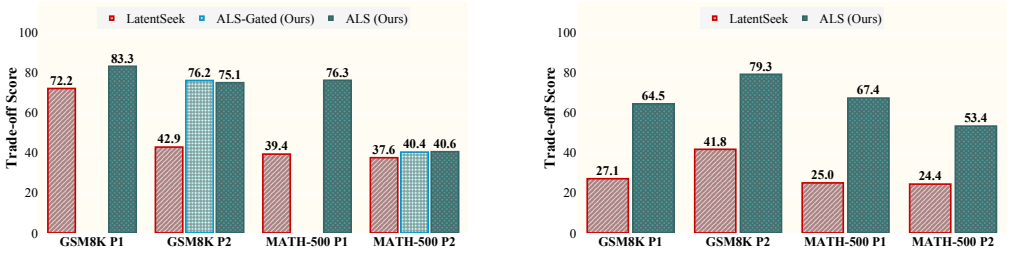
Amortized Latent Steering: Low-Cost Alternative to Test-Time Optimization

Nathan Egbuna Saatvik Gaur Kevin Zhu
Sunishchal Dev Ashwinee Panda Maheep Chaudhary*
Algoverse AI Research
negbuna26@andover.edu, maheepchaudhary.research@gmail.com

Abstract

Test-time optimization remains impractical at scale due to prohibitive inference costs—techniques like iterative refinement and multi-step verification can require $10 - 100\times$ more compute per query than standard decoding. Latent space test-time optimization methods like LatentSeek offer a more direct approach by steering hidden representations, but still demand expensive per-query optimization loops with multiple backward passes. We propose AMORTIZED LATENT STEERING (ALS), which collapses this iterative optimization into a single offline-computed vector applied at constant cost during inference. ALS computes the mean difference between hidden states from successful versus unsuccessful generations, then uses this direction to calibrate the model's hidden representations: when decoding drifts away from the success manifold, ALS nudges activations back toward it. Across GSM8K and MATH-500 benchmarks, ALS achieves $2 - 5\times$ speed-up over iterative methods while matching or surpassing greedy Chain-of-Thought and Self-Consistency baselines, yielding up to 101% improvement in efficiency–accuracy trade-off. These results show that much of latent optimization's benefit can be captured offline, making sophisticated reasoning techniques viable for production deployment. Code is available at `https://github.com/negbuna/ALS`.



(a) Qwen-2.5-7B-Instruct (b) Llama-3.1-8B-Instruct

Figure 1: **ALS consistently outperforms LatentSeek across all evaluation settings while matching or exceeding Chain-of-Thought performance.** Results show efficiency–accuracy trade-offs on GSM8K and MATH-500 for both Qwen-2.5-7B and Llama-3.1-8B models.

*Project Lead

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: .

1 Introduction

Test-time optimization (TTO) methods require $10 - 100\times$ more compute than standard decoding, making them impractical for production deployment. Even sophisticated latent-space methods like LatentSeek demand expensive per-query optimization loops with multiple backward passes.

We propose Amortized Latent Steering (ALS), inspired by causal intervention techniques that identify which latent representations causally influence model outputs. Rather than iteratively optimizing during inference, ALS pre-computes steering directions by analyzing the causal relationship [6, 2] between hidden states and reasoning success. The key insight is that we can intervene on the model's internal representations using directions that causally promote correct reasoning, applying these interventions at constant cost during inference.

ALS achieves dramatic efficiency gains, with $2 - 5\times$ speed-up over iterative methods while matching or surpassing greedy Chain-of-Thought (CoT) and Self-Consistency baselines across GSM8K and MATH-500. Most significantly, on the challenging MATH-500 benchmark, ALS yields up to 101% improvement in efficiency–accuracy trade-off, demonstrating that the computational benefits of latent optimization can be fully amortized offline without performance degradation. We discuss related TTO and latent intervention methods in Appendix A.

Our contributions are: (1) A computationally practical latent steering method that amortizes expensive TTO into a single offline-computed vector, (2) Superior efficiency–accuracy trade-offs and up to 101% improvement on challenging benchmarks, and (3) Insights into how steering strength and model architecture determine intervention effectiveness.

