigma^2 I)$:

$$H^{(t+1)} = H^{(t)} + \eta\, R(H^{(t)} + \xi)\, \frac{\xi}{\sigma^2} \qquad (1)$$

where η is the learning rate and σ is decayed geometrically.

Confidence-based reward. The reward measures output distribution concentration over the top-k tokens:

$$R(H) = \frac{1}{K} \sum_{j=1}^{K} C(h_j) \qquad (2)$$

where $C(h_j) = -\frac{1}{k} \sum_{v \in \text{top}(k, P_j)} \log P_j(v)$.

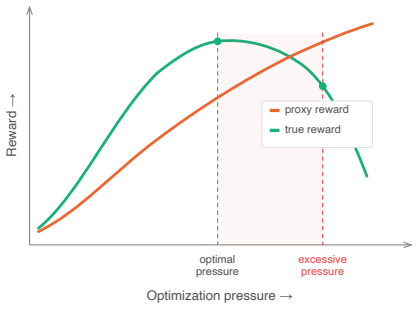

Figure 1: Goodhart's Law in test-time optimization: accuracy peaks at moderate optimization pressure and degrades under aggressive optimization.

After T optimization steps, the optimized latent thought representations are denoted by

$$H^* = H^{(T)}.$$

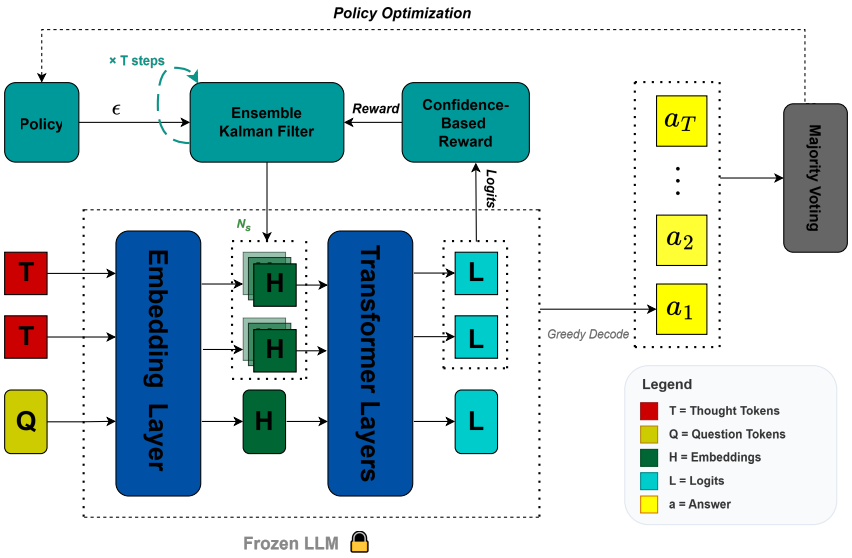

Figure 2: Overview of the GoATT-R framework.

These optimized thought-token embeddings are concatenated with the prompt embeddings H_Q and supplied to the frozen LLM for autoregressive decoding:

$$y = \text{Decoder}(M_\theta(H_Q \,\|\, H^*)), \tag{3}$$

where $\|$ denotes sequence concatenation. Thus, LTPO keeps the LLM parameters frozen and performs test-time optimization exclusively in the latent thought-token space, using the resulting optimized representations to generate the final answer.

3 Method: GoATT-R

3.1 Extremal Goodhart in Confidence Reward Optimization

We now formalize the overoptimization phenomenon described in Section 1. Manheim & Garrabrant (2018) model regime change with a fixed boundary a and constant proxy-goal gaps x and y (Appendix C, Eq. 26). We generalize this to allow the proxy-goal gap to vary with optimization pressure and problem. Let ϵ denote the optimization pressure, s the problem context, $M(\epsilon, s)$ the confidence reward (proxy), and $G(\epsilon, s)$ the resulting accuracy (true goal). We model their relationship as:

$$G(\epsilon, s) = \begin{cases} M(\epsilon, s) + x(\epsilon, s), & \epsilon \le \epsilon^*(s) \\ M(\epsilon, s) + y(\epsilon, s), & \epsilon > \epsilon^*(s) \end{cases} \tag{4}$$

where $x(\epsilon, s)$ and $y(\epsilon, s)$ are the proxy-goal gaps in each regime, and $\epsilon^*(s)$ is the problem-dependent boundary. The two regimes are characterized by:

$$\epsilon \le \epsilon^*(s): \quad \frac{\partial x}{\partial \epsilon} \ge -\frac{\partial M}{\partial \epsilon} \quad \Rightarrow \quad \frac{\partial G}{\partial \epsilon} \ge 0 \quad \text{(accuracy improves or holds)} \tag{5}$$

$$\epsilon > \epsilon^*(s): \quad \frac{\partial y}{\partial \epsilon} < -\frac{\partial M}{\partial \epsilon} \quad \Rightarrow \quad \frac{\partial G}{\partial \epsilon} < 0 \quad \text{(accuracy degrades)} \tag{6}$$

Below ϵ^*, both proxy $M(\epsilon, s)$ and accuracy $G(\epsilon, s)$ increase together. Above ϵ^*, accuracy drops even as confidence continues to rise. The boundary $\epsilon^*(s)$ is the point where the gap dynamics shift from x to y.

Since $\epsilon^*(s)$ is unobservable and varies across problems and datasets, no fixed $\epsilon(s)$ suffices. For this reason, we train a policy network online to learn ϵ^*, using majority-vote agreement as a label-free estimate of G.

3.2 Ensemble Kalman Filter for Latent Thought Optimization

We apply the EnKF from Appendix B to optimize latent thought embeddings. We modify the filter's target measurement by adding the optimization pressure ϵ from Eq. 4, turning ϵ into a throttle that governs how far the filter pushes the embeddings and naturally caps optimization when the target is reached. Section 3.4 introduces the policy network that selects ϵ.

