SUTA: TOWARDS ONLINE ADAPTATION FOR SELF-IMPROVING LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deployed language models are typically static: experience accumulated while processing one input does not improve their behavior on later inputs. Test-time adaptation offers a practical form of online self-improvement by allowing a model to update persistent parameters from unlabeled deployment data. For autoregressive LLMs, however, treating every token in a self-generated trajectory as an equally reliable update signal can reinforce confirmation bias and cause reasoning collapse. We introduce **S**elective **U**ncertainty-aware **T**est-time **A**daptation (SUTA), a lightweight parametric self-improvement mechanism for autoregressive LLMs. SUTA uses predictive entropy to identify localized uncertainty bottlenecks in its own generations and restricts cumulative adapter updates to these high-entropy positions. An information-gain contrastive objective encourages the updated model to reduce uncertainty through the input context rather than context-free shortcuts, while a frozen-prior preservation term constrains drift during online updates. Across mathematical reasoning, domain knowledge, and coding benchmarks, SUTA consistently outperforms representative test-time optimization baselines on pre-trained and instruction-tuned models. Its gains grow with model scale and remain complementary to inference-time sampling and voting. These results show that selectively resolving uncertainty in self-generated trajectories is an effective and stable route to test-time self-improvement under distribution shift. Code is available at `https://anonymous.4open.science/r/SUTA-B150`.

1 INTRODUCTION

Large language models (LLMs) have achieved strong performance in mathematical reasoning, coding, and scientific question answering (Brown et al., 2020; Achiam et al., 2023; Wei et al., 2022; Guo et al., 2025). Yet, after deployment, an LLM is typically static: signals produced while processing one input are not retained in the model that processes later inputs. This limitation becomes especially salient under distribution shift, evolving instructions, and long reasoning trajectories, where a model may repeatedly encounter uncertainty patterns that are poorly covered by its pre-training distribution (Liang et al., 2022; Ouyang et al., 2022). Test-time adaptation (TTA) offers a practical mechanism for addressing this gap. When its parameter updates are retained across a sequence of deployment inputs, TTA provides a form of online self-improvement: signals from earlier inference episodes affect the model used for later inference (Sun et al., 2020).

For autoregressive LLMs, however, online self-improvement raises a fundamental feedback-quality question: **which parts of a model's own generation should contribute to a persistent update?** Existing LLM adaptation methods use pseudo-labeling, perplexity minimization, or entropy-based objectives (Huang et al., 2023; Hu et al., 2025; Kang et al., 2026; Wang et al., 2020; Ma et al., 2026), while a prevailing trajectory-level formulation applies its optimization signal indiscriminately across generated tokens. This treatment implicitly regards all positions in a trajectory as equally useful self-supervision. Autoregressive reasoning does not have this property. Some tokens are largely deterministic syntactic transitions or repeated templates, whereas others mark uncertain knowledge boundaries, strategic choices, and logical transitions. Our analysis in Fig. 1 shows that indiscriminately reinforcing a trajectory can damage the very behavior that an online update is intended to improve: strict low-entropy optimization truncates reasoning, while broad low-entropy or unconstrained optimization can amplify repetitive and degenerate text. Thus, the challenge is not

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.

merely to update a model at test time, but to extract useful internal feedback from its own trajectories without reinforcing confirmation bias or collapsing later reasoning.

We address this problem with **S**elective **U**ncertainty-aware **T**est-time **A**daptation (SUTA), a lightweight mechanism for online self-improvement in LLMs. At adaptation step t, the system consists of a frozen reference model and a persistent LoRA adapter. Given an incoming batch B_t, it first produces autoregressive responses, then identifies token positions with high predictive entropy, and computes the adapter gradient only from these localized uncertainty bottlenecks. The resulting adapter update is retained for later inputs in the adaptation stream. In this sense, SUTA instantiates the following constrained improvement loop: $\Delta\theta_{t+1} = U_{\mathrm{SUTA}}(\Delta\theta_t; B_t, Y_t, F_t)$, where $\Delta\theta_t$ denotes the current adapter, Y_t is the model's self-generated trajectory, F_t comprises internal signals extracted from that trajectory, and U_{SUTA} is a fixed update rule. Crucially, SUTA does not treat predictive entropy as a correctness label. Instead, it uses high-entropy positions as a selective locus for parameter updates, based on the hypothesis that these positions expose localized uncertainty that is more useful for adaptation than confident, repetitive continuations.

Selective updates alone are insufficient: a model can reduce uncertainty by becoming insensitive to the input or by drifting away from useful pre-trained behavior. SUTA therefore introduces two constraints on its persistent self-updates. First, an *Information-Gain Contrastive Objective* compares the model's uncertainty under the original prompt with that under a semantically neutralized prompt, encouraging uncertainty reduction that remains grounded in the input context. Second, a *Prior-Preservation Constraint* anchors the adapted predictive distribution to a frozen reference model, constraining destructive drift during cumulative online adaptation. Together with selective high-entropy optimization, these objectives are designed to constrain cumulative self-updates to remain input-dependent and close to the frozen prior, mitigating destructive drift during online adaptation.

We evaluate SUTA across competition mathematics, formal quantitative reasoning, domain knowledge, and coding benchmarks. Based on pre-trained or instruction-tuned LLMs, SUTA consistently outperforms representative test-time optimization baselines. Its improvements extend across model families and scales and incur modest additional per-batch overhead. Notably, we observe that selective uncertainty-aware adaptation becomes increasingly effective as model scale grows. It also enables adapted pre-trained LLMs to surpass their instruction-tuned counterparts in several cases. Furthermore, SUTA remains complementary to inference-time scaling strategies like majority voting and pass@k sampling, yielding additional gains under larger test-time compute budgets.

Our contributions are summarized as follows:

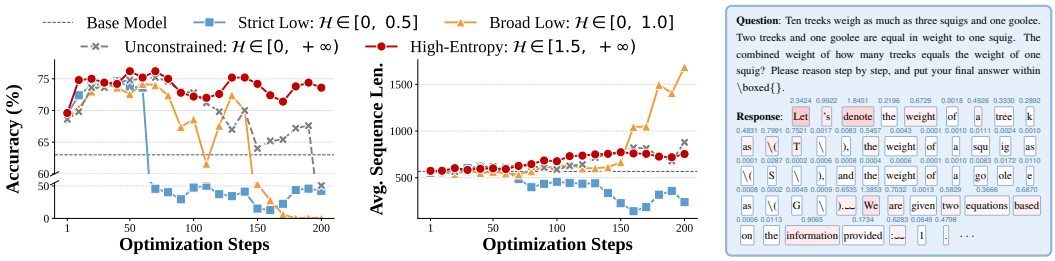
- We formulate online self-improvement in LLMs as an autoregressive test-time adaptation problem, in which a persistent adapter is cumulatively updated from self-generated trajectories without requiring external labels or rewards.
- We identify failure modes of uniform trajectory optimization: reinforcing low-entropy or indiscriminate token positions can lead to reasoning truncation, confirmation bias, and degenerate repetition. Our analysis shows that selecting high-entropy positions provides a more effective locus for constrained self-updates.
- We propose SUTA, which computes cumulative adapter updates from selected high-entropy positions and constrains them with input-grounded contrastive regularization and frozen-prior preservation.
- We conduct extensive experiments to demonstrate consistent improvements over existing baselines on both pre-trained and instruction-tuned LLMs, showing especially strong gains on larger-scale models.

2 Which Signals Drive Online Improvement?

Let M_θ denote an autoregressive large language model parameterized by θ. Given a prompt $x = [x_1, \ldots, x_M]$, it generates a response $y = [y_1, \ldots, y_T]$ according to

$$p_\theta(y|x) = \prod_{t=1}^{T} p_\theta(y_t|x, y_{<t})$$

where $y_{<t}$ denotes the preceding response tokens.



(a) Accuracy and Length by Entropy Range. (b) Token-Level Entropy.

Figure 1: **Token selection affects the stability of cumulative trajectory optimization.** **(a)** We optimize on the MATH-500 training set under four predictive-entropy-based token-selection settings, and evaluate accuracy and average generation length on the held-out test set across optimization steps. Strict Low ($\mathcal{H} \in [0, 0.5]$) shows a brief initial accuracy gain, followed by shorter generations and lower accuracy; Broad Low ($\mathcal{H} \in [0, 1.0]$) and Unconstrained ($\mathcal{H} \in [0, +\infty)$) produce large increases in generation length, while showing repetitive or degenerate continuations; Selective High-Entropy ($\mathcal{H} \in [1.5, +\infty]$) yields a more moderate increase in length and the highest accuracy among the four settings. **(b)** Token-level predictive entropy $\mathcal{H}$ on a prefix of one self-generated reasoning response. In this example, entropy is elevated at some transitions and variable-introduction positions but low on many template continuations.

In online test-time adaptation, unlabeled inputs arrive sequentially or in streaming mini-batches. For each incoming batch, the model first generates responses, then uses those trajectories as self-supervision to update a small set of parameters before processing later batches. These updates occur on the test distribution without ground-truth labels.

Existing approaches include self-training and entropy-based objectives (Huang et al., 2023; Hu et al., 2025; Kang et al., 2026; Ma et al., 2026). A common trajectory-level self-training objective minimizes the negative log-likelihood of the generated response:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(y_t|x, y_{<t}).$$

We examine how the choice of token positions affects performance during cumulative adaptation. Predictable continuations often have low entropy, whereas high entropy indicates uncertainty in the next-token distribution (Wang et al., 2026). This motivates the intuitive hypothesis that optimizing low-entropy positions could reinforce predictable parts of a reasoning trajectory without introducing excessive bias. For the predictive distribution at position t, we compute Shannon entropy as

$$\mathcal{H}_\theta(y_t) = -\sum_{v \in \mathcal{V}} p_\theta(v \mid x, y_{<t}) \log p_\theta(v \mid x, y_{<t})$$

where $\mathcal{V}$ is the vocabulary. We compare four partly overlapping token-selection settings: Strict Low ($\mathcal{H} \in [0, 0.5]$), Broad Low ($\mathcal{H} \in [0, 1.0]$), Unconstrained ($\mathcal{H} \in [0, +\infty)$), and Selective High-Entropy ($\mathcal{H} \in [1.5, +\infty)$). Fig. 1 (b) illustrates how predictive entropy varies across positions in one generated response. In Fig. 1 (a), we optimize on the MATH-500 training set and track accuracy and average generation length on the held-out test set across optimization steps. The comparison reveals two failure modes alongside a more stable high-entropy setting.

Strict Low-Entropy Optimization Eventually Shortens Responses and Reduces Accuracy.

Optimizing strictly low-entropy positions yields a short-lived increase in accuracy, followed by a marked decline in the accuracy panel of Fig. 1 (a). The generation-length panel shows a concurrent drop in response length. This pattern is consistent with truncated step-by-step reasoning, as illustrated by the qualitative example in Fig. 10.

Token Selection Shapes Whether Longer Responses Reflect Exploration or Repetition.

