Steering LLM Reasoning Through Bias-Only Adaptation

Viacheslav Sinii[1], Alexey Gorbatovski[1], Artem Cherepanov[1,2], Boris Shaposhnikov[1], Nikita Balagansky[1], Daniil Gavrilov[1],

[1]T-Tech, [2]Central University,

Correspondence: siniy.vyacheslav@gmail.com

Abstract

We show that training a single d-dimensional steering vector per layer with reinforcement learning, while freezing all base weights, matches the accuracy of fully RL-tuned reasoning models on mathematical-reasoning tasks. On an 8 billion-parameter model this adds only $\approx 0.0016\%$ additional parameters and reproduces performance across a range of base models and mathematical-reasoning benchmarks. These results tighten the upper bound on the parameter budget required for high-level chain-of-thought reasoning, indicating that millions of adapter weights are unnecessary. The minimal trainable footprint reduces optimizer memory and inter-GPU communication, lowering the overall cost of fine-tuning. Moreover, a logit-lens analysis shows that the learned vectors amplify coherent token directions, providing clearer insight into the model's internal computations.

1 Introduction

Reasoning models have recently made striking gains by learning to produce a chain-of-thought before the final answer Jaech et al. (2024); Guo et al. (2025). Much of this progress comes from reinforcement learning with verifiable rewards in mathematical domains, a now-standard setup for training reasoning models (Zeng et al., 2025; Hu et al., 2025; Venhoff et al., 2025). However, training large models is costly, and, because of the amount of parameters and complex internal computations, the mechanisms induced by reasoning training remain poorly understood.

In this work, we show that training just 0.0016% of parameters suffices to match the performance of a fully RL-tuned model. We train per-layer steering vectors that are added to each layer's output while keeping all base weights fixed (see Section A for visualization). Compared with LoRA (Hu et al., 2022) and BitFit (Zaken et al., 2021), this approach **(i)** requires orders of magnitude fewer resources and **(ii)** isolates a much smaller, more interpretable parameter set, making it easier to see what changes during reasoning training.

We present a preliminary study showing that these vectors amplify meaningful directions in representation space, aligning with interpretable token clusters such as causality ("Because", "However"), validation ("correctness", "necessity", "confirmation"), and programming-language tokens.

Taken together, these results provide a simple, resource-efficient training setup that both reduces the cost of adapting large models and simplifies the study of how reasoning training modifies pretrained models.

2 Related Work

The use of steering vectors, a technique within activation engineering, provides a direct way to probe and manipulate model behavior with minimal changes to the underlying weights. Traditionally, such vectors are constructed from activation differences on contrastive prompts (e.g., positive vs. negative sentiment) and are typically interpreted as feature amplifiers rather than creators of novel behaviors (Turner et al., 2023; Panickssery et al., 2023). They have already been used to identify and control "reasoning" behaviours (Venhoff et al., 2025; Ward et al., 2025). Other work demonstrates that these vectors can also be trained, not merely computed, allowing for more targeted control. For example, Cao et al. (2024) optimized steering directions using preference data, while Mack and Turner (2024) and Engels et al. (2025) (building on Betley et al. (2025)) showed that training simple additive vectors in an unsupervised manner can elicit complex latent behaviors, such as reasoning and self-awareness. We apply these ideas on a scale of a real GRPO-like training, and show that their simlicity and interpretability benefits do not stand

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 9202–9211
November 4-9, 2025