We therefore cast latent thought optimization as a state estimation problem. The state $H_t \in \mathbb{R}^{K\times d}$ represents the thought token embeddings at step t, with $\hat{H}_t$ denoting the posterior mean estimate. Unlike classical EnKF applications where the process model captures physical dynamics, the thought embeddings have no intrinsic dynamics; we wish to find a fixed point that maximizes answer accuracy. We therefore use an identity process model with exploration noise:

$$H_t = H_{t-1} + w_t, \quad w_t \sim \mathcal{N}(0, Q) \tag{7}$$

where $Q = q^2 I$ controls the exploration magnitude.

The observation model evaluates the thought embeddings through the frozen LLM:

$$y_t = R(H_t) + \nu_t, \quad \nu_t \sim \mathcal{N}(0, \Gamma) \tag{8}$$

where $R(\cdot)$ is the confidence reward (Eq. 2) and Γ is the observation noise variance.

The target observation is $\tilde{R} = R_0 + \epsilon$, where R_0 is the reward at the initial (unoptimized) state and $\epsilon \geq 0$ is the optimization pressure. The innovation at each step is:

$$e_t = \tilde{R} - \hat{y}_{t|t-1} \tag{9}$$

At each step, N_s ensemble members are sampled, evaluated through the frozen LLM, and the ensemble statistics (mean reward $\hat{y}_{t|t-1}$, innovation variance S_t, cross-covariance P_{HR}) are computed as described in Appendix B. The Kalman gain $\mathcal{K}_t = P_{HR}/S_t$ and the posterior mean update $\hat{H}_t = \hat{H}_{t|t-1} + \mathcal{K}_t\, e_t$ follow directly.

The key property is that the EnKF drives the embeddings toward the prescribed target $\tilde{R}$ rather than maximizing the reward without bound. As the ensemble mean reward approaches $\tilde{R}$, the innovation $e_t \to 0$ and the updates vanish naturally. This provides a principled mechanism to cap the optimization pressure at the level set by ϵ, directly addressing the Goodhart problem formalized in Section 3.1.

In practice, we maintain a diagonal state covariance $P \in \mathbb{R}^{K\times d}$ rather than the full $(Kd)^2$ matrix. Details are given in Appendix B.

3.3 Trajectory-Based Majority Voting

We decode an answer at each optimization step and apply majority voting across the trajectory. Let $\{H_0, \hat{H}_1, \ldots, \hat{H}_T\}$ be the sequence of embeddings across T optimization steps, with H_0 being the unoptimized thought embeddings. Decoding from each state yields answers $\{a_0, a_1, \ldots, a_T\}$. The trajectory-based majority vote is:

$$a^* = \arg\max_a \sum_{t=0}^{T} \mathbf{1}[a_t = a] \tag{10}$$

Each optimization step corresponds to a different level of optimization pressure: step 0 produces the unoptimized answer, early steps provide mild perturbations, and later steps explore more aggressively. Correct answers tend to be stable across nearby latent states, while incorrect answers vary unpredictably across steps, making majority voting a natural aggregation strategy. The agreement strength of the majority vote also provides a label-free signal of answer quality, which we use as the reward for the policy network that learns a suitable optimization pressure ϵ (Section 3.4).

Algorithm 1 GoATT-R: Goodhart-Aware Test-Time Reasoning

Input: Problems $\{q_1, \ldots, q_N\}$, frozen LLM, ensemble size N_s, steps T
Initialize policy π_θ, replay buffer $\mathcal{B} \leftarrow \emptyset$, exploration rate α
for $n = 1, \ldots, N$ **do**
  Embed prompt $\rightarrow \bar{H}_Q$; initialize thought tokens H_0
  Evaluate initial ensemble $\rightarrow R_0, P_0, S_0$
  Build history context $\mathbf{h}_n$ from $\mathcal{B}$
  Select pressure: $\epsilon_n \leftarrow \begin{cases} U(0, \epsilon_{\max}) & \text{w.p. } \alpha \\ \pi_\theta(\bar{H}_Q, R_0, P_0, S_0, \mathbf{h}_n) & \text{otherwise} \end{cases}$
  Set target: $\tilde{R} \leftarrow R_0 + \epsilon_n$
  for $t = 1, \ldots, T$ **do** ▷ EnKF optimization
    Sample, evaluate, and update ensemble toward $\tilde{R}$
  end for
  Decode answers $a_0, \ldots, a_T$ from each step
  $a_n^* \leftarrow \text{majority_vote}(a_0, \ldots, a_T)$
  $r_n \leftarrow \text{agreement}(a_0, \ldots, a_T)$ ▷ Label-free reward
  $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\bar{H}_Q, R_0, P_0, S_0, \epsilon_n, r_n)\}$
  Update π_θ on $\mathcal{B}$ via Eq. 11
  $\alpha \leftarrow \max(\alpha \cdot \gamma, \alpha_{\min})$
end for

3.4 Adaptive Optimization Pressure via Online Policy Learning

Since the optimal pressure is unknown and cannot be determined without labels, we formulate pressure selection as a contextual bandit that learns a suitable operating point online. At each problem the agent observes a context, selects one continuous action (the optimization pressure ϵ), and receives one scalar reward based on majority-vote agreement. We train a lightweight policy network to output this pressure, requiring no ground-truth labels. Since each dataset is processed in a single pass with few problems, the policy must learn from limited interactions, motivating a simple deterministic policy trained by direct regression rather than a more complex stochastic policy gradient method.

