ATLAS: Adaptive Test-Time Latent Steering with External Verifiers for Enhancing LLMs' Reasoning

Anonymous ACL submission

Abstract

Recent work on activation and latent steering has demonstrated that modifying internal representations can effectively guide large language models (LLMs) toward improved reasoning and efficiency without additional training. However, most existing approaches rely on fixed steering policies and static intervention strengths, which limit their robustness across problem instances and often result in over- or under-steering. We propose Adaptive Test-time Latent Steering, called (ATLAS), a task-specific framework that dynamically controls steering decisions at inference time using an *external, lightweight latent verifier*. Given intermediate hidden states, the verifier predicts the quality of ongoing reasoning and adaptively selects whether and how strongly to apply steering, enabling per-example and per-step adjustment with minimal overhead. To our knowledge, ATLAS is the first method to integrate learned latent verification into test-time steering for enhancing LLMs' reasoning. Experiments on multiple mathematical reasoning benchmarks show that ATLAS consistently outperforms both vanilla decoding and fixed steering baselines, achieving higher accuracy while substantially reducing test-time token usage. These results demonstrate that verifier-guided latent adaptation provides an effective and scalable mechanism for controlling reasoning efficiency without sacrificing solution quality. All source code will be publicly available.

1 Introduction

LLMs exhibit strong performance across a wide range of complex tasks, including planning (Valmeekam et al., 2023), mathematical problem solving (AlphaProof and AlphaGeometry, 2024; Ahn et al., 2024; Chen et al., 2025d), and code generation (Xia et al., 2025), particularly when augmented with structured prompting techniques that enhance task effectiveness, such as Chain-of-Thought (CoT) (Brown, 2020), Tree-of-Thought (ToT) (Yao et al., 2023), and Graph-of-Thought (GoT) (Besta et al., 2024). These approaches enable models to decompose tasks into intermediate reasoning steps and explore multiple solution paths to reach the correct conclusions, but sometimes at the cost of extra-long reasoning paths. However, lengthy reasoning is not always necessary. Studies have shown that LLMs often arrive at the correct final answer early in the reasoning process, yet continue generating excessive and redundant thought sequences (Fu et al., 2025a). Such inefficiencies can even impair overall performance, as models may become trapped in repetitive verification loops (Chen et al., 2025c) or engage in unnecessary detours that lead to underthinking (Wang et al., 2025).

While structured reasoning methods such as CoT enhance interpretability and accuracy, their reliance on natural language reasoning limits the model's expressive capacity. Latent reasoning overcomes this limitation by performing multi-step inference directly within the model's hidden states, enabling control over the reasoning process, bypassing token-level supervision, and supporting more efficient exploration of solution paths (Zhu et al., 2025; Chen et al., 2025b; Hao et al., 2025). Recent work has further investigated model steering in reasoning through the use of steering vectors. By identifying linear directions in an LLM's activation space corresponding to specific reasoning patterns such as backtracking, uncertainty estimation, or hypothesis testing, these vectors allow fine-grained modulation of the reasoning process (Venhoff et al., 2025; Chen et al., 2025a). This approach offers a practical and interpretable means of guiding how models explore solution paths, complementing latent space reasoning by enabling targeted manipulation of internal inference dynamics.

Despite the advances in structured and latent reasoning, existing approaches do *not* fully leverage the rich semantic and structural information embedded in a model's internal representations.

Existing evidence suggests that intermediate layer activations encode semantic content, task-specific features, and uncertainty estimates, positing that *LLMs' hidden states could help anticipate the quality of reasoning steps before text generation completes*. Such hypothesis can enable a more dynamic guidance of the reasoning process in LLMs, offering a promising avenue toward adaptive and efficient reasoning in LLMs.

To this end, we propose ATLAS, a novel framework that trains *external, lightweight latent verifiers* that operate directly on hidden states to dynamically steer reasoning. ATLAS trains compact neural networks to predict reasoning quality from intermediate hidden states extracted at strategic points during generation (e.g., paragraph boundaries). These verifiers are distilled using a *process reward model* (Malik et al., 2025) equipped with external, task-specific knowledge that provides step-level quality labels of the reasoning. Based on the predicted quality scores, ATLAS dynamically selects among multiple steering modes, for example, execution for straightforward steps, reflection for error-prone reasoning, and transition for strategic pivots by applying steering vectors that algebraically modify hidden state representations. Our main contributions are:

1. **Latent Verifier for Reasoning Quality.** We introduce a lightweight latent verifier learned from external knowledge that predicts step-level reasoning quality directly from intermediate hidden states, enabling early evaluation of reasoning trajectories without requiring full token generation.
2. **Adaptive Test-Time Latent Steering.** We propose ATLAS, an adaptive test-time steering framework that leverages latent verification signals to dynamically control execution, reflection, and transition behaviors during reasoning. This approach consistently improves both accuracy and inference efficiency over vanilla decoding and fixed steering baselines.
3. **Analysis of Latent Reasoning Control.** We conduct extensive empirical analyses on steering stability, verifier generalization, and computation efficiency, providing insights into the reliability and limitations of latent-space interventions for multi-step reasoning.

2 Related Works

Step-by-step Reasoning in LLMs. Step-by-step reasoning methods, such as CoT, ToT, and GoT prompting, have become central to eliciting structured reasoning capabilities in LLMs (Wei et al., 2022; Yao et al., 2023; Besta et al., 2024). Subsequent extensions enhance these capabilities: self-consistent decoding samples multiple reasoning trajectories and aggregates their outcomes (Yoon et al., 2025; Fu et al., 2025b), while least-to-most prompting decomposes complex problems into a sequence of simpler subproblems (Zhou et al., 2023). Despite their strengths, they rely on a fixed reasoning strategy applied uniformly throughout generation, limiting their ability to adapt to the varying reasoning demands within a single problem.

