AutoL2S: Auto Long-Short Reasoning for Efficient LLMs

Anonymous ACL submission

Abstract

Reasoning-capable large language models (LLMs) achieve strong performance on complex tasks but often exhibit overthinking after distillation, generating unnecessarily long chain-of-thought (CoT) reasoning even for simple inputs and incurring high inference cost. However, naively shortening reasoning length can degrade reasoning accuracy, as concise reasoning may be insufficient for certain inputs and lacks explicit supervision. We propose Auto Long-Short Reasoning (AutoL2S), a distillation framework that empowers non-reasoning LLMs to think thoroughly but only when necessary. AutoL2S first learns a lightweight switching token with verified long-short CoTs to enable instance-wise long-short reasoning selection. Then it leverages long-short reasoning rollouts induced by switching tokens within a GRPO-style loss to improve reasoning efficiency while maintaining accuracy. Experiments demonstrate that AutoL2S effectively reduces reasoning length up to 71% with minimal accuracy loss, yielding markedly better trade-off in token length and inference time while preserving accuracy. The code is available at https://anonymous.4open.science/r/AutoL2S-A72E

1 Introduction

Reasoning distillation is an effective approach for transferring complex reasoning abilities from strong teacher large language models (LLMs) to non-reasoning capable student LLMs (Guo et al., 2025; Labs, 2025; Muennighoff et al., 2025; Ye et al., 2025), but it often introduces a critical, inefficient overthinking issue (Sui et al., 2025). Distilled models tend to generate excessively long CoT reasoning paths even for inputs that admit concise solutions, resulting in substantial increases in decoding time, memory usage, and deployment cost (Chen et al., 2024). This behavior arises because distillation typically trains student models to imitate full long-form reasoning paths in order to preserve accuracy, implicitly treating long reasoning as uniformly necessary across instances. As a result, distilled models lack signals indicating when shorter reasoning would suffice. Existing approaches mitigate overthinking through manual post-distillation control of reasoning modes (e.g., prompting users to select short or long reasoning) (Yang et al., 2024; Anthropic, 2023), or by learning reasoning-mode selection via special tokens or reinforcement learning guided by outcome-based rewards (Luo et al., 2025; Ma et al., 2025a; Fang et al., 2025). However, these methods rely on strong reference behaviors and often suffer from accuracy degradation when reasoning is aggressively shortened.

The challenges lie in the nature of the trade-off between reasoning length and accuracy. First, short reasoning may lead to performance degradation compared to long ones, particularly on inputs that require multi-step inference or error correction. Without reliable signals indicating when compression is safe, enforcing shorter reasoning risks discarding intermediate steps that are critical to correctness, resulting in inferior and inconsistent behaviors across inputs (Luo et al., 2025; Fang et al., 2025). Second, reasoning length is difficult to regulate while preserving accuracy. For many inputs, multiple reasoning paths of varying lengths can lead to correct answers, and whether concise reasoning is sufficient is often only observable through outcome correctness (Ma et al., 2025b; Zhang et al., 2025b; Liu et al., 2024; Yu et al., 2024). These challenges suggest that effective CoT compression cannot be treated as uniform truncation, but must instead be framed as an instance-wise decision problem that dynamically balances correctness and token-generation cost. We ask: ***How can reasoning distillation allow reasoning length to vary across instances while maintaining correctness?***

To address these challenges, we propose Auto

Long-Short Reasoning (AutoL2S), a distillation framework that enables non-reasoning LLMs to reason thoroughly only when necessary. AutoL2S pairs long teacher reasoning with verified short reasoning paths and explicitly supervises a lightweight switching token (<EASY>) during distillation. The <EASY> token is generated as part of the reasoning sequence and induces instance-wise selection between long and short CoT reasoning paths. Building on this supervision, AutoL2S further leverages the induced long-short reasoning rollouts by <EASY> token to fine-tune the model with a GRPO-style loss. This stage encourages correctness of CoT reasoning paths while implicitly favoring shorter reasoning when sufficient, allowing the model to internalize instance-wise trade-offs between reasoning sufficiency and efficiency.

Across multiple reasoning benchmarks, we demonstrate that AutoL2S substantially reduces reasoning length while maintaining accuracy, achieving up to a 71% reduction in reasoning length with minimal loss in accuracy. The contributions are listed as follows:

- **Auto Long-Short Reasoning.** We propose AutoL2S, a distillation framework that pairs long and verified short reasoning paths for long-short reasoning selection, and further improves efficiency via GRPO-style fine-tuning under supervision-induced long-short rollouts.
- **Implicit Control via Joint Generation.** We introduce a lightweight <EASY> switching token that is jointly generated before the actual rollout, allowing long-short mode selection to be learned implicitly without explicit length constraints.
- **Reasoning Evaluation.** AutoL2S achieves substantial efficiency gains while maintaining accuracy across multiple reasoning benchmarks.

2 Preliminary

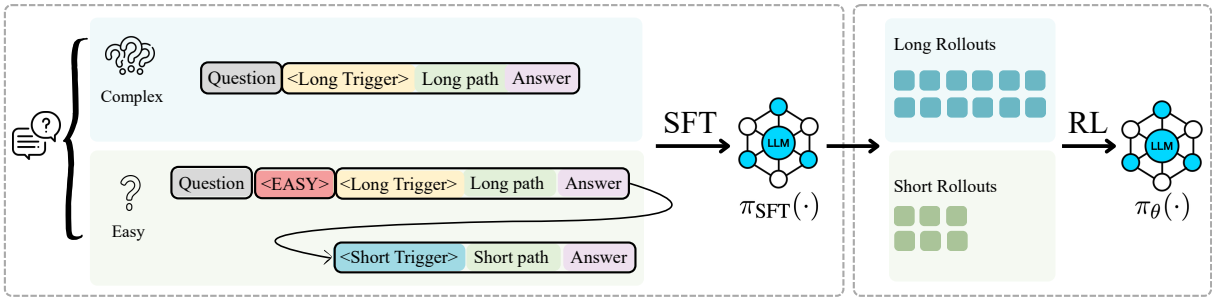
In this section, we first formally define the Auto Long-Short reasoning problem. We then illustrate the challenges in controlling the reasoning length of distilled large reasoning-capable LLMs.

2.1 Problem Definition

We aim to develop reasoning models $\pi(\cdot \mid \theta)$ with trainable parameters θ that complete tasks correctly while using reasoning paths as short as possible. The objective is to train $\pi(\cdot \mid \theta)$ in $\mathcal{D}$ to learn a policy that selects minimal sufficient reasoning for each input. We expect the outputs of $\pi(\cdot \mid \theta_{\mathcal{D}})$ to be sufficiently short while maintaining reasoning accuracy. This reduction in output length translates directly to fewer generated tokens and thus faster inference. To this end, we propose the *Auto Long-Short Reasoning* (AutoL2S) framework to enable efficient LLM reasoning through joint utilization of valid long and short CoT reasoning paths. We emphasize that reasoning length does not admit a unique ground-truth label: for many inputs, multiple reasoning paths of different lengths can lead to correct answers. AutoL2S aims to identify an effective trade-off between reasoning accuracy and generation efficiency. Therefore, AutoL2S is evaluated based on outcome preservation under reasoning length with accuracy preservation.

2.2 Challenges of Length Controlling

Balancing brevity and completeness in reasoning remains challenging. Aggressive compression of reasoning paths can omit essential intermediate steps, leading to degraded performance on complex inputs, while the absence of supervision signals for the minimally sufficient reasoning trace makes it difficult for models to determine when concise reasoning is appropriate. Recent SFT-based approaches mitigate overthinking by curating datasets with variable-length or information-dense reasoning traces and fine-tuning models to produce shorter reasoning (Ma et al., 2025a; Xia et al., 2025). However, these methods primarily encourage global compression and do not provide an explicit mechanism for instance-wise reasoning-length selection, often resulting in over-compression on inputs that require extended reasoning. Related work (Fang et al., 2025) explores reasoning-mode selection using switching tokens and reinforcement learning, but such approaches rely on outcome-based rewards rather than direct supervision of correctness-preserving brevity, and lack explicit signals indicating when shorter reasoning suffices. As a result, models do not receive explicit supervision, distinguishing cases where concise reasoning is sufficient from those that require extended reasoning, relying instead on outcome-based optimization, which can make instance-wise reasoning-length adaptation less reliable. These limitations suggest that effective reasoning compression cannot rely solely on uniform supervision or reward-driven mode selection, but instead requires structured supervision that enables reasoning length to vary across instances.



(a) Stage 1 (b) Stage 2

Figure 1: Training pipeline of AutoL2S in two stages. (a) Supervised with paired long and short CoT reasoning paths with <EASY> token. (b) <EASY> token is used to induce long-short reasoning rollout, enabling the model to generate short or long reasoning paths in an instance-dependent manner.

3 Auto Long-Short Reasoning

We systematically introduce the AutoL2S framework. AutoL2S aims to distill reasoning capabilities from reasoning-capable LLMs, allowing the model to learn effective reasoning patterns while reducing the length of reasoning paths required to arrive at correct reasoning answers. To achieve our goal, we propose a two-stage training pipeline: (1) supervised fine-tuning for adaptive reasoning rollout selection, and (2) leveraging GRPO-style optimization with supervision-induced rollouts.

3.1 SFT Stage of AutoL2S

AutoL2S constructs a diverse reasoning dataset containing both long and short CoT reasoning paths based on prediction correctness. The construction pipeline is illustrated in Figure 1. Long CoT reasoning paths are provided for all questions to capture complete reasoning, while short CoT reasoning paths are preferred whenever they still yield correct answers, offering more efficient representations. AutoL2S trains LLMs to learn both long and short reasoning paths and to identify EASY questions, enabling efficient reasoning when appropriate.

Constructing Long CoT Reasoning Paths. We use Bespoke-Stratos-17k (Labs, 2025) as the source of questions and employ a strong reasoning-capable LLM as teacher to generate long CoT reasoning paths together with final answers, forming the base long-CoT dataset. For an input X with ground-truth answer $y^\star$, we treat L as an effective long reasoning path if it yields the correct answer, without requiring token-level semantic optimality.

Constructing Short CoT Reasoning Paths. To avoid uniformly enforcing long reasoning when concise reasoning suffices, we generate a short reasoning path S such that $|S| \ll |L|$. Specifically, we employ a short CoT teacher to generate candidate short reasoning paths $\{S_j\}_{j=1}^{k}$ using rejection sampling with k trials. Among these candidates that yield the correct answer, we select the shortest path $S = \arg\min_{\{S_j|\hat{y}(S_j)=y\}} |S_j|$ as the effective short CoT reasoning path. This procedure yields concise reasoning traces that preserve correctness while minimizing reasoning length.