At each problem n, the policy network observes a state $s_n = (\bar{H}_Q, R_0, P_0, S_0)$ consisting of the mean-pooled question token embedding, projected to dimension d_p via a learned linear layer, and the initial EnKF statistics, along with a history context $\mathbf{h}_n$ encoding recent (state, action, reward) interactions. Here R_0 is the confidence reward at the unoptimized state, and P_0, S_0 are the prior variance and innovation variance of the initial ensemble (Appendix B). The policy outputs the optimization pressure directly: $\epsilon_n = \pi_\theta(s_n, \mathbf{h}_n)$, bounded to $[0, \epsilon_{\max}]$ via a sigmoid output layer. During an initial exploration phase, ϵ_n is instead drawn uniformly at random from $[0, \epsilon_{\max}]$ with decaying probability $\alpha_n = \max(\alpha_0 \cdot \gamma^n, \alpha_{\min})$, where α_0 is the initial exploration rate, γ is the decay factor, and $\alpha_{\min}$ is the exploration floor.

After each problem, the policy observes the majority vote agreement r_n (Eq. 10) as reward. We update the policy on a replay buffer $\mathcal{B}$ of all past interactions by minimizing an advantage-weighted squared error:

$$\mathcal{L} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} A_i \cdot (\pi_\theta(s_i, \mathbf{h}_i) - \epsilon_i)^2 \tag{11}$$

where ϵ_i is the optimization pressure that was used, and $A_i = (r_i - \bar{r})/\sigma_r$ is the standardized advantage. Positive advantages pull the policy toward optimization pressures that produced above-average agreement; negative advantages push it away from pressures that fell in the overoptimization region. The sigmoid output bounds π_θ to $[0, \epsilon_{\max}]$, preventing the negative-weight terms from driving the output to infinity. Architectural details are provided in Appendix A.

The full procedure is given in Algorithm 1.

4 Experiments

4.1 Experimental Setup

Models. We evaluate GoATT-R on four open-source instruction-tuned models spanning two families and multiple scales: Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen-2.5-7B-Instruct, and Qwen-2.5-14B-Instruct (Yang et al., 2024).

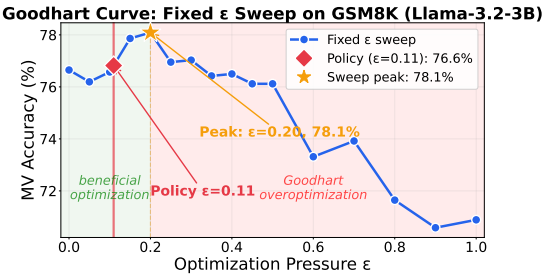

Figure 3: Empirical Goodhart curve on GSM8K (Llama-3.2-3B). Accuracy peaks at ϵ=0.20 and degrades beyond, confirming Eq. 4. The policy learns ϵ=0.11.

Datasets. We use five mathematical reasoning benchmarks: GSM8K (Cobbe et al., 2021) (1319 problems), MATH-500 (Hendrycks et al., 2021) (500 problems), ASDiv-Aug (Miao et al., 2020) (1038 problems), AIME 2024 (30 problems), and AIME 2025 (30 problems). GSM8K and ASDiv test multi-step arithmetic reasoning, MATH-500 covers diverse competition-level topics, and AIME 2024/2025 are high-difficulty competition problems.

Baselines. We compare against three baselines. (1) **CoT**: zero-shot chain-of-thought prompting with greedy decoding and no thought token optimization. (2) **LTPO** (Ye et al., 2026): REINFORCE-based latent thought optimization with best-reward step selection. (3) **LTPO-MV**: LTPO with trajectory majority voting across all optimization steps.

GoATT-R configuration. Following Ye et al. (2026), who find K=8 to be the most robust setting for LTPO, we adopt the same value to ensure a conservative comparison that favors the baseline. We set T=5 optimization steps, ensemble size N_s=50, process noise Q=0.001, observation noise Γ=0.01, and top-k=10 for the confidence reward. The policy network is trained online via SGD (lr = 0.01, no momentum) on the advantage-weighted loss (Eq. 11), updating once per problem on the full replay buffer. We use projection dimension d_p=32 and history length N_h=32. All results use greedy decoding and are reported as accuracy (%).

4.2 Main Results

Table 1 presents our main results. We report two aggregate metrics: Avg_3, the mean accuracy over the three large benchmarks (GSM8K, MATH-500, ASDiv), and Avg_5, the mean over all five benchmarks including AIME 2024 and 2025. GoATT-R achieves the highest Avg_3 and Avg_5 for all four models, demonstrating consistent improvement across both model families and scales.

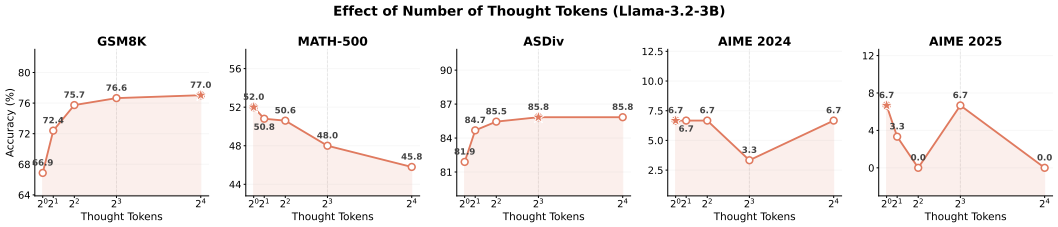

Figure 4: Effect of the number of thought tokens on MV accuracy (Llama-3.2-3B). Performance plateaus at K=4–8 on GSM8K and ASDiv, while MATH-500 peaks at K=1 and degrades with more tokens. The dashed line marks the default K=8 used in all other experiments. Full results across all four models are provided in Figure 5 (Appendix D).

Table 1: Comparison of Chain of Thought (CoT), LTPO, and GoATT-R across models and benchmarks.

