FIRST GENERATE, THEN DELETE: TEST-TIME THINKING-TOKEN DELETION FOR MULTIMODAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large reasoning models typically solve complex problems by generating long chain-of-thought traces before producing final answers. In this work, we observe a phenomenon that not every generated thinking token is equally beneficial for subsequent answer decoding. Motivated by this observation, we propose Test-time Token Deletion (TTD), a first-generate-then-delete framework for test-time reasoning intervention. Given a question, the model first produces an initial reasoning trace, and TTD then selectively deletes a subset of thinking tokens and re-decodes the final solution conditioned on the remaining trace. To select tokens for deletion, we instantiate TTD with a training-free entropy-guided rule and a test-time mask-learning variant optimized with forward-only evaluations, while keeping the base model frozen. Extensive experiments on various multimodal reasoning benchmarks show that TTD improves reasoning accuracy while reducing response length. Our findings suggest that selective thinking-token pruning provides a complementary axis of test-time scaling beyond generating more tokens.

1 INTRODUCTION

Recent advances in reasoning-oriented Large Language Models (LLMs) have substantially improved performance on complex reasoning tasks, especially mathematical reasoning, as exemplified by OpenAI's o1 and o3 (OpenAI, 2024; 2025), DeepSeek-R1 (Guo et al., 2025), and QwQ (Qwen Team, 2024; 2025). In parallel, Multimodal Large Language Models (MLLMs) have made rapid progress on visual and multimodal reasoning (Meng et al., 2025; Deng et al., 2025; Xiao & Gan, 2025). These improvements have been driven largely by two complementary directions, training-time reasoning enhancement, such as reinforcement-learning-based post-training (Guo et al., 2025; Hu, 2025; Shao et al., 2024), and inference-time computation, often referred to as test-time scaling (Hou et al., 2025; Xiang et al., 2025; Yue et al., 2025; Li et al., 2025b). Test-time scaling typically improves reasoning by allocating additional computation at inference, either through sequential refinement (Hou et al., 2025; Xiang et al., 2025) or through parallel sampling and self-consistency-based aggregation (Wang et al., 2022; Fu et al., 2025; Snell et al., 2025; Yue et al., 2025). Across these approaches, step-by-step Chain-of-Thought (CoT) traces serve as a common reasoning interface through which models generate long intermediate thinking tokens before producing final answers (Wei et al., 2022; Chen et al., 2025; Liu et al., 2025; Jiang et al., 2025).

However, the success of CoT-based reasoning also raises a fundamental question: are all generated intermediate tokens necessary or beneficial for subsequent final-answer decoding? Prior studies have investigated token selection during pre-training (Lin et al., 2024) or post-training (Wang et al., 2026), mainly focusing on which tokens should contribute to model optimization. In contrast, less is known about thinking tokens that have already been generated at test time and how they affect subsequent answer decoding. To investigate this question, we draw inspiration from in-context learning (Dai et al., 2023; Yang et al., 2024) and introduce a reasoning analysis framework that treats previously generated thinking tokens as in-context hints for producing a subsequent direct answer, as shown in Figure 1. Specifically, after a complete CoT trace has been produced, we condition the model on this trace and prompt it to produce the final answer without additional deliberate reasoning. Within this framework, we conduct *leave-one-out* deletion analysis, where each generated thinking token is removed in turn, to examine whether each token contributes positively to the conditioning context.

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.

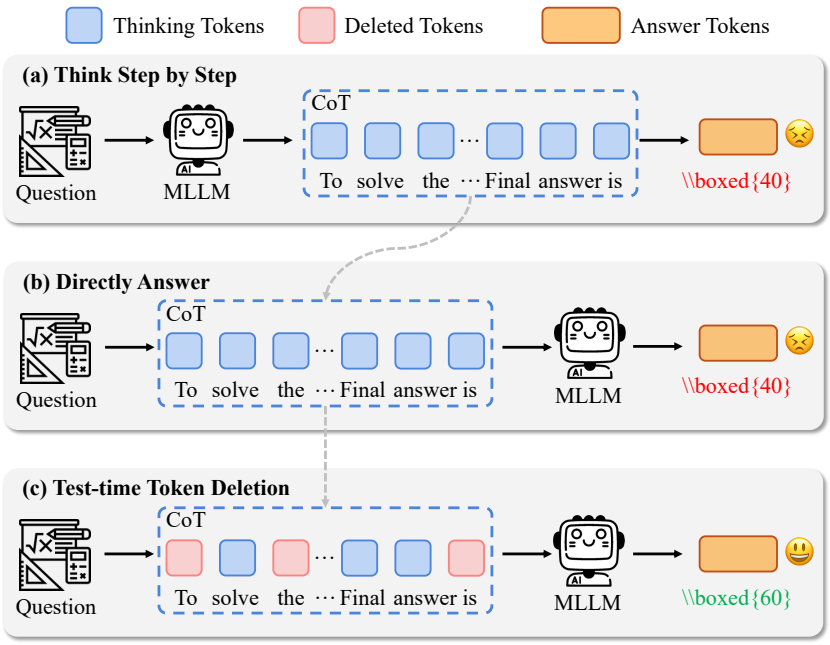


Figure 1: Illustration of our framework. (a) *Think Step by Step*: the MLLM generates chain-of-thought (CoT) tokens and then produces a final answer, which can still be incorrect. (b) *Directly Answer*: the MLLM is prompted to output the answer conditioned on the previously generated CoT without additional deliberate reasoning, often retaining the same error. (c) *Test-time Token Deletion*: we delete a small subset of tokens from the generated CoT at inference time and re-generate the answer, correcting the prediction.

Empirically, we observe that removing a few tokens, and even a single token, from the reasoning trace can significantly improve reasoning performance by flipping incorrect answers into correct ones. This suggests that generated CoT traces may contain tokens that are less useful, or even distracting, for final-answer decoding.

Building upon the discovery, we propose Test-time Token Deletion (TTD), a simple test-time reasoning intervention framework for MLLMs, as illustrated in Figure 1(c). TTD follows a first-generate-then-delete pipeline without modifying model parameters. Given an input question, the model first generates an initial reasoning trace. TTD then applies a binary mask over the generated thinking tokens, removes selected tokens, and re-decodes the final answer conditioned on the remaining trace. We instantiate TTD with two deletion strategies. The first is a training-free entropy-guided rule that selects high-entropy thinking tokens for deletion based on predictive uncertainty. The use of entropy differs from but does not contradict prior work that identifies high-entropy tokens as forking points determining reasoning trajectories (Wang et al., 2026; Qian et al., 2026), since our method first allows the model to generate a complete reasoning trace and then selectively deletes tokens less useful for decoding the final answer. The second is a test-time mask-learning variant that learns an instance-adaptive thinking-token mask through forward-only optimization. Overall, TTD provides a lightweight mechanism for intervening on reasoning traces and offers a complementary way to improve test-time reasoning beyond simply generating more tokens.

The main contributions of our work are summarized as follows:

- We reveal that not all generated thinking tokens are equally useful for final-answer decoding, and that deleting selected intermediate thinking tokens can correct some initially wrong predictions.
- We propose TTD, a simple yet effective test-time intervention framework with both a training-free entropy-guided rule and a forward-only learned-mask variant.
- Extensive experiments on eight multimodal reasoning benchmarks show that TTD consistently improves reasoning accuracy while reducing response length. Additional evaluations on pure-text reasoning and tool-augmented reasoning tasks, together with results across diverse model families and scales, further support its good transferability.

2 PRELIMINARY AND MOTIVATIONS

We first introduce the multimodal model decoding setup and the token-level entropy score used in our method. We then present two observations that motivate deleting selected thinking tokens at test time.

Multimodal large language model decoding. Given a visual question q and an input image, a visual encoder extracts visual features z. A MLLM π_θ with parameters θ then generates an answer autoregressively conditioned on (q, z). We treat the model-produced reasoning trace as a prefix $\mathcal{X} = (x_1, \ldots, x_L)$ and the final answer as a token sequence $\mathcal{Y} = (y_1, \ldots, y_N)$. Let $p_\theta(\cdot)$ denote the token distributions induced by π_θ. The conditional likelihood of generating the answer given the trace is as follows where $\mathcal{Y}_{<n} = (y_1, \ldots, y_{n-1})$.

$$p_\theta(\mathcal{Y} \mid q, z, \mathcal{X}) = \prod_{n=1}^{N} p_\theta(y_n \mid q, z, \mathcal{X}, \mathcal{Y}_{<n}), \tag{1}$$

Token entropy. At decoding step t, the model outputs a distribution over the vocabulary $\mathcal{V}$ denoted as $p_t(v) = p_\theta(v \mid \mathrm{ctx}_t)$ for $v \in \mathcal{V}$, where ctx_t is the conditioning context at step t (e.g., a prefix consisting of (q, z) and previously generated tokens). The token entropy is defined as

$$H_t = -\sum_{v \in \mathcal{V}} p_t(v) \log p_t(v). \tag{2}$$

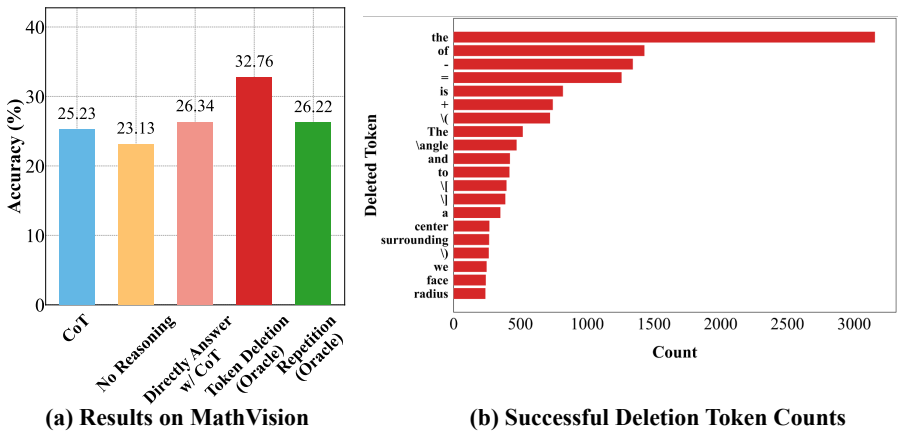
In practice, we use a truncated top-k entropy score to reduce noise from the long tail and to lower the memory footprint for large vocabularies (e.g., $|\mathcal{V}| > 150$K for Qwen2.5-VL-7B). Let $\mathcal{V}_t^{(k)}$ denote the set of top-k tokens under $p_t(\cdot)$. We define the truncated entropy score as

$$H_t^{(k)} = -\sum_{v \in \mathcal{V}_t^{(k)}} p_t(v) \log p_t(v). \tag{3}$$

Observation 1: Not all thinking tokens are equal for answer decoding. Our investigation begins by examining whether each generated thinking token is equally useful as conditioning context for final-answer decoding. As shown in Figure 2(a), standard Chain-of-Thought (CoT) prompting substantially improves accuracy over answering without reasoning. Moreover, prompting the model to directly answer while conditioning on the previously generated thinking trace achieves performance comparable to standard CoT. This setting separates the generation of the reasoning trace from the subsequent decoding of the final answer, allowing us to analyze how individual thinking tokens affect answer decoding. Within this framework, we conduct *leave-one-out* deletion analysis. Given a generated reasoning trace $\mathcal{X}$, we remove one token at a time and construct $\mathcal{X}^{(-\ell)} = (x_1, \ldots, x_{\ell-1}, x_{\ell+1}, \ldots, x_L)$, where x_ℓ is the deleted token. We then prompt the model to produce the final answer conditioned on $\mathcal{X}^{(-\ell)}$ without additional deliberate reasoning. For initially incorrect examples, we record whether deleting a single token can flip the final answer from incorrect to correct. This oracle leave-one-out analysis uses ground-truth correctness only for diagnosis and is not used by our TTD method.

Empirically, we observe that selectively removing individual thinking tokens can make the remaining trace a more effective conditioning context for final-answer decoding. This intervention corrects a non-trivial fraction of initially wrong predictions and improves MathVision accuracy from 25.23% to 32.76%. As a control to verify the gain is not explained by repeated independent generations alone, we repeatedly query the model with the unmodified trace using the same number of decoding attempts as in the leave-one-out setting. This repeated decoding control yields little improvement, suggesting that the observed corrections mainly arise from changes in the conditioning trace rather than from multiple generations (Yue et al., 2025) or decoding dynamics alone. Furthermore, Figure 2(b) shows that many successfully deleted tokens are function words, filler words, or punctuation marks. This suggests that even tokens with limited task-specific content can affect final-answer decoding.