SFT Training Strategy. AutoL2S follows Figure 1 to construct $\mathcal{D}$, where special tokens <EASY>, <Long Trigger>, <Short Trigger>, and <Answer Trigger> are used to hook the questions, long-short reasoning, and final answers. For inputs admitting both valid long and short CoT reasoning paths (L, S), we annotate the question with the <EASY> token; for inputs where no such valid short CoT exists, we retain the original long reasoning L and omit the <EASY> token. Formally, AutoL2S adopts the constructed dataset $\mathcal{D} = \{(x_i, \mathcal{R}_i, y_i^\star)\}_{i=1}^{N}$, where each reasoning path $\mathcal{R}_i \in \{\{L_i\}, \{\texttt{<EASY>}, L_i, S_i\}\}$, and trains the model $\pi(\cdot \mid \theta)$ by minimizing the next-token prediction loss function given as follows:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(x_i,\mathcal{R}_i,y_i^\star)\sim\mathcal{D}}\big[\log \pi(r_t \mid r_{<t}, x_i, \theta)\big],$$

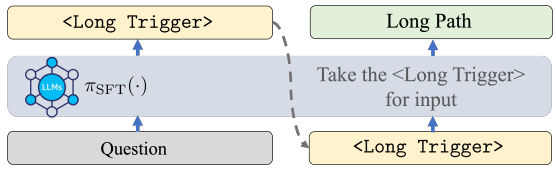
where $r_{<i}$ denotes the prefix tokens in $\mathcal{R}_t$ that precede position t. We denote the SFT-trained model as $\pi_{\text{SFT}}(\cdot)$.

3.2 AutoL2S: Long-Short Joint Rollouts

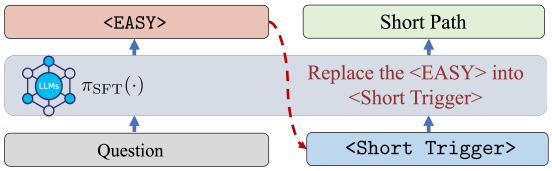
In this section, we correct residual length-accuracy misalignment induced by the SFT stage using positive and negative signals from long-short rollouts.

3.2.1 Long-short Rollout Generation

During the inference stage, the SFT model $\pi_{\text{SFT}}(\cdot)$ is able to determine whether short CoT reasoning is sufficient to solve questions, enabling adaptive length for rollout generation. Specifically, as illustrated in Figure 2, $\pi_{\text{SFT}}(\cdot)$ begins generation by



(a) Generating the long reasoning rollout



(b) Generating the short reasoning rollout

Figure 2: The SFT model $\pi_{\text{SFT}}(\cdot)$ generates (a) a long reasoning rollout when <Long Trigger> yields; or (b) a short reasoning rollout conditioned on <EASY> token. The long-short selection is determined by $\pi_{\text{SFT}}(\cdot)$.

producing either a <Long Trigger> or a <EASY> token, which determines the subsequent CoT generation. If the first generated token is a <Long Trigger> token (as shown in Figure 2(a)), it indicates that the question requires a long reasoning path, and then the model proceeds with standard autoregressive generation to complete the long reasoning and produce the final answer. In contrast, if $\pi_{\text{SFT}}(\cdot)$ initially generates an <EASY> token (as shown in Figure 2(b)), which suggests the question is solvable with a short CoT. Then, we leverage constraint decoding to directly inject <Short Trigger> token to enforce the model generating short reasoning paths.

3.2.2 Joint Rollouts Training

Leveraging automatic long-short selection with <EASY> token, AutoL2S proposes a GRPO-style framework to further improve reasoning efficiency while maintaining accuracy. Specifically, it leverages the long-short CoT rollouts generated by the SFT model $\pi_{\text{SFT}}(\cdot)$, and proposes a GRPO-style objective function that maximizes the model accuracy while keeping the output length distribution close to the SFT model, preserving the foundation capability for long-short selection. The objective function is formally defined by:

$$\mathcal{J}_{\text{AutoL2S}}(\theta) = \mathbb{E}_{\substack{z\sim\mathcal{D} \\ r\sim\pi_{\text{SFT}}}} \Big[\min\big(w(\theta)A(z,r), \text{clip}(w(\theta), 1-\varepsilon, 1+\varepsilon)A(z,r)\big)\Big],$$

where $w(\theta) = \frac{\pi_\theta(r|z)}{\pi_{\text{SFT}}(r|z)}$ indicates the ratio between the trainable model $\pi_\theta(\cdot)$ and SFT model $\pi_{\text{SFT}}(\cdot)$; $A(z,r) = \mathbf{U}(z,r) - \mathbb{E}_{r_i\sim\pi_{\text{SFT}}}[\mathbf{U}(z,r_i)]$ denotes the advantage of a rollout r, where $\mathbf{U}(z,r)$ takes 1 or 0, corresponding to whether r derives correct answers to a question z or not; and $\text{clip}(\cdot, 1-\varepsilon, 1+\varepsilon)$ is a clipping function with $0 \le \varepsilon \le 1$. The $\text{clip}(\cdot)$ operator constrains the output distribution to remain close to the SFT model.

The intuition behind AutoL2S is a combination of exploitation and exploration. *Exploitation:* $\pi_\theta(\cdot)$ is initialized from $\pi_{\text{SFT}}(\cdot)$, ensuring initial capability for long-short CoT selection for adaptive rollout generation. *Exploration:* The rollout advantage $A(z,r)$ serves as a correctness signal: $A(z,r) > 0$ if r yields correct answers, and $A(z,r) < 0$ otherwise. It encourages the model to explore correct solutions by reinforcing successful outcomes and discouraging incorrect ones with dynamic length of rollouts, thereby strengthening the adaptive reasoning behaviors learned after the SFT stage.

3.3 Theoretical Interpretation of AutoL2S

In this section, we provide a theoretical interpretation of AutoL2S to clarify the mechanisms underlying its training procedure. Lemma 1 in Appendix D offers an information-theoretic perspective, suggesting that conditioning short CoT reasoning paths on long reasoning can reduce learning uncertainty by supplying additional contextual information. This intuition naturally extends to the AutoL2S training setting, where optimization is performed via cross-entropy (equivalently, perplexity): concatenating long CoT paths with short ones effectively enriches the supervisory signal available for learning better short reasoning behaviors.

At the same time, reasoning length does not admit a unique ground-truth label. For many inputs, multiple reasoning paths of different lengths can yield correct answers, making long-short reasoning selection inherently instance-dependent. This motivates viewing reasoning-length choice as a trade-off between generation cost and uncertainty. We formalize this intuition in Theorem 1 in Appendix D, which frames long-short reasoning selection as a risk-cost trade-off: shorter reasoning reduces token cost but may increase uncertainty, while longer reasoning provides redundancy at higher computational expense.

From this unified perspective, paired long-short CoT supplies auxiliary information that reduces uncertainty when learning concise reasoning behaviors, thereby stabilizing short-path rollout generation while preserving correctness. This analysis is not used to derive the training objective of Au-

toL2S, nor does it claim optimality or theoretical guarantees. Rather, it serves as a conceptual lens for understanding why structured long-short supervision supports adaptive reasoning compression, consistent with the empirical results in Section 4.

4 Experiments

In this section, we evaluate AutoL2S as a length-aware reasoning method and, more importantly, analyze how long-short rollout design influences the trade-off between reasoning length, accuracy, and training stability. In addition, we conduct experiments to evaluate the performance of AutoL2S framework. We aim to answer the following three research questions: **RQ1**: How does AutoL2S perform on LLM reasoning tasks in terms of accuracy and efficiency, when long-short reasoning is explicitly modeled? **RQ2**: How do supervision-induced reasoning rollouts affect efficiency-accuracy trade-offs during fine-tuning? **RQ3**: What mechanisms govern long-short rollout enable AutoL2S to reliably preserve performance under compression?

4.1 Datasets and Baselines

Datasets We train the AutoL2S framework on the Bespoke-Stratos-17k dataset (Labs, 2025) and evaluate it on six reasoning benchmarks: Math500 (Hendrycks et al., 2021), GPQA-Diamond (GPQA) (Rein et al., 2024), GSM8K (Cobbe et al., 2021), OlympiadBench-Math (He et al., 2024), AIME24, and MMLU-pro (Wang et al., 2024). Additional dataset statistics and preprocessing details are provided in Appendix B. **Baseline Methods** We compare AutoL2S framework with the five state-of-the-art baselines to assess the effectiveness of length reduction and performance preservation. The baselines are listed as follows: R1-Distilled reasoning LLMs (Bespoke-Stratos-3B/7B) (Yeo et al., 2025), O1-pruner (Luo et al., 2025), CoT-Valve (Ma et al., 2025a), DPO (Rafailov et al., 2023), TokenSkip (Xia et al., 2025), and AlphaOne [1] (Zhang et al., 2025a). More details are listed in Appendix C.

4.2 Experimental Settings

Evaluation of Efficient LLM Reasoning. Following the settings of (Luo et al., 2025; Yeo et al., 2025), we evaluate reasoning efficiency using two metrics: (1) accuracy and (2) output token length, which directly reflects the amount of computation incurred during inference with efficiency-accuracy trade-off. The goal is to preserve reasoning performance while minimizing token usage, as shorter outputs under autoregressive decoding directly reduce inference computation. Shorter generations correspond to more concise reasoning and lower decoding cost, while longer generations indicate increased reasoning effort.

Implementation Details. To demonstrate the flexibility of AutoL2S across different LLM backbones, we train the framework using two non-reasoning base LLMs: Llama3.2-3B-Instruct (Touvron et al., 2023) and Qwen2.5-7B-Instruct. The short reasoning samples are generated via rejection sampling with sampling numbers $k \in \mathbb{N}$ using the Qwen2.5-Math-7B-Instruct model, following the settings of (Yeo et al., 2025; Yang et al., 2025). We filter out duplicate question-answer pairs that appear with both `<EASY>` and `<Long Trigger>` after rejection sampling, retaining only the pairs associated with `<EASY>` in such cases. We employ DeepSeek-R1 (Guo et al., 2025) as the strong reasoning-capable teacher model for generating L and Qwen2.5-Math-7B-Instruct (Yang et al., 2024) as a short CoT teacher for generating S. More details are in Appendix E.

4.3 Reasoning Efficiency of AutoL2S (RQ1)

We first report the overall accuracy and efficiency of AutoL2S across multiple reasoning benchmarks. Improvements in efficiency should be interpreted jointly with accuracy preservation and reasoning length. Additional results from repetition experiments are provided in Appendix F. We calculate the improvement percentile relative to the Bespoke-Stratos-3B/7B model, a strong baseline finetuned on the Bespoke-Stratos-17k. We conclude the observations as follows:

- **Baseline Comparison.** Table 1 reports reasoning accuracy and generated token length across benchmarks. AutoL2S achieves substantial reductions in reasoning length while preserving accuracy compared to strong baselines. Notably, these gains are obtained by explicitly modeling long and short reasoning as distinct rollout modes, rather than relying on implicit or fixed-length rollouts as in prior methods. Compared to baselines, AutoL2S achieves the best efficiency-accuracy trade-off and further compresses reasoning paths by up to 71.7% with negligible ac-

[1] Not compatible with Llama-family with vLLM.