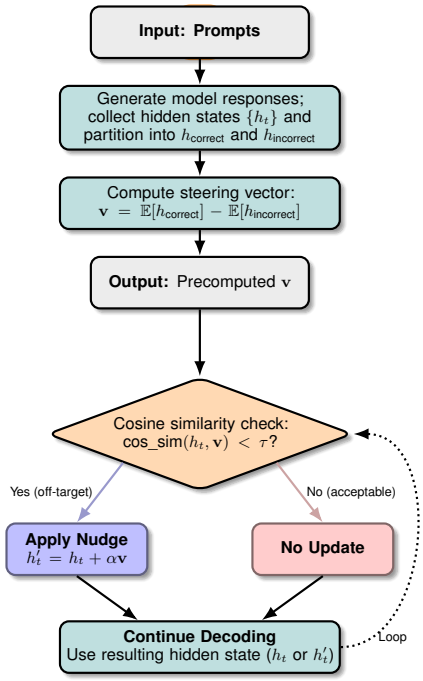


Figure 2: ALS workflow. Offline, the model computes a steering vector to nudge hidden states $\{h_t\}$ toward successful reasoning trajectories.

2 Method

Amortized Latent Steering ALS collapses iterative TTO into a single pre-computed vector. Offline, we generate model outputs from 1,000 GSM8K [1] and 500 MATH [4] examples (disjoint from evaluation) and extract the final token hidden state h_t at the penultimate layer for each generation, consistent with prior latent intervention work [3]. The combination of outputs yields two distributions of h_t (correct vs. incorrect). Correctness is determined by automatic answer verification against ground-truth, following LatentSeek [5]. For JSON prompts, exact format validity is also required. The steering vector is defined as

$$\mathbf{v} = \mathbb{E}[h_{\text{correct}}] - \mathbb{E}[h_{\text{incorrect}}]$$

capturing the latent difference between successful and failed trajectories.

At test time, we monitor cosine similarity between the current hidden state h_t and the steering vector $\mathbf{v}$ at each token. When similarity falls below threshold $\tau = 0.1$, we apply a lightweight additive update:

$$h'_t = h_t + \alpha\mathbf{v},$$

where α controls steering strength. The threshold $\tau = 0.1$ was selected through validation on 200 held-out examples, balancing intervention frequency (applied to $\sim$30% of tokens) with performance.

Higher values led to over-steering and degraded accuracy, while lower values missed correction opportunities. This monitoring and intervention process requires only vector operations with no backward passes, adding negligible computational overhead comparable to standard decoding.

We extract hidden states from the penultimate layer as this layer contains high-level semantic representations while remaining sufficiently close to the output to influence generation. We also explored injecting steering vectors at earlier and deeper layers. However, modifying hidden states outside the penultimate layer consistently caused attention mask mismatches in the `transformers` library due to residual connections expecting tensors with fixed shapes. This made evaluation unstable, so we restricted our analysis to the penultimate layer, which was both stable and semantically rich.

Experimental Setup For each dataset, we consider two prompt formats, following the experimental setup of LatentSeek. In the first (P1), a free-form CoT prompt instructs the model to reason step by step in natural language before producing a final answer. In the second (P2), a structured JSON prompt requires the model to return its answer in a rigid schema `{"thought process": "...", "final answer": "..."}`, introducing syntactic constraints in addition to correctness requirements.

Variants and Complexity We explored a gated variant (ALS-Gated) for structured outputs. At each decoding step, ALS-Gated parses the partial generation to check whether it remains valid JSON. If adding the next token would break the structure (i.e., cause a parsing error), the steering update is suppressed by setting $\alpha = 0$ for that step. This gating mechanism ensures that latent interventions do not corrupt required output formats while still allowing steering to improve reasoning quality within the valid schema.

Unlike LatentSeek, which requires $O(k \cdot B)$ operations at test time for k backward passes of cost B, ALS (and ALS-Gated) reduce this to $O(1)$ per token—only a cosine similarity computation and potential vector addition per step.

3 Experimentation and Results

We evaluate using a trade-off score that balances accuracy and efficiency:

$$\text{Trade-off} = \frac{\text{Accuracy} + (100 - \text{Normalized Time})}{2} \tag{1}$$

where normalized time scales the slowest method to 100. Table 1 compares ALS against baselines across models, datasets, and prompt styles using this trade-off score.

