

HintSteer: Predicting Context Imprints for Inference-Time Steering

Xinda Wang^{1*} Chenzhuo Zhao^{1*} You Hu² Guotong Li¹

Chuang Ou¹ Yuhao Zheng³

¹Peking University ²Zhejiang University

³University of Science and Technology of China
{nev_settle}@stu.pku.edu.cn

Abstract

Large language models rely on inference-time context including rationales and instructions to improve task performance. Prior work attempts to internalize such benefits via parameter-efficient fine-tuning, and also studies inference-time representation interventions, but these approaches typically update the base model or learn fixed directions tied to global attributes. As a result, it remains unclear how to capture and reuse the internal effect of a helpful context that is specific to a given query. We take a representation-level view and define a *context imprint* as the hidden-state displacement between context-conditioned and query-only forward passes. We propose **HintSteer**, which trains a lightweight query-conditioned predictor to estimate the imprint from the query representation and injects it into a frozen model at inference time, approximating the missing context effect without explicitly providing context. Experiments on mathematical reasoning and writing demonstrate the effectiveness of imprint-based steering under no-context inference and indicate that both imprint magnitude and predictability influence the resulting gains.

1 Introduction

Large language models are often improved by providing additional context at inference time (Chen et al.). Such context includes instructions, demonstrations, rationales, and task-specific hints (Feng et al., 2024). In reasoning tasks, short hints can change the solution path (Li et al., 2024b). In writing tasks, instructions can constrain style and content (Konen et al., 2024; Stolfo et al., 2025). A common concern is that these benefits depend on providing the context at inference time. This increases

prompt length and introduces dependence on context retrieval or generation (Li et al., 2024a). When context improves a model output, what change does it induce in the model representations (Panickssery et al., 2023), and can the same change be induced when the context is absent?

Existing approaches address context dependence mainly by improving the external context, through prompt engineering and in-context learning (Zou et al., 2024), or by internalizing context benefits into model parameters through fine-tuning. In parallel, activation steering shows that intervening on hidden states can change model behavior. Recent work has extended intervention to style control, debiasing, and multi-property control (Li et al., 2025; Scalena et al., 2024). However, most steering methods learn fixed directions associated with global attributes (Rahn et al., 2024). Dynamic or conditional steering methods make the intervention more adaptive, but they still mainly target behavior-level control rather than the hidden-state effect of a missing helpful context (Wang et al., 2025b).

In this work, we study context effects at the representation level. Given a query q and a helpful context c , we define the *context imprint* at layer ℓ as the hidden-state displacement between a context-conditioned forward pass and a query-only forward pass: $v_\ell(q, c) = h_\ell(q, c) - h_\ell(q)$. We propose **HintSteer**, which learns to predict the imprint from the query representation and inject it into a frozen base model during inference. We evaluate HintSteer on mathematical reasoning and writing. Results on both tasks indicate that predicting and injecting context imprints can improve performance without providing the context at inference time. Our contributions are:

- We formulate the effect of helpful context as a context imprint, and propose **HintSteer** to predict and inject imprints into frozen LLMs without providing context at inference time.

*Contributed equally.