Table 1: Accuracy (Acc) and Token Length (Len) across six reasoning benchmarks. Values in parentheses denote the accuracy improvement and token reduction relative to the Bespoke-Stratos-3B/7B model. "AutoL2S-SFT" defines AutoL2S with only SFT Stage, and "w/o RJ" defines AutoL2S without rejection sampling. Purple and blue cells highlight the best and second-best values, respectively.

	Average		MATH500		GPQA		GSM8K		Olympiad		AIME		MMLU-Pro	
	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len
Llama-3.2-3B-Instruct														
Llama-3.2-3B-Instruct	0.357	1015	0.404	740	0.293	498	0.729	203	0.147	2117	0.067	2053	0.500	477
Bespoke-Stratos-3B	0.413	10219	**0.574**	10148	0.273	8888	0.822	1387	**0.246**	15635	0.033	21341	**0.529**	3912
CoT-Valve	0.363 (-0.050)	11941 (+16.9%)	0.478 (-0.096)	10890 (+7.3%)	0.283 (+0.010)	9634 (+8.4%)	0.773 (-0.049)	2238 (+61.4%)	0.154 (-0.092)	18634 (+19.2%)	0.033 (+0.000)	26059 (+22.1%)	0.457 (-0.072)	4191 (+7.1%)
O1-pruner	0.402 (-0.011)	5995 (-41.3%)	0.562 (-0.012)	5295 (-47.8%)	0.308 (+0.035)	5394 (-39.3%)	0.816 (-0.006)	**860 (-38.0%)**	**0.236 (-0.010)**	**8622 (-44.9%)**	0.033 (+0.000)	**12074 (-43.4%)**	0.457 (-0.072)	3724 (-4.8%)
DPO	0.399 (-0.014)	7354 (-28.0%)	**0.574 (+0.000)**	5363 (-47.2%)	0.283 (+0.010)	6740 (-24.2%)	**0.832 (+0.010)**	911 (-34.3%)	0.227 (-0.019)	10441 (-33.2%)	0.033 (+0.000)	17456 (-18.2%)	0.443 (-0.086)	3215 (-17.8%)
TokenSkip	0.379 (-0.033)	10579 (+3.5%)	0.512 (-0.062)	10327 (+1.8%)	0.258 (-0.015)	9438 (+6.2%)	0.801 (-0.021)	2238 (+61.4%)	0.191 (-0.055)	15853 (+1.4%)	0.000 (-0.033)	21122 (-1.0%)	**0.514 (-0.015)**	4496 (+14.9%)
AutoL2S-SFT (w/o RJ)	**0.418 (+0.005)**	8280 (-19.0%)	0.552 (-0.022)	5990 (-41.0%)	**0.389 (+0.116)**	7520 (-15.4%)	0.823 (+0.001)	1166 (-15.9%)	0.206 (-0.040)	12941 (-17.2%)	**0.067 (+0.034)**	19158 (-10.2%)	0.471 (-0.058)	2906 (-25.7%)
AutoL2S (w/o RJ)	0.410 (-0.003)	**5954 (-41.7%)**	0.564 (-0.010)	6508 (-35.9%)	0.359 (+0.086)	**4569 (-48.6%)**	0.815 (-0.007)	964 (-30.5%)	0.230 (-0.016)	9269 (-40.7%)	0.033 (+0.000)	12494 (-41.5%)	0.457 (-0.072)	**1919 (-50.9%)**
AutoL2S-SFT	0.389 (-0.024)	6677 (-34.7%)	0.546 (-0.028)	**4181 (-58.8%)**	**0.369 (+0.096)**	6165 (-30.6%)	0.800 (-0.022)	1021 (-26.4%)	0.218 (-0.028)	10706 (-31.5%)	0.000 (-0.033)	14775 (-30.8%)	0.400 (-0.129)	3211 (-17.9%)
AutoL2S	**0.415 (+0.002)**	**4803 (-53.0%)**	0.520 (-0.054)	**4116 (-59.4%)**	0.273 (+0.000)	**3679 (-58.6%)**	**0.826 (+0.004)**	**819 (-41.0%)**	0.193 (-0.053)	**7199 (-54.0%)**	**0.167 (+0.134)**	**11538 (-45.9%)**	**0.514 (-0.015)**	**1469 (-62.4%)**
Qwen2.5-7B-Instruct														
Qwen2.5-7B-Instruct	0.520	529	0.748	556	0.308	27	0.902	260	0.384	896	0.133	1014	0.643	423
Bespoke-Stratos-7B	0.590	7430	0.824	5383	0.359	6049	0.926	1321	0.444	11322	0.200	18513	**0.786**	1989
CoT-Valve	0.543 (-0.047)	5942 (-20.0%)	0.730 (-0.094)	4483 (-16.7%)	0.369 (+0.010)	4930 (-18.5%)	0.898 (-0.028)	928 (-29.7%)	0.378 (-0.066)	8647 (-23.6%)	0.167 (-0.033)	14304 (-22.7%)	0.714 (-0.072)	2362 (+18.8%)
O1-pruner	0.581 (-0.009)	6773 (-8.8%)	**0.832 (+0.008)**	5104 (-5.2%)	0.399 (+0.040)	5312 (-12.2%)	**0.936 (+0.010)**	1065 (-19.4%)	0.433 (-0.011)	9586 (-15.3%)	0.200 (+0.000)	17655 (-4.6%)	0.686 (-0.100)	1916 (-3.7%)
DPO-Bespoke	**0.593 (+0.003)**	6073 (-18.3%)	0.806 (-0.018)	3688 (-31.5%)	0.374 (+0.015)	5961 (-1.5%)	0.920 (-0.006)	1576 (+19.3%)	**0.447 (+0.003)**	7364 (-35.0%)	**0.267 (+0.067)**	15991 (-13.6%)	0.743 (-0.043)	1858 (-6.6%)
TokenSkip	0.565 (-0.024)	7960 (+7.1%)	**0.826 (+0.002)**	5335 (-0.9%)	**0.434 (+0.075)**	5508 (-9.0%)	0.918 (-0.008)	1165 (-11.8%)	**0.447 (+0.003)**	10947 (-3.3%)	0.067 (-0.133)	22750 (+22.9%)	0.700 (-0.086)	2054 (+3.3%)
AlphaOne	0.519 (-0.071)	4441 (-40.2%)	0.732 (-0.092)	3867 (-28.2%)	0.313 (-0.046)	6278 (+3.8%)	0.907 (-0.019)	1943 (+47.1%)	0.356 (-0.088)	5252 (-53.6%)	0.133 (-0.067)	7162 (-61.3%)	0.671 (-0.115)	2146 (+7.9%)
AutoL2S-SFT (w/o RJ)	**0.600 (+0.010)**	6314 (-15.0%)	0.800 (-0.024)	3468 (-35.6%)	**0.434 (+0.075)**	4777 (-21.0%)	**0.934 (+0.008)**	735 (-44.4%)	**0.470 (+0.026)**	9068 (-19.9%)	**0.233 (+0.033)**	18332 (-1.0%)	0.729 (-0.057)	1504 (-24.4%)
AutoL2S (w/o RJ)	0.561 (-0.029)	**2299 (-69.1%)**	0.798 (-0.026)	**1601 (-70.3%)**	0.414 (+0.055)	**2666 (-55.9%)**	0.912 (-0.014)	707 (-46.5%)	0.439 (-0.005)	**3088 (-72.7%)**	0.100 (-0.100)	**4638 (-74.9%)**	0.700 (-0.086)	1091 (-45.1%)
AutoL2S-SFT	0.558 (-0.032)	4886 (-34.2%)	0.798 (-0.026)	2416 (-55.1%)	0.394 (+0.035)	3492 (-42.3%)	0.929 (+0.003)	**488 (-63.1%)**	0.436 (-0.008)	6459 (-43.0%)	0.133 (-0.067)	15399 (-16.8%)	0.657 (-0.129)	**1064 (-46.5%)**
AutoL2S	0.573 (-0.017)	**2103 (-71.7%)**	0.804 (-0.020)	**1405 (-73.9%)**	0.404 (+0.045)	**2798 (-53.7%)**	0.923 (-0.003)	**663 (-49.8%)**	0.435 (-0.009)	**2546 (-77.5%)**	0.100 (-0.100)	**4146 (-77.6%)**	**0.771 (-0.015)**	**1058 (-46.8%)**

curacy degradation, confirming the effectiveness of the proposed length-aware fine-tuning.

- **Rejection Sampling.** We observe that moderate rejection sampling in the SFT stage benefits the rollout generation for RL stage training. We conduct a study denoted as "w/o RJ," where rejection sampling is disabled by setting $k = 0$ during the SFT stage, and compare it against the default setting with AutoL2S with rejection sampling $k = 8$. Although applying rejection sampling may lead to a minor accuracy drop after supervised fine-tuning, it substantially improves the quality of the resulting long-short reasoning behavior during training. In particular, the generated reasoning becomes significantly shorter while the accuracy degradation is largely mitigated, yielding relative improvements of $\sim$71% and $\sim$53% for the 3B and 7B models, respectively. Overall, rejection sampling improves AutoL2S's accuracy–efficiency trade-off.

- **Efficiency Analysis.** Appendix F.1 presents the trade-off between accuracy vs. token usage, demonstrating the best efficiency-accuracy trade-offs compared to all other baselines.

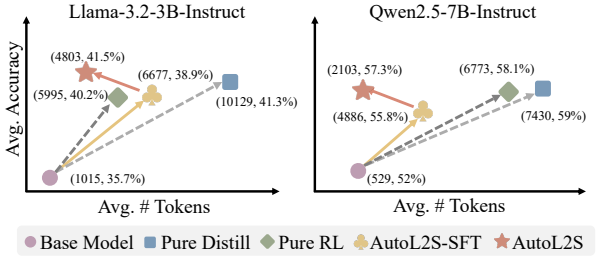


Figure 3: Optimization trajectories of AutoL2S, showing the trade-off between reasoning length and accuracy.