Activation Engineering. Activation engineering enables systematic control of LLMs' behaviors by modifying their internal activations, often achieved via so-called steering vectors constructed from contrasting representations across different prompt contexts or behavioral modes (Turner et al., 2023). This then allows targeted manipulation of attributes such as truthfulness, safety, stylistic preferences, and even specific reasoning behaviors *without* retraining (Burns et al., 2023; Stolfo et al., 2025). However, most steering methods rely on a fixed intervention throughout the generative process, assuming that a static modification is sufficient for all reasoning contexts. This rigidity limits their ability to accommodate the varying cognitive demands. Existing techniques also lack mechanisms for adaptive selection among multiple steering modes, leaving open the challenge of dynamically adjusting internal activations during generation.

Process Reward Models and Verification. Process Reward Models (PRMs) provide step-level supervision for assessing the quality of intermediate reasoning steps, offering finer-grained guidance than outcome-based rewards. These models are crucial for capturing nuanced signals in reasoning, and evaluating their accuracy is strongly predictive of downstream task performance (Malik et al., 2025). This makes PRMs a potential tool for guiding real-time reasoning in LLMs. However, directly applying PRMs is computationally expensive, as we will need to generate candidate reasoning steps and *then* verify them with PRMs. In this work, we bypass this bottleneck by learning to predict PRMs outputs *directly from hidden states*, enabling rapid, latent-space verification of reasoning quality without explicit text generation. This allows more efficient and adaptive steering of multi-step reasoning in LLMs.

Question: Marcell and Beatrice are having a contest to see who can eat the most [...] Beatrice's was two roll-ups wide and 24 roll-ups long while Marcell's was 3 roll-ups wide and 14 roll-ups long. If they both ate their entire amount, how many did they eat on average?
Output: *Okay, so Marcell and Beatrice are having a contest to see who can eat the most fruit roll-ups.*
.....
Wait, maybe the problem is saying that Beatrice's fruit roll-ups are two roll-ups wide and 24 roll-ups long, so each of her fruit roll-ups is $2 \times 24 = 48$. *But Marcell's is* $3 \times 14 = 42$.
Wait, maybe the problem is saying that Beatrice's fruit roll-ups are two roll-ups wide and 24 roll-ups long, so each of her fruit roll-ups is $2 \times 24 = 48$. *But Marcell's is* $3 \times 14 = 42$.
Wait, maybe the problem is saying that Beatrice's fruit roll-ups are two roll-ups wide and 24 roll-ups long, so each of her fruit roll-ups is $2 \times 24 = 48$. *But Marcell's is* $3 \times 14 = 42$.
.....

Figure 1: An example illustrating repetitive reasoning of Qwen-1.5B on the GSM8K dataset while executing single reasoning mode on SEAL (Chen et al., 2025a).

3 Motivation

Empirical analyses using multi-aspect evaluation frameworks (e.g., CaSE (Do et al., 2025)) show that LLMs frequently generate reasoning steps that are irrelevant to the problem context or incoherent with preceding steps. By segmenting reasoning into discrete *thought units* or contiguous spans of the reasoning trace separated by linguistic cues such as *"wait"*, *"alternatively"*, or *"let me verify"*, each unit can be associated with a distinct functional role, including *execution* (direct problem solving), *reflection* (self-verification or error checking), and *transition* (strategic shifts or exploration of alternative solution paths) (Chen et al., 2025a). When these thought types are misaligned or poorly sequenced, errors can propagate across the reasoning chain, degrading interpretability and leading to incorrect final answers even when the model internally encodes the correct solution.

A key limitation of existing steering approaches is that LLMs typically apply a uniform reasoning strategy throughout the entire inference process, failing to adapt to local changes during reasoning. Figure 1 illustrates a representative failure case in which enforcing only execution-type reasoning causes the model to repeatedly restate identical intermediate steps without making meaningful progress. In such cases, effective reasoning would require a transition or reflection step to redirect the reasoning trajectory, yet static steering prevents such adaptation. This observation highlights the limitations of current reasoning control methods and motivate the need for adaptive guidance mechanisms that can dynamically assess and steer reasoning quality at the granularity of individual thoughts.

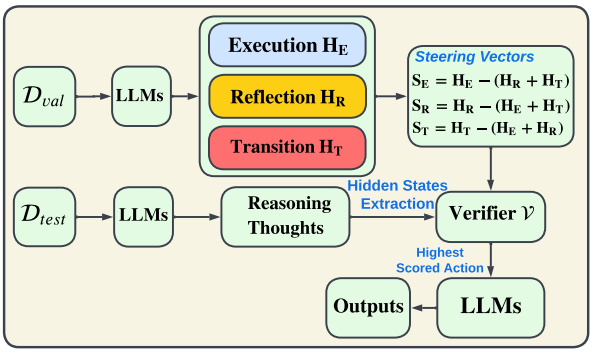

Figure 2: ATLAS pipeline: (Top) Offline extraction of contrastive steering vectors from execution, reflection, and transition reasoning modes. (Bottom) Test-time adaptive steering through lightweight latent verification of hidden states.

4 Method

Given a validation set $\mathcal{D}_{\text{val}}$, and a target LLM $\mathcal{M}$, we seek to learn an external verifier $\mathcal{V}$ that informs adaptive steering decisions during inference. The overall framework is shown in Figure 2.