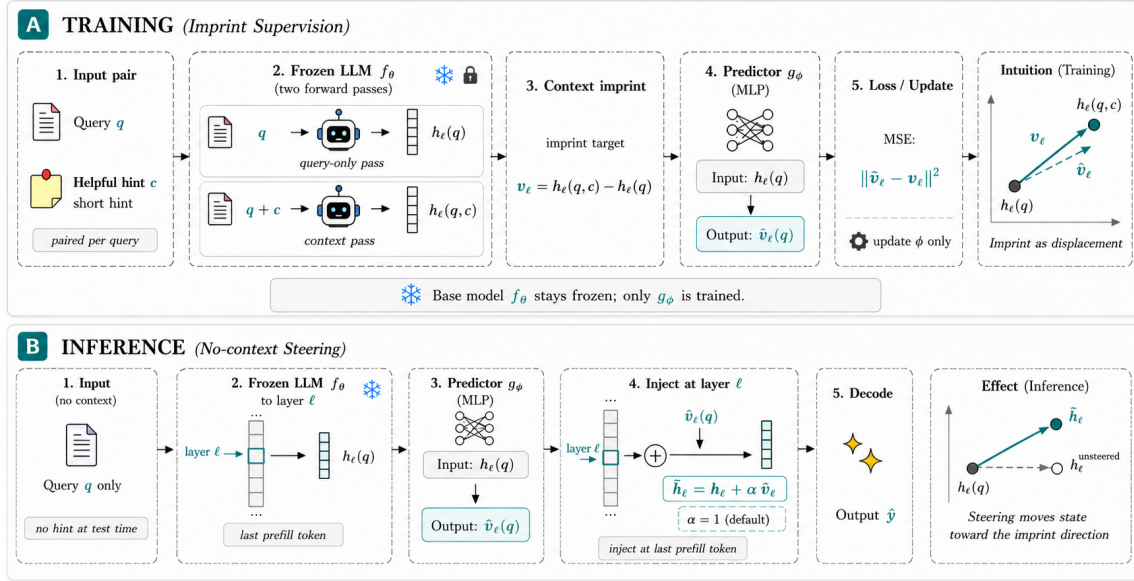


Figure 1: Overview of HintSteer. Training computes context imprints as hidden-state differences between context-conditioned and query-only forward passes and trains a lightweight predictor to estimate the imprint from the query representation. Inference uses no context and injects the predicted imprint into a frozen v model at layer ℓ .

- We demonstrate that HintSteer yields improvements of varying magnitudes on both mathematical reasoning benchmarks and Writing-Bench across different model architectures.
- We provide analyses of layer selection, output length sensitivity, and injection strategies, clarifying when imprint steering is effective.

2 Related Work

A growing line of work shows that model outputs can be influenced by intervening on internal activations at inference time. Activation Addition constructs steering vectors from contrastive prompt pairs and adds them to intermediate representations to guide generation (Turner et al., 2024). Additionally, recent methods explore alternative or adaptive intervention mechanisms, including conceptor-based steering, entropy-guided control, and conditional activation interventions (Postmus and Abreu, 2024; Lee et al., 2025). Representation engineering studies how high-level concepts can be read from and controlled through population-level representations (Zou et al., 2023). Activation interventions have also been applied to safety calibration, adversarial red-teaming, knowledge editing, and feature-level control with sparse or interpretable latent features (Cao et al., 2025; Scialanga et al., 2025; Soo et al., 2025; Yang et al., 2025). HintSteer differs by treating the context effect itself as supervision and learning a query-conditioned imprint

predictor for no-context inference.

3 Methodology

3.1 Problem Formulation

Let f_θ denote a frozen autoregressive language model with L transformer layers. Each training example consists of a query q , a helpful context c , and a target answer y . The context c can be a hint, rationale, or instruction that improves the model response when appended to the query. During training, both q and c are available. During inference, only q is provided. Our objective is to approximate the benefit of c without including it in the input at inference time. Instead of generating or retrieving a new context, we train an external predictor that maps the query representation to a steering vector. The base language model remains frozen throughout training and inference.

3.2 Context Imprint Extraction

We first characterize the internal effect of helpful context. For each query-context pair (q, c) , we run the frozen model under two input conditions: the query-only input and the context-conditioned input. At layer ℓ , let $h_\ell(q)$ denote the hidden state at the last prefill token for the query-only forward pass, and let $h_\ell(q, c)$ denote the corresponding hidden state when the model processes the query with the context. We define the *context imprint* as the

Methods	MATH-500	AMC	AIME	AIME25	OlympiadBench	Minerva	Total
Baseline	51.60	39.76	13.33	<u>3.33</u>	18.37	16.54	27.28
HintSteer (L16)	<u>58.80 (+7.20)</u>	48.19 (+8.43)	<u>16.67 (+3.33)</u>	6.67 (+3.33)	20.89 (+2.52)	14.34 (-2.21)	32.84 (+5.56)
L8	54.20	43.40	10.00	<u>3.33</u>	19.10	16.20	28.60 (+1.32)
L12	57.00	43.40	<u>16.67</u>	<u>3.33</u>	20.30	16.54	30.10 (+2.82)
L20	60.20	44.60	13.33	6.67	18.20	18.00	30.70 (+3.42)
L22	56.60	<u>47.00</u>	20.00	6.67	<u>20.60</u>	16.20	<u>32.70 (+5.42)</u>
L24	55.20	39.76	<u>16.67</u>	<u>3.33</u>	17.00	<u>17.30</u>	28.00 (+0.72)

Table 1: Math results on Qwen2.5-Math-7B across steerion layers on OOD benchmark.

hidden-state displacement induced by the context:

$$v_\ell(q, c) = h_\ell(q, c) - h_\ell(q). \quad (1)$$

This imprint represents the local change in model computation induced by the context at layer ℓ .