Broad Low and Unconstrained optimization instead produce large increases in generation length. The qualitative examples in Fig. 11 and Fig. 12 show repetitive or degenerate continuations under these settings. In contrast, Selective High-Entropy optimization produces a more moderate increase

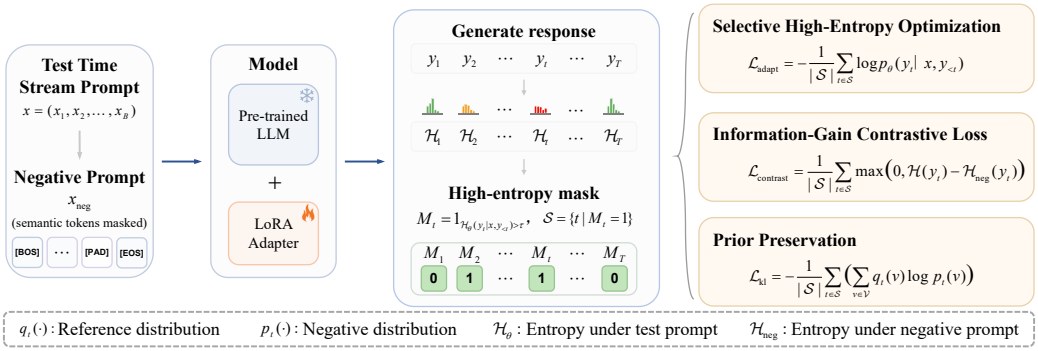


Figure 2: **Overall pipeline of SUTA.** For each incoming test batch, the model generates autoregressive responses and selects high-entropy reasoning tokens for adaptation. The objective combines (1) selective high-entropy optimization, (2) an information-gain contrastive term that grounds uncertainty reduction in the input context, and (3) a prior-preservation constraint that anchors predictions to the frozen pre-trained model.

in length and the highest accuracy among the four settings in Fig. 1 (a). This comparison motivates selecting uncertain positions for adaptation, rather than updating every position in a generated trajectory. Fig. 13 further illustrates step-by-step verification in a response generated by SUTA.

3 SUTA: Online Self-Improvement via Selective Adaptation

Online Self-Improving LLMs. It uses feedback from a system's own behavior or its environment to modify a persistent component that affects later behavior. This component may be model parameters, memory, or code and workflow; for example, SEAL studies self-edits to language-model parameters, ReasoningBank stores reusable reasoning memories, and the Darwin Gödel Machine evolves agent code and tools (Zweiger et al., 2026; Ouyang et al., 2026; Zhang et al., 2026).

We study the online improvement for autoregressive LLMs by formulating the problem as test-time adaptation. A lightweight adapter is updated from self-generated trajectories without task labels and retained across subsequent inputs. This setting raises a feedback-selection problem: trajectories mix localized uncertainty with deterministic or input-insensitive behavior, so writing every position into the persistent adapter can accumulate uninformative updates. SUTA addresses this problem by computing gradients over high-entropy positions, using an information-gain contrastive objective to preserve prompt dependence, and applying prior preservation to limit drift from the frozen reference distribution. From another perspective, SUTA selectively converts internal feedback into persistent adapter updates while constraining cumulative drift.

3.1 Uncertainty-Aware Selective Adaptation

Our preliminary experiments in Sec. 2 demonstrate that low-entropy optimization can lead to severe performance degradation by disturbing the model's decoding paths. We therefore avoid reinforcing these low-entropy (high-confidence) tokens and restrict adaptation to high-entropy positions, which typically correspond to localized uncertainty gaps or critical logical transitions.

For a generated response $y = [y_1, \ldots, y_T]$ to prompt $x = [x_1, \ldots, x_M]$, we construct a binary mask $M_t \in \{0, 1\}$ to isolate high-entropy tokens. A token is selected when the entropy of the current model's next-token distribution exceeds the threshold τ:

$$M_t = \mathbf{1}_{\mathcal{H}_\theta(y_t|x,y_{<t})>\tau} \tag{1}$$

Let $\mathcal{S} = \{t \in [T] \mid M_t = 1\}$ denote the set of selected token positions. SUTA then minimizes the negative log-likelihood of its self-generated tokens only at these positions:

$$\mathcal{L}_{\text{adapt}} = -\frac{1}{|\mathcal{S}|}\sum_{t\in\mathcal{S}} \log p_\theta(y_t \mid x, y_{<t}) \tag{2}$$

By masking out confident predictions, SUTA focuses adaptation gradients on high-entropy tokens, where reducing predictive ambiguity can provide a more informative update signal while reducing the risk of confirmation bias from reinforcing the model's own low-entropy continuations.

3.2 Information-Gain Contrastive Objective

A model can become more confident at selected positions without making better use of the input. To encourage uncertainty reduction that remains tied to the prompt, we introduce an *Information-Gain Contrastive Loss*. It compares the same response prefix under two contexts. The original prompt x contains the task information. The negative prompt x_{neg} is neutralized by replacing the semantic tokens of x with neutral tokens (such as [PAD]) while preserving special formatting tokens (such as [BOS]). We evaluate the negative-prompt entropy with the current model and the same prefix $y_{<t}$, without tracking gradients through this branch.

Intuitively, a model that uses the prompt effectively should be meaningfully less uncertain about the next token under informative x than under neutralized x_{neg}, given the same response prefix. Let $\mathcal{H}(y_t) = \mathcal{H}_\theta(y_t \mid x, y_{<t})$ and $\mathcal{H}_{\text{neg}}(y_t) = \mathcal{H}_{\text{neg}}(y_t \mid x_{\text{neg}}, y_{<t})$ denote predictive entropy under the original and negative prompts. On the selected positions, the information-gain contrastive loss is

$$\mathcal{L}_{\text{contrast}} = \frac{1}{|\mathcal{S}|} \sum_{t \in \mathcal{S}} \max\big(0, \mathcal{H}(y_t) - \mathcal{H}_{\text{neg}}(y_t)\big) \tag{3}$$

The detached negative-prompt entropy serves as a *non-differentiable baseline* that varies dynamically. The hinge contributes a penalty only when the informative prompt x leaves the model more uncertain than the neutralized prompt x_{neg}. It thus encourages the expected ordering of the two entropies and promotes input-dependent uncertainty reduction, while retaining the high-entropy selection rule from Sec. 3.1. The objective does not impose a fixed entropy margin.

3.3 Prior-Preserved Online Adaptation

Prompt-dependent updates can still accumulate drift when an adapter is retained across the test stream. Local self-generated trajectories may pull the model away from broader pre-trained behavior, creating risks of catastrophic forgetting or representation collapse.

To constrain this cumulative effect, SUTA keeps a frozen copy of the original pre-trained model, θ_{ref}. At each selected position $t \in S$, let $p_t(\cdot) = p_\theta(\cdot \mid x, y_{<t})$ be the adapted model's predictive distribution and $q_t(\cdot) = p_{\text{ref}}(\cdot \mid x, y_{<t})$ the reference distribution under the same prompt and response prefix. We use the reference-to-current cross-entropy as the prior-preservation loss:

$$\mathcal{L}_{\text{kl}} = -\frac{1}{|\mathcal{S}|} \sum_{t \in \mathcal{S}} \Big(\sum_{v \in \mathcal{V}} q_t(v) \log p_t(v) \Big) \tag{4}$$

At each selected position, $\mathcal{L}_{\text{adapt}}$ favors the sampled token, whereas $\mathcal{L}_{\text{kl}}$ penalizes departures from the frozen model's full next-token distribution. Together, they let the adapter learn from local ambiguity while reducing the risk that repeated self-generated optimization distorts the model's pre-trained continuation behavior.

Each loss is evaluated at the selected positions S: $\mathcal{L}_{\text{adapt}}$ uses the self-generated tokens, $\mathcal{L}_{\text{contrast}}$ encourages prompt-dependent uncertainty reduction, and $\mathcal{L}_{\text{kl}}$ regularizes predictions against the frozen reference. To update the adapter for an unlabeled test sample, we minimize the combined objective:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{adapt}} + \lambda \mathcal{L}_{\text{contrast}} + \mathcal{L}_{\text{kl}} \tag{5}$$

where λ is a hyperparameter controlling the strength of the contrastive context-grounding term.

Note: Both $\mathcal{L}_{\text{adapt}}$ and $\mathcal{L}_{\text{kl}}$ are cross-entropies averaged over the same selected positions S: the former uses the generated token as a hard target, and the latter uses the frozen model's distribution as a soft target. With this shared per-token normalization, we assign the two terms equal unit weights and use λ as the sole loss-weighting hyperparameter to balance them against the $\mathcal{L}_{\text{contrast}}$.

4 Experiments

Models. We validate the broad applicability and scalability of our SUTA across model scales and architectures using representative open-weight LLMs. We primarily adopt the Qwen family, includ-

Table 1: Main evaluation results across multiple domains, including competition mathematics, formal quantitative reasoning, domain knowledge, and coding task benchmarks. Best and second-best results per sub-group are highlighted with Best and Second Best.