4.1 Problem Formulation

Transformer Decoding Process. Given an input token sequence $\mathbf{x} = [x_1, \ldots, x_T]$, decoder-only transformer language models (Vaswani et al., 2017) map to a probability distribution over the vocabulary for next-token prediction. Each token x_i is associated with a sequence of residual stream activations $\mathbf{h}^{(l)}(x_i) \in \mathbb{R}^d$ across L layers, initialized by the token embedding $\mathbf{h}^{(0)}(x_i) = \text{Embed}(x_i)$. At each layer $l \in \{1, \ldots, L\}$, the residual stream $\mathbf{h}^{(l)}(x_i)$ is updated by combining the previous layer's activation $\mathbf{h}^{(l-1)}(x_i)$ with two components: (*i*) a multi-head self-attention mechanism, which computes $\mathbf{a}^{(l)}(x_{1:i})$ by attending to prior tokens $\{x_j : j \leq i\}$ using a causal mask to enforce autoregressive context flow; and (*ii*) a multi-layer perceptron (MLP), which applies non-linear transformations to the post-attention state $\mathbf{h}^{(l-1)}(x_i) + \mathbf{a}^{(l)}(x_{1:i})$ and produces $\mathbf{m}^{(l)}(x_i)$. The whole process is expressed as follows:

$$\mathbf{h}^{(l)}(x_i) = \mathbf{h}^{(l-1)}(x_i) + \mathbf{a}^{(l)}(x_{1:i}) + \mathbf{m}^{(l)}(x_i) \quad (1)$$

$$\mathbf{m}^{(l)}(x_i) = \text{MLP}\big(\mathbf{h}^{(l-1)}(x_i) + \mathbf{a}^{(l)}(x_{1:i})\big) \quad (2)$$

Through autoregressive aggregation, each $\mathbf{h}^{(l)}(x_i)$ aggregates context from prior tokens, with the final token's residual stream $\mathbf{h}^{(l)}(x) :\rightarrow \mathbf{h}^{(l)}(x_T)$ encapsulating the entire input's context.

Large Reasoning Models (LRMs). LRMs extend structured search in LLMs' inference through Tree-, Graph-, and Forest-of-Thought frameworks (Wei et al., 2022; Yao et al., 2023; Besta et al., 2024), including both proprietary models (e.g. *OpenAI's o-series, Google's Gemini-2.5-Flash, etc.*) (Jaech et al., 2024) and open-source models like *DeepSeek-R1* (Guo et al., 2025), Qwen3 (Yang et al., 2025), etc. LRMs are tailored for complex problem-solving and instruction-following, which leverage structured templates to handle user inputs:

```
<|begin_of_sentence|><|User|>
{instruction}<|Assistant|><think>
<think>··· </think>
```

The reasoning process is segmented by the delimiter "\n\n". These trajectories may include backtracking, self-verification, sub-goal formulation, and backward chaining, which closely mirror human cognitive behaviors and support effective multi-step problem solving (Gandhi et al., 2025). Current works categorize each reasoning step using keyword-based heuristics (Chen et al., 2025a; Zhang et al., 2025; Azizi et al., 2025), assigning each step to one of three functional types, such as *execution*, *transition*, or *reflection*.

4.2 Reasoning Steering Vectors Extraction

We construct the external verifier's training data from a validation set $\mathcal{D}_{val}$, which is drawn from standard reasoning benchmarks. By default, we sample from $\mathcal{D}_{val}$ and run the target model $\mathcal{M}$ to generate full reasoning traces $\mathcal{R}$. We define a "reasoning step" as the contiguous chunk of reasoning text between two occurrences of the special delimiter token "\n\n". Each trace is segmented into individual thoughts using line breaks and categorized into three types: *execution*, *reflection*, and *transition* via keyword-based heuristics. Please refer to the Appendix A.1 for more details.

To compute candidate steering vectors, we run inference over the entire chain-of-thought $\{\texttt{<think>}_1, \ldots, \texttt{<think>}_n\}$ as a single prefill and record the hidden states at the segment-terminating delimiter "\n\n". Such states provide a compact summary of the preceding reasoning segment. For each reasoning step $\texttt{<think>}_j$, we then extract the post-attention hidden representation from the i-th transformer block at the first delimiter token, denoted as $\mathbf{h}_i^j$. We then compute category-wise mean representations over the validation set $\mathcal{D}\text{val}$:

$$\bar{\mathbf{h}}_{C,i} = \frac{1}{N_C} \sum_{j \in C} \mathbf{h}_i^j, \tag{3}$$

where $C \in \{\text{Execution}, \text{Reflection}, \text{Transition}\}$. Based on these averages, we define three candidate reasoning steering vectors by contrasting each thought type against the remaining ones:

$$\begin{aligned} \mathbf{s}_E &= \bar{\mathbf{h}}_E - \bar{\mathbf{h}}_{RT}, \\ \mathbf{s}_R &= \bar{\mathbf{h}}_R - \bar{\mathbf{h}}_{ET}, \\ \mathbf{s}_T &= \bar{\mathbf{h}}_T - \bar{\mathbf{h}}_{RE}, \end{aligned} \tag{4}$$

where RT, ET, and RE denote the union of the corresponding complementary thought categories.

4.3 External Verifier Training

Data Construction. For each thought j, we extract the hidden representation $\mathbf{h}_i^j$ from the i-th transformer block at the line-break token, which serves as input to the latent verifier. To obtain supervision signals, we employ a PRM (Malik et al., 2025) that evaluates the full reasoning trace $\mathcal{R}$ and assigns a quality score vector $\mathbf{v}^j \in [0, 1]$ to each thought. This procedure yields paired training instances $(\mathbf{h}_i^j, \mathbf{v}^j)$. We then aggregate them into a dataset $\{\mathbf{H}, \mathbf{v}\}$ for training the external verifier, using a 6:2:2 split for training, validation, and testing.