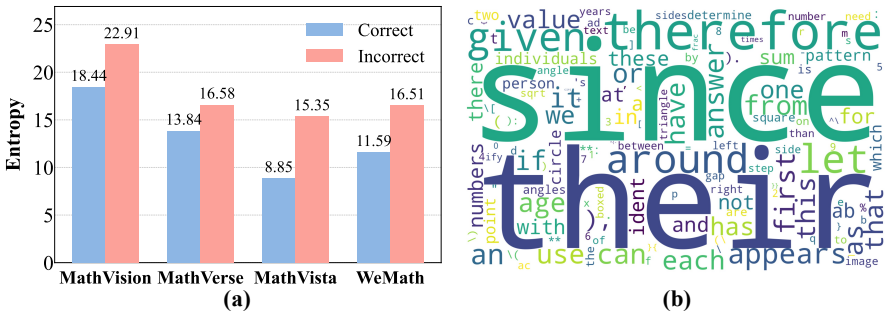
Observation 2: Thinking-token entropy correlates with answer correctness. We further analyze the relationship between reasoning uncertainty and answer accuracy. For each example, we prompt the model to generate a thinking trace $\mathcal{X}$ followed by a final answer $\mathcal{Y}$, and record the truncated entropy score for each generated thinking token. We then partition all traces into a correct set and an incorrect set based on final-answer correctness. As shown in Figure 3(a), reasoning traces that



(a) Results on MathVision **(b) Successful Deletion Token Counts**

Figure 2: Leave-one-out deletion analysis on MathVision using Qwen2.5-VL-7B. See Appendix E.1 for results on other benchmarks. (a) Accuracy under different prompting and intervention strategies. Token deletion removes one generated thinking token at a time and re-decodes the final answer. Repetition repeatedly decodes the answer with the unmodified trace using the same number of attempts. Oracle denotes that ground-truth labels are used to identify successful interventions for analysis. (b) Token-frequency statistics of the first successful deletion, with the top-20 tokens shown.

lead to incorrect answers tend to exhibit higher token entropy throughout generation. The consistent gap suggests that token entropy can serve as a useful uncertainty signal for identifying generated traces that are more likely to produce incorrect answers. Figure 3(b) further shows that many high-entropy tokens correspond to function words (e.g., articles and prepositions), discourse markers, and transition phrases. Together with the leave-one-out analysis, this observation motivates using entropy as the deletion signal. Importantly, a high-entropy token may be important during reasoning generation, for example, by acting as a branching point in a reasoning path, while still being less useful, or even distracting, when the completed trace serves as conditioning context for final-answer decoding. These findings motivate a test-time strategy that selects and removes a subset of generated thinking tokens, and then re-decodes the final answer conditioned on the remaining trace. In the next section, we introduce entropy-guided token deletion and a test-time mask-learning variant for adaptive thinking-token deletion.



(a) **(b)**

Figure 3: **Entropy correlates with answer correctness.** (a) Average thinking-token entropy across multimodal mathematical reasoning benchmarks with Qwen2.5-VL-7B, split by whether the final answer is correct. Reasoning traces leading to incorrect answers show higher entropy. (b) Thinking-token word cloud on MathVista, where token size is proportional to average entropy. High-entropy tokens often include function words (e.g., prepositions/articles) and transition phrases.

3 METHODOLOGY

Motivated by the observations, we propose Test-Time Token Deletion (TTD), a first-generate-then-delete pipeline for improving final-answer decoding without modifying model parameters. Given an input question, the model first generates a reasoning trace. TTD then constructs a binary mask over the generated thinking tokens, deletes selected tokens, and re-decodes the final answer conditioned

on the remaining trace. Formally, given a reasoning trace $\mathcal{X} = (x_1, \ldots, x_L)$, we aim to obtain a binary mask $\mathbf{m} \in \{0,1\}^L$ over thinking tokens, where $m_\ell = 1$ indicates that x_ℓ is kept and $m_\ell = 0$ indicates that it is deleted. Applying $\mathbf{m}$ yields a masked trace $\mathcal{X}^{(\mathbf{m})}$ that preserves the original order of the kept tokens. We then re-decode the final answer conditioned on the masked trace:

$$p_\theta(\mathcal{Y} \mid q, z, \mathcal{X}^{(\mathbf{m})}) = \prod_{n=1}^{N} p_\theta\Big(y_n \mid q, z, \mathcal{X}^{(\mathbf{m})}, \mathcal{Y}_{<n}\Big). \tag{4}$$

We next instantiate TTD in two ways. The first is a training-free entropy-guided deletion rule, and the second is a test-time mask-learning variant that optimizes an instance-adaptive mask.

3.1 Entropy-Guided Token Deletion

Inspired by the observation that there is a strong entropy gap between correct and incorrect samples driven by minor high-entropy thinking tokens, we propose a straightforward method that uses token uncertainty as a simple signal for selecting candidate tokens for deletion. During the initial generation of $\mathcal{X}$, we compute the truncated top-k token entropy score $H_\ell^{(k)}$ for each generated thinking token x_ℓ according to Eq. 3. We then delete high-entropy thinking tokens using a threshold τ:

$$m_\ell = \mathbb{I}\Big[H_\ell^{(k)} \le \tau\Big], \qquad \ell = 1, \ldots, L. \tag{5}$$

This yields a training-free mask that removes thinking tokens associated with high predictive uncertainty. Finally, we re-decode the answer using Eq. 4.

It is important to distinguish our use of entropy from prior studies that identify high-entropy tokens as forking points and optimize the model only on these tokens during post-training (Wang et al., 2026; Qian et al., 2026). In our setting, the model first generates a complete reasoning trace. Token entropy is then used only as a deletion signal for the already generated trace before final-answer generation. Therefore, deleting a token in our framework does not affect the forking process. TTD instead changes only the conditioning context used for decoding the final answer. A token may be important for forming a reasoning trajectory during generation, while still being less useful as conditioning context for final-answer re-decoding in some instances.

3.2 Test-Time Token Mask Learning

The entropy-guided rule is simple and efficient, but a fixed entropy threshold may be suboptimal because the token role can be instance-dependent and token interactions are not explicitly modeled. We therefore introduce a test-time mask-learning variant that learns an instance-adaptive thinking-token mask while keeping the MLLM parameters θ frozen.

Mask parameterization. We introduce mask logits $\mathbf{s} \in \mathbb{R}^L$ and define soft gates $\mathbf{g} = \sigma(\mathbf{s}) \in (0,1)^L$, where $\sigma(\cdot)$ is the sigmoid function. The gate g_ℓ controls the contribution of the thinking token x_ℓ to answer decoding. After optimization, we discretize the soft gates to obtain a hard mask

$$m_\ell = \mathbb{I}[g_\ell \ge \gamma], \tag{6}$$

with $\gamma = 0.5$ by default.

Forward-only optimization. Recent test-time adaptation methods often optimize model parameters on unlabeled test data using output-entropy objectives (Wang et al., 2020; Lee et al., 2024; Chen et al., 2022; Niu et al., 2022; Hu et al., 2025). Inspired yet different from the previous methods, we keep the base MLLM fixed and optimize only the mask applied to the generated thinking tokens. Since common inference engines (e.g., vLLM (Kwon et al., 2023)) do not support backpropagation through decoding, we adopt a zeroth-order optimization strategy that relies only on forward evaluations of the objective. Specifically, we optimize the mask logits $\mathbf{s}$ using Simultaneous Perturbation Stochastic Approximation (SPSA) (Spall, 2002; Malladi et al., 2023). At iteration t, we sample a perturbation vector $\mathbf{u}_t \in \mathbb{R}^L$ (e.g., $\mathbf{u}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_L)$), evaluate the objective $\mathcal{L}(\cdot)$ twice, and estimate the gradient as

$$\widehat{\nabla}\mathcal{L}(\mathbf{s}_t) = \frac{\mathcal{L}(\mathbf{s}_t + \epsilon\mathbf{u}_t) - \mathcal{L}(\mathbf{s}_t - \epsilon\mathbf{u}_t)}{2\epsilon}\,\mathbf{u}_t, \tag{7}$$

where ϵ is the perturbation scale. We then update the mask logits by $\mathbf{s}_{t+1} = \mathbf{s}_t - \eta\,\widehat{\nabla}\mathcal{L}(\mathbf{s}_t)$ with the step size η. Each SPSA step requires two objective evaluations, making it compatible with inference-engine-based decoding and with the non-differentiable hard mask in Eq. 6.

Optimization objective A crux design choice is the objective $\mathcal{L}$ used to optimize the mask. A natural option is to minimize the entropy of the answer-token distributions (Wang et al., 2020), encouraging the model to produce a more confident final answer conditioned on the pruned thinking trace. Given mask logits $\mathbf{s}$, we obtain a hard mask $\mathbf{m}$ via Eq. 6 and decode an answer from the remaining trace $\mathcal{X}^{(\mathbf{m})}$. We define the entropy objective as

$$\mathcal{L}_{\text{ent}}(\mathbf{s}) = \frac{1}{N}\sum_{n=1}^{N} H^{(k)}\Big(p_\theta(\cdot \mid q, z, \mathcal{X}^{(\mathbf{m})}, \mathcal{Y}_{<n})\Big), \tag{8}$$

where $H^{(k)}(\cdot)$ is the truncated entropy score defined in Eq. 3.

However, entropy minimization alone can be unreliable at test time. As shown in Table 1, compared with an oracle learner that uses ground-truth feedback, naive entropy minimization can suffer from severe error accumulation and drive optimization toward overconfident but incorrect predictions (Lee et al., 2024). The oracle learner updates the mask only when the predicted answer is correct, thereby avoiding error accumulation from erroneous pseudo-labels. The performance gap between the oracle learner and the entropy-only objective indicates that minimizing output entropy alone is not enough for test-time mask learning.

Table 1: Entropy minimization suffers from error accumulation at test time. Output-entropy minimization often drives test-time mask optimization toward overconfident but incorrect solutions, whereas an Oracle learner yields substantial gains. The Oracle uses a MathRuler grader function (hiyouga, 2025) to evaluate predictions against ground-truth answers and updates the mask only when the prediction is correct. Results are reported on MathVista with Qwen2.5-VL-7B.

Models	**MathVista**
Qwen2.5-VL-7B	67.20
+ Entropy-Guided TTD (threshold τ=0.1)	69.50
+ Masking Learning TTD (Oracle)	82.10
+ Masking Learning TTD (Entropy Min.)	70.70
+ Masking Learning TTD (Ours)	71.50

To improve the reliability of mask learning, we combine output entropy minimization with a robustness term that discourages masks whose predictions are overly sensitive to local perturbations of the thinking trace. Specifically, we apply a chunked random shuffle perturbation to thinking tokens with window size w, denoted by $\tilde{\mathcal{X}} = \text{Shuffle}_w(\mathcal{X})$. Let $\mathcal{Y}^*$ be the pseudo-answer predicted from the unshuffled pruned reasoning trace $\mathcal{X}^{(\mathbf{m})}$. We penalize the change in the model's probability of $\mathcal{Y}^*$ under the shuffled trace as

$$\mathcal{L}_{\text{rob}}(\mathbf{s}) = \Big|p_\theta(\mathcal{Y}^* \mid q, z, \mathcal{X}^{(\mathbf{m})}) - p_\theta(\mathcal{Y}^* \mid q, z, \tilde{\mathcal{X}}^{(\mathbf{m})})\Big|, \tag{9}$$

where larger values indicate the prediction is more sensitive to perturbations of the reasoning trace.

The final objective for the mask logits learning is

$$\mathcal{L}(\mathbf{s}) = \mathcal{L}_{\text{ent}}(\mathbf{s}) + \lambda\,\mathcal{L}_{\text{rob}}(\mathbf{s}), \tag{10}$$

where λ controls the strength of the robustness term. After test-time optimization, we discretize the learned gates using Eq. 6 and re-decode the final answer using Eq. 4. The complete algorithm of test-time masking learning is provided in Algorithm 1.

4 Experiments

We first elaborate on the experimental setup. Section 4.1 compares TTD with test-time intervention baselines and examines its transferability across tasks, model families, and scales. Section 4.2 analyzes key design choices. We present qualitative case studies in Appendix I.

Experimental setup. We use Qwen2.5-VL-7B (Bai et al., 2025b) as the base MLLM for our main experiments on eight benchmarks spanning two categories, including multimodal mathematical reasoning and general multimodal understanding and reasoning. We further validate the generalization of TTD to tool-augmented multimodal reasoning and text-only reasoning tasks. Benchmark details

Table 2: **Comparison with test-time intervention strategies.** We report performance on four multimodal mathematical reasoning benchmarks and four multimodal general reasoning benchmarks, together with the average performance within each benchmark group. The base model is Qwen2.5-VL-7B (Bai et al., 2025b). Pass@1 denotes accuracy with a single sampled reasoning trace, and Length denotes the average number of generated tokens per response. Higher Pass@1 and lower Length are better. Results are reported as the mean and standard deviation over three runs with different random seeds. Oracle uses ground-truth answers to learn masks only for analysis.