Method	COMPETITION		QUANTITATIVE		DOMAIN KNOW.		CODE		AVG.
	AIME24	AMC23	MATH500	Minerva	Olym.	GPQA	MBPP+	LCB v5	
Vanilla Base Models									
Qwen3-4B-Base	$10.0_{\pm 0.00}$	$37.5_{\pm 0.00}$	$63.0_{\pm 0.00}$	$27.9_{\pm 0.00}$	$33.7_{\pm 0.00}$	$22.1_{\pm 0.00}$	$65.9_{\pm 0.00}$	$13.5_{\pm 0.00}$	$34.2_{\pm 0.00}$
TLM	$13.3_{\pm 0.00}$	$47.5_{\pm 0.00}$	$53.0_{\pm 0.00}$	$15.1_{\pm 0.00}$	$25.9_{\pm 0.00}$	$1.86_{\pm 1.49}$	$38.2_{\pm 0.00}$	$13.8_{\pm 0.00}$	$26.1_{\pm 0.19}$
TTSV	$3.33_{\pm 0.00}$	$30.0_{\pm 8.66}$	$67.6_{\pm 0.69}$	$28.8_{\pm 0.85}$	$31.9_{\pm 1.97}$	$12.4_{\pm 1.42}$	$58.6_{\pm 0.12}$	$13.2_{\pm 0.07}$	$30.7_{\pm 1.19}$
SUTA	$13.3_{\pm 0.00}$	$50.0_{\pm 0.00}$	$75.4_{\pm 0.35}$	$31.7_{\pm 0.56}$	$37.0_{\pm 0.12}$	$24.3_{\pm 0.26}$	$67.3_{\pm 0.17}$	$14.5_{\pm 0.07}$	$39.2_{\pm 0.02}$
Qwen3-8B-Base	$6.67_{\pm 0.00}$	$52.5_{\pm 0.00}$	$74.4_{\pm 0.00}$	$33.1_{\pm 0.00}$	$36.7_{\pm 0.00}$	$24.6_{\pm 0.00}$	$69.6_{\pm 0.00}$	$15.5_{\pm 0.00}$	$39.1_{\pm 0.00}$
TLM	$10.0_{\pm 0.00}$	$55.0_{\pm 0.00}$	$74.6_{\pm 0.00}$	$35.3_{\pm 0.00}$	$36.9_{\pm 0.00}$	$22.5_{\pm 0.46}$	$44.2_{\pm 0.00}$	$15.5_{\pm 0.00}$	$36.7_{\pm 0.06}$
TTSV	$10.0_{\pm 0.00}$	$55.0_{\pm 4.33}$	$73.9_{\pm 2.89}$	$31.4_{\pm 5.52}$	$35.6_{\pm 2.57}$	$19.7_{\pm 0.46}$	$68.3_{\pm 0.00}$	$14.7_{\pm 0.06}$	$38.6_{\pm 0.44}$
SUTA	$10.0_{\pm 0.00}$	$59.2_{\pm 2.89}$	$78.7_{\pm 0.31}$	$44.6_{\pm 0.21}$	$38.7_{\pm 0.34}$	$36.2_{\pm 0.77}$	$69.6_{\pm 0.00}$	$15.6_{\pm 0.07}$	$44.1_{\pm 0.51}$
Instruct Models									
Qwen3-4B-Instruct	$16.7_{\pm 0.00}$	$50.0_{\pm 0.00}$	$77.8_{\pm 0.00}$	$37.5_{\pm 0.00}$	$41.9_{\pm 0.00}$	$30.6_{\pm 0.00}$	$57.1_{\pm 0.00}$	$23.8_{\pm 0.00}$	$41.9_{\pm 0.00}$
TLM	$16.7_{\pm 0.00}$	$52.5_{\pm 0.00}$	$77.0_{\pm 0.00}$	$37.9_{\pm 0.00}$	$43.1_{\pm 0.00}$	$35.4_{\pm 0.25}$	$55.8_{\pm 0.00}$	$23.2_{\pm 0.00}$	$42.7_{\pm 0.03}$
TTSV	$10.0_{\pm 0.00}$	$40.0_{\pm 0.00}$	$67.0_{\pm 2.77}$	$32.4_{\pm 7.01}$	$29.9_{\pm 2.72}$	$35.1_{\pm 0.34}$	$54.5_{\pm 0.00}$	$20.6_{\pm 0.06}$	$36.2_{\pm 0.91}$
SUTA	$22.2_{\pm 1.92}$	$62.5_{\pm 0.00}$	$80.5_{\pm 0.58}$	$39.8_{\pm 0.56}$	$43.6_{\pm 0.14}$	$35.6_{\pm 0.90}$	$57.7_{\pm 0.00}$	$24.7_{\pm 0.65}$	$45.8_{\pm 0.53}$
Qwen3-8B-Instruct	$20.0_{\pm 0.00}$	$55.0_{\pm 0.00}$	$78.0_{\pm 0.00}$	$36.8_{\pm 0.00}$	$42.8_{\pm 0.00}$	$36.6_{\pm 0.00}$	$65.3_{\pm 0.00}$	$24.2_{\pm 0.00}$	$44.8_{\pm 0.00}$
TLM	$20.0_{\pm 0.00}$	$60.0_{\pm 0.00}$	$78.5_{\pm 1.27}$	$35.9_{\pm 0.21}$	$43.3_{\pm 0.43}$	$39.0_{\pm 1.03}$	$65.6_{\pm 0.00}$	$22.6_{\pm 0.00}$	$45.6_{\pm 0.27}$
TTSV	$13.3_{\pm 0.00}$	$57.5_{\pm 0.00}$	$74.1_{\pm 2.19}$	$37.1_{\pm 1.27}$	$42.9_{\pm 1.99}$	$38.8_{\pm 0.34}$	$53.3_{\pm 0.12}$	$21.9_{\pm 0.19}$	$42.4_{\pm 0.31}$
SUTA	$20.0_{\pm 0.00}$	$60.0_{\pm 0.00}$	$80.0_{\pm 0.80}$	$37.4_{\pm 1.29}$	$43.7_{\pm 0.23}$	$39.8_{\pm 0.64}$	$66.3_{\pm 0.17}$	$24.8_{\pm 0.39}$	$46.5_{\pm 0.36}$

ing Qwen3-1.7B, Qwen3-4B, Qwen3-8B, Qwen3-14B (Yang et al., 2025) and Qwen3.5-4B (Qwen Team, 2026). We also incorporate Llama-3.1-8B (Grattafiori et al., 2024) to verify generalization to distinct pre-training paradigms and tokenizer designs.

Benchmarks. We conduct experiments across four primary task categories. 1) Competition Mathematics: AMC23 (Li et al., 2024) and AIME24 (Li et al., 2024); 2) Formal Quantitative Reasoning: MATH500 (Hendrycks et al., 2021) and Minerva Math (Lewkowycz et al., 2022); 3) Domain Knowledge: Olympiad Bench (He et al., 2024) and GPQA Diamond (Rein et al., 2023); 4) Coding: LiveCodeBench v5 (Jain et al., 2025) and MBPP+ (Liu et al., 2023).

Implementations. To ensure high reproducibility, we employ greedy decoding across all tasks, except for the pass@k evaluations. For standard generation (non-thinking mode), the maximum sequence length is set to 2,048 tokens, whereas for reasoning-intensive thinking modes, this limit is extended to 16,384 tokens. During pass@k evaluations, we utilize nucleus sampling with a temperature of 0.6 and a top-p threshold of 0.95.

For optimizer, we adopt AdamW with a learning rate of 5×10^{-5} and a batch size of 8. All experiments are run across 3 independent random seeds to account for variance. Unless otherwise specified, the default hyperparameters of our proposed framework are set to $\tau = 1.5$ and $\lambda = 0.5$. **Note**: We further examine the sensitivity of the entropy threshold τ and contrastive weight λ in Sec. 4.2 and Fig. 4. Ablation studies of τ and λ across multiple models, including Qwen3-4B-Base, Qwen3.5-4B-Base, and Qwen3-8B-Base, show that $\tau = 1.5$ and $\lambda = 0.5$ are the best-performing settings, suggesting some robustness across model architectures and sizes. This combination provides a good starting point for hyperparameter tuning on other models.

Baselines. We compare SUTA against several representative test-time online adaptation baselines: i) Vanilla: Direct inference without any adaptation or training. ii) TLM(Hu et al., 2025): A test-time learning baseline that adapts model parameters by minimizing the input perplexity of unlabeled test data. iii) TTSV(Kang et al., 2026): A non-invasive framework that minimizes output entropy to tune input-prepended steering vectors under a frozen LLM. In addition, we further evaluate the compatibility of SUTA with inference-time scaling strategies, including majority voting (maj@k) and pass@k sampling. Details are in the Appendix B.

4.1 Main Results

Overall Performance Comparison. Across all benchmarks in Table 1, SUTA consistently outperforms existing baselines on both pre-trained and instruction-tuned LLMs, with especially large gains on reasoning-intensive tasks such as MATH500, AMC23, and Minerva. These results validate the

Table 2: Generalization results across different model families and pre-training paradigms.

Method	Comp.	Quant.	Domain	Avg.
Qwen3.5-4B-Base	29.6	56.4	33.8	39.9
TLM	30.8	57.1	35.6	41.2
TTSV	30.0	57.4	31.7	39.7
SUTA	33.8	60.6	37.5	44.0
Qwen3.5-4B-Instruct	38.8	64.7	41.2	48.2
TLM	38.8	64.7	41.0	48.2
TTSV	37.5	64.3	40.1	47.3
SUTA	42.9	65.9	42.1	50.3
Llama3.1-8B-Instruct	15.8	39.3	17.4	24.2
TLM	12.5	38.3	14.1	21.6
TTSV	3.80	16.7	2.80	7.70
SUTA	18.8	41.8	17.6	26.0

Table 3: Ablation study of the proposed objectives on MATH-500.

$\mathcal{L}_{adapt}$	$\mathcal{L}_{kl}$	$\mathcal{L}_{contrast}$	Acc. (%)	Avg. Len.
			63.0	567.8
✓			73.0	583.6
✓	✓		73.6	585.0
✓		✓	74.0	643.8
✓	✓	✓	75.2	593.0

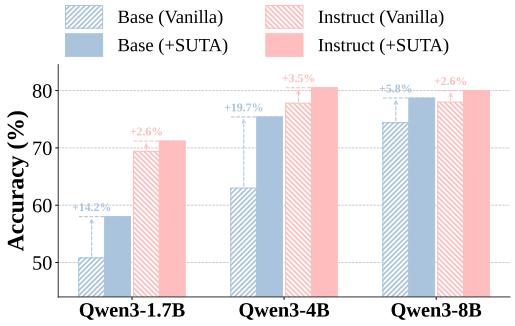


Figure 3: Scaling behavior of SUTA on MATH-500 benchmark across different model sizes.

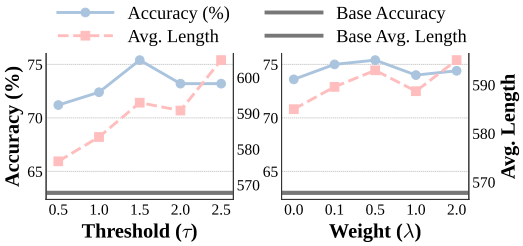


Figure 4: Hyperparameter sensitivity analysis on MATH-500 for Qwen3-4B-Base.

effectiveness of selectively adapting high-uncertainty reasoning regions instead of uniformly optimizing the entire generation trajectory. The improvements on coding benchmarks are comparatively modest, because code generation follows more rigid formatting and stronger logical constraints, leaving fewer high-uncertainty tokens for selective adaptation. Compared with prior methods, SUTA also exhibits substantially stronger stability, avoiding the severe degradation frequently observed on knowledge and coding benchmarks.

Generalization Across Architectures. To evaluate robustness across architectures and pre-training paradigms, we further test SUTA on the Qwen3.5 and Llama3.1 series (Table 2). SUTA consistently improves performance across all settings, e.g., improving Qwen3.5-4B-Base from 39.9% to 44.0% and Llama3.1-8B-Instruct from 24.2% to 26.0% average accuracy.

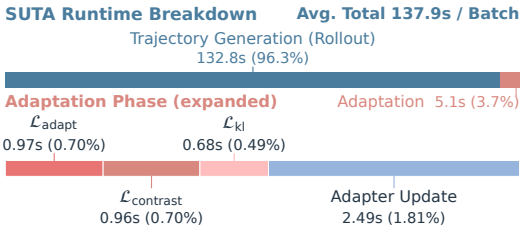
Scaling Behavior. Fig 3 shows the scaling behavior of SUTA across 1.7B, 4B, and 8B models (See Appendix C.4 for 14B models). SUTA consistently improves Base models at all scales, with gains becoming more pronounced as model size increases. Notably, adapted Base models can sometimes approach the performance of their instruction-tuned counterparts, suggesting that SUTA can help unlock the latent reasoning capacity of larger pretrained models during inference.