Training External Verifier $\mathcal{V}$. We investigate lightweight verifier architectures, specifically a two-layer MLP with hidden dimension 256 and ReLU activations. For each thought j, the verifier computes a scalar quality score s^j as:

$$s^j = \sigma\Big(W_2\,\text{ReLU}(W_1\,\mathbf{h}_i^j + b_1) + b_2\Big), \tag{5}$$

where $\mathbf{h}_i^j$ is the hidden representation of thought j from the i-th transformer block, W_1, W_2 and b_1, b_2 are learnable weights and biases, and σ denotes the sigmoid function. The verifier is distilled from the PRM-provided labels v^j using binary cross-entropy loss, enabling efficient step-level quality prediction using latent representations without requiring expensive reward models at inference time.

4.4 Adaptive Steering Policy

At each reasoning step, the system evaluates four candidate steering configurations: *execution* $\mathbf{s}_E$

(direct problem solving), *reflection* $\mathbf{s}_R$ (error checking), *transition* $\mathbf{s}_T$ (strategy pivoting), and *none* $\mathbf{s}_0 = \mathbf{0}$ (no intervention). Motivated by prior works such as (Chen et al., 2025a; Vu and Nguyen, 2025), which highlight that the steering direction is the key factor in intervention effectiveness, our system selects the steering vector according to the direction of each steering vector option:

$$\mathbf{s}^* = \arg\max_{\mathbf{s}\in S} \mathcal{V}\big(h_{\text{current}} + \mathbf{s}\big), \tag{6}$$

where h_{current} denotes the current hidden state and $S = \{\mathbf{s}_E, \mathbf{s}_R, \mathbf{s}_T, \mathbf{s}_0\}$ is the set of candidate steering vectors.

5 Experiments Set-up

Datasets. We evaluate our approach on three widely used mathematical reasoning benchmarks. GSM8K (Cobbe et al., 2021) contains 8,500 grade-school math word problems requiring multi-step arithmetic reasoning. MATH500 (Lightman et al., 2024) comprises 12,500 competition-level problems spanning algebra, geometry, and calculus, with a challenging subset used for evaluation. AIME2024 (Sun et al., 2025) consists of problems from the 2024 American Invitational Mathematics Examination, designed to assess advanced multi-step problem-solving skills, alongside AMC2023 (Mathematical Association of America, 2023) to cover a range of difficulty levels.

Target Models $\mathcal{M}$ and RPMs. We evaluate the effectiveness of ATLAS using several widely adopted reasoning models, including DeepSeek-R1-Distill-Qwen-1.5B/7B/32B (R1-Distill-1.5B/7B/32B) (Guo et al., 2025), and QwQ-32B-Preview (Team, 2024; Yang et al., 2024). For a fair comparison, we use the same instruction prompt and set the max length equal to 8192 tokens for all the models. For PRMs, we use Math-Shepherd[1], a 7B-parameter model featuring the process-oriented Math-Shepherd reward model, which assigns a reward score to each step of a math problem solution (Wang et al., 2024).

Baselines. We compare ATLAS against training-free methods and include three groups of representative baselines. *(i) Vanilla (No Steering)* uses the base model to generate solutions without any steering or intervention. *(ii) Fixed Steering* methods apply a single, non-adaptive steering strategy throughout the entire reasoning process, without conditioning on intermediate reasoning quality. This category includes:

- *ASC* (Azizi et al., 2025), which applies a fixed activation steering vector to encourage more concise chain-of-thought generation.
- *SEAL* (Chen et al., 2025a), which performs task arithmetic in latent space to down-regulate internal representations associated with reflection and transition behaviors.
- *CREST* (Zhang et al., 2025), a training-free test-time method that identifies cognition-related attention heads and applies calibrated latent-space rotations to suppress inefficient reasoning behaviors such as overthinking and backtracking, improving both accuracy and token efficiency.

Finally, we evaluate *(iii) Adaptive Steering* approaches. *ATLAS(T)* selects steering actions based on post-hoc verification of generated text using PRMs, while *ATLAS(L)* uses signals from the external latent verifier to guide adaptive steering decisions during inference.

Evaluation Metrics. We evaluate all systems along multiple complementary dimensions. Final accuracy (*Acc*) measures the percentage of problems for which the final answer matches the ground truth. Steering effectiveness captures inference efficiency, quantified by the total number of tokens generated during reasoning (*#Tok*).

6 Results

We evaluate ATLAS along two orthogonal generalization axes. First, we examine **the transferability of steering vectors** by comparing vectors learned from a single source development set with vectors learned in-domain for each target benchmark. Second, we analyze **the generalization of the latent verifier** by contrasting verifiers trained on hidden states extracted from one dataset with verifiers trained on in-domain development sets. Together, these axes disentangle the effects of representation-level transfer from dataset-specific adaptation in adaptive test-time steering.

6.1 In-Domain Performance

We show overall performance for the **in-domain** setting in Table 1. Across all benchmarks, adaptive steering methods consistently improve accuracy while substantially reducing test-time token usage compared to Vanilla models, whereas fixed steering