Method	MathVision		MathVerse		MathVista		WeMath		Average	
	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length
Base Model	25.06±0.11	529.72±4.52	43.64±0.79	398.84±1.33	66.83±0.26	281.97±7.87	63.54±0.68	286.07±2.84	49.77±0.20	374.15±3.58
Random Deletion	24.34±0.21	352.96±3.01	43.68±0.53	277.54±0.92	66.67±0.25	138.92±3.88	65.03±0.29	153.68±1.52	49.93±0.19	230.78±2.03
MTI (Yang et al., 2025c)	26.02±0.13	548.45±2.01	45.68±0.14	406.06±2.71	66.60±0.21	306.94±6.39	63.48±0.22	299.08±3.14	50.44±0.06	390.13±2.20
TTI (Shen et al., 2025)	25.37±0.10	996.14±3.70	44.51±0.11	759.01±4.15	67.13±0.12	544.44±1.02	63.83±0.07	556.16±1.67	50.21±0.04	713.94±2.40
s1 (Muennighoff et al., 2025)	25.90±0.29	1357.06±0.57	44.16±0.62	1224.78±1.13	68.43±0.80	770.85±33.44	63.29±0.75	1037.28±9.08	50.45±0.57	1097.49±7.54
TTD (Oracle)	50.44±0.63	380.82±2.20	56.87±0.35	301.19±1.58	81.63±0.41	183.85±2.20	81.28±0.39	175.46±2.21	67.56±0.30	260.33±0.96
TTD (Ours)	**27.16**±0.05	**352.96**±3.01	**46.30**±0.15	**277.54**±0.92	**71.53**±0.28	**138.92**±3.88	**67.64**±0.09	**153.68**±1.52	**53.16**±0.08	**230.77**±2.03

Method	MMMU		ScienceQA		A-OKVQA		MMBench		Average	
	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length
Base Model	51.44±0.18	423.91±6.83	85.21±0.06	191.59±0.91	82.32±0.71	145.63±1.47	84.79±0.16	159.79±1.20	75.94±0.17	230.23±1.90
Random Deletion	49.55±0.77	308.82±4.98	84.46±0.12	132.67±0.62	83.19±0.46	43.22±0.43	84.90±0.11	84.63±0.63	75.52±0.24	142.33±1.23
MTI (Yang et al., 2025c)	51.25±0.90	422.16±16.78	85.12±0.46	194.32±1.04	82.04±0.04	153.26±3.69	85.06±0.11	163.20±1.35	75.86±0.27	233.23±5.22
TTI (Shen et al., 2025)	51.62±0.26	779.97±1.70	84.88±0.16	361.76±1.01	79.14±0.41	272.23±4.45	83.71±0.17	314.39±2.64	74.84±0.24	432.09±1.43
s1 (Muennighoff et al., 2025)	51.47±0.66	949.45±42.91	84.77±0.13	605.04±13.59	82.26±0.60	744.95±1.49	84.14±0.10	654.61±14.10	75.66±0.25	738.52±18.02
TTD (Oracle)	72.74±0.60	265.50±4.18	91.04±0.22	116.59±2.87	85.97±0.25	97.75±3.02	88.73±0.46	102.51±2.74	84.62±0.34	145.59±2.35
TTD (Ours)	**53.41**±0.22	**308.82**±4.98	**86.00**±0.16	**132.67**±0.62	**84.89**±0.25	**43.22**±0.43	**85.78**±0.08	**84.63**±0.63	**77.52**±0.13	**142.33**±1.23

are summarized in Table 3 in Appendix C.2. For all benchmarks, we report accuracy (Pass@1) and response length. Further implementation details of TTD, prompt templates, and evaluation details are provided in Appendix C.1, Appendix C.4 and Appendix C.3, respectively. We compare TTD with representative test-time intervention baselines under the same evaluation protocol. The implementation details of baseline methods are elaborated in Appendix C.5. A comprehensive hyperparameter study is presented in Appendix D.

4.1 Comparison with Baselines

Effectiveness on multimodal reasoning. Table 2 shows that TTD achieves the highest accuracy on all eight multimodal mathematical and general reasoning benchmarks, improving accuracy over the base model up to 4.7% while reducing average response length by approximately 38%. The results suggest that selective thinking-token deletion can improve final-answer accuracy while making responses more concise. The oracle results provide an empirical upper bound for analysis, highlighting the potential for further gains with more accurate deletion signals. Although Random Deletion can achieve the same length reduction, it yields only a marginal improvement in average accuracy and consistently underperforms TTD, indicating that the gains of TTD depend on which thinking tokens are removed rather than on context shortening alone, since naive deletion can remove useful reasoning content. MTI (Yang et al., 2025c) provides modest gains on some benchmarks but does not consistently reduce response length. TTI (Shen et al., 2025) and s1 (Muennighoff et al., 2025) substantially increase response length, while yielding little accuracy improvement. Further inspection shows that answer-revision behavior can sometimes turn initially correct answers into incorrect predictions, suggesting repeated reconsideration may introduce additional errors.

Generalization to tool-augmented and text-only reasoning. We further evaluate whether TTD generalizes to broader reasoning settings. For tool-augmented multimodal reasoning, we apply TTD to Thyme (Zhang et al., 2026), which supports thinking with images through a code-as-tool interface. For pure-text reasoning, we use a text-only LLM Qwen2.5-7B (Yang et al., 2025a) as the base model. As shown in Tables 6 and Table 5 of Appendix E.2, TTD consistently improves average accuracy over the base models while reducing average response length in both settings. Detailed results and experimental settings are provided in Appendix E.2. The above results indicate that selectively deleting tokens from completed reasoning traces, including those produced through tool interactions, offers an effective direction for test-time intervention beyond steering token generation, extending reasoning, or prompting revision.

Architecture-scale generalization. To evaluate whether TTD transfers across model families and scales, we compare it with the test-time intervention baselines using Qwen2.5-VL-3B (Bai et al., 2025b), ThinkLite-VL-7B (Wang et al., 2025c), and MM-Eureka-7B (Meng et al., 2025). As shown

in Table 7, Table 8, and Table 9 of Appendix E.3, TTD achieves the highest average accuracy in both benchmark groups for all three backbones, while reducing average response length. Furthermore, evaluations on LLaMA3.2-Vision-11B (Grattafiori et al., 2024), ARES-7B (Chen et al., 2026), and Qwen3-VL-4B-Thinking (Bai et al., 2025a) in Table 10 of Appendix E.3 show consistent average improvements over the corresponding base models. The consistent advantage of TTD supports that test-time thinking-token deletion is not tied to a specific MLLM architecture or scale, and can also benefit smaller and stronger multimodal reasoners. Detailed results and analyses are provided in Appendix E.3.

4.2 More Analysis

Deletion ratio analysis. We further analyze entropy-guided test-time token deletion by sweeping the deletion ratio $r \in [0, 1]$. Instead of applying a global entropy threshold, we determine an instance-specific threshold so that the top-r fraction of high-entropy thinking tokens is removed for each example. As shown in Figure 4, entropy-guided deletion achieves higher accuracy than both low-entropy deletion and random deletion under moderate deletion ratios. This indicates that the improvement is not merely caused by shortening the reasoning trace, but depends on which tokens are removed. Additionally, deletion is not universally beneficial. When the deletion ratio becomes too large, especially beyond 30%, all methods begin to disrupt the conditioning trace and lead to a sharp accuracy drop.

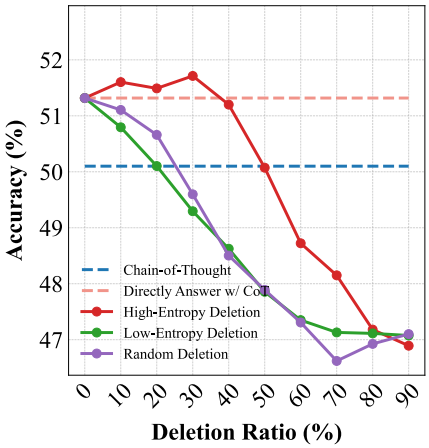


Figure 4: Deletion ratio analysis. Average accuracy under entropy-guided deletion with different deletion ratios.

Difficulty-controlled analysis. To examine the confounding factor of problem difficulty, we further conduct difficulty-controlled analysis. The results in Table 11 of Appendix F.1 demonstrate the entropy gap persists within each difficulty group, and TTD improves accuracy across different difficulty levels, confirming that the benefit does not merely coincide with difficult examples.

Design choice analysis. We further investigate whether TTD benefits from the proposed masking mechanism and entropy-minimization objective. As shown in Table 13 of Appendix F.3, using the complement mask, which keeps only the tokens that TTD would delete, reduces the average accuracy by 8.03 points. This indicates that the tokens retained by TTD are more effective conditioning context for final-answer decoding than those removed. Moreover, reversing the mask-learning objective with entropy maximization degrades the average accuracy by 9.84 points, validating the proposed design.

Computational analysis. We analyze the computational overhead of TTD, with detailed results provided in Appendix F.2. The training-free entropy-guided variant only requires one additional forward pass after deleting selected thinking tokens, taking around 0.125s per example on an NVIDIA A6000 GPU. The test-time mask-learning variant with SPSA optimization iterations takes around 20s per example. We refer to the detailed discussion in Appendix F.2.

Comparison with model-based test-time adaptation. As shown in Table 14 in Appendix F.4, the base model with TTD outperforms MM-UPT (Wei et al., 2025), an unsupervised RL post-training method, on all four mathematical benchmarks, without updating model parameters. And MM-UPT incurs an estimated cost of 60.48 seconds per sample. Besides, applying TTD to the MM-UPT-adapted model further improves reasoning performance, indicating the complementarity of model adaptation and reasoning-context intervention.

Complementarity with sampling-based test-time scaling. We further evaluate whether TTD can be combined with self-consistency (Wang et al., 2022). As shown in Figure 10 of Appendix F.5, applying entropy-guided TTD to each sampled reasoning trace before answer aggregation improves performance under the same sampling budget. The results suggest that thinking-token deletion provides a complementary axis to sampling-based test-time scaling.

5 Related Work

We discuss the most relevant line of work here and defer broader comparisons to Appendix B.

Multimodal reasoning models. Motivated by the recent success of reasoning-oriented language models (OpenAI, 2025; Guo et al., 2025), Multimodal Large Language Models (MLLMs) have increasingly moved from general vision-language understanding (Liu et al., 2024b;a; McKinzie et al., 2025; Zhang et al., 2024a) toward deliberate reasoning over images and text (Li et al., 2025a). A growing body of work improves multimodal reasoning by eliciting explicit reasoning traces or by training models with supervised fine-tuning (SFT), reinforcement learning-based post-training, or their combinations (Xu et al., 2025; Deng et al., 2025; Xiao & Gan, 2025; Huang et al., 2025; Li et al., 2025c; Yang et al., 2025b). We ground our proposed method on these multimodal reasoning models and study a complementary direction. We intervene at test time on the reasoning trace that has already been generated rather than updating model parameters or training models to generate better reasoning traces. TTD selectively deletes generated thinking tokens and re-decodes the final answer conditioned on the remaining trace, providing an inference-time mechanism for improving multimodal mathematical reasoning.

Test-time scaling and efficient chain-of-thought. Test-time scaling improves model performance by allocating additional computation at inference without changing the model architecture (Welleck et al., 2024; Li et al., 2025b). Existing methods can be broadly grouped into parallel inference, where multiple samples are generated and aggregated (Wang et al., 2022; Brown et al., 2024; Irvine et al., 2023; Toh et al., 2024), or sequential inference, where models iteratively refine, extend, or verify their reasoning (Hou et al., 2025; Lee et al., 2025; Zelikman et al., 2022; Wang et al., 2023; Muennighoff et al., 2025; Ye et al., 2025). Some methods further combine parallel exploration and sequential refinement through tree search (Wu et al., 2024b) or reward-guided inference (Snell et al., 2025; Chu et al., 2025). Another related line studies efficient Chain-of-Thought (CoT), which aims to reduce the cost of long reasoning traces by skipping (Xia et al., 2025), compressing (Zhang et al., 2025), or distilling intermediate reasoning steps into latent tokens (Goyal et al., 2023a; Deng et al., 2024; Hao et al., 2024). Most of these approaches aim to match the performance of full CoT while lowering computation or verbosity. TTD differs from these approaches in both mechanism and goal. Instead of generating more trajectories or compressing CoT primarily for efficiency, TTD edits a completed reasoning trace at test time by deletion and uses the remaining trace to improve final-answer decoding. The reduction in response length is a useful side effect, while the main goal is to improve answer accuracy through selective thinking-token deletion.

6 Discussions and Conclusions

Our results suggest that generated reasoning tokens are useful but not uniformly beneficial for final-answer decoding. The correlation between thinking-token entropy and answer correctness, together with the gains from selective deletion, indicates that some tokens may become less useful, or even distracting, when the reasoning trace serves as conditioning context for decoding the answer. We do not claim that high-entropy tokens are universally irrelevant. Some high-entropy tokens may be critical during reasoning generation, for example, as branching points, while being less useful as conditioning context for predicting the final answer. The proposed perspective also connects to context management in agentic systems (Zhou et al., 2025). In long-horizon settings, an agent may iteratively plan, retrieve information, write memory, call tools, and accumulate long histories of states (Gao et al., 2025). Such accumulated context contains useful evidence, but it may also include redundant or misleading residues. Although our experiments focus primarily on multimodal mathematical reasoning, thinking-token deletion may be useful for managing generated context, which we leave for future work.

In conclusion, we propose TTD, a first-generate-then-delete framework for test-time thinking-token deletion. TTD includes a training-free entropy-guided deletion rule and a forward-only mask-learning variant, both of which keep the base model frozen. Across multimodal reasoning benchmarks, TTD improves reasoning accuracy while reducing response length, and further transfers across model families and scales. Additional results on multimodal tool-use and pure-text reasoning tasks suggest broader applicability. Overall, our findings show selectively editing reasoning traces by deletion provides a complementary axis of test-time scaling beyond simply generating more tokens.

AI USE STATEMENT

In this work, we used Large Language Models (LLMs) to polish writing, focusing on grammar, clarity, and readability. All AI-assisted text was reviewed by the authors. The authors take full responsibility for the final content of this work, including any text produced with AI assistance.

REFERENCES

Jameel Abdul Samadh, Mohammad Hanan Gani, Noor Hussein, Muhammad Uzair Khattak, Muhammad Muzammal Naseer, Fahad Shahbaz Khan, and Salman H Khan. Align your prompts: Test-time prompting with distribution alignment for zero-shot generalization. *Advances in Neural Information Processing Systems*, 2023.