4.2 Ablation Studies

Effects of Online Adaptation Objectives. Table 3 compares combinations of SUTA's objectives on MATH-500. Selective High-Entropy Optimization ($\mathcal{L}_{\text{adapt}}$) alone raises accuracy from 63.0% to 73.0%. It uses self-generated tokens as targets only at high-entropy positions, supporting the value of selecting localized uncertainty as feedback for persistent adapter updates. Adding the Prior-Preservation Constraint ($\mathcal{L}_{\text{kl}}$) raises accuracy to 73.6% while leaving average response length nearly unchanged (583.6 versus 585.0 tokens). This term encourages the adapted model's predictions to stay close to a frozen reference distribution at selected positions and is intended to limit drift during cumulative updates. Adding the Information-Gain Contrastive Loss ($\mathcal{L}_{\text{contrast}}$) to $\mathcal{L}_{\text{adapt}}$ instead raises accuracy to 74.0%, but average length also rises to 643.8 tokens. By comparing predictive entropy under the original and neutralized negative prompts, this term encourages prompt-dependent uncertainty reduction. With all three objectives, accuracy reaches the highest value in this ablation, 75.2%, while average length falls to 593.0 tokens from the 643.8 observed without $\mathcal{L}_{\text{kl}}$. These

Table 4: (**Left**) Average runtime overhead per batch (seconds). (**Right**) SUTA runtime breakdown for Qwen3-4B-Base.

Models	Vanilla	TLM	TTSV	SUTA
Qwen3-4B-Base	120.0	140.5	184.1	137.9
Qwen3-4B-Instruct	157.9	181.7	164.5	175.7



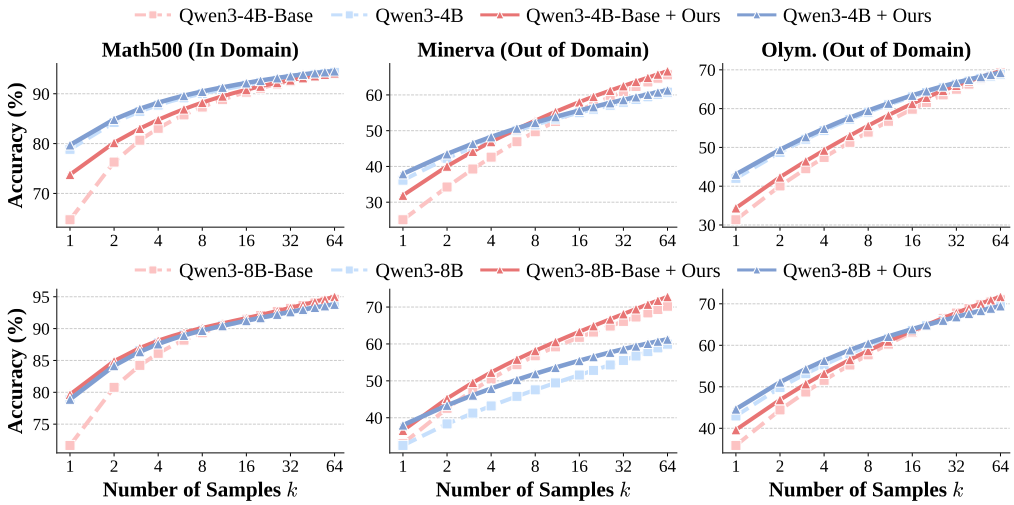


Figure 5: Pass@k scaling performance under different sampling budgets.

results support SUTA's design of selecting where self-generated feedback enters the update and using prompt contrast and the frozen prior to constrain persistent adaptation.

Hyperparameter Sensitivity. Fig 4 (left) examines the entropy threshold τ. A low τ admits many predictable tokens into the adaptation set, which can reinforce shallow continuation patterns and oversimplify reasoning, consistent with the low-entropy behavior observed in Sec. 2. Raising τ beyond 1.5 leaves fewer uncertain positions available for updates; accuracy falls at $\tau \geq 2.0$, while the longer trajectories at $\tau = 2.5$ bring no corresponding accuracy gain, suggesting less directed exploration when update signals become sparse. In this setting, $\tau = 1.5$ achieves the highest accuracy by retaining uncertainty bottlenecks that can support reasoning exploration and self-correction. Fig 4 (right) shows the effect of the contrastive weight λ. A moderate λ encourages the model to reduce uncertainty using the informative prompt while preserving flexibility. Increasing λ to 1.0 or 2.0 lowers accuracy relative to $\lambda = 0.5$, consistent with excessive context-grounding pressure constraining useful exploration.

4.3 Per-Batch Runtime Overhead

Table 4 reports the average total time per batch. Vanilla measures inference-only latency for autoregressively generating the batch responses, without an adaptation update. For each incoming batch, SUTA first generates responses (rollout trajectories) using the model state carried over from the preceding batch, then evaluates its losses on these fixed trajectories. The required distributions can be obtained with three full-sequence forward passes: one under the current model and original prompt, shared by all three loss terms; one gradient-free pass under the neutralized negative prompt; and one gradient-free pass through the frozen reference model. The current-model branch is then backpropagated to update the LoRA adapter. Unlike autoregressive rollout, these passes evaluate the known trajectories in parallel across token positions. For Qwen3-4B-Base, trajectory generation takes 132.8s per batch and the entire adaptation phase takes 5.1s, including 2.6s for the calculation of three losses (Table 4 right). Consequently, trajectory generation dominates SUTA's 137.9s total per batch, and the added adaptation cost remains modest relative to long-chain decoding.

4.4 Reasoning-Oriented Thinking Mode

Table 5: Majority voting (maj@k) results under different inference-time sampling budgets k.

Method	MATH-500				Minerva				Olym.			
	8	16	32	64	8	16	32	64	8	16	32	64
Qwen3-4B-Base	76.7	78.4	79.2	79.6	29.8	31.2	31.7	31.9	38.8	40.3	41.1	41.7
SUTA	78.5	79.3	79.7	80.4	36.9	38.1	38.8	39.2	39.7	40.9	41.5	41.7
Qwen3-4B-Instruct	82.7	83.2	83.5	83.8	39.6	40.8	41.5	42.0	45.2	45.5	45.6	45.5
SUTA	83.5	84.6	85.1	85.5	42.5	43.3	44.0	44.4	49.4	50.5	51.2	51.5
Qwen3-8B-Base	80.6	81.7	82.4	82.7	40.0	41.6	42.5	42.7	43.0	44.3	44.8	44.8
SUTA	83.3	83.8	83.9	84.0	40.5	41.5	42.4	43.0	43.9	44.9	45.4	45.7
Qwen3-8B-Instruct	82.8	83.4	83.6	83.7	35.2	35.8	36.0	36.0	46.6	47.1	47.4	47.5
SUTA	83.3	84.0	84.4	84.6	40.2	40.6	40.7	40.7	47.9	48.5	48.8	48.9

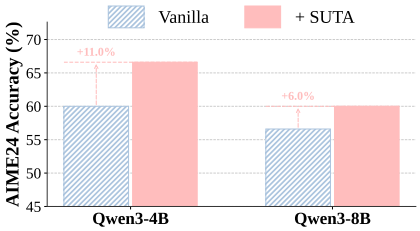


Figure 6: Performance under thinking mode on AIME24.

Recent reasoning-centric LLMs increasingly rely on extended "thinking" trajectories to improve multi-step problem solving. Under thinking mode, autoregressive generations naturally contain a larger proportion of high-entropy reasoning states, including intermediate verification and self-correction. To evaluate whether SUTA remains effective in such long-chain reasoning settings, we further conduct experiments using the native thinking mode of Qwen3 models. As shown in Fig 6, SUTA consistently improves reasoning accuracy under thinking mode for both Qwen3-4B and Qwen3-8B. These results suggest that selectively optimizing uncertain reasoning regions effectively guides the model toward more productive reasoning exploration while preserving stable long-chain reasoning behavior.

4.5 Compatibility with Inference-Time Scaling

SUTA preserves the multi-path exploration space. As shown in Fig 5, under pass@k evaluation, SUTA consistently maintains or improves reasoning accuracy across different sampling budgets k, indicating that the proposed adaptation strategy does not degrade the model's underlying reasoning capacity. Detailed experimental settings are provided in Appendix B.4.

SUTA improves the consistency of generating correct reasoning. Inference-time scaling methods such as majority voting (maj@k) further improve reasoning performance by allocating additional sampling compute during decoding. To evaluate the compatibility between SUTA and test-time scaling strategies, Table 5 reports the maj@k results under progressively increasing sampling budgets. Across Base and Instruct models, SUTA consistently outperforms Vanilla decoding across nearly all sampling scales and benchmarks, demonstrating that SUTA remains effective under larger inference-time search budgets. The improvements remain stable as the sampling budget increases, suggesting that selective adaptation may not only improve single-trajectory accuracy, but also encourage more consistent convergence toward correct reasoning paths across diverse sampled trajectories.

5 Conclusion

This work asks which self-generated signals should drive persistent online self-improvement in autoregressive LLMs. Strict low-entropy optimization can truncate reasoning, while broader low-entropy or unconstrained optimization can amplify repetition. SUTA uses predictive entropy to identify high-entropy positions as localized uncertainty bottlenecks for persistent adapter updates. An information-gain contrastive objective encourages input-grounded uncertainty reduction; a prior-preservation constraint limits drift from the frozen reference model.

Across evaluated mathematical reasoning, domain knowledge, and coding benchmarks, SUTA improves group-average accuracy over representative online test-time optimization baselines on pre-trained and instruction-tuned models. Ablations support selective updates and both constraints, with benefits across tested model families and sizes. SUTA remains broadly complementary to evaluated inference-time sampling and voting. In reported analysis, SUTA's online adaptation accounts for only 3.7% of the total time per batch. These findings suggest that selecting and constraining self-generated feedback can support online self-improvement without external labels or rewards.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing and editing the manuscript to improve readability. We have not used generative AI tools for conducting experiments, analyzing results and drawing conclusions, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for language polishing, grammar checking, and editing the manuscript to improve readability. We have reviewed all AI-assisted work. All LLM-edited text was carefully checked by the authors to ensure technical accuracy, and no AI tools were used in the design, execution, or analysis of experiments. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We release an anonymous implementation of SUTA at `https://anonymous.4open.science/r/SUTA-B150`. Algorithm 1 in Appendix B gives the complete procedure. Section 4 and Appendix B document all optimization settings (AdamW, learning rate 5×10^{-5}, batch size 8, LoRA rank $r = 8$ and target modules), the configuration of each baseline, and the hardware used. All benchmarks are publicly available; Appendix B lists their sizes and preprocessing, and Appendix B.4 describes the construction of the adaptation stream, which is drawn from the MATH training split so that it never overlaps any evaluation set, together with the sampling configuration ($T = 0.6$, top-$p = 0.95$) for the pass@k and maj@k experiments. Appendix C.5 provides token-level validation of the core hypothesis.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. The unreasonable effectiveness of entropy minimization in llm reasoning. *Advances in Neural Information Processing Systems*, 38:107150–107180, 2026.