[1] https://huggingface.co/peiyi9979/math-shepherd-mistral-7b-prm

	Methods	MATH				GSM8K			
		Acc	ΔAcc	#Tok	ΔTok	Acc	ΔAcc	#Tok	ΔTok
R1-Distill-1.5B	*Vanilla*	73.76	–	3941	–	79.30	–	2390	–
	SEAL	79.78	↑8.4	3034	↓23.0	82.41	↑3.9	1438	↓39.8
	ASC	73.98	↑0.3	3252	↓17.5	79.61	↑0.4	2267	↓5.1
	CREST	69.06	↓6.2	4016	↑1.9	75.44	↓4.9	2142	↓10.4
	ATLAS(T)	82.20	↑11.4	2783	↓29.4	85.29	↑7.6	1452	↓39.2
	ATLAS(L)	**82.28**	↑11.6	**2754**	↓30.1	**85.37**	↑7.7	**1316**	↓44.9
R1-Distill-7B	*Vanilla*	86.34	–	3395	–	89.23	–	841	–
	SEAL	88.96	↑3.0	3050	↓10.2	88.55	↓0.8	655	↓22.1
	ASC	86.38	↑0.0	3026	↓10.9	86.50	↓3.1	795	↓5.5
	CREST	86.48	↑0.2	3130	↓7.8	89.38	↑0.2	1243	↑47.8
	ATLAS(T)	**90.70**	↑5.1	2975	↓12.4	89.46	↑0.3	672	↓20.1
	ATLAS(L)	90.68	↑5.0	**2895**	↓14.7	**89.61**	↑0.4	**637**	↓24.3
R1-Distill-32B	*Vanilla*	91.54	–	2413	–	92.87	–	444	–
	SEAL	92.42	↑1.0	2030	↓15.9	92.95	↑0.1	436	↓1.8
	ASC	91.60	↑0.1	2014	↓16.5	93.03	↑0.2	493	↑11.0
	CREST	91.60	↑0.1	2156	↓10.6	92.95	↑0.1	1143	↑157.4
	ATLAS(T)	93.02	↑1.6	1937	↓19.7	93.48	↑0.7	473	↑6.5
	ATLAS(L)	**93.04**	↑1.6	**1896**	↓21.4	**93.56**	↑0.7	**427**	↓3.8
QwQ-32B-Pre	*Vanilla*	90.02	–	2114	–	94.54	–	687	–
	SEAL	90.20	↑0.2	2043	↓3.4	94.39	↓0.2	508	↓26.1
	ASC	90.10	↑0.1	2015	↓4.7	94.24	↓0.3	539	↓21.5
	CREST	90.16	↑0.2	2032	↓3.9	94.31	↓0.2	1034	↑50.5
	ATLAS(T)	90.40	↑0.4	1962	↓7.2	**95.07**	↑0.6	511	↓25.6
	ATLAS(L)	**90.42**	↑0.4	**1959**	↓7.3	94.84	↑0.3	**498**	↓27.5

Table 1: Accuracy (↑), test-time token usage (↓) and relative improvements (in percent) on in-domain reasoning benchmarks. **Bold** and Underlining denotes the best and second-best result.

approaches exhibit mixed gains and occasionally sacrifice accuracy for efficiency. From a model-scaling perspective, the advantages of adaptive steering become increasingly pronounced as model size grows. While smaller models (1.5B) already benefit from reduced token usage, larger models (7B and 32B) show stronger accuracy preservation alongside significant efficiency gains, highlighting the importance of capacity-aware steering. In particular, *ATLAS(T)* consistently improves over fixed steering baselines by dynamically selecting execution modes using text-level verification signals, whereas *ATLAS(L)* achieves the strongest and most stable gains across model scales by relying on a lightweight latent verifier.

Notably, *ATLAS(L)* is implemented as a small MLP operating on hidden states, making it substantially more efficient than text-based verifiers such as ThinkPRM[2]. Despite its simplicity, *ATLAS(L)* maintains or improves accuracy while achieving the largest reductions in token usage, particularly for larger models where inference cost is amplified. Overall, lightweight, latent-aware steering scales favorably with model capacity and provide an efficient alternative to text-based verification without compromising reasoning quality.

[2] https://huggingface.co/peiyi9979/math-shepherd-mistral-7b-prm

6.2 Out-of-Domain Transferability

Competing adaptive methods exhibit mixed or unstable behavior under distribution shift, while both ATLAS variants maintain favorable accuracy–efficiency trade-offs even on challenging benchmarks that differ significantly from the training distribution (Table 2). From a model-scaling perspective, the advantages of latent-based adaptive steering become increasingly pronounced as model capacity grows. On the AIME2024 benchmark, smaller models (1.5B) achieve a 6.7% accuracy improvement with a 51.0% reduction in token usage using ATLAS(L), while larger models (QwQ-32B-Preview) still realize a 0.7% absolute accuracy gain alongside a 36.2% token reduction. A similar trend is observed on AMC2023, where ATLAS(L) improves the 1.5B model's accuracy by 36.8% while reducing token usage by 26.8%, and *delivering consistent gains competitively with model of much larger size of 7B*. In contrast, improvements on GSM8K are smaller as this benchmark is less challenging and baseline models already produce concise and accurate solutions.

On more difficult benchmarks, such as AIME, where baseline models struggle to find correct solutions initially, adaptive steering enables the model to explore and revise multiple solution trajectories before converging on a final answer, providing additional evidence that purely focusing on a single reasoning strategy (e.g., execution thought) is insufficient. Additionally, for smaller models (1.5B and 7B), ATLAS(L) consistently achieves the best overall performance among all baselines. However, as model size increases, ATLAS(T) begins to outperform ATLAS(L), suggesting that the effectiveness of external verification may become constrained at larger scales, whereas text-level adaptive control can better leverage the stronger intrinsic reasoning capabilities of larger models.

6.3 Accuracy-Efficiency Tradeoff

Beyond reporting standard accuracy metrics and test-time token usage as often done in existing work(Chen et al., 2025a; Azizi et al., 2025; Zhang et al., 2025), we want to contextualize these metrics to meaningfully compare their final performance. For instance, a method that achieves comparable accuracy while reducing token usage by 20% on