Sotiris Anagnostidis, Dario Pavllo, Luca Biggio, Lorenzo Noci, Aurelien Lucchi, and Thomas Hofmann. Dynamic context pruning for efficient and interpretable autoregressive transformers. *Advances in Neural Information Processing Systems*, 36:65202–65223, 2023.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025b.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 295–305, 2022.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*, 2025.

Shuang Chen, Hangyu Guo, Yimeng Ye, Shijue Huang, Wenbo Hu, Jiayu Chen, Manyuan Zhang, Haoxi Li, Song Guo, and Nanyun Violet Peng. Ares: Multimodal adaptive reasoning via difficulty-aware token-level entropy shaping. In *International Conference on Learning Representations*, volume 2026, pp. 147474–147512, 2026.

Yilong Chen, Guoxia Wang, Junyuan Shang, Shiyao Cui, Zhenyu Zhang, Tingwen Liu, Shuohuan Wang, Yu Sun, Dianhai Yu, and Hua Wu. Nacl: A general and effective kv cache eviction framework for llms at inference time. *arXiv preprint arXiv:2408.03675*, 2024.

Rewon Child. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 719–729, 2024.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can gpt learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Association for Computational Linguistics*, 2023.

Yihe Deng, Hritik Bansal, Fan Yin, Nanyun Peng, Wei Wang, and Kai-Wei Chang. Openvlthinker: Complex vision-language reasoning via iterative sft-rl cycles. In *Advances in Neural Information Processing Systems*, 2025.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. *arXiv preprint arXiv:2508.15260*, 2025.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. *arXiv preprint arXiv:2310.02226*, 2023a.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. *arXiv preprint arXiv:2310.02226*, 2023b.

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

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

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

hiyouga. Mathruler. `https://github.com/hiyouga/MathRuler`, 2025.

Zhenyu Hou, Xin Lv, Rui Lu, Jiajie Zhang, Yujiang Li, Zijun Yao, Juanzi Li, Jie Tang, and Yuxiao Dong. Advancing language model reasoning through reinforcement learning and inference scaling. *arXiv preprint arXiv:2501.11651*, 2025.

Jian Hu. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*, 2025.

Jinwu Hu, Zhitian Zhang, Guohao Chen, Xutao Wen, Chao Shuai, Wei Luo, Bin Xiao, Yuanqing Li, and Mingkui Tan. Test-time learning for large language models. *arXiv preprint arXiv:2505.20633*, 2025.

Chen Huang, Wei Lu, and Wenxuan Zhang. Pear: Phase entropy aware reward for efficient reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 31141–31159, 2026.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025.

Robert Irvine, Douglas Boubert, Vyas Raina, Adian Liusie, Ziyi Zhu, Vineet Mudupalli, Aliaksei Korshuk, Zongyi Liu, Fritz Cremer, Valentin Assassi, et al. Rewarding chatbots for real-world engagement with millions of users. *arXiv preprint arXiv:2303.06135*, 2023.

Simon Jegou and Maximilian Jeblick. Kvzap: Fast, adaptive, and faithful kv cache pruning. *arXiv preprint arXiv:2601.07891*, 2026.

Gangwei Jiang, Yahui Liu, Zhaoyi Li, Wei Bi, Fuzheng Zhang, Linqi Song, Ying Wei, and Defu Lian. What makes a good reasoning chain? uncovering structural patterns in long chain-of-thought reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6501–6525, 2025.

Muhammad Uzair Khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. Maple: Multi-modal prompt learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Eunki Kim, Sangryul Kim, and James Thorne. Learning to insert [pause] tokens for better reasoning. In *Association for Computational Linguistics*, 2025a.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. KVzip: Query-agnostic KV cache compression with context reconstruction. In *Advances in Neural Information Processing Systems*, 2025b.

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

Jonghyun Lee, Dahuin Jung, Saehyung Lee, Junsung Park, Juhyeon Shin, Uiwon Hwang, and Sungroh Yoon. Entropy is not enough for test-time adaptation: From the perspective of disentangled factors. *arXiv preprint arXiv:2403.07366*, 2024.

Kuang-Huei Lee, Ian Fischer, Yueh-Hua Wu, Dave Marwood, Shumeet Baluja, Dale Schuurmans, and Xinyun Chen. Evolving deeper llm thinking. *arXiv preprint arXiv:2501.09891*, 2025.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

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*, 2024.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Xianzhi Li, Ethan Callanan, Abdellah Ghassel, and Xiaodan Zhu. Entropy-gated branching for efficient test-time reasoning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5054–5069, 2026.

Yunxin Li, Zhenyu Liu, Zitao Li, Xuanyu Zhang, Zhenran Xu, Xinyu Chen, Haoyuan Shi, Shenyuan Jiang, Xintong Wang, Jifang Wang, et al. Perception, reason, think, and plan: A survey on large multimodal reasoning models. *arXiv preprint arXiv:2505.04921*, 2025a.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, et al. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*, 2025b.

Zongxia Li, Wenhao Yu, Chengsong Huang, Rui Liu, Zhenwen Liang, Fuxiao Liu, Jingxi Che, Dian Yu, Jordan Boyd-Graber, Haitao Mi, et al. Self-rewarding vision-language model via reasoning decomposition. *arXiv preprint arXiv:2508.19652*, 2025c.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, Weizhu Chen, et al. Not all tokens are what you need for pretraining. In *Advances in Neural Information Processing Systems*, 2024.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26296–26306, 2024a.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, 2024b.

Yexiang Liu, Zekun Li, Zhi Fang, Nan Xu, Ran He, and Tieniu Tan. Rethinking the role of prompting strategies in llm test-time scaling: A perspective of probability theory. *arXiv preprint arXiv:2505.10981*, 2025.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? *arXiv preprint arXiv:2307.06281*, 2023.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12):220102, 2024c.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Advances in Neural Information Processing Systems*, 2022.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations*, 2024.

Xiaosong Ma, Jie Zhang, Song Guo, and Wenchao Xu. Swapprompt: Test-time prompt adaptation for vision-language models. *Advances in Neural Information Processing Systems*, 36:65252–65264, 2023.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075, 2023.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics*, pp. 2263–2279, 2022.

Brandon McKinzie, Zhe Gan, Jean-Philippe Fauconnier, Sam Dodge, Bowen Zhang, Philipp Dufter, Dhruti Shah, Xianzhi Du, Futang Peng, Anton Belyi, et al. Mm1: methods, analysis and insights from multimodal llm pre-training. In *European Conference on Computer Vision*, pp. 304–323. Springer, 2025.

Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Tiancheng Han, Botian Shi, Wenhai Wang, Junjun He, et al. Mm-eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2503.07365*, 2025.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*, 2024.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

Jingcheng Niu, Xingdi Yuan, Tong Wang, Hamidreza Saghir, and Amir H Abdi. Llama see, llama do: A mechanistic perspective on contextual entrainment and distraction in llms. *arXiv preprint arXiv:2505.09338*, 2025.

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, 2022.

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

OpenAI. Introducing openai o3 and o4-mini, 2025. URL https://openai.com/index/introducing-o3-and-o4-mini/.

Chen Qian, Dongrui Liu, Haochen Wen, Zhen Bai, Yong Liu, and Jing Shao. Demystifying reasoning dynamics with mutual information: Thinking tokens are information peaks in LLM reasoning. In *Advances in Neural Information Processing Systems*, 2026.

Runqi Qiao, Qiuna Tan, Guanting Dong, MinhuiWu MinhuiWu, Chong Sun, Xiaoshuai Song, Jiapeng Wang, Zhuoma Gongque, Shanglin Lei, Yifan Zhang, et al. We-math: Does your large multimodal model achieve human-like mathematical reasoning? In *Association for Computational Linguistics*, pp. 20023–20070, 2025.

Qwen Team. QwQ: Reflect deeply on the boundaries of the unknown, November 2024. URL https://qwenlm.github.io/blog/qwq-32b-preview/.

Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL https://qwenlm.github.io/blog/qwq-32b/.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763, 2021.

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

Shuvendu Roy and Ali Etemad. Consistency-guided prompt learning for vision-language models. *arXiv preprint arXiv:2306.01195*, 2023.

Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. A-okvqa: A benchmark for visual question answering using world knowledge. In *European Conference on Computer Vision*, 2022.

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.

Aman Sharma and Paras Chopra. Think just enough: Sequence-level entropy as a confidence signal for llm reasoning. *arXiv preprint arXiv:2510.08146*, 2025.

Junhong Shen, Hao Bai, Lunjun Zhang, Yifei Zhou, Amrith Setlur, Shengbang Tong, Diego Caples, Nan Jiang, Tong Zhang, Ameet Talwalkar, et al. Thinking vs. doing: Agents that reason by scaling test-time interaction. *arXiv preprint arXiv:2506.07976*, 2025.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pp. 31210–31227, 2023.

Manli Shu, Weili Nie, De-An Huang, Zhiding Yu, Tom Goldstein, Anima Anandkumar, and Chaowei Xiao. Test-time prompt tuning for zero-shot generalization in vision-language models. *Advances in Neural Information Processing Systems*, 35:14274–14289, 2022.

Charlie Victor 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*, 2025.

James C Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, 37(3):332–341, 2002.

Vernon YH Toh, Deepanway Ghosal, and Soujanya Poria. Not all votes count! programs as verifiers improve self-consistency of language models for math reasoning. *arXiv preprint arXiv:2410.12608*, 2024.

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.

Guowei Wang, Fan Lyu, and Changxing Ding. Partition-then-adapt: Combating prediction bias for reliable multi-modal test-time adaptation. *Advances in Neural Information Processing Systems*, 2025a.

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset. *Advances in Neural Information Processing Systems*, 2024.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2023.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. In *Advances in Neural Information Processing Systems*, 2026.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pp. 7907–7915, 2025b.

Xiyao Wang, Zhengyuan Yang, Chao Feng, Hongjin Lu, Linjie Li, Chung-Ching Lin, Kevin Lin, Furong Huang, and Lijuan Wang. Sota with less: Mcts-guided sample selection for data-efficient visual reasoning self-improvement. *arXiv preprint arXiv:2504.07934*, 2025c.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

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.

Lai Wei, Yuting Li, Chen Wang, Yue Wang, Linghe Kong, Weiran Huang, and Lichao Sun. Unsupervised post-training for multi-modal llm reasoning via grpo. *Advances in Neural Information Processing Systems*, 2025.

Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models. *arXiv preprint arXiv:2406.16838*, 2024.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13084–13094, 2024.

Siye Wu, Jian Xie, Jiangjie Chen, Tinghui Zhu, Kai Zhang, and Yanghua Xiao. How easily do irrelevant inputs skew the responses of large language models? *arXiv preprint arXiv:2404.03302*, 2024a.

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*, 2024b.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*, 2025.

Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, et al. Towards system 2 reasoning in llms: Learning how to think with meta chain-of-thought. *arXiv preprint arXiv:2501.04682*, 2025.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

Wenyi Xiao and Leilei Gan. Fast-slow thinking grpo for large vision-language model reasoning. In *Advances in Neural Information Processing Systems*, 2025.

Zehao Xiao, Shilin Yan, Jack Hong, Jiayin Cai, Xiaolong Jiang, Yao Hu, Jiayi Shen, Qi Wang, and Cees GM Snoek. Dynaprompt: Dynamic test-time prompt tuning. *arXiv preprint arXiv:2501.16404*, 2025.

Guowei Xu, Peng Jin, Ziang Wu, Hao Li, Yibing Song, Lichao Sun, and Li Yuan. Llava-cot: Let vision language models reason step-by-step. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2087–2098, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025a.

Jiaxi Yang, Binyuan Hui, Min Yang, Bailin Wang, Bowen Li, Binhua Li, Fei Huang, and Yongbin Li. Iterative forward tuning boosts in-context learning in language models. In *Association for Computational Linguistics*, 2024.

Yi Yang, Xiaoxuan He, Hongkun Pan, Xiyan Jiang, Yan Deng, Xingtao Yang, Haoyu Lu, Dacheng Yin, Fengyun Rao, Minfeng Zhu, et al. R1-onevision: Advancing generalized multimodal reasoning through cross-modal formalization. *arXiv preprint arXiv:2503.10615*, 2025b.

Zhen Yang, Mingyang Zhang, Feng Chen, Ganggui Ding, Liang Hou, Xin Tao, and Ying-Cong Chen. Less is more: Improving llm reasoning with minimal test-time intervention. *arXiv preprint arXiv:2510.13940*, 2025c.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.

Hee Suk Yoon, Eunseop Yoon, Joshua Tian Jin Tee, Mark Hasegawa-Johnson, Yingzhen Li, and Chang D Yoo. C-tpt: Calibrated test-time prompt tuning for vision-language models via text feature dispersion. *arXiv preprint arXiv:2403.14119*, 2024.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. Making retrieval-augmented language models robust to irrelevant context. *arXiv preprint arXiv:2310.01558*, 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, 2025.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9556–9567, 2024.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *Advances in Neural Information Processing Systems*, 2025.

Maxime Zanella and Ismail Ben Ayed. On the test-time zero-shot generalization of vision-language models: Do we really need prompt learning? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Zihao Zeng, Bokai Lin, Tianqi Hou, Hao Zhang, and Zhijie Deng. In-context kv-cache eviction for llms via attention-gate. *arXiv preprint arXiv:2410.12876*, 2024.

Haotian Zhang, Mingfei Gao, Zhe Gan, Philipp Dufter, Nina Wenzel, Forrest Huang, Dhruti Shah, Xianzhi Du, Bowen Zhang, Yanghao Li, et al. Mm1.5: Methods, analysis & insights from multimodal llm fine-tuning. *arXiv preprint arXiv:2409.20566*, 2024a.

Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. Lightthinker: Thinking step-by-step compression. *arXiv preprint arXiv:2502.15589*, 2025.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, et al. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? In *European Conference on Computer Vision*, pp. 169–186, 2024b.

YiFan Zhang, Xingyu Lu, Shukang Yin, Chaoyou Fu, Wei Chen, Xiao Hu, Bin Wen, Kaiyu Jiang, Changyi Liu, Tianke Zhang, Haonan fan, Kaibing Chen, Jiankang Chen, Haojie Ding, Kaiyu Tang, Zhang Zhang, Liang Wang, Fan Yang, Tingting Gao, and Guorui Zhou. Thyme: Think beyond images. In *International Conference on Learning Representations*, 2026.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16816–16825, 2022a.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022b.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*, 2025.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, et al. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

First Generate, Then Delete: Test-Time Thinking-Token Deletion for Multimodal Reasoning

-Supplementary Material-

A Broader Impacts

This paper proposes a test-time reasoning intervention framework for multimodal large language models. By selectively removing intermediate reasoning tokens, our method aims to improve inference-time reasoning accuracy while reducing the effective context length. This may contribute to better multimodal AI systems, especially in applications that require complex visual reasoning.

B Additional Related Work

Entropy-guided post-training and reasoning. Prior work has explored the use of entropy to guide model optimization and reasoning generation. During RL post-training, entropy-related mechanisms are used to maintain exploration (Yu et al., 2025; Cui et al., 2025), select tokens for policy-gradient updates (Wang et al., 2026), and shape reasoning behavior through reward design (Chen et al., 2026; Huang et al., 2026). Specifically, strategies for mitigating entropy collapse include relaxing the upper clipping bound (Yu et al., 2025) and constraining policy updates at high-covariance tokens (Cui et al., 2025) Other work identifies high-entropy tokens as reasoning forks and concentrates policy-gradient updates on these tokens (Wang et al., 2026). ARES (Chen et al., 2026) and PEAR (Huang et al., 2026) further balance exploration and exploitation for adaptive reasoning behaviors by incorporating difficulty-aware and phase-specific entropy rewards, respectively. At inference time, another line of work uses entropy as an uncertainty signal to allocate computation, either by selectively expanding reasoning branches at high-uncertainty positions (Li et al., 2026) or by triggering early stopping based on a sequence-level entropy (Sharma & Chopra, 2025). These policy-optimization and generation-control mechanisms primarily regulate how models learn to reason or generate reasoning trajectories. TTD instead focuses on the utility of already-generated thinking tokens as conditioning context for final-answer decoding and edits this context through selective token deletion at test time. Motivated by our leave-one-out deletion analysis, we instantiate this post-generation intervention with both entropy-guided deletion and instance-adaptive test-time mask learning, while keeping the base model frozen. Rather than treating entropy itself as a new confidence measure, we distinguish a token's role during reasoning generation from its utility when the completed trace is reused for final-answer decoding.

Distraction. Prior work has shown that LLMs can be distracted by contextual information in prompts or in-context examples (Yoran et al., 2023; Shi et al., 2023; Niu et al., 2025). For example, models may fail on questions they otherwise answer correctly when provided with additional in-context examples or with irrelevant and misleading information (Shi et al., 2023; Mirzadeh et al., 2024). Besides, distraction and interference effects have also been reported in retrieval-augmented generation (RAG) and information retrieval settings, where retrieved passages can introduce noise that degrades reasoning (Yoran et al., 2023; Wu et al., 2024a; Cuconasu et al., 2024). These studies mainly focus on externally provided context. In contrast, our work studies the context generated by the model itself. We ask whether some tokens in a completed reasoning trace can be less useful, or even distracting, when served as conditioning context for final-answer decoding.

Pruning-based efficient inference. A large body of work reduces the memory and computational cost of long-context decoding through token pruning, sparse attention, or KV-cache compression (Child, 2019; Zhang et al., 2023; Xiao et al., 2023; Chen et al., 2024; Zeng et al., 2024). Since KV-cache memory grows with sequence length, several training-free methods are proposed to retain informative tokens with high accumulated attention or other importance scores (Zhang et al., 2023; Xiao et al., 2023; Chen et al., 2024; Kim et al., 2025b; Jegou & Jeblick, 2026). Other methods learn token dropping or gating policies with additional training or fine-tuning stage (Anagnostidis et al., 2023; Zeng et al., 2024). TTD shares the broad idea of token deletion, but differs in motivation and implementation. Rather than accelerating decoding or reducing KV-cache memory, we target improvements in reasoning performance, motivated by our leave-one-out deletion analysis. To this

end, TTD edits generated thinking tokens via entropy-guided deletion or test-time mask learning to improve final-answer decoding, while keeping the base model frozen.

Prompt optimization. Prompt optimization adapts pre-trained language (Liu et al., 2019) or vision-language models (Radford et al., 2021) on downstream tasks by learning discrete or continuous prompts (Lester et al., 2021; Li & Liang, 2021; Zhou et al., 2022b; Roy & Etemad, 2023; Khattak et al., 2023). The learnable prompts are often prepended or inserted into the input embedding space to improve downstream transfer (Zhou et al., 2022a). Recent work further explores test-time prompt tuning on unlabeled data, with TPT (Shu et al., 2022) establishing a common paradigm of multi-view augmentation and entropy minimization. Later methods improve this paradigm using calibration (Yoon et al., 2024), contrastive (Ma et al., 2023), alignment (Abdul Samadh et al., 2023), ensemble (Zanella & Ben Ayed, 2024), or dynamic buffer (Xiao et al., 2025) strategies. TTD is related in spirit because it adapts the conditioning context at inference time. However, instead of adding or tuning prompt tokens, TTD removes selected thinking tokens from the model's own generated reasoning trace.

Test-time adaptation and training. Test-time adaptation aims to adapt a pre-trained model to a target distribution using online test data without ground-truth labels (Wang et al., 2020; Lee et al., 2024; Wang et al., 2025a; Hu et al., 2025). Prior work addresses challenges such as noisy pseudo-labels, biased entropy signals, and catastrophic forgetting via pseudo-label refinement (Chen et al., 2022), sample reweighting or selective updates (Niu et al., 2022; Lee et al., 2024), and filtering criteria such as input perplexity (Hu et al., 2025). Recent studies also explore test-time training on unlabeled mathematical reasoning tasks with reinforcement-learning objectives (e.g., GRPO) and majority-vote supervision (Zuo et al., 2025; Wei et al., 2025). Unlike these methods, TTD does not update model parameters, and test-time optimization is performed only over a binary mask applied to generated intermediate thinking tokens.

C More Detailed Experimental Setup

C.1 Implementation Details

We use Qwen2.5-VL-7B (Bai et al., 2025b) as the base MLLM for all main experiments. The Chain-of-Thought and Directly Answer prompt templates are provided in Figure 5 and Figure 6 of Appendix C.4. For the test-time mask learning variant, we adopt the MeZO (Malladi et al., 2023) implementation of SPSA for memory-efficient forward-only optimization. We set the objective weight to $\lambda = 1$, the perturbation scale to $\epsilon = 0.3$, the step size to $\eta = 0.2$, the number of optimization iterations to $T = 80$, and the chunked shuffle window size to $w = 5$. We conduct a comprehensive hyperparameter study in Appendix D.

C.2 Evaluation Benchmarks

In our main experiments, we evaluate TTD on four multimodal mathematical reasoning benchmarks, comprising MathVision (Wang et al., 2024), MathVerse (Zhang et al., 2024b), MathVista (Lu et al., 2024), and We-Math (Qiao et al., 2025), and four general multimodal understanding and reasoning benchmarks, including MMMU (Yue et al., 2024), ScienceQA (Lu et al., 2022), A-OKVQA (Schwenk et al., 2022), and MMBench (Liu et al., 2023). The multimodal mathematical benchmarks are designed to assess whether MLLMs can jointly interpret visual information, such as diagrams, charts, tables, and geometric figures, and perform mathematical reasoning under both multiple-choice and free-form answer settings. Specifically, MathVision consists of competition-level mathematical problems with visual contexts and covers a broad range of mathematical disciplines and difficulty levels. MathVerse focuses on visual mathematical reasoning with different degrees of visual and textual dependency, making it suitable for evaluating whether a model truly leverages diagrams rather than relying only on textual cues. MathVista combines diverse mathematical and visual tasks, including geometry, algebra, chart understanding, logic puzzles, and scientific figure reasoning. We-Math further decomposes visual mathematical reasoning into hierarchical knowledge concepts and multiple reasoning granularities, which enables a more fine-grained analysis of model performance. MMMU evaluates college-level multidisciplinary reasoning over heterogeneous visual inputs, covering domains such as science, engineering, medicine, business, humanities, and art. ScienceQA focuses on elementary- and high-school science question answering and requires models

Table 3: Summary of evaluation benchmarks used in our experiments. The reported sample counts correspond to the specific evaluation split.

Benchmark	Evaluation Type	# Eval. Samples
MathVision (Wang et al., 2024)	Multimodal Math	3,040
MathVerse (Zhang et al., 2024b)	Multimodal Math	3,940
MathVista (Lu et al., 2024)	Multimodal Math	1,000
We-Math (Qiao et al., 2025)	Multimodal Math	1,740
MMMU (Yue et al., 2024)	Multimodal General	900
ScienceQA (Lu et al., 2022)	Multimodal Science	2,017
A-OKVQA (Schwenk et al., 2022)	Multimodal Commonsense	1,145
MMBench (Liu et al., 2023)	Multimodal General	4,329
V^* (Wu & Xie, 2024)	Multimodal Tool-Use	191
ChartQA (Masry et al., 2022)	Multimodal Tool-Use	2,500
HRBench-4K (Wang et al., 2025b)	Multimodal Tool-Use	800
OCRBench (Liu et al., 2024c)	Multimodal Tool-Use	1,000
AIME (Li et al., 2024)	Text-only Math	30
AMC (Li et al., 2024)	Text-only Math	83
MATH500 (Hendrycks et al., 2021)	Text-only Math	500
GPQA-Diamond (Rein et al., 2024)	Text-only Science	198

to combine textual, visual, and scientific knowledge. A-OKVQA evaluates visual question answering that requires commonsense and world knowledge beyond the image content. MMBench provides a broad evaluation of multimodal perception and reasoning abilities through multiple-choice questions across fine-grained ability dimensions.

To assess generalization to tool-augmented multimodal reasoning, we additionally evaluate TTD using Thyme (Zhang et al., 2026), a framework that enables *thinking with images* through a *code-as-tool interface*. Specifically, Thyme interleaves textual reasoning with Python code execution for image manipulation, such as cropping, zooming, rotation, and contrast enhancement, as well as numerical computation. The generated code is executed in a sandboxed environment, and the resulting feedback guides subsequent reasoning, allowing the model to actively inspect and transform visual inputs during problem solving. We conduct these experiments on four benchmarks: V^* (Wu & Xie, 2024), ChartQA (Masry et al., 2022), HRBench-4K (Wang et al., 2025b), and OCRBench (Liu et al., 2024c), covering high-resolution visual perception, chart understanding, and text reading. Specifically, V^* evaluates fine-grained visual recognition and spatial relationship reasoning in high-resolution images. ChartQA assesses question answering about charts through visual and logical reasoning. HRBench-4K evaluates fine-grained single-instance perception (FSP) and fine-grained cross-instance perception (FCP) using 4K-resolution images. Each question is evaluated with four answer-option orderings. OCRBench assesses text recognition and text-centric visual understanding.

Finally, we study the generalization of TTD on four challenging pure-text reasoning tasks, including AIME 2024 (Li et al., 2024), AMC (Li et al., 2024), MATH500 (Hendrycks et al., 2021), and GPQA (Rein et al., 2024). AIME and AMC evaluate mathematical problem solving at the competition level, while MATH-500 measures general mathematical reasoning over challenging problems sampled from the MATH benchmark. GPQA evaluates graduate-level scientific reasoning in biology, physics, and chemistry. A summary of all benchmarks used in our work is presented in Table 3.

C.3 Evaluation Metric

For all benchmarks, we report accuracy (pass@1) and response length. We prompt models to place the final answer in a designated boxed format, and then use a stronger judge model, Qwen2.5-32B-Instruct (Bai et al., 2025b), to assess correctness by comparing the extracted answer with the ground truth, which often includes complex mathematical expressions. The evaluation prompt template is provided in Figure 8 of Appendix C.4.

C.4 Prompt Templates

We provide the prompt templates used throughout our experiments. The default step-by-step Chain-of-Thought (CoT) prompting template is shown in Figure 5. The prompt template for Direct Answer conditioned on previously generated reasoning tokens is shown in Figure 6. For inference-time check-again verification, we adopt the prompt from TTI (Shen et al., 2025), as shown in Figure 7. For answer evaluation, we follow prior work and use the standard evaluation template in Figure 8. To ensure fair comparisons, we use the same evaluation prompts for all base models.

Chain-of-Thought Prompt

system
Please reason step by step, and put your final answer within \boxed{}.

user
{question}

assistant

Figure 5: Prompt template for step-by-step Chain-of-Thought (CoT) generation..