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

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. *arXiv preprint arXiv:2412.09078*, 2024.

Yi Bin, Tianyi Jiang, Yujuan Ding, Kainian Zhu, Fei Ma, Jingkuan Song, Yang Yang, and Heng Tao Shen. Explore briefly, then decide: Mitigating llm overthinking via cumulative entropy regulation. *arXiv preprint arXiv:2510.02249*, 2025.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Guohao Chen, Shuaicheng Niu, Deyu Chen, Jiahao Yang, Zitian Zhang, Mingkui Tan, Pengcheng Wu, and Zhiqi Shen. Zerosiam: An efficient siamese for test-time entropy optimization without collapse. *arXiv preprint arXiv:2509.23183*, 2025.

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

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 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 54625–54640, 2024.

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.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Jinwu Hu, Zitian Zhang, Guohao Chen, Xutao Wen, Chao Shuai, Wei Luo, Bin Xiao, Yuanqing Li, and Mingkui Tan. Test-time learning for large language models. In *International Conference on Machine Learning*, pp. 24823–24849. PMLR, 2025.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 1051–1068, 2023.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International conference on learning representations*, volume 2024, pp. 32808–32824, 2024.

Jonas Hübotter, Sascha Bongni, Ido Hakimi, and Andreas Krause. Efficiently learning at test-time: Active fine-tuning of llms. In *International Conference on Learning Representations*, volume 2025, pp. 74978–75035, 2025.

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.

Xinyue Kang, Diwei Shi, and Li Chen. Model whisper: Steering vectors unlock large language models' potential in test-time. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 31392–31400, 2026.

Adilbek Karmanov, Dayan Guan, Shijian Lu, Abdulmotaleb El Saddik, and Eric Xing. Efficient test-time adaptation of vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14162–14171, 2024.

Vanshaj Khattar, Moumita Choudhury, Jing Liu, Toshiaki Koike-Akino, Ming Jin, Ye Wang, et al. Amplification effects in test-time reinforcement learning: Safety and reasoning vulnerabilities. *arXiv preprint arXiv:2603.15417*, 2026.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.

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.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Jia Liu, ChangYi He, YingQiao Lin, MingMin Yang, FeiYang Shen, and ShaoGuo Liu. Ettrl: Balancing exploration and exploitation in llm test-time reinforcement learning via entropy mechanism. *arXiv preprint arXiv:2508.11356*, 2025.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572, 2023.

Jing Ma, Hanlin Li, and Xiang Xiang. Ptta: Purifying malicious samples for test-time model adaptation. In *Forty-second International Conference on Machine Learning*, 2025.

Jing Ma, Hanlin Li, and Xiang Xiang. Decoupled entropy minimization. *Advances in Neural Information Processing Systems*, 38:161195–161230, 2026.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*, pp. 16888–16905. PMLR, 2022.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. *arXiv preprint arXiv:2302.12400*, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, volume 2026, pp. 94327–94354, 2026.

Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

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. *arXiv preprint arXiv:2311.12022*, 2023.

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.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025.

Woomin Song, Saket Dingliwal, Sai Muralidhar Jayanthi, Bhavana Ganesh, Jinwoo Shin, Aram Galstyan, and Sravan Babu Bodapati. Accelerated test-time scaling with model-free speculative sampling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 30611–30624, 2025.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pp. 9229–9248. PMLR, 2020.

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

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *Advances in Neural Information Processing Systems*, 38:115452–115486, 2026.

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

Yikun Wang, Rui Zheng, Liang Ding, Qi Zhang, Dahua Lin, and Dacheng Tao. Uncertainty aware learning for language model alignment. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11087–11099, 2024.

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

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 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.

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

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin gödel machine: open-ended evolution of self-improving agents. In *International Conference on Learning Representations*, volume 2026, pp. 104223–104294, 2026.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Xinwei Long, Ermo Hua, Biqing Qi, et al. Ttrl: Test-time reinforcement learning. *Advances in Neural Information Processing Systems*, 38:131459–131483, 2026.

Adam Zweiger, Jyo Pari, Han Guo, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. *Advances in Neural Information Processing Systems*, 38:74084–74115, 2026.

APPENDIX

A Related Work **14**

B Implementation Details **16**

B.1 Baselines 16

B.2 Computational Budget 16

B.3 Benchmarks 16

B.4 Sampling 17

C More Analysis **18**

C.1 Effect of LoRA Target Modules 18

C.2 Response Length 18

C.3 Comparison with Test-Time Reinforcement Learning and GRPO 18

C.4 Scalability to Larger Models 19

C.5 Token-Level Validation of the Core Hypothesis 19

C.6 Error Analysis of Selective Adaptation 20

C.7 Additional Ablations 20

C.8 Robustness and Efficiency 21

D Qualitative Examples **21**

D.1 Visualization 21

D.2 Qualitative Analysis of Entropy Intervals and Failure Modes 22

E Limitations and Future Directions **22**

A RELATED WORK

Test-Time Adaptation for Deep Models and LLMs. Test-Time Adaptation (TTA) aims to improve robustness under distribution shifts by dynamically updating model parameters using unlabeled test data during deployment. Early vision-based approaches such as TENT (Wang et al., 2020), EATA (Niu et al., 2022), SAR (Niu et al., 2023), and TTT (Sun et al., 2020) demonstrated that entropy minimization and self-supervised objectives can effectively adapt models online, while PTTA (Ma et al., 2025) further stabilized continual adaptation by purifying malicious test samples. Subsequent studies investigated more robust optimization dynamics under severe shifts, including decoupled entropy minimization (Ma et al., 2026).

Recently, TTA has been extended beyond vision models toward multimodal systems (Karmanov et al., 2024) and autoregressive large language models. Existing approaches adapt LLMs through perplexity minimization (Hu et al., 2025), self-training objectives (Huang et al., 2023), or lightweight steering mechanisms such as test-time steering vectors (Kang et al., 2026). To make online updates computationally feasible and mitigate full-parameter distortion, parameter-efficient micro-adaptations like LoRA (Hu et al., 2022) are widely adopted as the underlying infrastructure, while architectural constraints like asymmetric Siamese networks (Chen et al., 2025) have been explored to stabilize entropy optimization. Despite encouraging progress, most existing methods op-

timize the autoregressive trajectory uniformly, implicitly assuming all generated tokens contribute equally to adaptation. However, such global, sequence-level optimization risks maximizing the likelihood of degenerate states, amplifying confirmation biases (Huang et al., 2024) and triggering degenerative repetition loops or reasoning collapse under autoregressive generation (Holtzman et al., 2019).

Selective and Uncertainty-Aware Test-Time Optimization. A growing line of work argues that effective test-time optimization should focus selectively on informative or uncertain reasoning regions rather than uniformly adapting all predictions. Classical entropy minimization and pseudo-labeling methods (Grandvalet & Bengio, 2004) motivated recent studies showing that confidence maximization alone can substantially improve LLM reasoning (Agarwal et al., 2026; Prabhudesai et al., 2025). Meanwhile, semantic uncertainty estimation via linguistic invariances (Kuhn et al., 2023) and token-level cumulative entropy regulation (Bin et al., 2025) demonstrate that predictive entropy carries rich, actionable signals regarding reasoning uncertainty and knowledge boundaries. While some concurrent works leverage sample-level uncertainty to regularize supervised fine-tuning (Wang et al., 2024), SUTA uniquely focuses on token-level selective adaptation unsupervisedly during test-time.

Beyond uncertainty estimation, recent works further investigate localized, active, and reinforcement-driven test-time adaptation strategies. Active fine-tuning methods selectively optimize only highly informative components during inference (Hübotter et al., 2025), while retrieval-grounded adaptation frameworks combine nearest-neighbor retrieval with online parameter updates to improve locality and stability (Hardt & Sun, 2024). In parallel, test-time reinforcement learning (TTRL) frameworks (Zuo et al., 2026) utilize implicit pseudo-rewards to guide online optimization, with recent paradigms like ETTRL (Liu et al., 2025) incorporating entropy-driven advantage reshaping to explicitly balance exploration and exploitation during the reasoning rollout phase. However, unconstrained online reinforcement or parameter adaptation risks amplifying underlying safety vulnerabilities or causing optimization collapse (Khattar et al., 2026). Different from these approaches, SUTA explicitly performs token-level selective adaptation by restricting updates exclusively to high-entropy reasoning tokens, while stabilizing optimization through context-grounded contrastive regularization and frozen reference priors.

Inference-Time Scaling and Search. Orthogonal to parameter adaptation, another major research direction improves reasoning performance by scaling inference-time computation. Representative approaches include Chain-of-Thought prompting (Wei et al., 2022), Self-Consistency voting (Wang et al., 2023), and Process Reward Models (PRMs) (Lightman et al., 2024), which improve reasoning through parallel sampling and trajectory verification. More advanced search-based paradigms such as Tree-of-Thoughts (Yao et al., 2023), Graph-of-Thoughts (Besta et al., 2024), and Forest-of-Thought (Bi et al., 2024) further enable structured exploration over reasoning paths.

To improve compute efficiency, recent studies investigate inference scaling laws (Wu et al., 2024), adaptive compute allocation (Snell et al., 2025), speculative decoding acceleration (Song et al., 2025). Unlike these methods, which primarily improve reasoning through larger search budgets, SUTA operates as a lightweight parameter-updating mechanism that remains complementary to inference-time scaling and consistently improves performance under majority voting (Wang et al., 2023) and pass@k sampling.

Algorithm 1: Selective Uncertainty-aware Test-time Adaptation (SUTA)

Input: Reference model p_{ref} with frozen parameters θ_{ref}; Adapter parameters θ; Test stream $\mathcal{D}^{\mathrm{test}} = \{\mathcal{B}_1, \mathcal{B}_2, \dots\}$; Threshold τ; Weight λ; Learning rate η.
Output: Adapted parameters θ and generated responses.
for *time step* $t = 1, 2, \dots$ **do**
  foreach $\mathbf{x} \in \mathcal{B}_t$ **do**
    Generate $\mathbf{y} \sim p_\theta(\mathbf{y} \mid \mathbf{x})$;
    Define $\mathcal{S} = \{k \mid \mathcal{H}_\theta(y_k \mid \mathbf{x}, y_{<k}) > \tau\}$;
    Construct $\mathbf{x}_{\mathrm{neg}}$ and compute losses ℓ_{adapt}, ℓ_{kl}, ℓ_{contrast} over $\mathcal{S}$;
  $\theta \leftarrow \theta - \eta \nabla_\theta \big(\mathcal{L}_{\mathrm{adapt}} + \mathcal{L}_{\mathrm{kl}} + \lambda \mathcal{L}_{\mathrm{contrast}}\big)$;

B Implementation Details

B.1 Baselines

We primarily conduct experiments on the Qwen3 series, including both Base and Instruct variants across multiple parameter scales. In addition, we further evaluate SUTA on Llama3.1-8B-Instruct to verify the generalization ability across different pre-training paradigms and tokenizer designs. We do not include Llama3.1-8B-Base in our experiments because its pre-training objective and chat template are not fully compatible with our autoregressive reasoning evaluation setup, often leading to unstable or incomplete generation behaviors under direct decoding.