	Methods	GSM8K				AIME2024				AMC2023			
		Acc(↑)	ΔAcc	#Tok(↓)	ΔToken	Acc(↑)	ΔAcc	#Tok(↓)	ΔToken	Acc(↑)	ΔAcc	#Tok(↓)	ΔToken
R1-Distill-1.5B	*Vanilla*	79.30	–	2390	–	20.00	–	7396	–	47.50	–	5241	–
	SEAL	83.32	↑5.1	1432	↓40.1	23.33	↑16.7	6820	↓7.8	52.50	↑10.5	5111	↓2.5
	ASC	77.41	↓2.4	2647	↑10.8	10.00	↓50.0	7744	↑4.7	42.50	↓10.5	5590	↑6.7
	CREST	75.28	↓5.1	1799	↓24.7	23.33	↑16.7	7063	↓4.5	57.50	↑21.1	5349	↑2.1
	ATLAS(T)	84.46	↑6.5	1189	↓50.3	30.00	↑50.0	6361	↓14.0	62.50	↑31.6	3843	↓26.7
	ATLAS(L)	**84.53**	↑6.6	**1171**	↓51.0	**33.33**	↑66.7	**6304**	↓14.8	**65.00**	↑36.8	**3837**	↓26.8
R1-Distill-7B	*Vanilla*	89.23	–	841	–	40.00	–	7269	–	82.50	–	4458	–
	SEAL	88.63	↓0.7	657	↓21.9	46.67	↑16.7	6222	↓14.4	82.50	↑0.0	3822	↓14.3
	ASC	88.70	↓0.6	848	↑0.8	36.67	↓8.3	7001	↓3.7	70.00	↓15.2	4624	↑3.7
	CREST	**89.84**	↑0.7	1604	↑90.7	40.00	↑0.0	6835	↓6.0	72.50	↓12.1	4743	↑6.4
	ATLAS(T)	89.46	↑0.3	623	↓25.9	**60.00**	↑50.0	5721	↓21.3	82.50	↑0.0	3752	↓15.8
	ATLAS(L)	89.61	↑0.4	**611**	↓27.4	56.67	↑41.7	**5694**	↓21.7	**87.50**	↑6.1	**3749**	↓15.9
R1-Distill-32B	*Vanilla*	92.87	–	444	–	40.00	–	6654	–	80.00	–	4221	–
	SEAL	92.04	↓0.9	425	↓4.3	46.67	↑16.7	6110	↓8.2	**90.00**	↑12.5	3442	↓18.5
	ASC	91.74	↓1.2	454	↑2.3	36.67	↓8.3	6432	↓3.3	75.00	↓6.3	4217	↓0.1
	CREST	93.03	↑0.2	1132	↑155.0	46.67	↑16.7	6082	↓8.6	82.50	↑3.1	3427	↓18.8
	ATLAS(T)	**93.93**	↑1.1	**419**	↓5.6	**66.67**	↑66.7	**5438**	↓18.3	87.50	↑9.4	**3084**	↓26.9
	ATLAS(L)	93.63	↑0.8	425	↓4.3	60.00	↑50.0	5442	↓18.2	82.50	↑3.1	3134	↓25.8
QwQ-32B-Pre	*Vanilla*	94.54	–	687	–	33.33	–	5977	–	82.50	–	3778	–
	SEAL	92.80	↓1.8	696	↑1.3	43.33	↑30.0	**5465**	↓8.6	80.00	↓3.0	**3352**	↓11.3
	ASC	94.01	↓0.6	812	↑18.2	30.00	↓10.0	5902	↓1.3	77.50	↓6.1	3792	↑0.4
	CREST	95.00	↑0.5	1242	↑80.8	40.00	↑20.0	5694	↓4.7	82.50	↑0.0	3521	↓6.8
	ATLAS(T)	**95.22**	↑0.7	**438**	↓36.2	**60.00**	↑80.0	5532	↓7.4	**87.50**	↑6.1	3482	↓7.8
	ATLAS(L)	95.00	↑0.5	449	↓34.6	56.67	↑70.0	5621	↓6.0	85.00	↑3.0	3521	↓6.8

Table 2: Accuracy (↑), test-time token usage (↓), and relative improvements (in percent) on cross-domain reasoning benchmarks. "Red" highlights degraded performance, while "blue" denotes substantial improvements relative to the baseline. **Bold** indicates the best performance, while underlining denotes the second-best result.

Methods	R1-Distill-1.5B			R1-Distill-7B			R1-Distill-32B			QwQ-32B-Pre		
	GSM8K	AIME	AMC	GSM8K	AIME	AMC	GSM8K	AIME	AMC	GSM8K	AIME	AMC
Vanilla	22.4	22.1	11.5	33.4	9.4	38.7	74.0	5.5	17.0	70.5	5.5	26.5
SEAL	45.4	30.0	22.7	10.4	29.6	45.7	56.0	39.0	84.5	34.0	72.0	62.5
ASC	11.5	0.0	0.0	11.8	3.9	1.3	47.5	9.0	0.0	50.5	7.5	0.0
CREST	1.4	29.6	33.9	50.0	7.0	7.5	30.5	40.0	60.0	46.0	44.0	56.0
ATLAS(T)	51.8	45.3	46.9	46.4	**60.9**	46.5	**100.0**	**100.0**	**92.0**	**100.0**	**93.0**	**85.0**
ATLAS(L)	**52.4**	**52.4**	**52.4**	**50.5**	54.0	**61.0**	93.0	88.5	73.0	95.0	79.5	68.5

Table 3: EC scores across models and benchmarks. **Bold**, underlining denote the best and second-best results.

a larger model should be considered superior to alternatives that require substantially more computation. Motivated by this observation, we propose an *Efficiency–Capacity Trade-off (EC) metric*, scaled from 0 to 100, that jointly accounts for accuracy, test-time token usage, and model size. This metric enables a more principled and deployment-aware comparison of steering methods under realistic computational constraints. Specifically, given the relative change in performance (ΔAcc) and token reduction (ΔToken), we define the trade-off score of a method evaluated on model m as:

$$\mathrm{EC}(m)=\frac{1}{2}(\overline{\Delta\mathrm{Acc}}(m)+\overline{\Delta\mathrm{Token}}(m)\times\frac{P_m}{P_{\max}}), \tag{7}$$

where $\overline{\Delta\mathrm{Acc}}$, $\overline{\Delta\mathrm{Token}}$ denote the min–max normalized changes in accuracy and token usage, respectively, and P_m, $P_{\max}$ denote the # parameters of model m and the largest model considered.