Model	Method	GSM8K	MATH-500	ASDiv	AIME 24	AIME 25	Avg_3	Avg_5
Llama-3.2-3B	CoT	65.6	46.4	84.3	3.3	**6.7**	65.4	41.3
	LTPO	67.6	46.8	76.5	**10.0**	0.0	63.6	40.2
	LTPO-MV	69.6	**52.8**	78.6	**10.0**	3.3	67.0	42.9
	GoATT-R	**76.6**	48.0	**85.8**	3.3	**6.7**	**70.1**	**44.1**
Llama-3.1-8B	CoT	76.9	46.0	79.0	3.3	0.0	67.3	41.0
	LTPO	78.2	48.8	73.0	**10.0**	0.0	66.7	42.0
	LTPO-MV	83.2	53.0	78.6	**10.0**	0.0	71.6	45.0
	GoATT-R	**84.8**	**54.2**	**85.2**	6.7	0.0	**74.7**	**46.2**
Qwen-2.5-7B	CoT	88.8	73.8	91.6	10.0	6.7	84.7	54.2
	LTPO	**89.4**	73.4	**91.7**	6.7	6.7	84.8	53.6
	LTPO-MV	89.2	75.0	91.4	13.3	10.0	85.2	55.8
	GoATT-R	89.1	**75.4**	91.3	**16.7**	**16.7**	**85.3**	**57.8**
Qwen-2.5-14B	CoT	92.1	78.8	**92.5**	13.3	**20.0**	87.8	59.3
	LTPO	92.3	77.4	91.7	16.7	6.7	87.1	57.0
	LTPO-MV	93.0	79.8	92.2	16.7	13.3	88.3	59.0
	GoATT-R	**93.3**	**81.0**	92.0	**16.7**	16.7	**88.8**	**59.9**

On the three large benchmarks, GoATT-R outperforms all baselines across every model. The gains are largest on the smaller models where there is more room for improvement. On Llama-3.2-3B, GoATT-R improves GSM8K accuracy by 7.0 points over LTPO-MV and 11.0 points over CoT. On Llama-3.1-8B, GoATT-R reaches 85.2% on ASDiv, surpassing both CoT (79.0%) and LTPO-MV (78.6%) by a wide margin. As model capacity increases, the baselines strengthen and the margins narrow, but GoATT-R maintains a consistent edge on Avg_3 and Avg_5.

On Llama-3.2-3B ASDiv, LTPO scores 76.5% while CoT scores 84.3%, a case where unbounded optimization actively harms performance. GoATT-R recovers this loss and exceeds the CoT baseline (85.8%), demonstrating that controlled optimization avoids the degradation caused by unbounded reward maximization.

AIME 2024 and 2025 (30 problems each) exhibit high variance, where a single problem flip changes accuracy by 3.3%. Despite this, GoATT-R achieves 16.7% on both AIME 2024 and AIME 2025 with Qwen-7B, outperforming all baselines on these benchmarks for that model.

On Qwen-2.5-14B, CoT is a strong baseline that benefits from the model's inherent reasoning capability. GoATT-R still outperforms all methods, achieving the best Avg_3 (88.8%) and Avg_5 (59.9%) across all methods on this model.

4.3 Discussion

Effect of number of thought tokens. Figure 4 varies the number of latent thought tokens $K \in \{2^0, 2^1, 2^2, 2^3, 2^4\}$. On GSM8K and ASDiv, performance increases steadily from K=1 to K=4 and plateaus beyond that across all models. On Llama-3.2-3B GSM8K, the gain from K=1 to K=8 is 9.7 points (66.9% to 76.6%), while increasing to K=16 adds only 0.4 points. MATH-500 behaves differently: on Llama-3.2-3B, performance peaks at K=1 (52.0%) and declines monotonically with more tokens, reaching 45.8% at K=16. A similar but milder pattern appears on Llama-3.1-8B, where the peak shifts to K=4 (57.2%). For the stronger Qwen models, MATH-500 performance is stable across all token counts. This suggests the optimal number of thought tokens may depend on the model-task pair, and K=8 offers a reasonable default across settings. The AIME benchmarks show high variance due to their small size (30 problems) and do not reveal a clear trend.The full sweep across all four models is provided in Figure 5 (Appendix D).

Effect of top-k in the confidence reward. Accuracy is largely insensitive to $k \in \{10, 1000, 10000\}$ across all four models, with differences within 1–2 points on the large benchmarks (Table 3, Appendix E). We retain k=10 following Ye et al. (2026).

Verifying the Goodhart boundary. To verify the regime change predicted by Eq. 4, we run the EnKF (Section 3.2) on GSM8K with Llama-3.2-3B using fixed optimization pressure $\epsilon \in \{0.0, 0.05, 0.10, \ldots, 1.0\}$. Each value of ϵ is held constant across all problems in the dataset, with the policy network disabled. All other hyperparameters remain at their defaults (N_s=50, T=5, Q=0.001, Γ=0.01, k=10). Figure 3 shows the resulting accuracy as a function of ϵ. The inverted-U is clearly visible, with MV accuracy peaking at ϵ=0.20 (78.1%) and declining steadily beyond, losing over 8 points by ϵ=0.90. To compare, we extract the mean ϵ selected by the policy network from the GoATT-R run on the same dataset, obtaining $\bar{\epsilon}$=0.11. The policy operates in the beneficial region of the curve, avoiding the steep degradation beyond the peak while still providing meaningful optimization pressure above the unoptimized baseline.

Effect of ensemble size and compute cost. Table 2 varies the ensemble size $N_s \in \{10, 50, 100\}$. Accuracy improves from N_s=10 to N_s=50 and saturates beyond that, with N_s=100 offering no meaningful gain; we retain N_s=50 as the default. Wall-clock cost is comparable across all methods: GoATT-R at N_s=50 averages 110.2 s per problem versus 88.7 s for LTPO (1.24×) and 85.7 s for CoT (1.29×). Increasing from N_s=10 to N_s=100 adds only 7.2 s (6.6%), as ensemble evaluations are fully batched on a single GPU.

Table 2: Effect of ensemble size N_s on MV accuracy (%, Llama-3.2-3B) and average wall-clock time per problem.