4.4 Impact of Training Dynamics (RQ2)

In this section, we analyze the training dynamics of AutoL2S to better understand how the training paradigm affects the trade-off between reasoning efficiency and accuracy. Figure 3 illustrates the optimization trajectories of AutoL2S on both the 3B and 7 B-based models, showing the trade-off between reasoning length (measured by the average number of generated tokens) and accuracy. Compared to the base instruction model and long-only

Table 2: Comparison of Different Rollout Configurations for Training AutoL2S under Qwen2.5-7B-Instruct

Method	Average		MATH500		GPQA		GSM8K		Olympiad		AIME24		MMLU-pro	
	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len
AutoL2S-SFT	0.558	4886	0.798	2416	0.394	3492	0.929	488	0.436	6459	0.133	15399	0.657	1064
AutoL2S	0.573	2103	0.804	1405	0.404	2798	0.923	663	0.435	2546	0.100	4146	0.771	1058
w/ Force-short	0.547	2399	0.778	1305	0.399	2544	0.924	578	0.438	2598	0.100	5898	0.643	1468
Δ vs AutoL2S	-0.026	+14.1%	-0.026	-7.1%	-0.005	-9.1%	+0.001	-12.8%	+0.003	+2.0%	+0.000	+42.3%	-0.128	+38.8%
w/ Force-long	0.570	2428	0.796	1606	0.369	2529	0.923	620	0.453	2726	0.167	5695	0.714	1389
Δ vs AutoL2S	-0.003	+15.4%	-0.008	+14.3%	-0.035	-9.6%	+0.000	-6.5%	+0.018	+7.1%	+0.067	+37.4%	-0.057	+31.3%

Table 3: Performance of different annotation format.

Method	Avg. Acc	Avg. Len
Long-only Distill	0.668	5213
Long-short Separated Distill	0.644	3682
Short-Long Distill	0.622	2059
Long-Short Distill (w/o RJ)	0.674	3910
Long-Short Distill	0.643	2784

distillation, AutoL2S leverages paired long-short distillation to achieve both higher accuracy and shorter reasoning paths. Compared to Pure RL, AutoL2S leverages adaptive long-short rollouts for training and achieves a better trade-off between accuracy and efficiency. Overall, these trajectories show that enabling instance-wise long-short reasoning allows AutoL2S to substantially reduce reasoning length while preserving task performance.

4.5 Impact of Annotation Format (RQ2)

We next analyze how the annotation format influences the effective long-short rollout behavior learned during training. Rather than directly controlling rollout composition, we reinterpret long-short reasoning annotation as a proxy that modulates the optimization signal for short-mode learning. By varying the availability and quality of short-mode supervision, we implicitly alter how frequently and reliably the model adopts short rollouts at inference time. To further disentangle these effects, we additionally report accuracy and reasoning length conditioned on rollout mode (long vs. short), revealing distinct performance profiles across modes. (1) *Long-only Distill* represents the original distillation from only long reasoning in the Bespoke-Stratos-17k reasoning dataset; (2) *Short-long Distill* switches the position of long and short reasoning path; and (3) *Long-short Separated Distill* separately constructs the long and short CoT reasoning paths. All results are demonstrated in Table 3. Compared with other formats of long-short term annotation, we observe that *Long-Short Distill* achieves the best performance in terms of accuracy preservation and output length.

4.6 Adaptive Rollout Behaviors (RQ2)

In this section, we analyze the reliability of the <EASY> token as a rollout selector, examining when the model chooses short reasoning and whether such decisions preserve correctness that benefits training AutoL2S. We conduct the ablation studies: (1) "w/ Force-Short" refers to the setting where `<Short Trigger>` is always used to generate only short rollout, and (2) "w/ Force-Long" denotes the setting where `<Longer Trigger>` is consistently used to initiate long-only rollout generation. The results are showcased in Table 2. Compared to the "Force-Long" case, AutoL2S obtains a similar reasoning accuracy on average while generating around 15% shorter reasoning lengths. Furthermore, we compare AutoL2S with "w/ Force-Short" variants. We observe that AutoL2S outperforms the "w/ Force-Short" in both reasoning accuracy and length. An explanation may be that uniformly enforcing shorter reasoning introduces noisy training signals: aimless truncating reasoning paths can remove necessary intermediate steps, leading to substantial accuracy degradation. In contrast, AutoL2S instead reinforces reasoning behaviors with adaptive rollouts, allowing the model to learn when shorter reasoning is appropriate. This enables more reliable trade-offs between accuracy and efficiency during training.

4.7 Impact of Rejection Sampling (RQ2)

In this section, we conduct a sensitivity analysis on the rejection size k. Specifically, we evaluate $k \in \{0, 4, 8\}$ under both the Llama-3.2-3B-Instruct and Qwen2.5-7B-Instruct models. The results are reported in Figure 5. Recall that the larger size of rejection sampling k would lead to a larger proportion of short CoT paths appearing in training data $\mathcal{D}$. In Figure 5-(a), we observe that using a larger size k leads to a more favorable trade-off between reasoning accuracy and generation length under AutoL2S. This trend is consistent across both base models, with $k = 8$ achieving the best overall trade-off compared to $k = 0$ and $k = 4$. Addi-

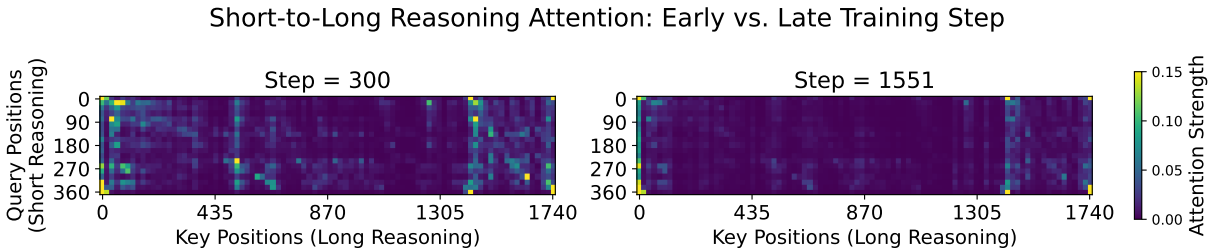


Figure 4: Comparison of attention maps at early and late training steps of AutoL2S. Step 1551 corresponds to the final training step. Given the long sequence lengths, we group every 20 tokens together to calculate attention scores between long and short reasoning paths for better visualization.

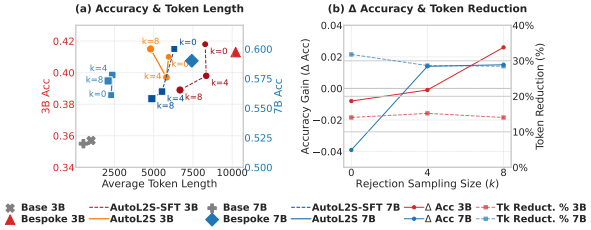


Figure 5: Accuracy and Efficiency Trade-off on different rejection sampling sizes k. (a) The Trade-off between Accuracy and Efficiency. (b) Performance Improvement from AutoL2S-SFT to AutoL2S

tionally, a larger size k results in shorter reasoning lengths on average in the SFT stage, but causes more accuracy degradation. This may occur because minimizing reasoning length increases the information bottleneck in supervision, reducing redundancy that can otherwise stabilize learning for complex reasoning cases. In Figure 5-(b), the results reveal that $k = 8$ leads to the best improved rate of accuracy–efficiency trade-offs compared to $k = 0$ and $k = 4$ for the SFT model. One possible reason is that a larger k encourages in SFT supervision tends to favor aggressively shorter reasoning paths, resulting in shorter but less robust reasoning, and consequently, lower accuracy with shorter reasoning lengths when used alone. When such SFT models (i.e., larger k) are used as reference policies for RL fine-tuning, outcome-based feedback restores accuracy while retaining the efficiency-oriented behavior, resulting in improved accuracy–efficiency trade-offs with even shorter reasoning lengths. Thus, we select $k = 8$ as our final configuration for AutoL2S in Table 1.

4.8 Mechanism behind the Auto Long-short Reasoning (RQ3)

In this section, we discuss the mechanism explanation of AutoL2S training. To assess the mechanism behind, Figure 4 presents the attention map comparisons across different training steps of AutoL2S, highlighting the benefit of the concatenation order used in *Long-Short Distill*. In the early stages of training (i.e., Figure 4 left side: training step 300), we observe that long CoT reasoning paths significantly impact the attention patterns of short CoT reasoning paths, indicating that long-form reasoning benefits the learning of short reasoning generation. As training progresses till the end (i.e., Figure 4 right side: training step 1551), the correlation between long and short CoT reasoning paths significantly diminishes, indicating that they evolve into two distinct components. This separation explains why AutoL2S is effective and flexible in switching to easy questions simply using the `<Short Trigger>` when the `<EASY>` token is presented during inference. The phenomenon again meets the properties of Lemma 1, where long CoT reasoning paths provide auxiliary information for short-path learning. This also explains the reason why the direct use of `<Short Trigger>` remains effective, without introducing dummy key-value pairs or modifying positional encodings.

5 Conclusion

This paper presents Auto Long–Short Reasoning (AutoL2S), a distillation framework that mitigates overthinking while preserving accuracy. By pairing verified long and short CoT paths, AutoL2S learns a lightweight switching token that enables instance-wise selection between concise and extended reasoning, and further refines efficiency through post-distillation GRPO-style training without enforcing rigid length constraints. Experiments show that AutoL2S reduces reasoning length by up to 71% with minimal performance loss, achieving favorable trade-offs between token usage, inference time, and accuracy. These results demonstrate the effectiveness of structured long–short supervision, combined with the proposed AutoL2S framework, for efficient reasoning in distilled LLMs.

Limitations

While AutoL2S effectively balances reasoning accuracy and efficiency, its performance depends on the availability and quality of paired long and short reasoning annotations, which may be costly to obtain for some domains. In addition, the binary distinction between long and short reasoning paths may not fully capture more fine-grained variations in reasoning complexity. Our efficiency measurements focus on autoregressive decoding settings, and the gains may differ under alternative inference paradigms. We leave extensions to richer reasoning taxonomies and broader deployment settings to future work.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.

Anthropic. 2023. Claude.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yingqian Cui, Pengfei He, Jingying Zeng, Hui Liu, Xianfeng Tang, Zhenwei Dai, Yan Han, Chen Luo, Jing Huang, Zhen Li, and 1 others. 2025. Stepwise perplexity-guided refinement for efficient chain-of-thought reasoning in large language models. *arXiv preprint arXiv:2502.13260*.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025. Thinkless: Llm learns when to think. *arXiv preprint arXiv:2505.13379*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Tingxu Han, Chunrong Fang, Shiyu Zhao, Shiqing Ma, Zhenyu Chen, and Zhenting Wang. 2024. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*.

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

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*.

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

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*.

Bespoke Labs. 2025. Bespoke-stratos: The unreasonable effectiveness of reasoning distillation. https://www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable/-effectiveness-of-reasoning-distillation. Accessed: 2025-01-22.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2024. Can language models learn to skip steps? *arXiv preprint arXiv:2411.01855*.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025a. Cot-valve: Length-compressible chain-of-thought tuning. *arXiv preprint arXiv:2502.09601*.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025b. Cot-valve: Length-compressible chain-of-thought tuning, 2025. *URL https://arxiv. org/abs/2502*, 9601.

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

OpenAI. Learning to reason with llms. urlhttps://openai.com/index/learning-to-reason-with-llms/.