3.3 Query-Conditioned Imprint Prediction

At inference time, the context c is unavailable, so the oracle imprint $v_\ell(q, c)$ cannot be computed. We therefore train a predictor g_ϕ to estimate the context imprint from the query representation alone:

$$\hat{v}_\ell(q) = g_\phi(h_\ell(q)). \quad (2)$$

We instantiate g_ϕ as a two-layer bottleneck MLP:

$$g_\phi(h) = W_2 \cdot \text{GELU}(W_1 h + b_1) + b_2, \quad (3)$$

where the bottleneck dimension is set to 256. The output projection is initialized to zero so that the initial intervention is close to the original model behavior. The predictor is trained with mean squared error against the oracle imprint:

$$\mathcal{L}_{\text{MSE}} = \mathbb{E}_{(q,c)} \left[\|g_\phi(h_\ell(q)) - v_\ell(q, c)\|^2 \right]. \quad (4)$$

Only the predictor parameters ϕ are updated. The LLM parameters θ remain fixed.

3.4 Inference-Time Steering

During inference, HintSteer receives only the query q . We run the model to layer ℓ and obtain the hidden state $h_\ell(q)$ at the last prefill token. The predictor produces an estimated context imprint $\hat{v}_\ell(q)$, which is added to the hidden state:

$$\tilde{h}_\ell(q) = h_\ell(q) + \alpha \hat{v}_\ell(q), \quad (5)$$

where α is a scaling coefficient and is set to 1 unless specified otherwise. Generation then proceeds using the original frozen model. Compared with standard activation steering, HintSteer differs in

two aspects. First, the steering vector is query-conditioned rather than shared across inputs. Second, the supervision signal is the hidden-state effect induced by helpful context, rather than a contrastive label for a global attribute. We analyze layer selection and steering behavior in Section 4.

4 Experiments

4.1 Experimental Setup

Models, datasets, and benchmarks. For mathematical reasoning, we train the imprint predictor on MathLightEval and evaluate on its test split. We evaluate out-of-distribution (OOD) generalization on AMC, AIME, AIME25, MATH-500, OlympiadBench, and Minerva. For writing, we train on DeepWriting (Wang et al., 2025a) and evaluate on WritingBench (Wu et al., 2026). The helpful contexts used for imprint extraction are short hints generated by GPT5.4 conditioned on the query. We use Qwen2.5-Math-7B for math experiments and Llama-3.1-8B-Instruct for cross-model comparison. For writing experiments, we use Qwen2.5-7B-Instruct.

Training and inference protocol. For each training example, we compute the context imprint as the hidden-state difference between context-conditioned and query-only forward passes at layer ℓ , and train a bottleneck MLP to predict this imprint from the query representation using MSE loss. The base model is frozen throughout. At inference time, the model receives only the query. Unless stated otherwise, we apply single-steer by injecting the predicted imprint once at the last prefill token of layer ℓ with $\alpha = 1$. For writing, we additionally evaluate multi-steer, which repeats imprint injection at multiple decoding steps to mitigate attenuation over long generations.

4.2 Math: Main Results and Layer Effects

Table 1 reports OOD accuracy on Qwen2.5-Math-7B across injection layers. HintSteer (L16)

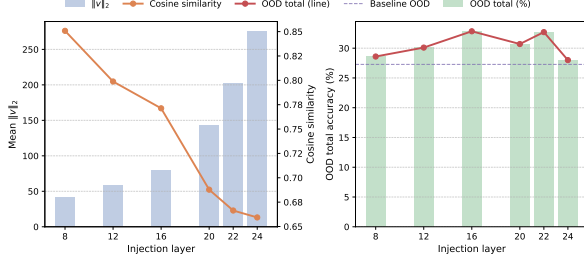


Figure 2: Layer-wise statistics on Qwen2.5-Math-7B. Left: mean imprint norm $\|v\|_2$ (bar) and cosine similarity between predicted and oracle imprints (line). Right: OOD total accuracy across injection layers.

achieves the best overall OOD total (32.84%), improving over the baseline (27.28%) by 5.56 points. Improvements are concentrated on AMC and MATH-500, while AIME and AIME25 show smaller gains. Minerva decreases at L16, suggesting that the injected signal can shift performance unevenly across distributions. A key pattern is that OOD performance is not monotonic in the injection layer. L16 and L22 yield the strongest OOD totals, whereas deeper layers such as L24 provide limited improvement. This non-monotonicity is consistent with the layer-wise statistics in Figure 2: the imprint norm increases with depth, while cosine similarity decreases, indicating that larger imprints at later layers may be harder to predict reliably.