Method	Sec/prob	GSM8K	MATH-500	ASDiv	AIME 24	AIME 25	Avg_5
CoT	85.7	65.6	46.4	84.3	3.3	6.7	41.3
LTPO-MV	88.7	69.6	52.8	78.6	10.0	3.3	42.9
GoATT-R (N_s=10)	108.5	74.4	46.0	84.9	6.7	3.3	43.0
GoATT-R (N_s=50)	110.2	76.6	48.0	85.8	3.3	6.7	44.1
GoATT-R (N_s=100)	115.7	76.7	48.2	85.8	3.3	6.7	44.1

5 Related Work

LTPO (Ye et al., 2026) introduced the idea of injecting special "thought" tokens into a frozen LLM's input and optimizing their embeddings at test time via a zeroth-order REINFORCE-style update. The approach requires no weight modification, no training data, and applies to any frozen model. But, it maximizes a confidence-based proxy reward without any stopping criterion, which we show leads to Goodhart overoptimization.

LatentSeek (Li et al., 2025) replaces the REINFORCE update with policy gradient methods but inherits the same unbounded maximization problem. Furthermore, it requires autoregressive generation at each optimization step, unlike LTPO which optimizes the confidence reward of the first token alone. SoftCoT (Xu et al., 2025a) and SIM-CoT (Wei et al., 2026) also operate on continuous representations but require supervised training to learn the soft prompts or step-level alignments beforehand, tying them to specific models and training distributions. SoftCoT++ (Xu et al., 2025b) adds test-time diversity by perturbing the trained soft tokens and voting over the resulting outputs. In contrast, LTPO and our method are entirely training-free: the latent embeddings are optimized from scratch on each problem, making them applicable to any frozen model without prior data collection or fine-tuning.

Other works reason in continuous spaces at a deeper level. COCONUT (Hao et al., 2024) replaces intermediate token predictions with hidden state propagation, allowing a model to "think" in latent space. While promising, COCONUT requires supervised training.

A separate line of work improves reasoning by updating model weights at test time rather than optimizing input representations. Test-Time Training (TTT) (Sun et al., 2020; 2024) adapts parameters on each test sample using a self-supervised auxiliary loss. TTRL (Zuo et al., 2025) bridges TTT and RL by using majority-voted pseudo-labels as rewards to fine-tune weights on unlabeled test data, demonstrating that models can self-evolve without ground-truth supervision. CoVo (Zhang et al., 2025a) and EMPO (Zhang et al., 2025b) similarly use voting agreement and entropy-based signals as intrinsic rewards for weight-level

RL. Intuitor (Zhao et al., 2026) takes this further by replacing external rewards entirely with the model's own confidence—the same type of signal LTPO uses to optimize latent tokens—but applies it to update model weights via GRPO, matching supervised RL performance on math benchmarks without any labels. These weight-update methods operate at a fundamentally different level than ours. They modify the model's parameters through RL, while we optimize additional thought token embeddings prepended to the input of a frozen model.

While prior work establishes that latent representations can be optimized at test time, it does not provide an explicit mechanism for controlling how far the optimization should proceed. We address this by formulating latent thought optimization as a target-tracking problem: rather than maximizing the confidence reward indefinitely, the latent representation is iteratively updated so that the resulting confidence approaches a prescribed target. The Ensemble Kalman Filter (EnKF) provides a natural ensemble-based formulation of this problem. Originally developed for data assimilation in high-dimensional dynamical systems (Evensen, 1994; 2003), the EnKF requires only forward model evaluations and has subsequently been applied to machine learning. For example, Kovachki & Stuart (2019) used the EnKF as an alternative to backpropagation for training neural networks, demonstrating competitive performance on small-scale problems.

To our knowledge, we are the first to apply the EnKF to test-time optimization of LLM representations. We treat the frozen LLM as an uncertain nonlinear system whose internal dynamics are unknown, with the thought-token embedding $H \in \mathbb{R}^{K \times d}$ serving as the control variable that steers the model's output. The EnKF acts as a gradient-free tracking controller, iteratively updating the thought-token embeddings to steer the LLM's confidence output toward a prescribed target.

6 Conclusion

We presented GoATT-R, a Goodhart-aware approach to test-time latent thought optimization. We replace LTPO's unbounded REINFORCE update with an Ensemble Kalman Filter that drives the confidence-based reward toward a prescribed target rather than maximizing it without bound, providing a built-in safeguard against over-optimization.The Goodhart boundary is unobservable, and as our epsilon sweep on GSM8K confirms (Figure 3), accuracy degrades when optimization pressure exceeds a critical level. To address this, a lightweight policy network is trained online during evaluation to predict a suitable optimization pressure, using majority-vote agreement as a label-free reward signal without requiring ground-truth labels or updates to the frozen model. We aggregate answers across the optimization trajectory via majority voting, which consistently outperforms LTPO's best-reward selection. GoATT-R achieves the best average accuracy across four models and five benchmarks, confirming that adaptive pressure control is effective and practical for mitigating Goodharting at test time without requiring labeled validation data.

AI use statement

In this work, we used generative AI tools for text editing, correcting grammatical mistakes, implementing auxiliary code for experiment management and visualization, and interpreting results through discussion of experimental findings. We have not used generative AI tools to generate synthetic data sets, develop theoretical models or conceptual frameworks, formulate mathematical claims, provide ingredients for or assist in the writing of proofs, propose or refine hypotheses, or design the research methodology or experiments. Data cleaning, translation, and qualitative data analysis are not applicable to this work. All AI-assisted work was reviewed and verified by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

References

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training

verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Geir Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, 99(C5):10143–10162, 1994.

Geir Evensen. The ensemble Kalman filter: Theoretical formulation and practical implementation. *Ocean Dynamics*, 53(4):343–367, 2003.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. *Proceedings of ICML*, 2023.

Charles AE Goodhart. Monetary theory and practice: The UK experience. 1984.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Goshvadi, Shunyu Hao, Kaituo Liu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

Jacek Karwowski, Oliver Hayman, Xingjian Bai, Klaus Kiendlhofer, Charlie Griffin, and Joar Skalse. Goodhart's law in reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 24546–24576, 2024.