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

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

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, and Shiguo Lian. 2025. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv preprint arXiv:2503.04472*.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Qwen Team. Qwq-32b-preview. urlhttps://qwenlm.github.io/blog/qwq-32b-preview/.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.

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

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, and 22 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Wang Yang, Hongye Jin, Jingfeng Yang, Vipin Chaudhary, and Xiaotian Han. 2025. Thinking preference optimization. *arXiv preprint arXiv:2502.13173*.

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

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. 2024. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*.

Zhaojian Yu, Yinghao Wu, Yilun Zhao, Arman Cohan, and Xiao-Ping Zhang. 2025. Z1: Efficient test-time scaling with code. *Preprint*, arXiv:2504.00810.

Junyu Zhang, Runpei Dong, Han Wang, Xuying Ning, Haoran Geng, Peihao Li, Xialin He, Yutong Bai, Jitendra Malik, Saurabh Gupta, and 1 others. 2025a. Alphaone: Reasoning models thinking slow and fast at test time. *arXiv preprint arXiv:2505.24863*.

Yifan Zhang and Team Math-AI. 2024. American invitational mathematics examination (aime) 2024.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025b. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.

Appendix

A Related Work

Reasoning-capable LLMs. Recent advancements in LLMs have significantly enhanced their reasoning capabilities, exemplified by large reasoning models such as OpenAI o1 (OpenAI) and DeepSeek-R1 (Guo et al., 2025), and QwQ-32B (Team). OpenAI o1 (OpenAI) introduces advanced reasoning mechanisms designed to tackle complex problems, such as mathematical and programming tasks. Similarly, DeepSeek-R1 (Guo et al., 2025) enhances reasoning abilities by employing RL to incentivize effective reasoning behaviors. Additionally, DeepSeek-R1 curates specialized reasoning datasets, enabling the explicit distillation of reasoning capabilities into smaller models through SFT.

Efficient LLM Reasoning. Thinking steps of LLMs have become longer, leading to the "overthinking problem" (Chen et al., 2024; Sui et al., 2025). To mitigate lengthy responses and reasoning processes, several works have been conducted to shorten the thinking steps and produce more concise reasoning (Sui et al., 2025). *RL-based methods* aim to encourage full-length reasoning models to generate concise thinking steps or train non-reasoning models to learn efficient reasoning by incorporating a length-aware reward (Team et al., 2025; Luo et al., 2025; Aggarwal and Welleck, 2025; Yeo et al., 2025; Shen et al., 2025; Hou et al., 2025). Specifically, they propose designing a length-based score to penalize excessively lengthy responses, complementing original rewards (e.g., format reward and accuracy reward). Kimi K1.5 (Team et al., 2025) calculates a length reward based on the response length relative to the shortest and longest responses. L1 (Aggarwal and Welleck, 2025) modifies the training data with the designated length constraint instruction, and then add the length reward. O1-Pruner (Luo et al., 2025) introduces the length-harmonizing reward, which calculates the ratio of lengths between the reference model and predicted model along with the accuracy-based constraints.

SFT-based methods curate variable-length CoT training datasets to fine-tune overthinking reasoning models for shorter reasoning paths or to equip non-reasoning models with efficient reasoning capabilities (Han et al., 2024; Xia et al., 2025; Ma et al., 2025a; Yu et al., 2025; Cui et al., 2025). Specifically, based on long CoT reasoning paths, they curate shorter yet accurate CoT reasoning paths as training data. Token-skip (Xia et al., 2025) leverages LLMLingua (Jiang et al., 2023) to compress lengthy CoT responses into shorter ones based on semantic scores, and then fine-tunes the model for efficient reasoning. CoT-Valve (Ma et al., 2025a) controls the magnitude of LoRA (Hu et al., 2022) weights to generate variable-length CoT training data, which are then used to fine-tune an efficient reasoning model. Token-Budget (Han et al., 2024) assigns specific token budgets to prompts in order to generate shorter reasoning steps, and these concise CoT examples are then used for model fine-tuning.

B Details of Evaluation Dataset

We train the AutoL2S framework under the Bespoke-Stratos-17k (Labs, 2025) dataset and assess the framework on the long-to-short reasoning task under four different reasoning datasets. The details of the assessment datasets are provided as follows:

- **Math500** (Hendrycks et al., 2021): A challenging benchmark consisting of 500 high-quality math word problems that require multi-step symbolic reasoning.
- **GPQA-Diamond (GPQA)** (Rein et al., 2024): The Graduate-Level Physics Question Answering (GPQA) dataset contains 198 multiple-choice questions from graduate-level physics exams.
- **GSM8K** (Cobbe et al., 2021): A widely-used benchmark comprising 1319 grade school-level math word problems.
- **Olympiad Bench Math (Olympiad)** (He et al., 2024): A collection of 674 math competition problems inspired by middle and high school mathematics Olympiad competitions.
- **AIME** (Zhang and Math-AI, 2024): A benchmark consisting of 30 problems from the 2024 American Invitational Mathematics Examination.

- **MMLU-Pro** (Wang et al., 2024): A robust multi-task benchmark that enhances evaluation rigor by expanding answer options and curating high-quality problems across diverse disciplines to mitigate random guessing.

C Details of Baseline Implementation

C.1 Bespoke-Stratos

We implement this baseline by fully fine-tuning language models on the Bespoke-Stratos-17k dataset, which comprises 17,000 examples of questions, long-form reasoning traces, and corresponding answers. The resulting model serves as an oracle reference for reasoning performance.

Following standard SFT procedures, training is performed by minimizing the standard cross-entropy loss over the input sequence. We employ the AdamW optimizer with a learning rate of $1\mathrm{e}{-5}$ and a batch size of 32. Fine-tuning is conducted for three epochs on two NVIDIA A100 80GB GPUs with mixed-precision training enabled. For the 7B base model, we directly utilize the publicly released checkpoint VanWang/Bespoke-Stratos-7B-repro-SFT.

C.2 O1-pruner

O1-pruner introduces a Length-Harmonizing Reward, integrated with a GRPO-style loss, to optimize the policy model π_θ and reduce the length of generated chain-of-thought (CoT) reasoning. Considering the effectiveness of off-policy training with pre-collected data, O1-pruner adopts an off-policy training approach by sampling from the reference model π_{SFT} rather than from π_θ. Specifically, the training procedure consists of two steps: (1) generating CoT samples using π_{SFT}, and (2) fine-tuning the policy model with the proposed GRPO-style objective based on the generated samples.

In our implementation, we follow the original experimental setting and reproduce the method based on its official repository.[2] For training, we sample 5,000 problems from the Bespoke-Stratos-17k dataset and generate 16 solutions for each problem. We then perform length-harmonizing fine-tuning for one epoch to jointly optimize both output length and answer correctness. To ensure fair comparison with our method, we use Bespoke-Stratos-3B/7B as the reference model and set the maximum sequence length to 10,240 tokens when training.

C.3 CoT-Valve

COT-Valve is designed to enable models to generate reasoning chains of varying lengths. It controls the length of reasoning by linearly combining the LoRA weights of the distilled long-form reasoning CoT and the non-reasoning model. For the specific Long to Short CoT task, it has three stages: (1) finetune the LLM base model on a long-cot dataset using Lora to identify a direction in the parameter space that control the length of generated CoT(2) merge Lora weights with the base model at varying interpolation ratios generate models and use them construct datasets containing CoT of decreasing lengths (3) finetuning the distilled reasoning model with the generated dataset in a progressive way, where the model is trained with shorter reasoning path samples between epochs. This progressive training strategy enables the model to gradually compress its reasoning while maintaining correctness.

In our implementation, we follow the original configuration in CoT-Valve. The LoRA rank and LoRA alpha are set to 32 and 64, respectively, for both the first and third stages. In the first stage, we finetune the non-reasoning models Llama-3.2-3B-Instruct/Qwen2.5-7B-Instruct on the Bespoke-Stratos-17k dataset for three epochs using Lora. The learning rate is 4e-5 and the batch size is 64. In the second stage, we apply LoRA weight interpolation with coefficients 0.8 and 0.6. Due to resource constraints, we randomly sample 2,000 questions for each interpolated model to generate responses, and retain only those samples with correct answers. In the third stage, the model obtained in the first stage is further fine-tuned for 2 epochs on each type of generated dataset, using the same learning rate of 4e-5 and a batch size of 64.

[2] https://github.com/StarDewXXX/O1-Pruner

D Theorem Statements and Proofs

In this section, we present and prove Lemma 1 and Theorem 1, with accompanying remarks to provide intuitive explanations.

Lemma 1 (Concatenation Advantage for Long–Short CoT Training). *Let X denote the input, $L = (\ell_1, \ldots, \ell_{T_L})$ the long-CoT token sequence, and $S = (s_1, \ldots, s_{T_S})$ the short-CoT token sequence, with training order L to S. Then, the conditional entropy $H(\cdot|\cdot)$ of the next short token satisfies:*

$$H(S_t \mid X, L, S_{<t}) \;\le\; H(S_t \mid X, S_{<t}), \quad \forall\, t \in [1, T_S]. \tag{1}$$

Equivalently, averaging across all positions with the improvement quantified as

$$\frac{1}{T_S}\sum_{t=1}^{T_S}\Big[H(S_t \mid X, S_{<t}) - H(S_t \mid X, L, S_{<t})\Big] = \frac{1}{T_S}\sum_{t=1}^{T_S} I(S_t; L \mid X, S_{<t}) \;\ge 0. \tag{2}$$

Thus, the long CoT reasoning path L provides additional mutual information $I(\cdot|\cdot)$ that strictly increases the entropy of the short CoT reasoning path S whenever L is informative about S.

Proof. The inequality follows directly from the fact that conditioning reduces entropy: adding L to the conditioning set cannot increase the uncertainty of S_t. Formally, for each $t \in [T_S]$,

$$H(S_t \mid X, L, S_{<t}) \;\le\; H(S_t \mid X, S_{<t}).$$

Averaging over t yields the stated inequality.