Direct-Answer Prompt

system
Output only the final answer. Do not include any reasoning, steps, explanations, or additional text. Put your final answer within \boxed{}.

user
{question}
{generated thinking tokens}

assistant

Figure 6: Prompt template for Direct Answer conditioned on previously generated reasoning tokens

Check-Again Prompt

user
{question}
{generated thinking tokens}{generated answer}
You returned an answer in the last step. Let's pause, check the question, and think again. If you still think the task is finished, double-check your answer, revise it if need, and return a final answer. If not, continue the task. You should still put your final answer within \boxed{}.

assistant

Figure 7: Prompt template for inference-time check-again verification used in TTI (Shen et al., 2025).

C.5 Baseline Implementation Details

In this section, we provide a more detailed explanation of the baseline methods used in our experiments. All baselines use the same base model, initial Chain-of-Thought prompt, decoding setting, answer extraction rule, and evaluation protocol as TTD, unless otherwise specified.

Random Deletion serves as a simple control for testing whether the improvement comes merely from shortening the reasoning trace. We randomly remove thinking tokens with the same deletion number as TTD for each example. The tokens are sampled uniformly at random from the generated thinking trace, and the final answer is then re-decoded conditioned on the remaining trace.

MTI (Yang et al., 2025c) is a training-free test-time intervention method that uses token entropy to identify unstable generation positions. MTI inserts an additional token, [Output Error],

Evaluation Prompt

user
Your task is to judge whether the response expresses the same meaning as the answer of a question.
The question is: {question}
The answer is: {ground truth}
The response is: {model answer}
Please check and compare them and then judge. If the response is correct, your output should be Yes. Otherwise, your output should be No. Directly give me your output.

assistant

Figure 8: Prompt template for evaluating the predicted answer against the ground truth.

as a prefix cue around high-entropy positions and prompts the model to regenerate the potentially erroneous tokens. We re-implement MTI in our multimodal mathematical reasoning setting. For each evaluation benchmark, we sweep the entropy-based position selection threshold over $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$.

TTI (Shen et al., 2025) applies a check-again prompt at inference time. Given the question, the generated reasoning trace, and the previous final answer, the model is asked to reconsider its reasoning and either reaffirm or revise the answer if necessary. We re-implement TTI in the multimodal mathematical reasoning setting using the prompt template shown in Figure 7. The final answer produced after the check-again step is used for evaluation.

s1 (Muennighoff et al., 2025) uses a budget-forcing strategy that encourages the model to continue reasoning before producing the final answer by suppressing the end-of-thinking delimiter and appending a continuation cue such as "Wait" to the current reasoning trace. We adapt this strategy to our multimodal reasoning setting by appending "Wait" four times to the generated reasoning trace before asking the model to continue and produce the final answer.

D Hyper-parameter Studies of TTD

To better understand the sensitivity and design choices of TTD, we conduct controlled hyper-parameter studies using Qwen2.5-VL-7B as the base model. Unless otherwise specified, we keep the default setting used from the main experiments and vary only one factor at a time. As shown in Table 4, the variance across different random seeds is small, suggesting that the mask-learning variant is stable under different SPSA perturbation noises. Additionally, We study the number of SPSA optimization iterations. Performance increases from 69.30 at 20 iterations to 71.50 at 80 iterations, and then gradually saturates. Increasing the budget to 160 iterations yields only a marginal improvement to 71.60. Therefore, we use $T = 80$ as the default setting, which provides a good trade-off between accuracy and computational cost. We further study the perturbation scale ϵ. When ϵ is too small or too large, performance slightly decreases, suggesting that the perturbation should be large enough to explore useful mask changes but not so large that it destabilizes optimization. The best accuracy is obtained at $\epsilon = 0.3$, validating our choice. Similarly, for step size η, stable mask optimization requires a moderate update magnitude, and $\eta = 0.2$ achieves the best performance. Overall, the results support the default hyper-parameter choices used in the main experiments.

E More Results

E.1 More Leave-One-Out Deletion Results

Figure 9 provides additional results of the leave-one-out deletion analysis on four multimodal mathematical reasoning benchmarks. In this setting, we remove one generated thinking token in turn, re-decode the final answer conditioned on the remaining trace, and use ground-truth answers only to identify whether a single-token deletion can correct an initially wrong prediction. This analysis helps quantify the potential effect of token-level interventions on generated reasoning traces. The leave-one-out deletion analysis yields accuracy improvements of $+7.53$, $+6.34$, $+6.50$, and $+9.42$

Table 4: **Hyper-parameter studies of TTD.** We report Pass@1 accuracy using Qwen2.5-VL-7B on MathVista. Each experiment varies one factor while keeping the remaining hyper-parameters fixed to the default setting. The default configuration is $T = 80$, $\epsilon = 0.3$, and $\eta = 0.2$.

(a) Random seed

Seed	Accuracy
Seed 1	71.50
Seed 2	71.90
Seed 3	71.20

(b) Iterations

T	Accuracy
20	69.30
40	69.90
80	**71.50**
160	71.60

(c) Perturbation scale

ϵ	Accuracy
0.1	69.40
0.3	**71.50**
0.5	71.40
0.7	70.70

(d) Step size

η	Accuracy
0.1	70.10
0.2	**71.50**
0.3	71.10
0.4	70.60

points on MathVision, MathVerse, MathVista, and WeMath, respectively. The results show that, across different multimodal mathematical reasoning benchmarks, deleting even a single selected thinking token can sometimes make the remaining trace a more effective conditioning context for final-answer decoding. This further supports our observation that generated thinking tokens are not equally beneficial.

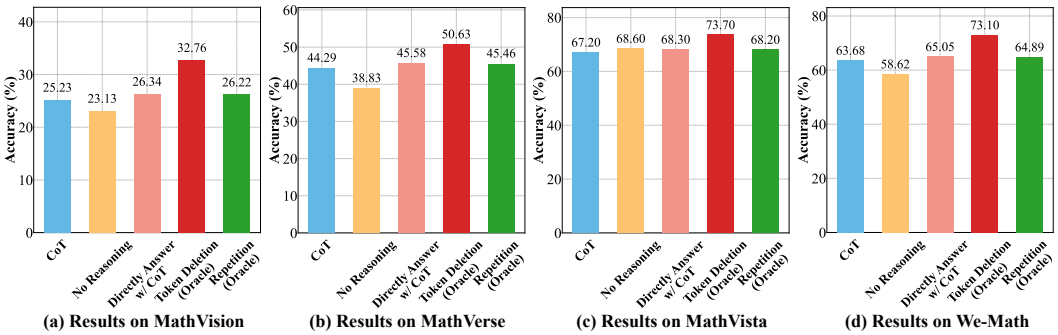


Figure 9: Additional leave-one-out deletion results on MathVision, MathVerse, MathVista, We-Math using Qwen2.5-VL-7B (Bai et al., 2025b). Oracle denotes that ground-truth labels are used only to identify successful single-token deletions for analysis.

E.2 Generalization Beyond Multimodal Mathematical Reasoning Tasks

Although our main experiments focus on multimodal mathematical reasoning, we further evaluate whether TTD transfers to broader reasoning settings. For general multimodal reasoning, we evaluate TTD on MMMU, ScienceQA, A-OKVQA, and MMBench. For pure-text reasoning, we evaluate TTD on AIME, AMC, MATH, and GPQA using Qwen2.5-7B as the base text-only LLM.

As shown in Table 2, TTD consistently improves the average performance on general multimodal reasoning tasks. Specifically, TTD improves the average Pass@1 from 75.94% to 77.52%, while reducing the average response length from 230.23 tokens to 142.33 tokens. In contrast, Random Deletion reduces response length but slightly hurts average accuracy, decreasing Pass@1 from 75.89% to 75.27%. This comparison suggests that the benefit of TTD is not simply caused by length reduction alone, but depends on selectively removing thinking tokens that are less useful for final-answer decoding.

Table 5 further evaluates TTD on pure-text reasoning tasks. Using Qwen2.5-7B (Yang et al., 2025a) as the base model, TTD improves the average Pass@1 from 44.21% to 46.52%, while reducing the average response length from 435.24 tokens to 366.16 tokens. Again, Random Deletion substantially shortens responses but degrades average accuracy to 38.89%, suggesting that naive deletion can remove useful reasoning context. The oracle variants in both tables serve as analysis-only upper bounds, where ground-truth feedback is used to identify beneficial deletions, highlighting the potential of test-time thinking-token intervention when more accurate deletion signals are available.

We evaluate TTD on Thyme (Zhang et al., 2026) across four multimodal benchmarks in Table 6 Thyme supports *thinking with images* through a*code-as-tool interface*, interleaving textual reasoning with sandboxed Python execution for image manipulation (e.g., cropping, zooming, rotation, and contrast enhancement) and numerical computation, and using execution feedback to guide subsequent reasoning.

Table 5: **Generalization to pure-text mathematical and science reasoning tasks.**

Method	AIME 2024		AMC		MATH500		GPQA		Average	
	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length
Qwen2.5-7B (Yang et al., 2025a)	20.00	665.50	43.37	432.73	77.60	284.32	35.86	358.40	44.21	435.24
Random Deletion	16.67	510.87	39.76	368.75	63.80	253.26	35.35	331.74	38.89	366.16
MTI (Yang et al., 2025c)	16.67	1205.41	43.37	855.22	77.40	600.96	36.36	806.07	43.45	866.92
TTI (Shen et al., 2025)	16.67	1027.44	44.57	643.32	77.60	456.31	35.35	651.78	43.55	694.71
s1 (Muennighoff et al., 2025)	16.67	1982.83	42.17	1204.42	76.20	1570.84	32.82	1702.58	41.97	1615.17
TTD (Oracle)	26.67	539.57	59.04	316.93	81.20	170.47	52.53	283.66	54.86	327.66
TTD	**23.33**	**510.87**	**46.99**	**368.75**	**78.40**	**253.26**	**37.37**	**331.74**	**46.52**	**366.16**

Table 6: **Generalization to tool-augmented multimodal reasoning tasks.**

Method	V*		ChartQA		HRBench-4K		OCRBench		Average	
	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length
Thyme (Zhang et al., 2026)	82.20	211.14	86.10	130.45	77.00	185.33	86.30	187.55	82.90	178.62
Random Deletion	83.24	171.62	86.48	92.06	77.62	109.73	83.40	148.87	82.69	130.57
TTD	**84.29**	**171.62**	**88.24**	**92.06**	**78.13**	**109.73**	**86.90**	**148.87**	**84.39**	**130.57**

E.3 Cross-Architecture-Scale Generalization

To examine whether the advantages of selective thinking-token deletion persist across different backbones and parameter scales, we here extend the baseline comparisons to additional multimodal backbones, including Qwen2.5-VL-3B (Bai et al., 2025b), ThinkLite-VL-7B (Wang et al., 2025c), and MM-Eureka-7B (Meng et al., 2025). All methods are evaluated on the same eight benchmarks used in our main comparison.

As shown in Table 7, TTD increases average Pass@1 over Qwen2.5-VL-3B from 39.99 to 44.93 on the mathematical benchmarks and from 69.73 to 72.67 on the general benchmarks, corresponding to gains of 4.94 and 2.94 percentage points, respectively. Interestingly, Random Deletion is the strongest competing baseline in both groups, but remains below TTD by a large margin. The other interventions provide smaller or inconsistent gains. These comparisons show that the benefits of TTD also extend to a smaller multimodal reasoner.

As shown in Table 8 and Table 9, TTD improves average reasoning performance for both base models and outperforms the strongest competing intervention in each group. Overall, these results support the advantages of TTD effectively extend beyond the backbone used in our main experiments.

F More Analyses

F.1 Difficulty-Controlled Analysis

In Figure 3, we observe that there is a strong entropy gap between correct and incorrect samples. The entropy gap between correct and incorrect traces may be confounded by problem difficulty, since harder problems can induce both higher uncertainty and lower accuracy. To examine this possibility, we conduct a difficulty-controlled analysis on MMMU, which provides difficulty annotations. As shown in Table 11, the average thinking-token entropy gap between correct and incorrect samples persists within each difficulty group. Moreover, TTD improves accuracy over the base model on easy, medium, and hard examples. These results suggest that the benefit of TTD does not merely coincide with easy or difficult examples.

F.2 Computational Analysis

We analyze the inference cost of the entropy-guided and mask-learning variants of TTD, together with their complementary roles within our first-generate-then-delete framework.

Table 7: **Comparison with test-time intervention strategies on Qwen2.5-VL-3B (Bai et al., 2025b).** We report Pass@1 performance on four multimodal mathematical reasoning benchmarks and four general multimodal reasoning benchmarks.

Method	MathVision		MathVerse		MathVista		WeMath		Average	
	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length
Base Model	19.47	748.88	33.58	485.26	56.30	301.02	50.63	346.91	39.99	470.51
Random Deletion	20.07	343.47	33.96	404.54	58.90	215.95	53.79	216.65	41.68	295.15
MTI (Yang et al., 2025c)	19.05	741.39	34.59	476.21	56.00	283.19	49.89	364.39	39.88	466.29
TTI (Shen et al., 2025)	20.00	1268.31	34.14	812.72	54.80	483.46	52.13	592.44	40.26	789.23
s1 (Muennighoff et al., 2025)	21.71	1479.92	29.77	1327.84	56.50	1205.05	54.14	1352.13	40.53	1341.24
TTD (Ours)	**22.60**	**343.47**	**35.99**	**404.54**	**61.20**	**215.95**	**59.94**	**216.65**	**44.93**	**295.15**

Method	MMMU		ScienceQA		A-OKVQA		MMBench		Average	
	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length
Base Model	43.55	472.41	76.35	202.11	80.96	39.57	78.05	202.32	69.73	229.10
Random Deletion	46.55	366.43	78.14	128.78	81.13	20.59	80.91	102.56	71.68	154.59
MTI (Yang et al., 2025c)	46.55	499.01	75.85	197.25	80.08	52.71	78.03	207.45	70.13	239.11
TTI (Shen et al., 2025)	44.22	772.01	76.40	298.14	79.03	55.65	78.19	279.03	69.46	351.21
s1 (Muennighoff et al., 2025)	45.11	1212.34	76.54	934.01	81.13	789.99	78.12	1052.86	70.23	997.30
TTD (Ours)	**47.56**	**366.43**	**79.13**	**128.78**	**82.18**	**20.59**	**81.82**	**102.56**	**72.67**	**154.59**

Table 8: **Comparison with test-time intervention strategies on ThinkLite-VL-7B (Wang et al., 2025c).** We report Pass@1 performance on four multimodal mathematical reasoning benchmarks and four general multimodal reasoning benchmarks.