We evaluated ALS on two open source instruction-tuned models, Qwen-2.5-7B-Instruct and Llama-3.1-8B-Instruct, using the full GSM8K test set (1,319 examples) and MATH-500 dataset (500 examples). Baselines include greedy CoT decoding [10], Self-Consistency [9] with k=5 samples, and LatentSeek, representing standard decoding and optimization-heavy latent methods. Table results show best α per setting, as performance varies with hyperparameter α, as expected (see Table 2). All methods use identical random seeds and problem splits for reproducibility. Accuracy and average inference latency serve as evaluation metrics, with trade-off scores combining both for comparison under efficiency–accuracy constraints.

On GSM8K, ALS variants consistently outperform LatentSeek and compete closely with CoT. On the more challenging MATH-500 dataset, ALS shows greater sensitivity to α but consistently outperforms both LatentSeek and CoT while significantly reducing inference time.

3.1 Efficiency, Task, and Model Sensitivity

Notably, when ALS underperforms relative to CoT, the margins are modest (0.4-14% decrease), but when ALS succeeds, the improvements are substantial (19-101% increase), indicating that the method either maintains competitive performance or delivers significant gains.

On GSM8K, ALS demonstrates strong performance parity with CoT across both models and prompt types. However, on the more challenging MATH-500 benchmark, ALS shows dramatic improvements, particularly for Qwen-2.5-7B. This suggests ALS provides greater benefits on harder reasoning tasks where latent steering can more effectively guide the model away from incorrect solution paths.

Table 1: Trade-off, accuracy (Acc.), and average inference time (s) for ALS (best α) and standard baselines. The best method is highlighted in bold, and the second best is underlined. Teal numbers indicate the percent change of ALS trade-off relative to CoT.

			Qwen-2.5-7B-Instruct			**Llama-3.1-8B-Instruct**		
Dataset	Prompt	Method	Acc.	Time	Trade-off	Acc.	Time	Trade-off
MATH-500	P1	LatentSeek	75.4	47.0	39.4	50.0	106.8	25.0
		CoT	76.0	9.9	<u>77.8</u>	52.0	10.8	**70.8**
		SC	76.0	48.6	38.0	52.0	54.0	26.0
		ALS (Ours)	91.0	5.2	**93.1** (+19.7%)	44.8	10.6	<u>67.4</u> (-4.8%)
	P2	LatentSeek†	56.0	43.1	37.6	48.7	116.5	24.4
		CoT	3.5	10.7	<u>41.7</u>	5.0	11.5	<u>44.8</u>
		SC	3.0	53.3	1.5	5.0	56.2	1.4
		ALS-Gated (Ours)	3.0	11.8	40.4	-	-	-
		ALS (Ours)	68.5	2.2	**83.8** (+101.0%)	16.0	10.7	**53.4** (+19.2%)
GSM8K	P1	LatentSeek	90.4	9.8	72.2	54.1	18.8	27.1
		CoT	91.0	4.3	**85.5**	54.0	4.5	**74.8**
		SC	91.0	21.3	45.5	54.0	19.8	36.0
		ALS (Ours)	90.6	5.1	<u>83.3</u> (-2.6%)	51.4	4.2	<u>64.5</u> (-13.8%)
	P2	LatentSeek	85.8	13.3	42.9	83.6	30.6	41.8
		CoT	66.0	1.7	**76.5**	60.0	3.2	<u>78.4</u>
		SC	66.0	8.6	50.5	60.0	15.5	37.3
		ALS-Gated (Ours)	72.0	2.6	<u>76.2</u> (-0.4%)	-	-	-
		ALS (Ours)	70.4	2.7	75.1	68.8	3.1	**79.3** (+1.1%)

† Evaluated on a 224-example stratified subset due to compute cost.

Cross-architectural analysis reveals distinct sensitivities to latent interventions. Qwen-2.5-7B consistently shows larger improvements from ALS steering, particularly on MATH-500 where accuracy jumps from 76% (CoT) to 91% (ALS) on free-form prompts. In contrast, Llama-3.1-8B shows more modest accuracy improvements but maintains the efficiency gains, suggesting architecture-specific differences in latent geometry and steering effectiveness.

The structured JSON format (P2) presents unique challenges, with both models showing degraded CoT performance compared to free-form prompts. However, ALS demonstrates remarkable robustness: on MATH-500 P2, ALS achieves 68.5% accuracy compared to CoT's 3.5%, a 65 percentage point improvement that translates to a 101% trade-off score increase, showing ALS can maintain reasoning coherence even under syntactic constraints where standard decoding fails.