Model	A&E	F&B	P&L	L&A	Edu	A&M
Baseline	44.02	41.96	41.56	39.64	46.38	44.50
Single-steer	44.50	41.77	41.56	40.14	47.17	44.77
Multi-steer	44.47	41.99	41.41	40.05	46.95	45.36

Table 2: Domain1 results on Qwen2.5-7B-Instruct. A&E: Academic & Engineering; F&B: Finance & Business; P&L: Politics & Law; L&A: Literature & Arts; Edu: Education; A&M: Advertising & Marketing.

4.3 Writing: Results and Length Sensitivity

We evaluate HintSteer on WritingBench using Qwen2.5-7B-Instruct. Table 2 reports Domain1 scores for the baseline, single-steer, and multi-steer. The results show that steering can improve writing performance across several domains, while the effect varies across configurations. Both steering variants improve multiple domains over the baseline. We further analyze performance by response length to understand how the intervention behaves during generation. Table 3 reports score differences against the baseline within each length bin. Single-steer shows larger gains for short-to-medium responses and smaller gains for longer generations.

This motivates multi-steer, which injects the predicted imprint multiple times during decoding and improves several bins where single-steer provides limited gains. Δ is computed against the baseline.

Response length	Single-steer Δ	Multi-steer Δ
<500	+0.089	+0.200
500–1k	+0.157	+0.333
1k–2k	+0.042	+0.263
2k–4k	-0.017	-0.400
4k–8k	+0.014	+0.173

Table 3: WritingBench length sensitivity.

4.4 Cross-Model and Imprint Strength

We apply HintSteer to Llama-3.1-8B-Instruct under the same evaluation protocol. Table 4 reports OOD results at two injection layers. Compared with Qwen2.5-Math-7B, the gains are smaller. Notably, cosine similarity remains relatively high at L16, while the imprint norm is much smaller than in Qwen2.5-Math-7B. Together with the layer-wise trend in Figure 2, this comparison indicates that prediction similarity alone is insufficient, and that imprint magnitude can limit the effect size in this cross-model setting.

Layer	$\ v\ _2$	Cos.	Baseline	Steered	Δ
L16	4.13	0.8785	15.71%	16.42%	+0.71
L31	24.52	0.7325	15.71%	16.82%	+1.11

Table 4: Math OOD results on Llama-3.1-8B-Instruct.

5 Conclusion

We studied whether the effect of helpful context can be reproduced at inference time without providing the context. We defined a *context imprint* as the hidden-state displacement between context-conditioned and query forward passes, and proposed **HintSteer**, which trains a lightweight query-conditioned predictor to estimate this imprint from the query representation and injects it into a frozen base model. Experiments on mathematical reasoning and writing show that imprint steering can improve performance under a no-context inference setting. Layer comparisons indicate that gains are not monotonic across layers, consistent with a trade-off between imprint magnitude and predictability, and cross-model results show that limited imprint strength can constrain the achievable improvement. These results support a

representation-level view of context use and suggest that predicting context-induced hidden-state effects is a viable alternative to relying on inference-time prompting.

6 Limitations

Despite the empirical results reported in this paper, our study has two main limitations. First, in writing, the helpful contexts are generated automatically and we do not have a reliable procedure to verify whether a generated hint is genuinely helpful for a given prompt. This uncertainty can introduce noise into imprint extraction and supervision, because the imprint is defined relative to a context-conditioned pass whose utility is not guaranteed. As a result, even when the steering mechanism is implemented correctly, the predictor may be trained on contexts that provide limited guidance, which can reduce the achievable gains in writing. Second, the effectiveness of HintSteer depends on the base model exhibiting a sufficiently strong internal response to the helpful context. When the context-induced imprint magnitude is small, predicted steering tends to yield limited improvements even if cosine similarity is high, which constrains performance transfer across model families and tasks.