Method	MathVision		MathVerse		MathVista		WeMath		Average	
	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length
Base Model	26.94	498.25	46.58	363.37	69.0	270.40	67.99	289.64	52.63	355.41
Random Deletion	25.36	421.78	44.59	305.35	70.0	171.99	65.11	228.27	51.26	281.85
MTI (Yang et al., 2025c)	26.57	495.38	47.31	386.27	71.0	288.26	67.47	296.46	53.08	366.59
TTI (Shen et al., 2025)	27.00	957.96	48.40	696.53	70.3	529.05	67.01	575.80	53.18	689.84
s1 (Muennighoff et al., 2025)	26.28	1143.36	48.45	991.57	70.9	614.59	66.89	790.74	53.12	885.07
TTD (Ours)	**27.24**	**421.78**	**49.06**	**305.35**	**73.6**	**171.99**	**68.16**	**228.27**	**54.51**	**281.85**

Method	MMMU		ScienceQA		A-OKVQA		MMBench		Average	
	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length
Base Model	52.33	418.91	88.20	192.74	83.75	147.12	86.32	164.91	77.65	230.92
Random Deletion	50.66	340.82	87.50	141.45	83.84	55.36	86.41	79.16	77.10	154.20
MTI (Yang et al., 2025c)	52.00	432.34	88.49	196.56	83.40	152.61	86.07	172.13	77.49	238.41
TTI (Shen et al., 2025)	52.66	799.26	88.74	363.48	83.66	277.50	86.02	324.66	77.77	441.23
s1 (Muennighoff et al., 2025)	53.78	694.61	88.24	519.10	83.23	666.47	86.39	564.06	77.91	611.06
TTD (Ours)	**55.11**	**340.82**	**88.89**	**141.45**	**85.07**	**55.36**	**86.81**	**79.16**	**78.97**	**154.20**

Table 9: **Comparison with test-time intervention strategies on MM-Eureka-7B (Meng et al., 2025).** We report Pass@1 performance on four multimodal mathematical reasoning benchmarks and four general multimodal reasoning benchmarks.

Method	MathVision		MathVerse		MathVista		WeMath		Average	
	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length
Base Model	28.06	611.07	**50.46**	488.49	69.4	330.21	64.48	371.94	53.10	450.43
Random Deletion	27.47	542.76	47.43	447.92	70.7	300.02	64.54	303.57	52.54	398.56
MTI (Yang et al., 2025c)	28.22	646.21	49.49	500.12	70.1	328.45	65.22	386.34	53.25	465.28
TTI (Shen et al., 2025)	29.17	1187.40	48.83	982.59	70.4	646.67	63.39	737.12	52.95	888.45
s1 (Muennighoff et al., 2025)	28.28	1215.05	48.27	1137.77	70.5	799.66	65.63	993.49	53.17	1036.49
TTD (Ours)	**29.31**	**542.76**	50.10	**447.92**	**72.1**	**300.02**	**66.44**	**303.57**	**54.49**	**398.56**

Method	MMMU		ScienceQA		A-OKVQA		MMBench		Average	
	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length
Base Model	51.77	524.77	86.11	210.63	82.96	161.17	86.27	172.64	76.78	267.30
Random Deletion	50.11	463.08	85.37	154.11	84.27	64.37	86.18	115.28	76.48	199.21
MTI (Yang et al., 2025c)	51.88	539.81	85.51	221.34	82.53	173.77	85.86	179.55	76.44	278.62
TTI (Shen et al., 2025)	52.00	1005.01	85.67	421.47	82.79	322.93	85.81	349.42	76.57	524.71
s1 (Muennighoff et al., 2025)	52.22	978.96	85.47	687.18	82.18	519.53	86.02	701.52	76.47	721.80
TTD (Ours)	**54.56**	**463.08**	**86.47**	**154.11**	**85.68**	**64.37**	**86.58**	**115.28**	**78.32**	**199.21**

Computational cost. For entropy-guided TTD, the main additional cost comes from re-decoding the final answer conditioned on the pruned reasoning trace. In our implementation, this step takes approximately 0.125 seconds per example on an NVIDIA A6000, excluding the cost of generating the initial reasoning trace.

Table 10: Ablation study using additional multimodal reasoning models with different scales.

Models	MathVision	MathVerse	MathVista	WeMath	Average	MMMU	ScienceQA	A-OKVQA	MMBench	Average
Qwen2.5-VL-3B	19.47	33.58	56.30	50.63	39.99	43.55	76.35	80.96	78.05	69.73
Qwen2.5-VL-3B + TTD (Ours)	22.60	35.99	61.20	59.94	44.93	47.56	79.13	82.18	81.82	72.67
ThinkLite-VL-7B	26.94	46.58	69.00	67.99	52.63	52.33	88.20	83.75	86.32	77.65
ThinkLite-VL-7B + TTD (Ours)	27.24	49.06	73.60	68.16	54.51	55.11	88.89	85.07	86.81	78.97
MM-Eureka-7B	28.06	50.46	69.40	64.48	53.10	51.77	86.11	82.96	86.27	76.78
MM-Eureka-7B + TTD (Ours)	29.31	50.10	72.10	66.44	54.49	54.56	86.47	85.68	86.58	78.32
LLaMA3.2-Vision-11B	16.02	24.64	44.70	42.07	31.86	44.66	85.91	83.75	82.14	74.12
LLaMA3.2-Vision-11B + TTD (Ours)	19.84	27.39	48.50	48.05	35.95	47.78	86.91	84.19	83.60	75.62
ARES-7B	47.76	51.31	67.30	71.09	59.36	57.78	86.11	82.09	85.58	77.89
ARES-7B + TTD (Ours)	49.90	52.79	71.40	74.54	62.16	61.56	86.66	84.63	86.07	79.73
Qwen3-VL-4B-Thinking	58.88	58.95	75.90	80.86	68.64	62.66	88.74	83.84	88.68	80.98
Qwen3-VL-4B-Thinking + TTD (Ours)	59.93	61.47	78.30	82.64	70.59	65.89	91.47	85.41	89.28	83.01

Table 11: **Difficulty-controlled entropy analysis on MMMU.** We report the average thinking-token entropy for correct and incorrect samples within each difficulty group, together with the accuracy of the base model and TTD. The entropy gap persists across difficulty levels, and TTD improves accuracy in all groups.

Difficulty	Average Thinking-Token Entropy		Accuracy	
	Correct Samples	Incorrect Samples	Base Model	TTD
Easy	18.33	21.63	65.76	66.44
Medium	21.63	25.05	45.75	46.70
Hard	26.46	27.02	42.54	43.65
Overall	21.19	24.62	51.67	52.56

For mask-learning TTD, the dominant cost comes from forward-only optimization of an instance-specific deletion mask. We use SPSA with $T = 80$ iterations, each requiring two objective evaluations. The robustness term introduces an additional forward evaluation of a perturbed reasoning trace, but this evaluation does not require autoregressive generation. As shown in Table 12, mask-learning TTD takes approximately 20 seconds per example, compared with 18 seconds for MTI, 16 seconds for TTI, and 34 seconds for s1. Its runtime is therefore higher than those of MTI and TTI but lower than that of s1.

Complementary roles and optimization potential. Entropy-guided and mask-learning TTD are complementary instantiations of the same framework, rather than alternatives designed for identical computational budgets. The entropy-guided variant provides a lightweight, training-free intervention, whereas the mask-learning variant explores adaptive token selection through test-time optimization. Both variants keep the underlying model parameters frozen. Furthermore, the performance gaps among the entropy-guided, mask-learning, and oracle variants motivate better self-improvement signals for reasoning-context selection. In particular, on MathVision, the mask-learning TTD oracle achieves 49.57% accuracy, compared with 32.76% for the token-deletion oracle considered in our analysis. This result is consistent with the importance of interactions among tokens during test-time learning, since the utility of a retained token may depend on which other tokens remain in the reasoning context.

F.3 Design Choice Analysis

We conduct additional analyses to examine whether TTD benefits from the proposed masking mechanism and to evaluate the importance of the entropy-minimization objective. Given the binary mask $\mathbf{m}$ produced by TTD, where $m_\ell = 1$ indicates that token x_ℓ is kept, we construct a complement mask $\mathbf{1} - \mathbf{m}$. The complement mask keeps only the tokens that TTD would delete and removes the tokens that TTD would retain. The resulting reasoning trace preserves the original order of the kept tokens and is used to re-decode the final answer under the same evaluation protocol. As shown in Table 13, using the original TTD mask improves the average accuracy from 50.10 to 53.18. In contrast, using the complement mask reduces the average accuracy to 45.15, which is 4.95 points below the base model and 8.03 points below the original TTD mask. The substantial degradation

Table 12: **Runtime analysis of TTD and comparison methods.** All runtimes are approximate and measured on an NVIDIA A6000. Panel (a) reports the additional answer re-decoding overhead of entropy-guided TTD, excluding initial reasoning-trace generation. Panel (b) compares per-example runtimes across methods. The two panels use different timing scopes and should not be compared directly.

Method	**Time (s/example)**
(a) Additional answer re-decoding overhead	
Entropy-guided TTD	0.125
(b) Per-example runtime comparison	
MTI	18
TTI	16
s1	34
Mask-learning TTD ($T = 80$)	20

indicates that the tokens retained by the mask in TTD are more useful for final-answer decoding than those removed. Moreover, we further evaluate an entropy-maximization counterexample. Specifically, we use the same test-time mask-learning procedure but reverse the entropy objective. The variant significantly degrades average performance to 43.34, which is 9.84 points below the original design. The results suggest that the direction of the deletion objective is important. Masks that preserve uncertain or less useful contexts can make final-answer decoding worse, even when the response length is reduced.

Table 13: **Design choice analysis.** The complement mask $\mathbf{1} - \mathbf{m}$ keeps only the tokens that TTD would delete and removes the tokens that TTD would retain. Entropy Maximization uses the same test-time mask-learning procedure but reverses the entropy objective. Both variants substantially degrade performance, validating the design choice of TTD.

Method	**MathVision**		**MathVerse**		**MathVista**		**WeMath**		**Average**	
	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length
Base Model	25.23	523.78	44.29	397.23	67.20	271.48	63.68	284.16	50.10	369.16
TTD (mask $\mathbf{m}$)	27.20	349.01	46.24	276.42	71.50	133.76	67.76	152.66	53.18	227.96
TTD (complement mask $\mathbf{1} - \mathbf{m}$)	21.67	174.76	37.69	120.81	61.90	137.72	59.36	131.50	45.15	141.20
TTD (Entropy Maximization)	20.39	244.98	35.58	188.67	59.40	129.17	57.99	136.58	43.34	174.85

F.4 Comparison with Model-based Test-Time Adaptation

We compare TTD with MM-UPT (Wei et al., 2025) to examine whether selective thinking-token deletion can provide an alternative or complementary approach to unsupervised model-based test-time training methods. MM-UPT performs unsupervised post-training through GRPO, using majority voting over sampled responses to construct self-rewarding signals without ground-truth answers. We re-implement it in our setting and perform 10 rollouts per data point with 15 training episodes. It incurs an estimated cost of 60.48 seconds per sample. In contrast, TTD keeps the model parameters frozen and selectively deletes tokens from a completed reasoning trace before re-decoding the final answer. Table 14 compares Qwen2.5-VL-7B (Bai et al., 2025b) and its MM-UPT-adapted counterpart, each with and without TTD. Applying TTD directly to Qwen2.5-VL-7B increases average Pass@1 from 50.10 to 53.18, exceeding the 51.76 achieved by MM-UPT alone by 1.42 percentage points. This advantage holds across all four benchmarks. These results show that selectively editing completed reasoning traces can even outperform the evaluated parameter-adapted model without updating the base MLLM. We further apply TTD to reasoning traces generated by the MM-UPT-adapted model, keeping its parameters frozen throughout token deletion and answer re-decoding. This combination enhances average Pass@1 from 51.76 to 53.31, a further gain of 1.55 percentage points. The results validate the complementarity of model-parameter adaptation and reasoning-context intervention.

Table 14: Comparison and complementarity with unsupervised model adaptation.