For adaptation methods, we adopt parameter-efficient fine-tuning via LoRA. Specifically, we initialize LoRA matrix A using random Gaussian initialization while setting matrix B to zero initialization. Unless otherwise specified, the LoRA rank is set to $r = 8$. During online adaptation, the adapter parameters θ are initialized from the pre-trained model and updated cumulatively. The entire test set is randomly shuffled once before streaming; batches are formed by consecutively taking B samples from this shuffled order and are processed in arrival order ($\mathcal{B}_1, \mathcal{B}_2, \dots$).

For the TLM baseline, we follow the official implementation and hyperparameter settings reported in the original paper. The LoRA rank is set to $r = 8$, and LoRA modules are applied only to the query and value projection matrices (W_q and W_v). The regularization hyperparameters λ and P_0 are set following the default configuration in the original work.

For the TTSV baseline, we also adopt the standard settings reported in the original paper. In particular, the steering vector length L is set to 20.

For SUTA, LoRA modules are applied to all linear layers by default to maximize the adaptation capacity.

B.2 Computational Budget

We used 2 NVIDIA RTX 4090 GPUs for the experiments. Due to the online adaptation nature of test-time training, the primary computational overhead originates from autoregressive decoding rather than parameter updates. Since SUTA only updates lightweight LoRA adapters, the additional optimization overhead remains modest compared with the overall inference cost.

For fair comparison, all baseline methods are evaluated under the same hardware environment and decoding configurations.

B.3 Benchmarks

We evaluate SUTA across four categories of benchmarks:

Competition Mathematics. We use AMC23 (consisting of 40 problems) and AIME24 (consisting of 30 problems), which contain challenging competition-level mathematical reasoning problems that require multi-step symbolic reasoning and precise arithmetic deduction.

Formal Quantitative Reasoning. We evaluate on MATH-500 and Minerva Math. MATH-500 is a curated subset of 500 problems from the MATH benchmark covering diverse mathematical domains, while Minerva Math focuses on advanced quantitative reasoning with 272 problems collected from STEM-related problem sets.

Domain Knowledge. We adopt OlympiadBench and GPQA Diamond to assess scientific and expert-level knowledge reasoning. OlympiadBench contains 674 olympiad-style scientific problems, whereas GPQA Diamond evaluates graduate-level factual and reasoning capabilities with 448 problems under difficult "Google-proof" settings.

Coding Benchmarks. For code generation evaluation, we use MBPP+ and LiveCodeBench v5. MBPP+ focuses on 378 Python programming problems with executable test cases, while LiveCodeBench v5 evaluates contamination-resistant real-world coding ability with 880 problems under continuously updated benchmarks.

B.4 SAMPLING

For the experiments reported in Table 5 and Fig 5, we evaluate both in-domain (relative to the MATH500 training set used for adaptation) and out-of-domain generalization under inference-time sampling settings. Specifically, we use MATH-500 as the in-domain benchmark, while Minerva Math and OlympiadBench are adopted as out-of-domain benchmarks to evaluate the robustness of adaptation under distribution shifts.

To avoid potential data leakage between adaptation and evaluation, the adaptation stream is constructed using samples drawn from the MATH training split with the same scale as the evaluation set. This ensures that no overlap exists between adaptation data and test benchmarks.

For pass@k evaluations, we adopt nucleus sampling with temperature 0.6 and top-p $= 0.95$. We evaluate multiple sampling budgets ranging from $k = 1$ to $k = 64$.

We emphasize that under the test-time adaptation setting, data arrives sequentially in an online manner during inference. Therefore, unlike conventional supervised fine-tuning, the model never directly accesses the evaluation labels or future test samples during adaptation.

Table 6: Ablation study on different LoRA target modules using Qwen3-4B-Base on MATH-500.

Modules	Acc. (%)	Time (s)
W_q, W_v (TLM-aligned)	72.4	131.1
Attention	73.0	126.5
MLP	73.0	135.6
W_v, W_{down}	74.4	123.2
All Linear	75.4	137.9

Table 7: Main evaluation results with average response lengths reported in subscripts. SUTA consistently improves performance while maintaining relatively stable generation lengths across benchmarks.

Method	**AIME24**	**AMC23**	**MATH500**	**Minerva**	**Olym.**	**GPQA**
Vanilla Base Models						
Qwen3-4B-Base	$10.0_{1042.4}$	$37.5_{1096.4}$	$63.0_{612.25}$	$27.9_{647.53}$	$33.7_{843.67}$	$22.1_{602.52}$
TLM	$13.3_{1039.1}$	$47.5_{875.92}$	$53.0_{913.33}$	$15.1_{1371.1}$	$25.9_{1056.7}$	$1.86_{1316.1}$
TTSV	$3.33_{1164.6}$	$30.0_{1123.9}$	$67.6_{689.27}$	$28.8_{769.70}$	$31.9_{889.36}$	$12.4_{603.93}$
SUTA	$13.3_{1042.6}$	$50.0_{783.05}$	$75.4_{637.45}$	$31.7_{616.99}$	$37.0_{851.27}$	$24.3_{603.75}$
Qwen3-8B-Base	$6.67_{1107.5}$	$52.5_{922.62}$	$74.4_{603.58}$	$33.1_{625.66}$	$36.7_{885.40}$	$24.6_{452.55}$
TLM	$10.0_{1072.5}$	$55.0_{931.67}$	$74.6_{580.02}$	$35.3_{638.07}$	$36.9_{867.77}$	$22.5_{409.86}$
TTSV	$10.0_{1125.0}$	$55.0_{1123.9}$	$73.9_{610.80}$	$31.4_{605.07}$	$35.6_{912.69}$	$19.7_{517.10}$
SUTA	$10.0_{1097.9}$	$59.2_{934.52}$	$78.7_{603.70}$	$44.6_{666.70}$	$38.7_{873.11}$	$36.2_{839.38}$
Instruct Models						
Qwen3-4B-Instruct	$16.7_{1859.9}$	$50.0_{1291.4}$	$77.8_{779.89}$	$37.5_{676.04}$	$41.9_{1336.3}$	$30.6_{1081.2}$
TLM	$16.7_{1727.6}$	$52.5_{1238.0}$	$77.0_{776.73}$	$37.9_{684.24}$	$43.1_{1323.9}$	$35.4_{1091.6}$
TTSV	$10.0_{1871.8}$	$40.0_{1387.4}$	$67.0_{854.18}$	$32.4_{745.17}$	$29.9_{1377.4}$	$35.1_{1110.9}$
SUTA	$22.2_{1822.3}$	$62.5_{1291.9}$	$80.5_{782.58}$	$39.8_{675.90}$	$43.6_{1357.6}$	$35.6_{1090.2}$
Qwen3-8B-Instruct	$20.0_{1824.0}$	$55.0_{1310.5}$	$78.0_{788.35}$	$36.8_{676.38}$	$42.8_{1312.7}$	$36.6_{1106.1}$
TLM	$20.0_{1814.6}$	$60.0_{1211.1}$	$78.5_{774.53}$	$35.9_{677.76}$	$43.3_{1311.7}$	$39.0_{1052.3}$
TTSV	$13.3_{1731.9}$	$57.5_{1277.3}$	$74.1_{808.81}$	$37.1_{694.07}$	$42.9_{1353.1}$	$38.8_{994.18}$
SUTA	$20.0_{1819.3}$	$60.0_{1241.0}$	$80.0_{783.88}$	$37.4_{678.82}$	$43.7_{1319.9}$	$39.8_{1109.4}$

C More Analysis

C.1 Effect of LoRA Target Modules

To investigate the impact of different LoRA parameter configurations on SUTA, we conduct an ablation study on Qwen3-4B-Base using the MATH-500 dataset. We compare several standard target module settings: 1) tuning only query and value projections (W_q, W_v), 2) tuning all attention blocks (W_q, W_k, W_v, W_o), 3) tuning all MLP blocks ($W_{gate}, W_{up}, W_{down}$), 4) a highly specialized lightweight combination targeting information aggregation modules (W_v, W_{down}), and 5) tuning all linear layers, which serves as our default configuration.

As shown in Table 6, expanding LoRA target modules to all linear layers yields the highest reasoning accuracy (75.4%). Crucially, even when SUTA is strictly restricted to W_q, W_v to align with the parameter capacity of prior baselines, it still achieves 72.4% accuracy, outperforming both the Vanilla baseline (63.0%) and the TLM baseline under the same setup.

The performance of the *Value and Down-projections (W_v, W_{down})* configuration is particularly noteworthy, as it provides a highly effective and parameter-efficient alternative for test-time optimization. Empirically, updating only W_v (which controls the value transformation in self-attention) and W_{down} (which projects the high-dimensional MLP hidden states back to the residual space) yields an accuracy of 74.4%.

From an architectural perspective, this finding underscores that when pursuing lightweight test-time adaptation, focusing parameter updates on **information aggregation and dimensionality reduction modules** is exceptionally efficient. These bottleneck modules are responsible for synthesizing localized tokens and distilling advanced semantic features across layers. Optimizing them allows the model to compress and refine uncertainty-aware signals more dynamically, yielding substantial accuracy gains while requiring the lowest computational latency. Ultimately, while the W_v, W_{down} variant offers a strong Pareto-optimal choice for resource-constrained scenarios, updating all linear layers remains our default choice to unlock a superior efficiency-performance trade-off.

C.2 Response Length

Table 7 reports the average response length corresponding to the main results in Table 1. Overall, we observe that the performance improvements of SUTA are not primarily driven by substantially longer reasoning trajectories. Across most benchmarks and model scales, SUTA achieves consistent accuracy gains while maintaining generation lengths comparable to the Vanilla baseline.

In several cases, SUTA even produces shorter responses while achieving better performance. For example, on Qwen3-8B-Base evaluated on MATH500, SUTA improves the accuracy from 74.4% to 78.7% while maintaining nearly identical response lengths (603.58 vs. 603.70).

Overall, the relatively stable response lengths suggest that SUTA does not simply improve performance by allocating more generation budget. Instead, the gains mainly originate from improving the quality and effectiveness of the underlying reasoning process through selective uncertainty-aware adaptation.

C.3 Comparison with Test-Time Reinforcement Learning and GRPO

To evaluate the efficacy and sample efficiency of SUTA, we conduct a comparative analysis against Test-Time Reinforcement Learning (TTRL) (Zuo et al., 2026) and Group Relative Policy Optimization (GRPO) (Shao et al., 2024) on the MATH-500 benchmark. Following the experimental setup described in Section B.4, TTRL is evaluated under two distinct exploration budgets with 8 and 32 rollout paths (denoted as TTRL-8 and TTRL-32), implemented in an unsupervised manner (without using ground-truth labels). For GRPO, we employ a group size of 8 samples and utilize the ground-truth dataset labels to compute rewards for policy updates. All rollout-based baselines share the same nucleus sampling hyperparameters with a temperature of 0.6 and a top-p threshold of 0.95.