References

- Zouying Cao, Yifei Yang, and Hai Zhao. 2025. Scans: Mitigating the exaggerated safety for llms via safety-conscious activation steering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23523–23531.
- Tong Chen, Qirun Dai, Zhijie Deng, and Dequan Wang. Demonstration distillation for efficient in-context learning.
- Kaituo Feng, Changsheng Li, Xiaolu Zhang, Jun Zhou, Ye Yuan, and Guoren Wang. 2024. Keypoint-based progressive chain-of-thought distillation for llms. *arXiv preprint arXiv:2405.16064*.
- Kai Konen, Sophie Jentzsch, Diaoulé Diallo, Peer Schütt, Oliver Bensch, Roxanne El Baff, Dominik Opitz, and Tobias Hecking. 2024. Style vectors for steering generative large language models. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 782–802.
- Bruce W Lee, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Erik Miehl, Pierre Dognin, Manish Nagireddy, and Amit Dhurandhar. 2025. Programming refusal with conditional activation steering. In *International conference on learning representations*, volume 2025, pages 90960–90985.
- Yichen Li, Zhiting Fan, Ruizhe Chen, Xiaotang Gai, Luqi Gong, Yan Zhang, and Zuozhu Liu. 2025. Fairsteer: Inference time debiasing for llms with dynamic activation steering. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11293–11312.
- Yichuan Li, Xiyao Ma, Sixing Lu, Kyumin Lee, Xiaohu Liu, and Chenlei Guo. 2024a. Mend: Meta demonstration distillation for efficient and effective in-context learning. *arXiv preprint arXiv:2403.06914*.
- Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Bin Sun, Xinglin Wang, Heda Wang, and Kan Li. 2024b. Turning dust into gold: Distilling complex reasoning capabilities from llms by leveraging negative data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18591–18599.
- Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. 2023. Steering llama 2 via contrastive activation addition. *arXiv preprint arXiv:2312.06681*.
- Joris Postmus and Steven Abreu. 2024. Steering large language models using conceptors: Improving addition-based activation engineering. *arXiv preprint arXiv:2410.16314*.
- Nate Rahn, Pierluca D’Oro, and Marc G Bellemare. 2024. Controlling large language model agents with entropic activation steering. *arXiv preprint arXiv:2406.00244*.
- Daniel Scalena, Gabriele Sarti, and Malvina Nissim. 2024. Multi-property steering of large language models with dynamic activation composition. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 577–603.
- Marco Scialanga, Thibault Laugel, Vincent Grari, and Marcin Detyniecki. 2025. Sake: Steering activations for knowledge editing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15966–15978.
- Samuel Soo, Chen Guang, Wesley Teng, Chandrasekaran Balaganesh, Tan Guoxian, and Yan Ming. 2025. Interpretable steering of large language models with feature guided activation additions. *arXiv preprint arXiv:2501.09929*.
- Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. 2025. Improving instruction-following in language models through activation steering. In *International Conference on Learning Representations*, volume 2025, pages 55790–55823.
- Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Ulisse Mini, and Monte MacDiarmid. 2024. Activation addition: Steering language models without optimization.

- Haozhe Wang, Haoran Que, Qixin Xu, Minghao Liu, Wangchunshu Zhou, Jiazhan Feng, Wanjun Zhong, Wei Ye, Tong Yang, Wenhao Huang, and 1 others. 2025a. Reverse-engineered reasoning for open-ended generation. *arXiv preprint arXiv:2509.06160*.
- Weixuan Wang, Jingyuan Yang, and Wei Peng. 2025b. Semantics-adaptive activation intervention for llms via dynamic steering vectors. In *International Conference on Learning Representations*, volume 2025, pages 79334–79351.
- Yuning Wu, Jiahao Mei, Ming Yan, Chenliang Li, Shaopeng Lai, Yuran Ren, Zijia Wang, Ji Zhang, Mengyue Wu, Qin Jin, and 1 others. 2026. Writing-bench: A comprehensive benchmark for generative writing. *Advances in Neural Information Processing Systems*, 38.
- Jingyuan Yang, Rongjun Li, Weixuan Wang, Ziyu Zhou, Zhiyong Feng, and Wei Peng. 2025. Lf-steering: Latent feature activation steering for enhancing semantic consistency in large language models. *arXiv preprint arXiv:2501.11036*.
- Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, and 1 others. 2023. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*.
- Jiaru Zou, Mengyu Zhou, Tao Li, Shi Han, and Dongmei Zhang. 2024. Promptintern: Saving inference costs by internalizing recurrent prompt during large language model fine-tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10288–10305.