The gap between the two sides can be expressed as the conditional mutual information:

$$\frac{1}{T_S}\sum_{t=1}^{T_S}\big[H(S_t \mid X, S_{<t}) - H(S_t \mid X, L, S_{<t})\big] = \frac{1}{T_S}\sum_{t=1}^{T_S} I(S_t; L \mid X, S_{<t}) \;\ge 0.$$

In the realizable training case under long CoT reasoning path distillation, the model is optimized with the per-token cross-entropy objective

$$\mathrm{CE}(S \mid \mathcal{C}) = \frac{1}{T_S}\sum_{t=1}^{T_S}\mathbb{E}\big[-\log p_\theta(s_t \mid \mathcal{C}, S_{<t})\big],$$

where the context $\mathcal{C}$ is either (X) or (X, L). When p_θ matches the true distribution, the cross-entropy coincides with the entropy above. Thus, the same inequality carries over to cross-entropy:

$$\mathrm{CE}(S \mid X, L) \;\le\; \mathrm{CE}(S \mid X),$$

with the gap equal to the average conditional mutual information. Finally, since perplexity is defined as $\mathrm{PPL}(S \mid \mathcal{C}) = \exp(\mathrm{CE}(S \mid \mathcal{C}))$, the inequality extends directly to perplexity:

$$\mathrm{PPL}(S \mid X, L) \;\le\; \mathrm{PPL}(S \mid X).$$

□

Theorem 1 (Rollout Adaptation with <EASY> Token). *Let $p_\theta^L(\cdot \mid x)$ and $p_\theta^S(\cdot \mid x)$ denote the predictive distributions when decoding with the long and short CoT reasoning paths $L = (\ell_1, \ldots, \ell_{T_L})$ and $S = (s_1, \ldots, s_{T_S})$, respectively. Given an input $x \in \mathcal{Y}$, define the per-instance risks as*

$$J_S(x) = \mathbb{E}\Big[D\Big(p_\theta^S(\cdot \mid x) \,\Big\|\, p_\theta^L(\cdot \mid x)\Big)\Big] \;+\; \lambda\,\mathbb{E}[T_S(x)], \tag{3}$$

$$J_L(x) = \lambda\,\Big(\mathbb{E}[T_L(x)] + c_\pi\Big), \tag{4}$$

where $D(\cdot\|\cdot)$ is a statistical divergence, $T_S(x)$ and $T_L(x)$ denote the token lengths of the short and long CoT reasoning paths, $\lambda > 0$ is the per-token cost, and $c_\pi \geq 0$ is a fixed overhead for invoking the long path. Then the optimal adaptation policy is

$$\pi^*(x) = \begin{cases} 0 & \textit{if } J_S(x) < J_L(x) \quad (\textit{choose short}), \\ 1 & \textit{otherwise} \quad (\textit{choose long}). \end{cases} \tag{5}$$

Proof. We provide the proof within the following six steps.

Assumptions from AutoL2S Design. Let $\mathcal{Y}$ be the input space with data distribution $\mathcal{D}$. Assume $D(\cdot\|\cdot) \geq 0$ is a statistical divergence for which $\mathbb{E}[D(p_\theta^S(\cdot \mid x) \,\|\, p_\theta^L(\cdot \mid x))]$ exists, and the token lengths $T_S(x), T_L(x)$ are nonnegative random variables with finite expectations. Let an adaptation policy be a measurable mapping $\pi : \mathcal{Y} \to \{0, 1\}$, where $\pi(x)$=0 chooses short reasoning CoT and $\pi(x)$=1 chooses long reasoning CoT. For a policy π, define the population risk

$$\mathcal{R}(\pi) := \mathbb{E}_{x\sim\mathcal{D}}\Big[\, J_S(x)\, \mathbf{1}\{\pi(x) = 0\} + J_L(x)\, \mathbf{1}\{\pi(x) = 1\} \,\Big].$$

By the assumptions above, $\mathcal{R}(\pi)$ is well-defined and finite.

Step 1 (Reduction to deterministic policies). Consider any *randomized* policy that, for a fixed x, chooses short with probability $\alpha(x) \in [0, 1]$ and long with probability $1-\alpha(x)$. Its *conditional* (on x) contribution to risk equals

$$\alpha(x)\, J_S(x) + (1 - \alpha(x))\, J_L(x) = J_L(x) + \alpha(x)\, \Delta(x), \quad \text{where } \Delta(x) := J_S(x) - J_L(x).$$

Since this expression is linear in $\alpha(x)$, its minimum over $\alpha(x) \in [0, 1]$ is always achieved at an extreme point $\alpha(x) \in \{0, 1\}$:

$$\alpha^\star(x) = \begin{cases} 1, & \text{if } \Delta(x) < 0, \\ 0, & \text{if } \Delta(x) > 0, \\ \text{any in } [0,1], & \text{if } \Delta(x) = 0. \end{cases}$$

Hence, randomization cannot improve over a deterministic rule, and it suffices to prove that it optimizes over a deterministic policy π.

Step 2 (Pointwise decomposition). For any deterministic π,

$$\mathcal{R}(\pi) = \mathbb{E}\big[J_L(x)\big] + \mathbb{E}\big[\Delta(x)\, \mathbf{1}\{\pi(x) = 0\}\big].$$

The first term does not depend on π, so minimizing $\mathcal{R}(\pi)$ reduces to minimizing the second term. Because the expectation is taken with respect to $\mathcal{D}$ and the integrand depends on π only through the indicator, this is a pointwise decision:

Step 3 (Pointwise optimal action). For a fixed x:

$$\min_{a\in\{0,1\}} \big\{\, \Delta(x)\, \mathbf{1}\{a = 0\} \,\big\} = \begin{cases} \Delta(x), & \text{if } a = 0 \text{ and } \Delta(x) < 0, \\ 0, & \text{if } a = 1 \text{ or } \Delta(x) \geq 0, \end{cases}$$

which is achieved by choosing a=0 (short) when $\Delta(x) < 0$, and a=1 (long) otherwise. Thus, the Bayes-optimal policy is

$$\pi^\star(x) = \begin{cases} 0, & \text{if } \Delta(x) < 0 \;\; \big(\text{i.e., } J_S(x) < J_L(x)\big), \\ 1, & \text{otherwise.} \end{cases}$$

This is exactly the threshold rule stated in the theorem.

Step 4 (Existence and uniqueness). Existence follows because the pointwise minimum is always attained by an action in $\{0, 1\}$. Uniqueness holds everywhere except on the *tie set* $\{x : \Delta(x) = 0\}$ where both actions yield the same risk; changing $\pi^\star$ on this set does not alter $\mathcal{R}(\pi^\star)$. Hence, the optimal policy is unique almost surely (up to ties).

Step 5 (Explicit threshold and interpretation). Expanding $\Delta(x)$ gives

$$\Delta(x) = \underbrace{\mathbb{E}\Big[D\Big(p_\theta^S(\cdot \mid x) \,\Big\|\, p_\theta^L(\cdot \mid x)\Big)\Big]}_{\text{predictive distribution divergence}} + \lambda \Big(\mathbb{E}[T_S(x)] - \mathbb{E}[T_L(x)] - c_\pi\Big).$$

Thus $\pi^\star(x){=}0$ (choose short) iff the divergence penalty is outweighed by the token savings:

$$\mathbb{E}\Big[D\Big(p_\theta^S \,\Big\|\, p_\theta^L\Big)\Big] \;<\; \lambda \Big(\mathbb{E}[T_L(x)] + c_\pi - \mathbb{E}[T_S(x)]\Big).$$

Equivalently, *choose short when predicted distributions are sufficiently close* and *the token savings are large enough*.

Step 6 (Comparative statics). The decision boundary moves monotonically: increasing c_π or the long/short length gap $\mathbb{E}[T_L] - \mathbb{E}[T_S]$ makes short more favorable; increasing the divergence or decreasing the length gap makes long more favorable. Increasing λ amplifies the weight on token savings, thus favoring short when $\mathbb{E}[T_L] + c_\pi > \mathbb{E}[T_S]$. □

Remark 1. *Theorem 1 establishes that an optimal adaptation strategy between long and short CoT reasoning paths always exists and is essentially unique, reducing to a deterministic threshold rule. The policy selects the short path whenever the predictive distribution of the short rationale is sufficiently close to that of the long reasoning while offering enough token savings to offset the overhead of using the long path. This shows that the* `<EASY>` *token is not an ad hoc mechanism, but corresponds to a Bayes-optimal decision that balances semantic fidelity and inference efficiency. Together with Lemma 1, this highlights that the long reasoning paths not only improve the learnability of the short reasoning paths during training, but also guide optimal switching at inference time.*

E Details of Implementation and Instruction Prompt and Triggers

In this section, we introduce the format of instruction prompts and triggers that we utilized in our AutoL2S framework.

E.1 Details of Implementation Settings

All experiments for the 7B base model are conducted using four NVIDIA A100 80G GPUs, while those for the 3B base model utilize two NVIDIA A100 80GB GPUs. We leverage the Transformers library for fine-tuning and vLLM for efficient inference. Fine-tuning is performed using the AdamW optimizer with a learning rate of 1e-5. The temperature is fixed at 0.7 in both AutoL2S and baselines, ensuring that the output reasoning sequences are fully generated without truncation.

E.2 System prompt and Inference Prompt

System Prompt / Inference Prompt

Your role as an assistant is to solve problems using one of two ways.
First, think step-by-step and give the answer. The solution should be formatted as follows:
`<|begin_of_solution|>`
`{precise and concise solution}`
`<|end_of_solution|>`

Second, your output thoroughly explores questions through a systematic, long-term thinking process before providing the final, precise, and accurate solutions. This requires engaging in a comprehensive cycle of analysis, summarizing, exploration, reassessment, reflection, backtracing, and iteration to develop a well-considered thinking process.

Please structure your response into two main sections: Thought and Solution.

In the Thought section, detail your reasoning process using the specified format:
`<|begin_of_thought|>{thought with steps separated with '\n\n'}<|end_of_thought|>`

Each step should include detailed considerations such as analyzing questions, summarizing relevant findings, brainstorming new ideas, verifying the accuracy of the current steps, refining any errors, and revisiting previous steps.

In the Solution section, based on various attempts, explorations, and reflections from the Thought section, systematically present the final solution that you deem correct. The solution should remain a logical, accurate, concise expression style and detail the necessary steps needed to reach the conclusion, formatted as follows:
`<|begin_of_solution|>`
`{final formatted, precise, and clear solution}`
`<|end_of_solution|>`

Now, try to solve the following question through the above guidelines:

E.3 Special Token and Long-short Trigger

Special Easy Token and Long-short Trigger

`<EASY>` Token: `<specialLong>`

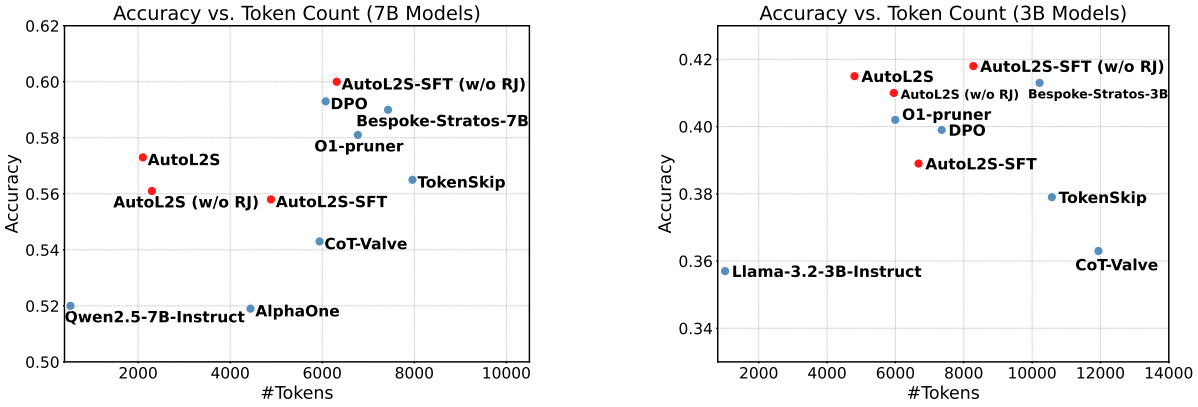
`Short Trigger`: This is a trigger to ensure the model's upcoming output `<EASY>`.