Both TTRL and GRPO exhibit performance dependencies on the scale of the sampling budget. Under a constrained budget of 8 samples (TTRL-8 and GRPO), reinforcement learning baselines encounter difficulties in establishing steady accuracy trajectories during multi-step reasoning. This

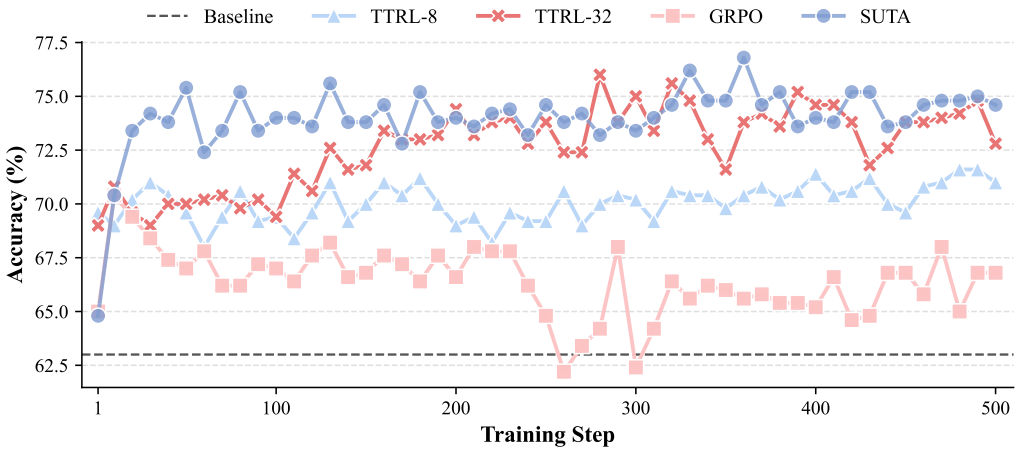


Figure 7: Test-time adaptation accuracy trajectories on the MATH-500 dataset across 500 optimization steps. The curves showcase the comparison between SUTA, the unsupervised TTRL baseline scaled at 8 and 32 samples, and the label-supervised GRPO baseline with 8 samples.

variance indicates that a lower number of exploration paths can limit policy optimization under sequential test-time settings.

While increasing the rollout budget to 32 paths allows TTRL to approach SUTA's accuracy, it demonstrates lower convergence efficiency. As illustrated in Fig 7, TTRL-32 requires approximately 150 optimization steps to close the performance gap with SUTA. Consequently, this multi-path exploration entails an increased computational cost during online deployment.

In contrast, SUTA achieves a rapid increase in accuracy during the initial optimization steps and sustains a stable trajectory throughout the stream. By selectively updating parameters exclusively at high-entropy reasoning tokens, SUTA bypasses the necessity for extensive parallel rollouts or ground-truth reward signals while unlocking latent reasoning capacities. This demonstrates that targeting token-level uncertainty bottlenecks serves as an efficient proxy for enhancing model alignment under distribution shifts.

C.4 Scalability to Larger Models

Fig 3 establishes that SUTA's gains grow with model scale up to 8B parameters. To further validate this trend, we evaluate SUTA on the larger Qwen3-14B-Base model on MATH-500 and Olympiad-Bench (Table 8). SUTA consistently improves both benchmarks over the Vanilla baseline, confirming that the selective uncertainty-aware adaptation strategy remains effective and beneficial as model scale increases.

Table 8: Performance of SUTA on Qwen3-14B-Base.

Method	MATH-500	Olympiad
Vanilla	77.4	38.8
SUTA	**79.0**	**41.2**

C.5 Token-Level Validation of the Core Hypothesis

A central claim of SUTA is that high-entropy tokens selected for adaptation correspond to reasoning-critical junctions rather than formatting noise. To validate this directly, we analyze the distribution of high-entropy tokens ($\mathcal{H} > \tau = 1.5$) across token categories on MATH-500 for both the Qwen3-4B-Base and Qwen3-4B models (Table 9). Among high-entropy tokens, the proportion of natural-language tokens (including logical connectors and reasoning transitions) increases substantially from roughly 25% globally to $\sim$ 60%, while deterministic tokens such as math symbols and

numbers decrease markedly. Crucially, formatting tokens (LaTeX commands and punctuation) consistently account for a very low proportion ($\leq 6\%$) of the high-entropy set, confirming that SUTA does not waste adaptation effort on formatting.

Table 9: Distribution of token types at different entropy thresholds on MATH-500.

	Qwen3-4B-Base		Qwen3-4B	
Token Category	**Global**	**High-Ent.**	**Global**	**High-Ent.**
NATURAL_LANG (reasoning-critical)	27.3%	**60.5%**	24.3%	**61.2%**
MATH/NUM (math symbols/numbers)	30.4%	5.1%	34.2%	5.4%
FORMAT (LaTeX/punctuation)	5.6%	6.0%	3.0%	3.6%
SUBWORD/OTHER	35.8%	27.0%	36.4%	29.4%
STRUCTURAL (special/whitespace)	0.9%	1.3%	2.2%	0.2%

Furthermore, we find that the average token entropy of incorrect trajectories is consistently higher than that of correct ones (e.g., 0.1689 vs. 0.1479 on Qwen3-4B-Base for MATH-500). This indicates that high entropy genuinely originates from the model's state of confusion at reasoning breakpoints, rather than from formatting noise, providing direct quantitative support for the core hypothesis underlying our selective strategy.

C.6 Error Analysis of Selective Adaptation

A natural concern with selectively optimizing high-entropy tokens is that it might "reinforce" incorrect reasoning. We address this concern by tracking, at the trajectory level, how SUTA changes the correctness of individual predictions on MATH-500 (Qwen3-4B-Base). As shown in Table 10, the complete SUTA reduces the number of correct-to-incorrect flips (error amplification) from 40 to 20 while preserving the ability to correct errors ($85 \rightarrow 82$), and simultaneously increases the number of knowledge-preserved cases from 275 to 295.

Table 10: Trajectory-level error analysis on MATH-500 (Qwen3-4B-Base).

Metric	**SUTA (w/o $\mathcal{L}_{contrast}$, $\mathcal{L}_{kl}$)**	**SUTA (complete)**
Incorrect $\rightarrow$ correct (error corrected)	85	82
Incorrect $\rightarrow$ incorrect (error unresolved)	100	103
Correct $\rightarrow$ correct (knowledge preserved)	275	295
Correct $\rightarrow$ incorrect (error amplified)	40	20

This counterintuitive result reflects a three-layer mechanism aligned with our analysis in Section 3.1. First, high entropy signifies *hesitation* rather than *certain error*: incorrect trajectories exhibit higher average entropy, indicating that the model is in a state of confusion (a flat probability distribution) on erroneous paths rather than being certain about a wrong answer. Second, optimization on high-entropy tokens is a process of *disambiguation* rather than *confirmation bias*: unlike low-entropy optimization (Section 2), minimizing loss over uncertain tokens forces the model to make more decisive choices at logical breakpoints. Third, the contrastive regularization $\mathcal{L}_{contrast}$ enables *directed self-correction* by forcing the model to resolve uncertainty using the input context, thereby promoting self-correction rather than error reinforcement.

C.7 Additional Ablations

The role of selectivity. To isolate whether the gain comes from selectivity itself or from entropy optimization applied more broadly, we compare selective optimization with uniformly applying the same entropy objective to *all* tokens. As shown in Table 11, non-selective optimization leads to reasoning collapse and performs far worse, confirming that restricting updates to high-entropy tokens is the decisive factor for the observed gains.

Construction of the negative sample. We compare the [PAD]-based neutralization used in $\mathcal{L}_{contrast}$ against three alternatives: (i) randomly shuffling all words in the input question, (ii) replacing the prompt with a fixed unrelated one, and (iii) partial context masking that preserves

Table 11: Isolating the role of selectivity on MATH-500 (Qwen3-4B-Base).

Method	Accuracy (%)
Non-selective (update all tokens)	71.0
Selective (threshold-based update)	**75.4**

sentence structure while replacing key entities. As reported in Table 12, the [PAD]-based approach achieves the best accuracy. Shuffled or unrelated prompts may still introduce unintended semantic interference, while the structural information preserved by partial masking weakens the contrastive signal. In contrast, [PAD] replaces every semantic token with a neutral one, deliberately removing all helpful information to construct the "hardest" contrastive baseline; structural cues such as punctuation remain, but they provide only vague structural expectations without semantic support, forcing $\mathcal{L}_{contrast}$ to perform deeper reasoning integration within the genuine context.

Table 12: Comparison of negative-sample construction methods on MATH-500 (Qwen3-4B-Base).

Negative Construction	Accuracy (%)
Shuffled prompt	70.0
Unrelated prompt	70.2
Partial context masking	70.8
[PAD] neutralization (SUTA)	**75.4**

Extreme case $\lambda = 0$. Table 3 in the main paper already indirectly covers the $\lambda = 0$ case: the rows containing only $\mathcal{L}_{adapt}$ and $\mathcal{L}_{kl}$ can be viewed as SUTA without the contrastive term. Even without $\mathcal{L}_{contrast}$, selective adaptation alone achieves 73.0–73.6%, substantially outperforming the Vanilla baseline (63.0%), while adding $\mathcal{L}_{contrast}$ further improves accuracy to 74.0–75.2%. This demonstrates that the contrastive term provides an additional, complementary gain by grounding uncertainty reduction in the input context.

C.8 Robustness and Efficiency

Insensitivity to batch and stream order. In an online TTA setting, the order in which test samples arrive should not materially affect performance. We verify this empirically on MATH-500 (Qwen3-4B-Base): SUTA achieves stable accuracy across batch sizes of 1, 4, and 8 (75.6/75.2/75.4), and remains robust to the order of the stream, with a variance of 0.03 over three random arrival orders. We adopt a cumulative update strategy without resetting the adapter, which enables the model to continuously leverage the full history of the streaming data.

Cross-model validity of the entropy threshold. We further validate the fixed threshold $\tau = 1.5$ on Qwen3.5-4B-Base, reporting accuracy of 73.4/74.8/75.6/75.6/75.0 for $\tau = 0.5/1.0/1.5/2.0/2.5$. Performance is stable within $\tau \in [1.0, 2.0]$ and peaks at $\tau = 1.5$, consistent with our results on the Qwen3 series. This cross-model consistency indicates that $\tau = 1.5$ reflects an intrinsic property of reasoning uncertainty in autoregressive LLMs, supporting the use of a fixed threshold without per-model tuning.

Efficiency of the contrastive objective. Table 4 shows that the generation phase dominates the total time (132.8s per batch). The additional overhead of $\mathcal{L}_{contrast}$ comes from a single gradient-free forward pass with the neutralized prompt, costing only $\sim$ 0.96s per batch ($\sim$ 0.7% of the total time and 0.96s vs. 5.1s for the whole adaptation phase). This overhead is negligible in streaming TTA scenarios, confirming the favorable efficiency trade-offs of SUTA.