Models	MathVision	MathVerse	MathVista	WeMath	Average
Qwen2.5-VL-7B (Bai et al., 2025b)	25.23	44.29	67.20	63.68	50.1
Qwen2.5-VL-7B + TTD (Ours)	27.20	46.24	71.50	67.76	53.18 (**3.08%**↑)
MM-UPT-7B (Wei et al., 2025)	26.31	43.29	70.20	67.24	51.76
MM-UPT-7B + TTD (Ours)	27.01	45.46	71.80	68.97	53.31 (**1.55%**↑)

F.5 Complementarity with Sampling-Based Test-Time Scaling

To investigate whether TTD provides a complementary axis to sampling-based test-time scaling, we compare standard self-consistency (SC) (Wang et al., 2022) with entropy-guided TTD. Self-consistency improves reasoning performance by sampling multiple independent reasoning traces and aggregating their final answers, typically through majority voting. TTD edits each sampled thinking trajectory by deleting selected thinking tokens before final-answer re-decoding. Therefore, TTD can be naturally combined with self-consistency by applying thinking-token deletion to each sampled reasoning trace before answer aggregation. We evaluate both methods under different sampling budgets on MathVision, MathVerse, MathVista, and WeMath using Qwen2.5-VL-7B (Bai et al., 2025b). As shown in Figure 10(a), applying TTD consistently improves the accuracy of self-consistency under the same number of sampled trajectories. In several settings, self-consistency with TTD reaches comparable or better accuracy using $2\times$ to $4\times$ fewer sampled trajectories than standard self-consistency. The results suggest that thinking-token deletion is complementary to parallel-inference-style test-time scaling. Rather than only increasing the number of sampled trajectories, selectively editing each generated trace can make the available trajectory samples more effective for answer aggregation.

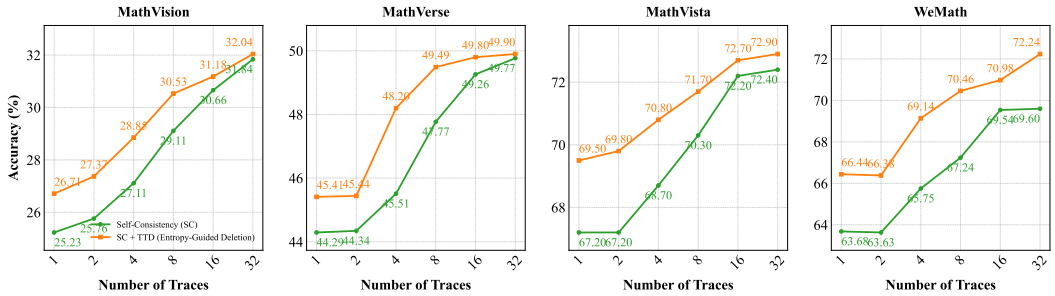
We further evaluate inference efficiency in terms of wall-clock time in Figure 10(b). Entropy-guided TTD requires one additional forward pass per sampled trace, whereas self-consistency scales inference cost by autoregressively generating additional independent traces. Nevertheless, TTD maintains a favorable accuracy–efficiency trade-off, demonstrating that it can strengthen sampling-based test-time scaling by improving the utility of each trajectory before answer aggregation.

G Additional Discussion on Pause-Token Methods

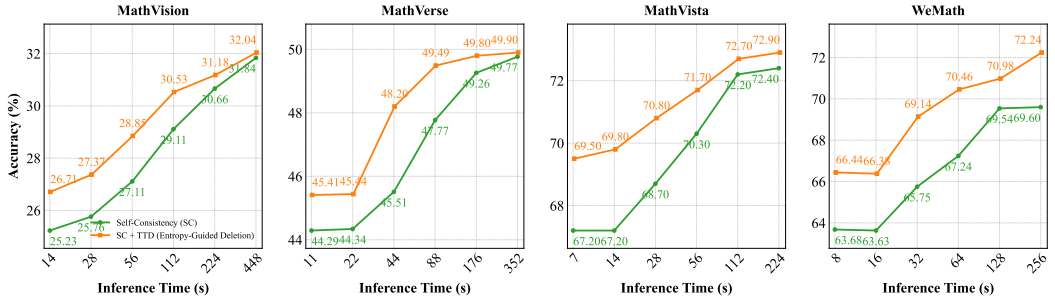
We further discuss the relation between TTD and recent pause-token methods (Kim et al., 2025a; Goyal et al., 2023b). These methods study whether inserting special-purpose pause tokens can provide models with additional computation steps. Typically, the pause tokens are introduced through pre-training or fine-tuning, so the model is explicitly trained to interpret them as useful intermediate computation tokens. In contrast, TTD keeps the base MLLM frozen and intervenes only at test time on the model's naturally generated thinking tokens. It does not introduce new special tokens, nor does it require the model to learn how to use additional tokens for computation. After the model has generated a complete reasoning trace, TTD selectively removes generated thinking tokens and re-decodes the final answer conditioned on the remaining trace. Note that we do not argue that more intermediate tokens are always harmful. In fact, our experiments show that CoT reasoning is beneficial overall. Rather, we show that not all generated thinking tokens are equally useful for final-answer decoding after the reasoning trace has already been produced. In our main experiments, we also compare TTD with MTI (Yang et al., 2025c), which inserts an additional cue token, [Output Error], around high-entropy positions to prompt token regeneration. The results demonstrate that our TTD can enhance reasoning performance without introducing additional tokens.

H Algorithm of Test-time Token Mask Learning

The complete algorithm of our test-time thinking-token mask learning mechanism is presented in Algorithm 1.



(a) Accuracy versus the number of sampled reasoning traces.



(b) Accuracy versus wall-clock inference time.

Figure 10: **Complementarity with sampling-based test-time scaling.** We compare standard self-consistency (Wang et al., 2022) with entropy-guided TTD on MathVision, MathVerse, MathVista, and WeMath using Qwen2.5-VL-7B (Bai et al., 2025b). **(a)** Under the same sampling budget, TTD improves the accuracy of self-consistency and often achieves comparable or better performance with $2\times$ to $4\times$ fewer sampled thinking traces. **(b)** TTD preserves a favorable accuracy–efficiency trade-off in terms of wall-clock inference time. The entropy-guided variant introduces a small overhead, requiring only one additional forward pass to edit each sampled trace via deletion. Overall, the results suggest that thinking-token deletion is complementary to sampling-based test-time scaling.

I VISUALIZATIONS

We provide two qualitative examples to illustrate how thinking-token deletion can change final-answer decoding. In each example, the base model first generates a complete reasoning trace and produces an incorrect final answer. After removing the selected thinking token from the generated trace, TTD re-decodes the final answer conditioned on the remaining trace and obtains the correct prediction. These examples show that even small interventions on the completed reasoning trace, including the deletion of a single token, can correct the final answer.

Algorithm 1: Test-time Token Mask Learning

Require: initial thinking trace $\mathcal{X} = (x_1, \ldots, x_L)$, mask logits $\mathbf{s} \in \mathbb{R}^L$ (one logit per thinking token), step budget T, perturbation scale ϵ, learning rate η_t, hard threshold γ.

Initialize: $\mathbf{s} \leftarrow s_0 \cdot \mathbf{1} \in \mathbb{R}^L$ ▷ Mask logits initialized to constant s_0

for $t = 1, \ldots, T$ **do**
- Sample random seed r
- $\mathbf{s} \leftarrow$ `PerturbMaskLogits`$(\mathbf{s}, +\epsilon, r)$ ▷ In-place: $\mathbf{s} \leftarrow \mathbf{s} + \epsilon\mathbf{u}$
- $\mathcal{X}^{(\mathbf{m})} \leftarrow$ `ThinkingTokenDeletion`$(\mathcal{X}, \mathbf{s})$
- $\ell_+ \leftarrow \mathcal{L}(\mathbf{s})$ ▷ Objective at $\mathbf{s} + \epsilon\mathbf{u}$ via Eq. 10
- $\mathbf{s} \leftarrow$ `PerturbMaskLogits`$(\mathbf{s}, -2\epsilon, r)$ ▷ Now at $\mathbf{s} - \epsilon\mathbf{u}$
- $\mathcal{X}^{(\mathbf{m})} \leftarrow$ `ThinkingTokenDeletion`$(\mathcal{X}, \mathbf{s})$
- $\ell_- \leftarrow \mathcal{L}(\mathbf{s})$ ▷ Objective at $\mathbf{s} - \epsilon\mathbf{u}$ via Eq. 10
- $\mathbf{s} \leftarrow$ `PerturbMaskLogits`$(\mathbf{s}, +\epsilon, r)$ ▷ Reset back to original $\mathbf{s}$
- `projected_grad` $\leftarrow (\ell_+ - \ell_-)/(2\epsilon)$
- Reset random number generator with seed r ▷ Resample the same $\mathbf{u}$ for update
- **for** $i = 1, \ldots, L$ **do**
  - $u_i \sim \mathcal{N}(0, 1)$
  - $s_i \leftarrow s_i - \eta_t \cdot$ `projected_grad` $\cdot\, u_i$ ▷ update mask logits
- **end**

end

Return: `ThinkingTokenDeletion`$(\mathcal{X}, \mathbf{s})$

Subroutine `PerturbMaskLogits`$(\mathbf{s}, \epsilon, r)$
- Reset random number generator with seed r
- **for** $i = 1, \ldots, L$ **do**
  - $u_i \sim \mathcal{N}(0, 1)$
  - $s_i \leftarrow s_i + \epsilon \cdot u_i$ ▷ Modify mask logits in place
- **end**
- **return** $\mathbf{s}$

Subroutine `ThinkingTokenDeletion`$(\mathcal{X}, \mathbf{s})$
- $\mathbf{g} \leftarrow \sigma(\mathbf{s})$ ▷ Soft gates $\mathbf{g} \in (0, 1)^L$
- $\mathbf{m} \in \{0, 1\}^L$ where $m_i = \mathbb{I}\big[\sigma(s_i) \geq \gamma\big]$ ▷ Hard mask via Eq. 6
- $\mathcal{X}^{(\mathbf{m})} \leftarrow ()$ ▷ Initialize empty masked trace
- **for** $i = 1, \ldots, L$ **do**
  - **if** $m_i = 1$ **then**
    - Append x_i to $\mathcal{X}^{(\mathbf{m})}$ ▷ Keep token order
  - **end**
- **end**
- **return** $\mathcal{X}^{(\mathbf{m})}$

Case Study Example 1 in WeMath

Prompt



Question: Assemble the shapes below into a rectangle. What are the length and width of the resulting rectangle?
Choices: A. 12, 4 B. 12, 8 C. 8, 4 D. No correct answer

Base Model Answer (False)

To solve this problem, let's follow these steps:
1. Identify the dimensions of the shapes given:
   - The first shape is a rectangle with a length of 8 cm and a width of 4 cm .
   - The second shape is a square with a side length of 4 cm .
2. Determine how to assemble these shapes into a rectangle:
   - The square can be placed either on top of the rectangle or on the side.
   - If we place the square on top of the rectangle, the resulting rectangle will have a length of 8 cm (the same as the original rectangle) and a width of 4 cm (the same as the square).
3. Check the options provided:
   - - A. 12, 4: This does not match the dimensions we calculated.
   - - B. 12, 8: This does not match the dimensions we calculated.
   - - C. 8, 4: This matches the dimensions we calculated.
   - - D. No correct answer: This is not applicable since we have a correct answer.

Therefore, the length and width of the resulting rectangle are 8 cm and 4 cm.
Final answer: \boxed{8, 4}

TTD Answer (True)

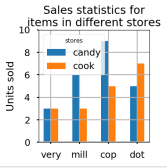
solve this problem, let's follow these steps:
1. Identify the dimensions of the shapes given:
   - The first shape is a rectangle with a length of 8 cm and a width of 4 cm .
   - The second shape is a square with a side length of 4 cm .
2. Determine how to assemble these shapes into a rectangle:
   - The square can be placed either on top of the rectangle or on the side.
   - If we place the square on top of the rectangle, the resulting rectangle will have a length of 8 cm (the same as the original rectangle) and a width of 4 cm (the same as the square).
3. Check the options provided:
   - - A. 12, 4: This does not match the dimensions we calculated.
   - - B. 12, 8: This does not match the dimensions we calculated.
   - - C. 8, 4: This matches the dimensions we calculated.
   - - D. No correct answer: This is not applicable since we have a correct answer.

Therefore, the length and width of the resulting rectangle are 8 cm and 4 cm.
Final answer: \boxed{12, 4}

Figure 11: **Case study example in WeMath.**

Case Study Example 2 in MathVista

Prompt



Question: How many items sold less than 5 units in at least one store?

Base Model Answer (False)

To determine how many items sold less than 5 units in at least one store, let's analyze the data for each item:

1. very: candy: 3 units - cook: 3 units - Both items sold less than 5 units.
2. mill: candy: 6 units - cook: 3 units - Only the cook item sold less than 5 units.
3. cop: candy: 9 units - cook: 5 units - The cook item sold exactly 5 units, not less than 5.
4. dot: candy: 5 units - cook: 7 units - Neither item sold less than 5 units.

Now, let's count the items that sold less than 5 units:

- very: 2 items (candy and cook)
- mill: 1 item (cook)

Adding these up, we have a total of 3 items that sold less than 5 units.

Final answer: \boxed{3}

TTD Answer (True)

To determine how many items sold less than 5 units in at least one store, let's analyze the data for each item:

1. very: candy: 3 units - cook: 3 units - items sold less than 5 units.
2. mill: candy: 6 units - cook: 3 units - Only the cook item sold less than 5 units.
3. cop: candy: 9 units - cook: 5 units - The cook item sold exactly 5 units, not less than 5.
4. dot: candy: 5 units - cook: 7 units - Neither item sold less than 5 units.

Now, let's count the items that sold less than 5 units:

- very: 2 items (candy and cook)
- mill: 1 item (cook)

Adding these up, we have a total of 3 items that sold less than 5 units.

Final answer: \boxed{2}

Figure 12: **Case study example in MathVista.**