Long Trigger: Let's consider this problem in a `<pureLong>` way.

F Additional Experimental Results of AutoL2S

F.1 Pareto Front of Reasoning Accuracy and Efficiency

In this section, we showcase the Pareto Front of different methods in terms of their accuracy and token counts. The results are shown in Figure 6. We observe that AutoL2S obtains the best trade-off between accuracy and reasoning efficiency.



(a) Accuracy vs. Token Count (7B Models)

(b) Accuracy vs. Token Count (3B Models)

Figure 6: Efficiency-accuracy trade-off of AutoL2S. AutoL2S variants consistently improve accuracy under substantially lower inference cost.

F.2 Robustness Analytics of AutoL2S-SFT

To assess the robustness of our method, we further evaluated AutoL2S on both 3B and 7B models under three different runs with different random seeds. The reported values correspond to the mean and standard deviation with the same settings presented in Section 4. The **bold** numbers represent the best performance, and underline refers to the second best among the settings.

Based on the average performance, AutoL2S-SFT outperforms CoT-Valve by achieving higher accuracy and generating shorter reasoning paths. Compared to O1-pruner, AutoL2S-SFT produces shorter reasoning paths while maintaining comparable average accuracy across all four reasoning benchmarks. Furthermore, AutoL2S-SFT achieves nearly the same average accuracy as the oracle SFT R1-distilled models (i.e., Bespoke-Stratos-3B/7B), while significantly reducing reasoning path length. This presents the same observation showcased in Section 4.

Considering standard deviation, AutoL2S-SFT continues to outperform both the oracle SFT R1-distilled models and other baselines, offering better accuracy and lower average token usage. For example, with AutoL2S based on Qwen2.5-7B-Instruct, the performance remains the best among all methods, while also achieving the shortest reasoning lengths. These results demonstrate that AutoL2S-SFT has both competitive and robust performance in efficient reasoning tasks.

Table 4: Evaluation results of AutoL2S based on Qwen2.5-3B-Instruct.(mean ± std)

	Average		**MATH500**		**GPQA**		**GSM8K**		**Olympiad**	
	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len
Qwen2.5-3B-Instruct	0.479	777	0.622	806	0.349	770	0.679	376	0.266	1158
Bespoke-Stratos	0.516	8931	0.636	9246	0.308	10129	0.848	1624	0.272	14724
CoT-Valve	0.484	5889	0.602	4980	0.258	6898	0.805	1660	0.270	10017
O1-pruner	**0.535**	6686	**0.704**	6769	0.283	7348	**0.859**	1210	**0.295**	11416
AutoL2S-SFT	0.523 ±0.006	5083 ±737	0.656 ±0.015	4287 ±605	0.322 ±0.003	**4018 ±941**	0.830 ±0.026	1109 ±224	0.284 ±0.023	10919 ±1293
AutoL2S-SFT(w/ RJ $k = 4$)	0.524 ±0.009	3569 ±506	0.646 ±0.016	2713 ±135	**0.347 ±0.015**	4118 ±514	0.826 ±0.003	503 ±4	0.278 ±0.007	6942 ±1915
AutoL2S-SFT(w/ RJ $k = 8$)	0.523 ±0.007	**3255 ±548**	0.671 ±0.021	**2523 ±200**	0.317 ±0.008	4135 ±598	0.825 ±0.004	**417 ±41**	0.280 ±0.005	**5947 ±1796**

Table 5: Evaluation results of AutoL2S based on Qwen2.5-7B-Instruct.(mean ± std)

	Average		**MATH500**		**GPQA**		**GSM8K**		**Olympiad**	
	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len
Qwen2.5-7B-Instruct	0.586	435	0.748	556	0.308	27	0.902	260	0.384	896
Bespoke-Stratos	0.638	6019	0.824	5383	0.359	6049	0.926	1321	0.444	11322
CoT-Valve	0.594	4747	0.730	4483	0.369	4930	0.898	928	0.378	8647
O1-pruner	0.650	5267	**0.832**	5104	0.399	5312	**0.936**	1065	0.433	9586
AutoL2S-SFT	**0.652 ±0.007**	4348 ±306	0.795 ±0.005	3278 ±240	**0.431 ±0.006**	4590 ±532	0.923 ±0.011	595 ±150	**0.460 ±0.010**	8932 ±335
AutoL2S-SFT(w/ RJ $k = 4$)	0.630 ±0.011	3233 ±474	0.788 ±0.017	2200 ±354	0.375 ±0.033	3103 ±494	0.915 ±0.003	439 ±68	0.442 ±0.009	7190 ±994
AutoL2S-SFT(w/ RJ $k = 8$)	0.626 ±0.013	**2746 ±496**	0.785 ±0.012	**2019 ±368**	0.380 ±0.019	**2587 ±799**	0.915 ±0.015	**415 ±75**	0.422 ±0.016	**5964 ±921**

F.3 AutoL2S with Length Penalty Reward

This experiment evaluates the effect of introducing an explicit length-penalty reward in AutoL2S. While the penalty further reduces reasoning length beyond the default setting, it also increases the risk of accuracy degradation by encouraging overly aggressive compression. These results highlight the trade-off between enforcing conciseness through explicit rewards and preserving reasoning correctness.

Table 6: Effect of length penalty on the performance of 3B models ($r_j = 8$).

Method	**Average**		**MATH500**		**GPQA**		**GSM8K**		**Olympiad**		**AIME**		**MMLU-Pro**	
	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len
Llama-3.2-3B-Instruct	0.357	1015	0.404	740	0.293	498	0.729	203	0.147	2117	0.067	2053	0.500	477
Bespoke-Stratos-3B	0.413	10219	0.574	10148	0.273	8888	0.822	1387	0.246	15635	0.033	21341	0.529	3912
AutoL2S-SFT	0.389	6677	0.546	4181	0.369	6165	0.800	1021	0.218	10706	0.000	14775	0.400	3211
AutoL2S	0.415	4803	0.520	4116	0.273	3679	0.826	819	0.193	7199	0.167	11538	0.514	1469
AutoL2S (w/ len)	0.398	2190	0.550	1819	0.273	2048	0.810	353	0.233	3099	0.067	4971	0.457	848

G Full Experimental Results of AutoL2S on Rejection Sampling Ablation Studies

This section presents the full experimental results of AutoL2S under varying rejection sampling sizes. We analyze how different values of k affect reasoning accuracy and generation length during supervised distillation and subsequent refinement. The results highlight the role of rejection sampling in shaping the accuracy–efficiency trade-off of AutoL2S across models and training stages.

Table 7: Accuracy (Acc) and Token Length (Len) for 3B and 7B models with different rejection sampling ratios across reasoning benchmarks.

Method	**Average**		**MATH500**		**GPQA**		**GSM8K**		**Olympiad**		**AIME**		**MMLU-Pro**	
	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len
Llama-3.2-3B-Instruct														
Llama-3.2-3B-Instruct	0.357	1015	0.404	740	0.293	498	0.729	203	0.147	2117	0.067	2053	0.500	477
Bespoke-Stratos-3B	0.413	10219	0.574	10148	0.273	8888	0.822	1387	0.246	15635	0.033	21341	0.529	3912
AutoL2S (w/o RJ)	0.418	8280	0.552	5990	0.389	7520	0.823	1166	0.206	12941	0.067	19158	0.471	2906
AutoL2S-Plus (w/o RJ)	0.410	5954	0.564	6508	0.359	4569	0.815	964	0.230	9269	0.033	12494	0.457	1919
AutoL2S (w/ RJ $k = 4$)	0.398	8347	0.574	5666	0.283	7546	0.812	1322	0.226	12185	0.067	18134	0.429	5227
AutoL2S-Plus (w/ RJ $k = 4$)	0.397	5810	0.562	5517	0.288	4510	0.815	876	0.230	8257	0.000	12669	0.486	3029
AutoL2S (w/ RJ $k = 8$)	0.389	6677	0.546	4181	0.369	6165	0.800	1021	0.218	10706	0.000	14775	0.400	3211
AutoL2S-Plus (w/ RJ $k = 8$)	0.415	4803	0.520	4116	0.273	3679	0.826	819	0.193	7199	0.167	11538	0.514	1469
Qwen2.5-7B-Instruct														
Qwen2.5-7B-Instruct	0.520	529	0.748	556	0.308	27	0.902	260	0.384	896	0.133	1014	0.643	423
Bespoke-Stratos-7B	0.590	7430	0.824	5383	0.359	6049	0.926	1321	0.444	11322	0.200	18513	0.786	1989
AutoL2S-SFT (w/o RJ)	0.600	6314	0.800	3468	0.434	4777	0.934	735	0.470	9068	0.233	18332	0.729	1504
AutoL2S (w/o RJ)	0.561	2299	0.798	1601	0.414	2666	0.912	707	0.439	3088	0.100	4638	0.700	1091
AutoL2S-SFT (w/ RJ $k = 4$)	0.564	5531	0.786	2560	0.409	3495	0.917	509	0.438	7991	0.133	17220	0.700	1409
AutoL2S (w/ RJ $k = 4$)	0.578	2361	0.788	1477	0.465	2707	0.921	609	0.445	2450	0.133	5503	0.714	1417
AutoL2S-SFT (w/ RJ $k = 8$)	0.558	4886	0.798	2416	0.394	3492	0.929	488	0.436	6459	0.133	15399	0.657	1064
AutoL2S (w/ RJ $k = 8$)	0.573	2103	0.804	1405	0.404	2798	0.923	663	0.435	2546	0.100	4146	0.771	1058

H Case Studies of AutoL2S

In this section, we present additional case studies to demonstrate the efficiency of AutoL2S. The AutoL2S responses shown in this section are generated by a model trained on Qwen2.5-7B-Instruct with $rj = 8$. The baseline responses in the first two examples are produced by Bespoke-Stratos-7B (Labs, 2025). We include three types of inference modes: Auto Long Reasoning, Auto Short Reasoning, and AutoL2S without <EASY>.