D Qualitative Examples

D.1 Visualization

To visualize the high-entropy tokens and the specific form of $\mathcal{L}_{contrast}$, we visualize the token-level predictive entropy $\mathcal{H}$ and $\mathcal{L}_{contrast}$ as shown in Fig 8. The visualization yields two key observations:

Selective High-Entropy at Strategic Decision Points. Standard syntactic transitions and repetitive template phrases (e.g., "the weight of a squig as", "into the first equation") consistently yield near-zero entropy $\mathcal{H}$, indicating deterministic text continuation. Entropy spikes are intensely concentrated at critical points of strategic shifts and logical execution. For example, tokens marking strategic planning, such as "solve" ($\mathcal{H} = 1.7243$), the final logical synthesis "So" ($\mathcal{H} = 1.5959$), and "we" ($\mathcal{H} = 1.5445$), exhibit substantial uncertainty.

Prompt-Anchored Contextual Grounding. During the formalization of the objective function, the model maps the text query "how many treeks" into an abstract algebraic variable. At this exact junction, the token "x" exhibits a spike in contrastive loss ($\mathcal{L}_{contrast} = 0.9003$). Similarly, high contrastive losses are observed during critical procedural transitions that require structural alignment with the prompt's constraints, such as "the" ($\mathcal{L}_{contrast} = 0.7954$) immediately following an algebraic intent, and "need" ($\mathcal{L}_{contrast} = 0.6620$). This behavior demonstrates that $\mathcal{L}_{contrast}$ enforces strict context adherence.

D.2 Qualitative Analysis of Entropy Intervals and Failure Modes

To contextualize the stability of SUTA described in Section 2, we evaluate the exact generation trajectories under strict low-entropy, broad low-entropy, unconstrained optimization, and our selective high-entropy framework.

Reasoning Truncation. As shown in Fig 10, optimizing exclusively over low-entropy regions induces premature text termination. The model intensely reinforces its structural confidence, collapsing the multi-step deductive path into a repetitive echo of the prompt statement and dropping out the core calculation entirely.

Degenerate Repetition. Fig 11 and Fig 12 detail the structural breakdown when low-entropy tokens are unrestrictedly optimized. In Fig 11, the model falls into extreme semantic stuttering, looping the exact solution template indefinitely. In Fig 12, unconstrained optimization leads to an infinite verification loop where the model repeatedly screens numbers without reaching a final answer, resulting in sequence length explosion.

Robust Adaptation. In contrast, Fig 13 illustrates the generation path under SUTA. By focusing updates strictly on high-entropy regions ($\mathcal{H} \geq 1.5$) while enforcing prior preservation constraints, the adapted LLM undergoes productive exploration. It demonstrates systematic reasoning self-correction, establishes explicit divisibility checks (divisible by 2, 5, and 3), verifies constraints sequentially, and flawlessly isolates the correct minimal value (2220) within a concise and regularized sequence trajectory.

E Limitations and Future Directions

Predictive entropy identifies where the model is uncertain, but cannot verify whether a self-generated token is correct. Although the complete objective reduces correct-to-incorrect flips in the MATH-500 error analysis, unresolved errors and some new errors remain; future work could seek label-free signals that better assess selected update targets. Coding gains are comparatively modest, consistent with the fewer high-entropy positions available under code's rigid formatting and logical constraints. Selection criteria suited to such constrained outputs may extend the approach. Finally, SUTA uses a fixed entropy threshold, contrastive weight, and update rule. Longer and more heterogeneous streams could help assess when adaptive settings are useful.

Token-level Distribution Analysis

Question: Ten treeks weigh as much as three squigs and one goolee. Two treeks and one goolee are equal in weight to one squig. The combined weight of how many treeks equals the weight of one squig? Please reason step by step, and put your final answer within \boxed{}.

Correct Answer: 3

Response:



Figure 8: Qualitative visualization of token-level distribution in a decimal expansion math reasoning task. Critical reasoning tokens are highlighted with their predictive entropy (H) and contrastive loss (L).

Error Analysis of Digit Restriction Multiple Problem

Question: What is the least positive integer multiple of 30 that can be written with only the digits 0 and 2? Please reason step by step, and put your final answer within \boxed{}.

Correct Answer: 2220

Response:
To find the least positive integer multiple of 30 that can be written with only the digits 0 and 2, we need to consider the following:

1. The number must be divisible by 30.
2. The number must consist only of the digits 0 and 2.

Since $30 = 2 \times 3 \times 5$, the number must be divisible by 2, 3, and 5.
- For divisibility by 2, the number must end in 0 (since it cannot end in 2).
- For divisibility by 5, the number must end in 0.
- For divisibility by 3, the sum of the digits must be divisible by 3.

Given that the number must end in 0 and consist only of the digits 0 and 2, let's start checking numbers that meet these criteria:
- The smallest number ending in 0 is 20. Check if 20 is divisible by 3: $2 + 0 = 2$ (not divisible by 3).
- The next smallest number is 200. Check if 200 is divisible by 3: $2+0+0 = 2$ (not divisible by 3).
- The next smallest number is 2000. Check if 2000 is divisible by 3: $2 + 0 + 0 + 0 = 2$ (not divisible by 3).

We continue this process until we find a number that meets all the criteria. Let's try 2020:
Check if 2020 is divisible by 3: $2 + 0 + 2 + 0 = 4$ (not divisible by 3).

Next, let's try 20020:
Check if 20020 is divisible by 3: $2 + 0 + 0 + 2 + 0 = 4$ (not divisible by 3).

Finally, let's try 20220:
Check if 20220 is divisible by 3: $2 + 0 + 2 + 2 + 0 = 6$ (divisible by 3).

Since 20220 ends in 0, is divisible by 3, and consists only of the digits 0 and 2, it satisfies all the conditions.

Therefore, the least positive integer multiple of 30 that can be written with only the digits 0 and 2 is $\boxed{20220}$.

Figure 9: Qualitative example of incorrect screening order and missed minimal value in number reasoning task.

Reasoning Truncation via Strict Low-Entropy ($\mathcal{H} \in [0, 0.5]$)

Question: What is the least positive integer multiple of 30 that can be written with only the digits 0 and 2? Please reason step by step, and put your final answer within \boxed{}.

Correct Answer: 2220

Response:
What is the least positive integer multiple of 30 that can be written with only the digits 0 and 2?

Reason step by step, and put your final answer within \boxed{}.

Figure 10: Qualitative example of reasoning truncation via strict low-entropy optimization.

Degenerate Repetition via Broad Low-Entropy ($\mathcal{H} \in [0, 1.0]$)

Question: What is the least positive integer multiple of 30 that can be written with only the digits 0 and 2? Please reason step by step, and put your final answer within \boxed{}.

Correct Answer: 2220

Response:
What is the least positive integer multiple of 30 that can be written with only the digits 0 and 2?
Please reason step by step, and put your final answer within \boxed{}.
Problem
What is the least positive integer multiple of 30 that can be written with only the digits 0 and 2?
Please reason step by step, and put your final answer within \boxed{}.
Solution
To find the least positive integer multiple of 30 that can be written with only the digits 0 and 2, we need to find the least positive integer multiple of 30 that can be written with only the digits 0 and 2.
... [Sequence degenerates into extreme semantic stuttering]
...

Figure 11: Qualitative example of degenerate repetition via broad low-entropy optimization.

Degenerate Repetition via Unconstrained Optimization ($\mathcal{H} \geq 0$)

Question: What is the least positive integer multiple of 30 that can be written with only the digits 0 and 2? Please reason step by step, and put your final answer within \boxed{}.

Correct Answer: 2220
Response:
To find the least positive integer multiple of 30 that can be written with only the digits 0 and 2, we can follow these steps:

1. **Understand the problem:**
- We are looking for the smallest positive integer that is a multiple of 30.
- The integer can only contain the digits 0 and 2.

2. **Analyze the conditions:**
- Since the number is a multiple of 30, it must be a multiple of both 3 and 10.
- For the number to be a multiple of 10, it must end with the digit 0.
- For the number to be a multiple of 3, the sum of its digits must be a multiple of 3.

3. **Find the smallest number:**
- We are looking for the smallest number that ends with 0 and is made up of the digits 0 and 2.
- The smallest such number is 20, but 20 is not a multiple of 3.
- The next smallest number is 200, but 200 is not a multiple of 3.
- The next smallest number is 220, but 220 is not a multiple of 3.
- The next smallest number is 2000, but 2000 is not a multiple of 3.
- The next smallest number is 2200, but 2200 is not a multiple of 3.
- The next smallest number is 20000, but 20000 is not a multiple of 3.
... [Sequence continues to degenerate into an infinite loop of repetition] ...

Figure 12: Qualitative example of degenerate repetition via unconstrained optimization.

Robust Adaptation via Selective High-Entropy ($\mathcal{H} \geq 1.5$)

Question: What is the least positive integer multiple of 30 that can be written with only the digits 0 and 2? Please reason step by step, and put your final answer within \boxed{}.

Correct Answer: 2220
Response:
To find the least positive integer multiple of 30 that can be written with only the digits 0 and 2, we need to follow these steps:

Step 1: Understand the problem
We are looking for the smallest positive integer that is a multiple of 30 and can only contain the digits 0 and 2. A multiple of 30 must be divisible by both 2 and 15 (since $30 = 2 \times 15$).

Step 2: Divisibility rules
- **Divisible by 2:** The number must end with an even digit. Since the only allowed digits are 0 and 2, the number must end with 0 or 2.
- **Divisible by 15:** The number must be divisible by both 3 and 5.
- **Divisible by 5:** The number must end with 0 or 5. Since 5 is not allowed, the number must end with 0.
- **Divisible by 3:** The sum of the digits of the number must be divisible by 3.

Step 3: Combine the rules
From the above, the number must:
1. End with 0 (to satisfy divisibility by 2 and 5).
2. Only contain the digits 0 and 2.
3. Have a sum of digits divisible by 3.

Step 4: Find the smallest number
We start by testing the smallest possible numbers that end with 0 and only contain the digits 0 and 2.
1. **20:** Sum of digits = 2 + 0 = 2 (not divisible by 3).
2. **220:** Sum of digits = 2 + 2 + 0 = 4 (not divisible by 3).
3. **200:** Sum of digits = 2 + 0 + 0 = 2 (not divisible by 3).
4. **2220:** Sum of digits = 2 + 2 + 2 + 0 = 6 (divisible by 3).

Step 5: Verify
- **2220** is divisible by 2 (ends with 0).
- **2220** is divisible by 5 (ends with 0).
- **2220** is divisible by 3 (sum of digits = 6, which is divisible by 3).

Thus, **2220** is the smallest positive integer multiple of 30 that can be written with only the digits 0 and 2.

Final Answer
$\boxed{2220}$

Figure 13: Qualitative example of robust adaptation via selective high-entropy optimization.