3.2 Mechanism: Confidence Calibration and Representation Disambiguation

Analysis suggests that ALS primarily improves confidence calibration. The steering vector, computed from successful versus unsuccessful trajectories, nudges hidden states toward regions historically associated with correct outputs, reducing the likelihood of over-committing to incorrect states and resolving competition between latent directions. ALS thus improves reliability while serving as a probe into latent-space geometry (see Figures 3 & 4 for visualizations).

4 Conclusion

We introduced ALS, which replaces per-input optimization with a precomputed vector derived from successful generations. On GSM8K and MATH-500, ALS achieves 2–5$\times$ faster inference than LatentSeek and matches or surpasses strong decoding baselines including CoT and Self-Consistency.

Like other methods, performance requires hyperparameter tuning of α. ALS offers a low-cost, interpretable approach for guiding hidden representations and probing latent-space geometry.

References

[1] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[2] Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, et al. Causal abstraction: A theoretical foundation for mechanistic interpretability. *Journal of Machine Learning Research*, 26(83):1–64, 2025.

[3] S. Hao, S. Sukhbaatar, D. Su, X. Li, Z. Hu, J. Weston, and Y. Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

[4] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

[5] H. Li, C. Li, T. Wu, X. Zhu, Y. Wang, Z. Yu, E.H. Jiang, S.-C. Zhu, Z. Jia, Y.N. Wu, and Z. Zheng. Seek in the dark: Reasoning via test-time instance-level policy gradient in latent space. *arXiv preprint arXiv:2505.13308*, 2025.

[6] Haoyang Liu, Maheep Chaudhary, and Haohan Wang. Towards trustworthy and aligned machine learning: A data-centric survey with causality perspectives. *arXiv preprint arXiv:2307.16851*, 2023.

[7] Sheng Liu, Tianlang Chen, Pan Lu, Haotian Ye, Yizheng Chen, Lei Xing, and James Zou. Fractional reasoning via latent steering vectors improves inference time compute. *arXiv preprint arXiv:2506.15882*, 2025.

[8] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering, 2024.

[9] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023.

[10] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[11] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to ai transparency, 2025.

A Related Work

Test-time optimization for reasoning spans a spectrum from expensive iterative methods to lightweight interventions. Optimization-heavy approaches like LatentSeek [5] and Fractional Reasoning [7] iteratively adjust hidden states at test time but require expensive query-specific operations and substantial computational overhead. Notably, LatentSeek itself was only evaluated on mathematical reasoning benchmarks, such as GSM8K, MATH-500, and AIME-2024, making our focus on GSM8K and MATH-500 directly comparable and consistent with prior work. Linear intervention methods including COCONUT [3] and activation steering [8, 11] show that hidden state modifications can shift model behavior, but typically require extensive task-specific tuning and are better suited for synthetic QA or stylistic control rather than complex symbolic reasoning tasks.

ALS bridges these approaches by using offline computation to derive a single steering vector from successful vs. unsuccessful trajectories, then applying lightweight linear interventions during inference. Unlike optimization-heavy methods, ALS requires no backward passes or large online overhead; unlike existing activation steering, it specifically targets symbolic reasoning accuracy through success/failure trajectory differences. This design enables ALS to achieve the accuracy benefits of iterative optimization and surpass standard decoding baselines while maintaining constant-time inference overhead.

B Additional information

B.1 Implementation Details

All experiments used $1\times$ NVIDIA H200. LatentSeek on Llama-3.1-8B proved prohibitively expensive ($>$100s per MATH-500 example), so we report results on a stratified 224-example subset for those settings. All methods used identical problem splits with fixed random seeds for reproducibility.

B.2 Ablation Studies

ALS ablations were conducted on Qwen-2.5-7B-Instruct for efficiency; Llama runs were omitted due to computational constraints. We ablate steering strength (α) and evaluate a gated variant. Table 2 shows that moderate values α improve free-form accuracy, while weak or strong steering is better suited for structured constraints.

Table 2: Ablation results for ALS on Qwen-2.5-7B-Instruct. We sweep steering strength $\alpha \in \{0.0, 0.1, 0.3, 0.6\}$ and report accuracy (%), average generation time (s), and trade-off score (Equation 1). For $\alpha = 0.0$, the model still uses offline training data for reasoning. Llama results are omitted due to computational constraints. The best performances are highlighted in bold.