The EC metric captures three key desiderata: (1) accuracy improvements are rewarded linearly through $\overline{\Delta\mathrm{Acc}}$; (2) token reductions are weighted by model capacity via the scaling factor $\frac{P_m}{P_{\max}}$, reflecting the higher inference cost of larger models; and (3) min–max normalization ensures that both components contribute comparably despite differing scales. Higher EC scores indicate more favorable trade-offs between accuracy and efficiency. We report the *EC score* in Table 3. Overall, ATLAS variants consistently achieve the highest *EC* scores across all configurations, with *ATLAS(L)*

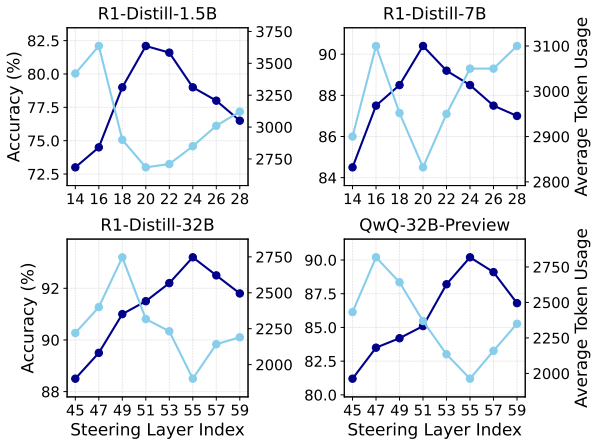

Figure 3: Layer-wise ablation study of steering effectiveness. Dark blue indicates the accuracy achieved by ATLAS(L), while sky blue denotes the corresponding test-time token usage.

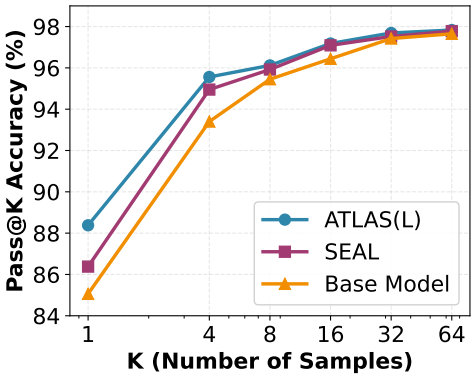

Figure 4: Pass@K performance of adaptive steering methods on the GSM8K dataset using R1-Distill-1.5B.

Model	Execution	Reflection	Transition	Neutral
R1-Distill-1.5B	37.6%	27.1%	21.2%	14.1%
R1-Distill-7B	49.4%	15.3%	18.8%	16.5%
R1-Distill-32B	62.3%	10.4%	14.3%	13.0%
QwQ-32B-Pre	58.8%	12.9%	15.3%	13.0%

Table 4: Steering mode distribution across model sizes.

ranking first in 8 out of 12 cases and *ATLAS(T)* leading in the remaining 4 on larger models. Notably, while CREST achieves superior accuracy on R1-Distill-7B (Table 2), its substantially higher token consumption results in poor efficiency-accuracy trade-offs relative to ATLAS methods.

6.4 Layer Ablation Study

We conduct a layer-wise ablation study to identify which transformer layers are most effective for intervention-based steering, with results shown in Figure 3. Overall, interventions applied to middle layers consistently yield the most effective reasoning behavior, achieving higher accuracy while reducing test-time token usage. This observation aligns with prior findings that middle layers primarily encode abstract and conceptual knowledge in large language models (Jin et al., 2025). These results highlight layer selection as a critical factor in adaptive steering and motivate the incorporation of layer-aware strategies into the proposed dynamic steering mechanism. Detailed experimental settings for the layer ablation study are provided in Appendix A.2.

6.5 Sampling Efficiency

We present an analysis of how adaptive model steering improves the sampling efficiency of LLMs, evaluated using the Pass@K metric (Brown et al., 2024), which measures the reasoning boundary of a model by revealing the proportion of problems it can potentially solve when given K attempts. We report preliminary Pass@K results on GSM8K using Qwen-1.5B, following recent evaluation protocols (Yue et al., 2025). As shown in Figure 4, ATLAS yields consistent gains across all K, including a modest 0.3% improvement at $Pass@64$, indicating that ATLAS explores the reasoning space and reaches the correct reasoning more effectively than other baselines.

6.6 Decision Statistics

We show in Table 4 the distribution of steering modes across different model sizes. Overall, larger models rely more heavily on execution mode, while smaller models require more reflection and transition steps. This pattern suggests that bigger models are more confident in solving new tasks, maintaining stable reasoning trajectories without frequently needing to transition between strategies or reflect on their reasoning process.

7 Conclusion

We presented ATLAS, a latent-space adaptive steering framework that improves both reasoning efficiency and robustness in LLMs. By using external verifier operating directly on hidden states, ATLAS substantially reduces inference overhead while retaining strong performance. Across in-domain and cross-domain reasoning benchmarks, ATLAS consistently improves accuracy and reduces token usage. These results demonstrate that latent verifier offers a practical and scalable approach to adaptive reasoning control in LLMs.

Limitations

While this work introduces an external guidance framework for automatically steering model behavior at inference time, it has several limitations. In particular, the selection of intervention layers and the calibration of steering strength for each steering vector are currently determined heuristically, rather than learned or optimized in a data-driven manner. Developing principled methods for adaptive layer selection and strength calibration remains an open challenge. In addition, our framework considers a limited set of discrete steering choices at each reasoning step, and does not model more fine-grained or continuous decision spaces over steering actions. This design simplifies inference-time control but may restrict the expressiveness of the intervention policy and limit potential performance gains. Exploring richer steering action spaces and joint optimization over layer selection, steering strength, and intervention timing is an important direction for future work.