Nikola B Kovachki and Andrew M Stuart. Ensemble Kalman inversion: A derivative-free technique for machine learning tasks. *Inverse Problems*, 35(9), 2019.

Hengli Li, Changshuo Li, Tianlong Wu, Xiangyu Zhu, Yuxuan Wang, Zitao Yu, Eric H. Jiang, Sicheng Zhu, Zhangyang Jia, Ying N. Wu, and Ziniu Zheng. Seek in the dark: Reasoning via test-time instance-level policy gradient in latent space. *arXiv preprint arXiv:2505.13308*, 2025.

Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *arXiv preprint arXiv:2502.06703*, 2025.

David Manheim and Scott Garrabrant. Categorizing variants of goodhart's law. *arXiv preprint arXiv:1803.04585*, 2018.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. *arXiv preprint arXiv:2106.15772*, 2020.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning (ICML)*, 2020.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. SIM-CoT: Supervised implicit chain-of-thought. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025a.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. SoftCoT++: Test-time scaling with soft chain-of-thought reasoning. *arXiv preprint arXiv:2505.11484*, 2025b.

An Yang, Baosong Yang, Binyuan Hui, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Wengao Ye, Yan Liang, and Lianlei Shan. Thinking on the fly: Test-time reasoning enhancement via latent thought policy optimization. In *International Conference on Learning Representations*, volume 2026, pp. 156213–156239, 2026.

Kongcheng Zhang, Qi Yao, Shunyu Liu, Yingjie Wang, Baisheng Lai, Jieping Ye, Mingli Song, and Dacheng Tao. Consistent paths lead to truth: Self-rewarding reinforcement learning for LLM reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025a.

Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question is already half the answer: Fully unsupervised LLM reasoning incentivization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. TTRL: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

A Policy Network Architecture

Prompt projection. The mean-pooled question embedding $\bar{H}_Q \in \mathbb{R}^{d_{\text{model}}}$ is projected via a learned linear layer $\mathbf{W}_\phi \in \mathbb{R}^{d_p \times d_{\text{model}}}$ (no bias), followed by LayerNorm, yielding $\tilde{H}_Q \in \mathbb{R}^{d_p}$. We use $d_p = 32$ throughout. The same projection is applied to both the current question and all history entries, ensuring a shared representation space.

History context. The policy conditions on $N_h = 32$ past interactions. Each entry is a tuple $(\tilde{H}_{Q,i}, R_0^{(i)}, P_0^{(i)}, S_0^{(i)}, \epsilon_i, r_i) \in \mathbb{R}^{d_p+5}$, where $\tilde{H}_{Q,i}$ is the projected question embedding and the remaining scalars are the initial EnKF statistics, the optimization pressure used, and the agreement reward obtained. The 32 entries are constructed as follows: the first 16 positions are filled with entries sampled uniformly at random from the replay buffer, and the last 16 are the most recent sequential interactions. Positions are zero-padded when fewer entries are available.

The history is processed by a two-layer 1D convolutional encoder:

$$\mathbf{h}' = \text{ReLU}(\text{Conv1D}_{d_p+5 \to 32,\, k=3}(\mathbf{H}))$$
$$\mathbf{h} = \text{AvgPool}(\text{ReLU}(\text{Conv1D}_{32 \to 16,\, k=3}(\mathbf{h}')))$$

where $\mathbf{H} \in \mathbb{R}^{N_h \times (d_p+5)}$ is the history matrix and both convolutions use padding of 1. Global average pooling over the N_h positions produces a fixed-length history representation $\mathbf{h} \in \mathbb{R}^{16}$.

Policy head. The current state $[\tilde{H}_Q; R_0; P_0; S_0] \in \mathbb{R}^{d_p+3}$ is concatenated with the history representation $\mathbf{h}$, yielding a $(d_p + 3 + 16)$-dimensional input. This is passed through a two-layer MLP: $\text{Linear}(d_p{+}19, 32) \to \text{ReLU} \to \text{Linear}(32, 1) \to \sigma$, where σ is the sigmoid function. The output is scaled by $\epsilon_{\max}$ to produce $\epsilon_n \in [0, \epsilon_{\max}]$.

Training. We use SGD with learning rate 0.01 and no momentum. The network is updated once per problem on the full replay buffer (no subsampling). All parameters — projection, LayerNorm, convolutional encoder, and policy head — are trained jointly. The total parameter count is approximately 138K, dominated by the projection layer ($d_{\text{model}} \times d_p$).

Exploration. During an initial phase, ϵ_n is drawn uniformly from $[0, \epsilon_{\max}]$ with probability $\alpha_n = \max(\alpha_0 \cdot \gamma^n, \alpha_{\min})$, where $\alpha_0 = 0.3$, $\gamma = 0.995$, and $\alpha_{\min} = 0.05$. This decays from 30% to 5% over the first $\sim$700 problems.

B The Ensemble Kalman Filter

The Kalman filter solves the fundamental problem of recovering hidden states from noisy observations of a system. The Ensemble Kalman Filter (EnKF; Evensen 1994; 2003) is a Monte Carlo approximation to the Kalman filter for nonlinear systems. It maintains an ensemble of N_s state vectors $\{z^{(1)}, \ldots, z^{(N_s)}\}$ and estimates the state distribution through their empirical moments. The filter's state estimate is the ensemble mean. At each step, the filter observes a noisy measurement y_t of the true state and corrects ensemble members $\{z_t^{(1)}, \ldots, z_t^{(N_s)}\}$.