H.1 Qualitative Analysis of AutoL2S

The AutoL2S response shown here is generated by a model trained in Qwen2.5-7B-Instruct with $rj = 8$, and the baseline response is produced by Bespoke-Stratos-7B. The case studies demonstrate clear improvements in AutoL2S output after training. Before training, responses were often verbose, meandering, and filled with self-doubt, with frequent backtracking and confusion in applying domain knowledge. This will lead to several rounds of redundant thinking and verification steps. We observe that in some simple questions, complex verification steps can be reduced or even omitted, where LLMs are strong enough to provide the answers. After training with AutoL2S, the outputs became more structured and coherent, with reasoning expressed in a logical step-by-step manner and conclusions stated concisely and decisively. The use of special triggers aligned better with the intended long-short reasoning split, domain knowledge was applied more accurately, and verbose exploration was replaced by tighter yet sufficiently detailed explanations. Overall, training enhanced fluency, confidence, and correctness while reducing redundancy, showing a clear shift from rambling exploration to disciplined, solution-oriented reasoning.

H.2 Auto Long Reasoning for Regular Questions

We showcase an example when encountering a regular question. AutoL2S automatically chooses to reason longer but eliminates the token usage compared to baseline responses.

Case Study: The Lollipop Packing Problem

question: Jean has 30 lollipops. Jean eats 2 of the lollipops. With the remaining lollipops, Jean wants to package 2 lollipops in one bag. How many bags can Jean fill?

Baseline Response (445 Tokens):

<|begin_of_thought|> Okay, let's see. Jean has 30 lollipops. She eats 2 of them. Then she wants to put the remaining ones into bags, with each bag holding 2 lollipops. The question is, how many bags can she fill? Hmm, first step, I need to figure out how many lollipops Jean has left after she eats 2. So originally, she has 30. If she eats 2, that means she subtracts 2 from 30. Let me write that down: 30 minus 2 equals 28. So she has 28 lollipops left. Now, she wants to put 2 lollipops in each bag. So, to find out how many bags she can fill, I need to divide the number of remaining lollipops by 2. That would be ...[251 tokens omitted]

AutoL2S Response (136 Tokens):

<|begin_of_solution|> **<EASY><Short Trigger>** This is a trigger to ensure the model's upcoming output <EASY>. Let's approach this step by step. Jean starts with 30 lollipops. She eats 2, so we need to subtract those from the total. 30 minus 2 equals 28. So, Jean has 28 lollipops left. Now, Jean wants to put 2 lollipops in each bag. To find out how many bags she can fill, we divide the number of remaining lollipops by the number of lollipops per bag. 28 divided by 2 equals 14. Therefore, Jean can fill 14 bags. <|end_of_solution|>

Case Study: Auto Long Reasoning for Regular Questions

question: Melissa works as a pet groomer. This week, she has 8 dogs that need to be bathed, 5 cats that need their nails clipped, 3 birds that need their wings trimmed, and 12 horses that need to be brushed. If she splits the grooming jobs evenly over the days, how many animals will she groom each day of the week?

Baseline Response (2207 Tokens):

<|begin_of_thought|>

Okay, let's see. Melissa is a pet groomer, and she has different animals to groom this week. The problem is asking how many animals she'll groom each day if she splits the jobs evenly over the days. Alright, so first, I need to figure out the total number of animals she has to groom. Let me list them out:

- Dogs: 8 that need bathing.
- Cats: 5 that need their nails clipped.
- Birds: 3 that need their wings trimmed.
- Horses: 12 that need brushing.

...[1889 tokens omitted]...

<|end_of_thought|>

<|begin_of_solution|>

Melissa has a total of 8 dogs, 5 cats, 3 birds, and 12 horses to groom. To find the total number of animals:

$$8\,(\text{dogs}) + 5\,(\text{cats}) + 3\,(\text{birds}) + 12\,(\text{horses}) = 28\,(\text{total animals})$$

The problem asks how many animals she will groom each day if the jobs are split evenly over the days. Assuming she works 7 days a week, the calculation is:

$$\frac{28\,(\text{total animals})}{7\,(\text{days})} = 4\,(\text{animals per day})$$

Thus, the final answer is $\boxed{4}$.

<|end_of_solution|>

AutoL2S Response (1351 Tokens):

<|begin_of_thought|>

Let's consider this problem in a <pureLong> way. Okay, let's see. So the problem is about Melissa, who's a pet groomer. She has different animals to groom: 8 dogs, 5 cats, 3 birds, and 12 horses. The question is asking how many animals she will groom each day if she splits the jobs evenly over the days of the week.

First, I need to figure out the total number of grooming jobs she has.

...[1112 tokens omitted]...

<|end_of_thought|>

<|begin_of_solution|>

Melissa has a total of 8 dogs, 5 cats, 3 birds, and 12 horses to groom. Adding these together gives:

$$8 + 5 + 3 + 12 = 28$$

She needs to split these 28 grooming jobs evenly over 7 days of the week. Dividing the total number of jobs by the number of days gives:

$$28 \div 7 = 4$$

Thus, the number of animals she will groom each day is:

$$\boxed{4}$$

<|end_of_solution|>

H.3 Auto Short Reasoning for EASY Questions

AutoL2S can automatically switch to from long to short path without sacrificing the reasoning performance.

Case Study: Auto Short Reasoning for EASY Questions

question: Andrew plans a road trip from New Jersey to Rochester. It takes 6 days to travel by bus and half as many days to travel by car from New Jersey to Rochester. How many days will he travel if he plans to go to Rochester by bus and return from Rochester by car?

Baseline Answer (573 Tokens):
<|begin_of_thought|>
Okay, so Andrew is planning a road trip from New Jersey to Rochester. The problem says it takes 6 days to travel by bus, and half as many days by car. Then he wants to go by bus and return by car. I need to figure out the total number of days he'll travel.
First, let me parse the information. The trip from New Jersey to Rochester by bus takes 6 days. Then, the return trip by car would take half as many days as the bus trip. So, half of 6 days is 3 days. That makes sense because the problem states "half as many days to travel by car," so it's half the bus time.
So, going by bus: 6 days. Coming back by car: 3 days. To find the total travel time, I need to add both legs of the journey. That would be 6 days + 3 days. Let me do the math: 6 + 3 equals 9. So, the total travel time is 9 days.
Wait a second, let me make sure I didn't miss anything. The problem says he plans to go to Rochester by bus and return from Rochester by car. So, it's a round trip. The key here is that the return trip is from Rochester back to where he started, which is New Jersey. So, the distances are the same, but the mode of transportation changes.
The bus trip to Rochester is 6 days, car trip back is half that, so 3 days. Adding them together gives 9 days total travel time. That seems straightforward. Is there any part of this that could be interpreted differently?
For example, maybe the problem is referring to something else, like including time in New Jersey or Rochester? But the problem specifically mentions the travel time by bus and car for the same distance. So, it's just the round trip, with each leg's time as given.
Another check: if the bus takes 6 days, then half of that is indeed 3 days. So, 6 + 3 is 9. Yeah, that seems right. I don't see any other factors here. The answer should be 9 days.
<|end_of_thought|>

<|begin_of_solution|>
Andrew plans to travel from New Jersey to Rochester by bus, which takes 6 days. The return trip by car takes half as many days as the bus trip, which is $6 \div 2 = 3$ days. The total travel time is the sum of both trips:

$$6 \text{ days (bus)} + 3 \text{ days (car)} = 9 \text{ days}$$

The final answer is $\boxed{9}$.
<|end_of_solution|>

AutoL2S Answer (200 Tokens): <|begin_of_solution|>
This is a trigger to ensure the model's upcoming output <EASY>. To determine the total number of days Andrew will travel, we need to consider both the trip to Rochester and the return trip from Rochester. Let's break it down step by step.
1. **Trip to Rochester by bus:** - It takes 6 days to travel by bus from New Jersey to Rochester.
2. **Return trip from Rochester by car:** - The problem states that it takes half as many days to travel by car as it does by bus. - Since the bus trip takes 6 days, the car trip will take $\frac{6}{2} = 3$ days.
3. **Total travel time:** - Add the time taken for both trips: 6 days (bus) + 3 days (car) = 9 days.
Therefore, the total number of days Andrew will travel is $\boxed{9}$.
<|end_of_solution|>

H.4 AutoL2S w/o <EASY>

Case Study: AutoL2S w/o <EASY>

question: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

AutoL2S w/o <EASY> (986 Tokens):

<|begin_of_thought|>

<specialLong> We will provide a detailed explanation and solution. Okay, let's see. So Janet's ducks lay 16 eggs every day. That's the starting point. Then she does a couple of things with those eggs. First, she eats three for breakfast every morning,

...[504 tokens omitted]...

<|end_of_thought|>

<|begin_of_solution|>

Janet starts with 16 eggs each day. She eats 3 eggs for breakfast and uses 4 eggs to bake muffins, totaling 3 + 4 = 7 eggs used per day. Subtracting this from the daily gives 16 - 7 = 9 eggs left to sell. Each egg is sold for $2, so the total money made from selling the eggs is 9 × 2 = 18 dollars.

The final answer is $\boxed{18}$.<|end_of_solution|><seperate><|begin_of_solution|>

This is a trigger to ensure the model's upcoming output <short>. To determine how much Janet makes every day at the farmers' market, we need to follow these steps:

1. **Calculate the total number of eggs laid by the ducks per day:** Janet's ducks lay 16 eggs per day.
2. **Determine the number of eggs Janet eats for breakfast each day:** Janet eats 3 eggs for breakfast every morning.
3. **Determine the number of eggs Janet uses to bake muffins each day:** Janet bakes muffins for her friends every day with 4 eggs.
4. **Calculate the total number of eggs Janet uses each day:**

$$3 \text{ (eggs for breakfast)} + 4 \text{ (eggs for muffins)} = 7 \text{ eggs}$$

5. **Calculate the number of eggs remaining after Janet uses some for breakfast and baking:**

$$16 \text{ (total eggs)} - 7 \text{ (eggs used)} = 9 \text{ eggs}$$

6. **Determine the amount of money Janet makes from selling the remaining eggs:** Janet sells the remaining 9 eggs at the farmers' market for $2 per egg.

$$9 \text{ eggs} \times 2 \text{ dollars per egg} = 18 \text{ dollars}$$

Therefore, the amount of money Janet makes every day at the farmers' market is $\boxed{18}$.

<|end_of_solution|>

AutoL2S w <EASY> (314 Tokens):

<|begin_of_solution|>

This is a trigger to ensure the model's upcoming output <EASY>. To determine how much Janet makes every day at the farmers' market, we need to follow these steps:

1. **Calculate the total number of eggs laid by the ducks per day:** Janet's ducks lay 16 eggs per day. ...[158 tokens omitted]...

4. **Determine how much Janet makes from selling the remaining eggs:** Janet sells the remaining 9 eggs at the farmers' market for $2 per egg. Therefore, the total amount she makes is:

$$9 \text{ eggs} \times 2 \text{ dollars per egg} = 18 \text{ dollars}$$

Thus, the amount Janet makes every day at the farmers' market is $\boxed{18}$ dollars.

<|end_of_solution|>