Dataset	Prompt	Alpha	Accuracy (%)	Time (s)	Trade-off
GSM8K	P1	$\alpha = 0.0$	76.0	6.8	84.6
GSM8K	P1	$\alpha = 0.1$	76.0	6.7	84.7
GSM8K	P1	$\alpha = 0.3$	90.6	5.1	**88.8**
GSM8K	P1	$\alpha = 0.6$	86.5	5.4	87.1
GSM8K	P2	$\alpha = 0.0$	23.0	16.4	54.3
GSM8K	P2	$\alpha = 0.1$	23.3	16.3	54.5
GSM8K	P2	$\alpha = 0.3$	70.4	2.7	**75.1**
GSM8K	P2	$\alpha = 0.6$	70.5	2.0	74.8
MATH-500	P1	$\alpha = 0.0$	91.0	5.0	**93.1**
MATH-500	P1	$\alpha = 0.1$	90.5	5.4	93.0
MATH-500	P1	$\alpha = 0.3$	73.4	10.1	76.3
MATH-500	P1	$\alpha = 0.6$	91.0	5.2	**93.1**
MATH-500	P2	$\alpha = 0.0$	68.5	2.2	**83.8**
MATH-500	P2	$\alpha = 0.1$	66.5	2.4	83.0
MATH-500	P2	$\alpha = 0.3$	1.8	11.0	40.6
MATH-500	P2	$\alpha = 0.6$	67.5	2.6	83.2

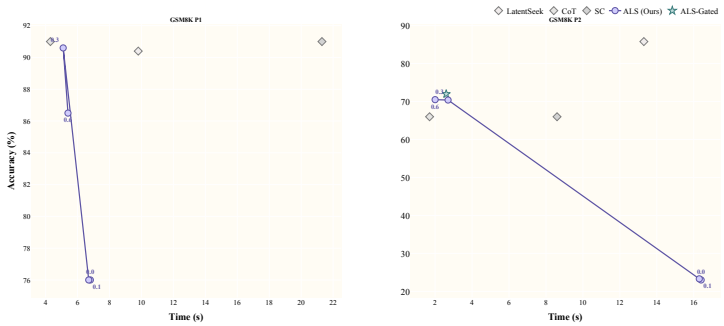


Figure 3: Pareto frontier for ALS on GSM8K. Each point corresponds to a steering strength α, showing the trade-off between accuracy and inference time. The curve highlights how intermediate α values yield the best balance of efficiency and accuracy.

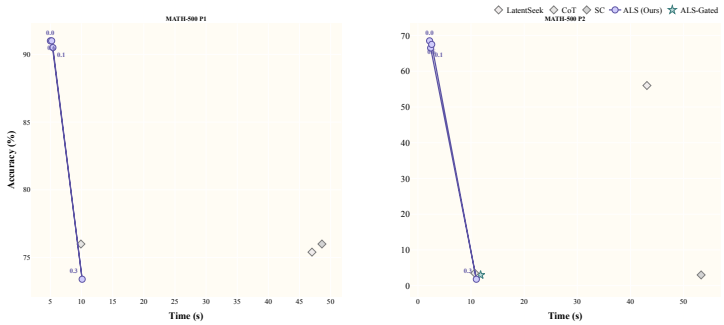


Figure 4: Pareto frontier for ALS on MATH-500. Results illustrate higher sensitivity to α compared to GSM8K, with both weak and strong steering outperforming moderate values depending on prompt style.

B.3 Limitations

ALS relies on a single global steering vector, which may not capture the full complexity of reasoning trajectories across diverse problem types. Effectiveness appears architecture-dependent, with different optimal α values across model families, suggesting limited generalizability. Following LatentSeek, our evaluation focuses on mathematical reasoning with two prompt formats, leaving broader domain applicability uncertain. The offline computation requires ground-truth labels for steering vector construction, which may not be available for all tasks.

B.4 Broader Impacts

ALS aims to improve computational efficiency of reasoning systems with several potential societal implications. Positive impacts include democratizing access to advanced reasoning capabilities through reduced computational costs and lower energy consumption at scale. However, more efficient reasoning systems could accelerate both beneficial and harmful AI applications. While our work focuses on mathematical reasoning, similar techniques could extend to other domains with dual-use implications. The method introduces no new safety concerns beyond those of the underlying language models.

B.5 Future Work

Future directions include extending ALS beyond a single global vector to multivector or task-adaptive steering, exploring dynamic or layer-wise injection strategies for multiconstraint reasoning, and applying ALS to domains beyond math, such as coding or scientific QA. We also see potential in using ALS as a diagnostic tool for latent-space geometry to inform future model design.