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *EACL*.

Team AlphaProof and Team AlphaGeometry. 2024. Ai achieves silver-medal standard solving international 178 mathematical olympiad problems. *DeepMind blog*.

Seyedarmin Azizi, Erfan Baghaei Potraghloo, and Massoud Pedram. 2025. Activation steering for chain-of-thought compression. *arXiv*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *AAAI*.

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

Tom B Brown. 2020. Language models are few-shot learners. *NeurIPS*.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2023. Discovering latent knowledge in language models without supervision. *ICLR*.

Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. 2025a. Seal: Steerable reasoning calibration of large language models for free. *COLM*.

Xinghao Chen, Anhao Zhao, Heming Xia, Xuan Lu, Hanlin Wang, Yanjun Chen, Wei Zhang, Jian Wang, Wenjie Li, and Xiaoyu Shen. 2025b. Reasoning beyond language: A comprehensive survey on latent chain-of-thought reasoning. *arXiv*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2025c. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *ICML*.

Zui Chen, Tianqiao Liu, Mi Tian, Qing Tong, Weiqi Luo, and Zitao Liu. 2025d. Advancing mathematical reasoning in language models: The impact of problem-solving data, data synthesis methods, and training stages. *ICLR*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv*.

Heejin Do, Jaehui Hwang, Dongyoon Han, Seong Joon Oh, and Sangdoo Yun. 2025. What defines good reasoning in llms? dissecting reasoning steps with multi-aspect evaluation. *arXiv*.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana Rosing, Ion Stoica, and 1 others. 2025a. Efficiently scaling llm reasoning with certaindex. *NeurIPS*.

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

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. 2025. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *COLM*.

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*.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2025. Training large language models to reason in a continuous latent space. *COLM*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv*.

Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, and 1 others. 2025. Exploring concept depth: How large language models acquire knowledge and concept at different layers? In *COLING*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *ICLR*.

Zhu Liu, Cunliang Kong, Ying Liu, and Maosong Sun. 2024. Fantastic semantics and where to find them: Investigating which layers of generative llms reflect lexical semantics. *ACL Findings*.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A Smith, Hannaneh Hajishirzi, and Nathan Lambert. 2025. Rewardbench 2: Advancing reward model evaluation. *arXiv*.

Mathematical Association of America. 2023. American mathematics competitions (amc) 10 and 12, 2023. Problems and Answer Keys.

Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. 2025. Improving instruction-following in language models through activation steering. *ICLR*.

Haoxiang Sun, Yingqian Min, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, Zheng Liu, Zhongyuan Wang, and Ji-Rong Wen. 2025. Challenging the boundaries of reasoning: An olympiad-level math benchmark for large language models. *arXiv*.

Qwen Team. 2024. Qwq: Reflect deeply on the boundaries of the unknown.

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

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023. On the planning abilities of large language models-a critical investigation. *NeurIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NeurIPS*.

Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. 2025. Understanding reasoning in thinking language models via steering vectors. *ICLR Workshop on Reasoning and Planning for Large Language Models*.

Hieu M Vu and Tan M Nguyen. 2025. Angular steering: Behavior control via rotation in activation space. *NeurIPS*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *ACL*.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, and 1 others. 2025. Thoughts are all over the place: On the underthinking of o1-like llms. *ICML*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. 2025. Demystifying llm-based software engineering agents. *The ACM on Software Engineering*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *NeurIPS*.

Dongkeun Yoon, Seungone Kim, Sohee Yang, Sunkyoung Kim, Soyeon Kim, Yongil Kim, Eunbi Choi, Yireun Kim, and Minjoon Seo. 2025. Reasoning models better express their confidence. *NeurIPS*.

Yang Yue, Zhiqi Chen, and 1 others. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv*.

Zhenyu Zhang, Xiaoxia Wu, Zhongzhu Zhou, Qingyang Wu, Yineng Zhang, Pragaash Ponnusamy, Harikaran Subbaraj, Jue Wang, Shuaiwen Leon Song, and Ben Athiwaratkun. 2025. Understanding and steering the cognitive behaviors of reasoning models at test-time. In *NeurIPS Workshop on Efficient Reasoning*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and 1 others. 2023. Least-to-most prompting enables complex reasoning in large language models. *ICLR*.

Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, Tianle Cai, Taylor Kergan, Assel Kembay, Andrew Smith, Chenghua Lin, Binh Nguyen, Yuqi Pan, Yuhong Chou, Zefan Cai, and 14 others. 2025. A survey on latent reasoning.

A Example Appendix

A.1 Criteria for different thoughts

Table 5 shows how thoughts can be categorized into different groups.

Table 5: Criteria for recognizing transition and reflection thoughts

Transition	"alternatively", "think differently", "another way", "another approach", "another method", "another solution", "another strategy", "another technique"
Reflection	"wait", "verify", "make sure", "hold on", "think again", "'s correct", "'s incorrect", "let me check", "seems right"

A.2 Layer Ablation Study Detail

For the layer ablation study, we randomly sample 1,000 examples from the MATH dataset. Prior work suggests that shallow layers primarily encode low-level lexical features. In contrast, middle layers capture higher-level semantic and conceptual representations that are more closely associated with reasoning processes (Liu et al., 2024; Jin et al., 2025; Chen et al., 2025a). Motivated by these findings, we focus our ablation on the middle layers of each model to determine where steering interventions have the most impact. Specifically, for R1-Distill-1.5B and R1-Distill-7B, which each contain 28 layers, we evaluate interventions applied to layers in the range $[15, 25]$. For larger models, including R1-Distill-32B and QwQ-32B-Preview, we perform ablations over layers $[45, 60]$. For each configuration, we record both task accuracy and test-time token usage to analyze the trade-offs between effectiveness and efficiency.