Consider a state-space model with state $z_t \in \mathbb{R}^n$, process model $z_t = f(z_{t-1}) + w_t$ with $w_t \sim \mathcal{N}(0, Q)$, and observation model $y_t = g(z_t) + \nu_t$ with $\nu_t \sim \mathcal{N}(0, \Gamma)$. The EnKF proceeds as follows:

Prediction step. Each ensemble member is propagated through the process model:

$$\tilde{z}^{(i)} = f(z_{t-1}^{(i)}) + w_t^{(i)}, \quad w_t^{(i)} \sim \mathcal{N}(0, Q) \tag{12}$$

The prior mean is estimated from the propagated ensemble:

$$\hat{z}_{t|t-1} = \frac{1}{N_s} \sum_{i=1}^{N_s} \tilde{z}^{(i)} \tag{13}$$

Update step. Each member is evaluated through the observation model to obtain predicted observations $\hat{y}^{(i)} = g(\tilde{z}^{(i)})$. The innovation statistics are:

$$\hat{y}_{t|t-1} = \frac{1}{N_s} \sum_{i=1}^{N_s} \hat{y}^{(i)} \tag{14}$$

$$S_t = \frac{1}{N_s - 1} \sum_{i=1}^{N_s} (\hat{y}^{(i)} - \hat{y}_{t|t-1})^2 + \Gamma \tag{15}$$

$$P_{zy} = \frac{1}{N_s - 1} \sum_{i=1}^{N_s} (\tilde{z}^{(i)} - \hat{z}_{t|t-1})(\hat{y}^{(i)} - \hat{y}_{t|t-1}) \tag{16}$$

The Kalman gain and per-member update are:

$$\mathcal{K}_t = P_{zy}\, S_t^{-1} \tag{17}$$

$$z_{t|t}^{(i)} = \tilde{z}^{(i)} + \mathcal{K}_t(y_t - \hat{y}^{(i)}) \tag{18}$$

Diagonal-covariance implementation. In our setting (Section 3.2), the state is the thought-token embedding $H \in \mathbb{R}^{K \times d}$ and the observation is the scalar confidence reward $R(H) \in \mathbb{R}$. We maintain a diagonal state covariance $P \in \mathbb{R}^{K \times d}$, initialized to $P_0 = \sigma^2 \mathbf{1}_{K \times d}$, avoiding the $\mathcal{O}((Kd)^2)$ cost of a full covariance matrix.

At each step t, ensemble members are sampled element-wise:

$$\tilde{H}^{(i)} = \hat{H}_t + \eta^{(i)} \odot \sqrt{P + Q}, \quad \eta^{(i)} \sim \mathcal{N}(0, I_{K \times d}) \tag{19}$$

where $\odot$ is the Hadamard product and $Q = q^2\mathbf{1}_{K\times d}$ is the isotropic process noise. Each member is evaluated through the frozen LLM to obtain scalar rewards $R(\tilde{H}^{(i)})$. The ensemble statistics are:

$$\hat{y}_{t|t-1} = \frac{1}{N_s}\sum_{i=1}^{N_s} R(\tilde{H}^{(i)}), \qquad S_t = \frac{1}{N_s-1}\sum_{i=1}^{N_s}\big(R(\tilde{H}^{(i)}) - \hat{y}_{t|t-1}\big)^2 + \Gamma \tag{20}$$

$$P_{HR} = \frac{1}{N_s-1}\sum_{i=1}^{N_s}\big(\tilde{H}^{(i)} - \hat{H}_t\big)\big(R(\tilde{H}^{(i)}) - \hat{y}_{t|t-1}\big) \ \in \mathbb{R}^{K\times d} \tag{21}$$

where $S_t \in \mathbb{R}$ is the scalar innovation variance and $P_{HR} \in \mathbb{R}^{K\times d}$ is the state–reward cross-covariance. Since S_t is scalar, the Kalman gain is computed element-wise:

$$\mathcal{K}_t = P_{HR} \,/\, S_t \ \in \mathbb{R}^{K\times d} \tag{22}$$

The posterior mean and diagonal covariance are updated as:

$$\hat{H}_{t+1} = \hat{H}_t + \mathcal{K}_t \cdot e_t, \qquad e_t = \tilde{R} - \hat{y}_{t|t-1} \tag{23}$$

$$P_{t+1} = \max\big(P_t + Q - \mathcal{K}_t \odot \mathcal{K}_t \cdot S_t,\ \delta\big) \tag{24}$$

C Goodhart's Law and Optimization Pressure

Goodhart's Law states that "when a measure becomes a target, it ceases to be a good measure" (Goodhart, 1984). Manheim & Garrabrant (2018) formalize overoptimization as occurring "when a metric which can be used to improve a system is used to such an extent that further optimization is ineffective or harmful," and identify four distinct failure modes: *regressional* (selecting for a proxy also selects for noise), *extremal* (optimization pushes into regions where the proxy-goal relationship no longer holds), *causal* (intervention changes the causal structure connecting proxy and goal), and *adversarial* (other agents game the metric).

Subsequent work has introduced specific parameterizations of the degree of optimization, termed *optimization pressure*. Gao et al. (2023) define it as $d = \sqrt{D_{\mathrm{KL}}(\pi\|\pi_{\mathrm{init}})}$ in the context of RLHF and empirically show that the true reward follows an inverted-U as d increases. In their setting, the proxy M is a learned reward model and the true goal G is a gold-standard reward model, both scoring the same quantity (text quality). The two are of the same nature, differing only by noise ($M = G + \text{noise}$), and they accordingly attribute the divergence to *regressional* Goodhart. Karwowski et al. (2024) define optimization pressure through an entropy-regularized objective:

$$\tilde{R}(s,a) = R_{\mathrm{proxy}}(s,a) + \alpha\, H(\pi(s)) \tag{25}$$

where $H(\pi(s))$ is the Shannon entropy of the policy at state s and $\alpha > 0$ controls regularization, with optimization pressure parameterized as $\lambda = e^{-\alpha} \in [0,1]$. They show that increasing λ beyond a critical point decreases true performance.