A Appendix

A.1 Comparison with Random and Static Steering

To examine whether the gains come from query-conditioned imprint prediction rather than generic activation perturbations, we compare HintSteer with two additional baselines: random steer and mean static steer. Random steer injects a randomly sampled vector, while mean static steer injects the average context imprint computed over the training set as a fixed steering direction. All methods follow the same math OOD evaluation protocol as the main experiment.

As shown in Table 5, random steering leads to unstable changes across benchmarks and only marginally improves the total score. Mean static steering improves some benchmarks but decreases the total score, suggesting that a single global imprint direction is insufficient. In contrast, HintSteer achieves the strongest total improvement, indicating that query-conditioned imprint prediction is important for effective no-context steering.

A.2 GPT-Generated Helpful Contexts

Goal. We construct a short helpful context for each query by prompting an external LLM to generate a concise hint. The hint is used only for context imprint extraction and predictor training, and it is never provided to the base model at test time.

Hint generation prompts. Table 6 and Table 7 summarize the system and user prompts used to generate math and writing hints, respectively.

Generation settings and post-processing. We generate both math and writing hints using `openai.gpt-5.4-2026-03-05` via the API. We use temperature 0.3 and max tokens 300 for math hints, and temperature 0.7 and max tokens 512 for writing hints; `top_p` is left unspecified (API default), and we do not use seeds or `n-best` sampling. For math hints, we parse the JSON field "hint" from the generator output (try `json.loads`, then regex extraction if parsing fails) and remove any leading code-fence prefixes. For writing hints, we use the raw text output after `strip()` with no additional processing.

Examples. Figure 3 and Figure 4 show representative math and writing hints used for imprint extraction.

Math Example 1
Query: Let $f(x) = ax+3$ if $x \geq 2$, $x-5$ if $-2 \leq x < 2$, $2x-b$ if $x < -2$. Find $a+b$ if f is continuous.
Hint: Set the two formulas meeting at $x=2$ equal: $2a+3=2-5$.

Math Example 2
Query: Find the center of the circle with equation $x^2 - 6x + y^2 + 2y = 9$.
Hint: Complete the square in both variables: group as $(x^2-6x)+(y^2+2y)=9$ and rewrite each quadratic expression.

Figure 3: Representative math hints (opening-step guidance).

Writing Example 1 (Chinese task; translated label)
Query: Write a micro-fiction using three cue words with a twist.
Hint: Specify audience/scene and a 3-part structure; do not write the story itself.

Writing Example 2 (English)
Query: Write a Fantasy/Comedy story: You are a Norse god of alcohol. A man challenges you...
Hint: Target lighthearted urban fantasy; open mid-challenge; use banter and escalating magical stakes.

Figure 4: Representative writing hints (context guidance).

A.3 Multi-steer Protocol

Single-steer. Given a query q , we run a prefill pass with the frozen base model and obtain the hidden state at layer ℓ and the last prefill token, denoted $h_\ell^{\text{prefill}}(q)$. We predict the imprint once as $\hat{v}_\ell(q) = g_\phi(h_\ell^{\text{prefill}}(q))$ and inject it a single time with $\alpha = 1.0$:

$$\tilde{h}_\ell^{\text{prefill}}(q) \leftarrow h_\ell^{\text{prefill}}(q) + 1.0 \cdot \hat{v}_\ell(q). \quad (6)$$

Decoding then proceeds normally using the frozen base model.

Multi-steer schedule. Multi-steer repeats injection during decoding to mitigate attenuation over long generations. We reuse the same imprint prediction $\hat{v}_\ell(q)$ computed from the prefill pass and do *not* recompute it during decoding. Specifically, after the prefill injection above, we maintain a decoding-step counter that starts immediately after prefill. During decoding, every 64 generated tokens we inject at layer ℓ with a smaller strength $\alpha = 0.5$. In implementation, letting counter denote the number of tokens generated so far after prefill, we inject whenever `counter % 64 == 0`. The decoding-time injection is:

$$\tilde{h}_\ell^{(t)} \leftarrow h_\ell^{(t)} + 0.5 \cdot \hat{v}_\ell(q), \quad (7)$$

where $h_\ell^{(t)}$ denotes the layer- ℓ hidden state for the current decoding token at step t . This schedule applies the full-strength imprint once at prefill and periodically refreshes the intervention during decoding with reduced strength.

A.4 WritingBench Evaluation Details

Judge protocol. We follow the official WritingBench criteria-based evaluation protocol for all main writing results reported in this paper. For

Methods	MATH-500	AMC	AIME	AIME25	OlympiadBench	Total
Baseline	51.60	39.76	13.33	3.33	18.37	27.28
Random steer	51.60 (+0.00)	38.36 (-1.40)	19.83 (+6.50)	3.73 (+0.40)	14.07 (-4.30)	28.08 (+0.80)
Mean static steer	54.00 (+2.40)	39.76 (+0.00)	7.73 (-5.60)	3.33 (+0.00)	19.47 (+1.10)	26.48 (-0.80)
HintSteer	58.80 (+7.20)	48.19 (+8.43)	<u>16.67 (+3.33)</u>	6.67 (+3.33)	20.89 (+2.52)	32.84 (+5.56)

Table 5: Comparison with random and static steering baselines on math OOD benchmarks. Random steer injects a randomly sampled vector, mean static steer injects the average training-set imprint as a fixed steering direction, and HintSteer predicts a query-conditioned imprint. Values in parentheses denote changes over the no-steering baseline.

Field	Content
System	You generate ONE concrete opening step (“Approach”) for a math problem. Rules: (i) output the first concrete action (substitution / equation / transform / setup); (ii) do not state the final answer; (iii) one sentence, ≤ 30 words; (iv) output JSON only: {"hint": "..."}.
User template	Problem: {problem} Reference solution (for your understanding only): {solution} Emit the JSON focused on the first concrete action from the reference solution.

Table 6: Math hint generation prompts. The reference solution is truncated to the first 3,500 characters before prompting.

Field	Content
System	You are a writing coach. Given a writing task, provide brief but specific context guidance. Include (as relevant): target audience and tone; recommended structure; domain conventions; stylistic techniques. Rules: 2–5 sentences; match the input language; do not write the actual content.
User template	Writing task: {query}

Table 7: Writing hint generation prompts.

each instance, WritingBench provides a per-sample checklist (3–6 criteria) via the `checklist` field. We score each criterion independently with a judge model and obtain an integer score from 1 to 10. We use `openai.gpt-5.4-2026-03-05` as the judge and validate the returned JSON format, retrying up to three times when the output is invalid.

A.5 Training and Inference Details

Predictor architecture and optimization. The imprint predictor g_ϕ is a two-layer bottleneck MLP with bottleneck dimension 256. The output projection is initialized to zero so that the initial intervention is close to the base model behavior. We train g_ϕ using mean squared error against the oracle imprint targets, updating only ϕ while keeping the base LLM frozen. We use AdamW with learning

Field	Content
System	You are an expert evaluator with extensive experience in evaluating response of given query.
User	Evaluate the Response based on the Query and Criteria provided following the Scoring Rules. Scoring range: integer 1–10. Be strict; do not be misled by format or length; penalize incomplete or fabricated responses. Criteria: {criteria} Query: {query} Response: {response} Return JSON: {"score": <1–10>, "reason": "..."}.

Table 8: WritingBench judge prompt (criteria-based). {criteria} is instance-specific and comes from the benchmark checklist field.

rate 10^{-3} , batch size 256, and weight decay 10^{-4} . We train for 100 epochs for the math setting, and 200 epochs for writing and Llama settings.

Imprint extraction and injection implementation.

For imprint extraction, we run the frozen base model with two inputs for each query: query-only and context-conditioned (query plus the generated hint). We record the hidden state at layer ℓ and the last prefill token, and define the oracle imprint as $v_\ell(q, c) = h_\ell(q, c) - h_\ell(q)$. At inference time, we compute $\hat{v}_\ell(q) = g_\phi(h_\ell(q))$ from the prefill pass and inject it into the layer- ℓ hidden state at the last prefill token. Unless stated otherwise, we use $\alpha = 1.0$ for single-steer. For multi-steer, we follow Appendix A.3: after the prefill injection, we inject every 64 generated tokens with $\alpha = 0.5$, reusing the same $\hat{v}_\ell(q)$ without recomputation.

Decoding settings. For math benchmarks, we use deterministic decoding (greedy) following the benchmark protocol. For writing, we use sampling with temperature 0.7 and top- p 0.8, and set `max_new_tokens` to 16,000. All decoding parameters are kept fixed across baseline and steered runs to isolate the effect of steering.