Manheim & Garrabrant (2018) note that extremal Goodhart can occur in two ways. The first is *model insufficiency* and the second is *regime change*: "the measure may be a correct estimator only in certain regions of the space. In regions where the generating process differs, the use of the measure leads to extremal Goodhart." They model this as:

$$G = \begin{cases} M + x, & M \le a \\ M + y, & M > a \end{cases} \tag{26}$$

where M is the proxy, G is the true goal, and a is the boundary beyond which the proxy-goal gap changes from x to y.

D Token Sweep: All Models

Figure 5 extends the token sweep from Figure 4 to all four models. On GSM8K and ASDiv, the pattern is consistent across models: accuracy rises from K=1 to K=4 and plateaus or

marginally improves through K=16. The gains are largest for the smaller Llama models and diminish as model capacity increases; Qwen-2.5-14B shows near-flat performance across all token counts on both benchmarks.

MATH-500 is the main exception. On Llama-3.2-3B, accuracy decreases monotonically from K=1 (52.0%) to K=16 (45.8%). Llama-3.1-8B shows a milder version of this pattern, peaking at K=4. The Qwen models are largely unaffected, with MATH-500 accuracy stable within 1–2 points across all token counts. AIME 2024 and 2025 fluctuate across all models due to their small size (30 problems) and do not exhibit a consistent trend.

Overall, K=8 sits in the plateau region for the benchmarks where token count matters most and avoids the degradation observed at K=16 on MATH-500 with weaker models, supporting its use as the default throughout our experiments.

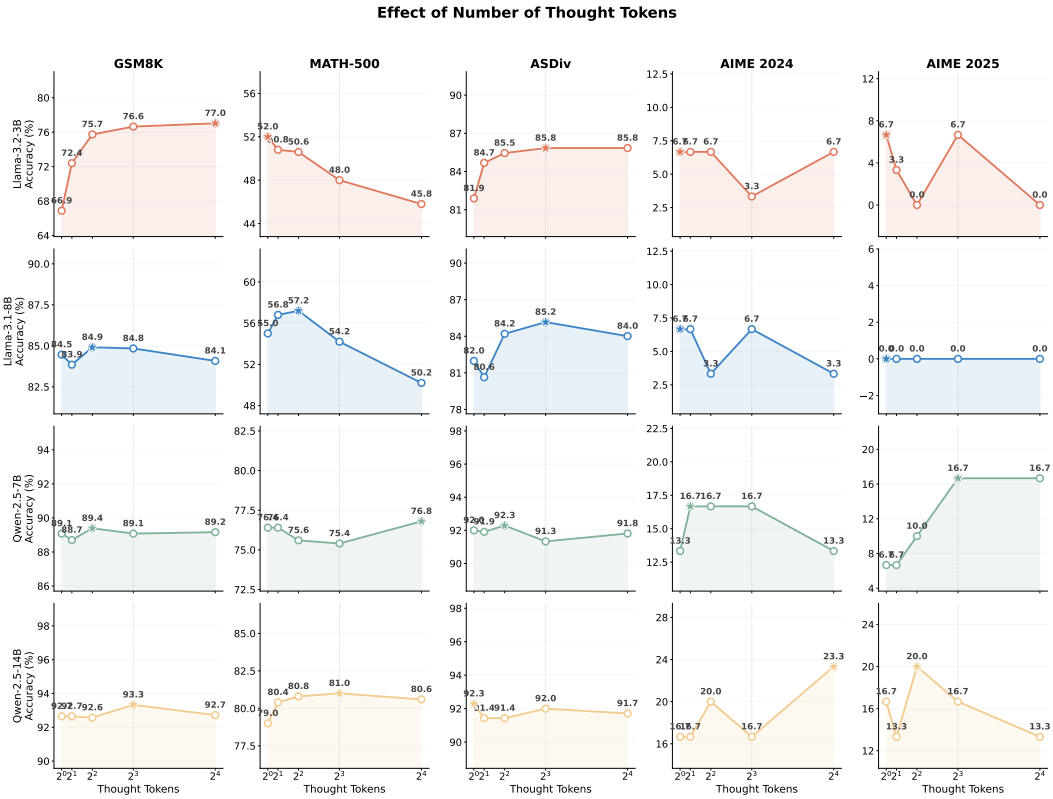

Figure 5: Effect of the number of thought tokens on MV accuracy across four models and five benchmarks. The dashed line marks the default K=8 (2^3) used in all other experiments.

E Top-k Ablation

Table 3 varies the number of tokens $k \in \{10, 1000, 10000\}$ used to compute the confidence reward (Eq. 2). On Llama-3.2-3B, k=10000 achieves the highest GSM8K accuracy (77.9%) and the best average (44.3%), slightly outperforming the default k=10 (44.1%). The pattern is similar across all models. Qwen-2.5-7B shows a mild drop on AIME when k is increased, but this is within the noise range of 30-problem datasets.

Table 3: Effect of top-k in the confidence reward on MV accuracy (%) across four models. Avg_5 is the mean over all five benchmarks.

Model	k	**GSM8K**	**MATH-500**	**ASDiv**	**AIME 24**	**AIME 25**	**Avg_5**
Llama-3.2-3B	10	76.6	48.0	85.8	3.3	6.7	44.1
	1000	76.5	48.2	85.0	3.3	3.3	43.3
	10000	77.9	48.6	85.0	3.3	6.7	44.3
Llama-3.1-8B	10	84.8	54.2	85.2	6.7	0.0	46.2
	1000	84.2	55.0	84.6	6.7	0.0	46.1
	10000	84.4	53.4	83.6	6.7	0.0	45.6
Qwen-2.5-7B	10	89.1	75.4	91.3	16.7	16.7	57.8
	1000	88.6	75.8	91.5	13.3	10.0	55.9
	10000	88.5	76.2	91.4	13.3	10.0	55.9
Qwen-2.5-14B	10	93.3	81.0	92.0	16.7	16.7	59.9
	1000	92.7	79.2	91.9	20.0	3.3	57.4
	10000	92.6	81.0	92.1	20.0	13.3	59.8