Model	Setup	AIME25	AIME24	AMC23	MATH500	MinervaMath	OlympiadBench	Avg.
Qwen2.5-1.5B	Base	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.9 ± 0.0	0.7 ± 0.0	0.3 ± 0.0	0.3 ± 0.0
Qwen2.5-1.5B	Steering	1.4 ± 0.0	2.8 ± 0.1	34.1 ± 0.1	57.5 ± 1.5	20.3 ± 1.8	23.3 ± 0.6	23.2 ± 0.3
Qwen2.5-1.5B	Full-Tune	1.2 ± 0.0	4.1 ± 0.2	29.9 ± 0.2	58.4 ± 0.7	20.6 ± 1.5	22.2 ± 1.1	22.7 ± 0.4
Qwen2.5-1.5B	SimpleRL-Zoo	—	4.2 ± 0.0	—	59.0 ± 0.0	20.2 ± 0.0	21.0 ± 0.0	—
Qwen2.5-1.5B	Open-Reasoner	—	—	—	58.0 ± 0.0	—	—	—
Qwen2.5-7B	Base	6.7 ± 0.0	13.3 ± 0.0	39.2 ± 0.0	45.9 ± 0.0	10.2 ± 0.0	29.9 ± 0.0	24.2 ± 0.0
Qwen2.5-7B	Steering	7.0 ± 0.3	12.8 ± 0.0	50.5 ± 0.1	74.7 ± 1.0	36.6 ± 1.1	36.6 ± 1.2	36.4 ± 0.3
Qwen2.5-7B	Full-Tune	6.6 ± 0.2	13.6 ± 0.2	53.8 ± 0.1	77.3 ± 0.7	34.6 ± 2.3	37.2 ± 0.5	37.1 ± 0.5
Qwen2.5-7B	Open-Reasoner	—	—	—	81.4 ± 0.0	—	—	—
Qwen2.5-7B	Open-Reasoner (from Oat-Zero)	—	—	—	82.2 ± 0.0	31.6 ± 0.0	47.9 ± 0.0	—
Qwen2.5-7B	SimpleRL-Zero	—	15.6 ± 0.0	—	78.2 ± 0.0	38.6 ± 0.0	40.4 ± 0.0	—
Qwen2.5-7B	R1-Distill	—	—	—	88.1 ± 0.0	35.9 ± 0.0	47.7 ± 0.0	—
Qwen2.5-14B	Base	3.3 ± 0.0	6.7 ± 0.0	37.5 ± 0.0	61.7 ± 0.0	20.3 ± 0.0	26.8 ± 0.0	26.1 ± 0.0
Qwen2.5-14B	Steering	15.4 ± 0.3	15.9 ± 0.4	59.8 ± 0.4	80.0 ± 0.2	39.2 ± 1.7	43.5 ± 1.1	42.3 ± 0.2
Qwen2.5-14B	Full-Tune	12.1 ± 0.6	15.9 ± 0.4	59.2 ± 0.2	80.5 ± 0.1	38.6 ± 0.3	41.3 ± 1.4	41.3 ± 0.1
Qwen2.5-Math-1.5B	Base	3.3 ± 0.0	13.3 ± 0.0	27.5 ± 0.0	32.2 ± 0.0	9.4 ± 0.0	22.9 ± 0.0	18.1 ± 0.0
Qwen2.5-Math-1.5B	Steering	8.1 ± 0.1	12.1 ± 0.0	48.8 ± 0.1	70.8 ± 0.8	27.5 ± 0.7	36.1 ± 0.5	33.9 ± 0.3
Qwen2.5-Math-1.5B	Full-Tune	8.4 ± 0.1	11.9 ± 0.1	49.1 ± 0.2	70.1 ± 0.2	29.4 ± 0.9	32.8 ± 0.5	33.6 ± 0.1
Qwen2.5-Math-1.5B	Oat-Zero	—	—	—	74.2 ± 0.0	25.7 ± 0.0	37.6 ± 0.0	—
Qwen2.5-Math-7B	Base	3.3 ± 0.0	16.7 ± 0.0	45.8 ± 0.0	52.2 ± 0.0	12.3 ± 0.0	18.6 ± 0.0	24.8 ± 0.0
Qwen2.5-Math-7B	Steering	12.6 ± 0.2	24.4 ± 0.2	62.5 ± 0.2	79.9 ± 1.2	36.0 ± 2.8	44.1 ± 0.8	43.3 ± 0.3
Qwen2.5-Math-7B	Full-Tune	14.0 ± 0.1	25.7 ± 0.1	64.2 ± 0.1	79.3 ± 0.6	36.9 ± 2.0	41.1 ± 0.9	43.5 ± 0.4
Qwen2.5-Math-7B	Oat-Zero	—	—	—	80.0 ± 0.0	30.1 ± 0.0	41.0 ± 0.0	—
Qwen2.5-Math-7B	SimpleRL-Zero	—	24.0 ± 0.0	—	80.2 ± 0.0	37.5 ± 0.0	39.0 ± 0.0	—
LLaMa3.1-8B-It	Base	0.0 ± 0.0	10.0 ± 0.0	28.3 ± 0.0	52.3 ± 0.0	21.1 ± 0.0	18.4 ± 0.0	21.7 ± 0.0
LLaMa3.1-8B-It	Steering	1.5 ± 0.2	10.3 ± 0.2	32.4 ± 0.7	58.7 ± 0.8	29.3 ± 1.5	24.6 ± 1.3	26.1 ± 0.6
LLaMa3.1-8B-It	Full-Tune	1.6 ± 0.2	10.8 ± 0.2	35.3 ± 0.4	58.0 ± 0.7	29.4 ± 0.6	21.8 ± 0.8	26.1 ± 0.1
LLaMa3.1-8B	Base	0.0 ± 0.0	0.0 ± 0.0	5.0 ± 0.0	11.2 ± 0.0	5.1 ± 0.0	2.0 ± 0.0	3.9 ± 0.0
LLaMa3.1-8B	Steering	0.3 ± 0.0	0.3 ± 0.1	8.9 ± 0.1	25.7 ± 0.9	12.4 ± 0.3	7.1 ± 0.4	9.1 ± 0.1
LLaMa3.1-8B	Full-Tune	0.5 ± 0.1	1.6 ± 0.0	9.0 ± 0.2	29.8 ± 1.6	18.4 ± 2.1	9.9 ± 0.8	11.5 ± 0.4

Table 1: Accuracies on six mathematical-reasoning benchmarks for three variants of each model: the Base model (no training), a Fully-Tuned model, and our Steering model that trains only per-layer steering vectors while freezing all other weights. For reference we also include numbers reported by SimpleRL-Zoo, Open-Reasoner, Oat-Zero, and R1-Distill. Across models and datasets, Steering matches the performance of Fully-Tuned models.

in constrast to quality of the trained models.

We implement steering vectors only by activating only layer-wise additive biases in MLPs while keeping all other model parameters frozen. This approach is aligned with BitFit (Zaken et al., 2021), which tunes only bias terms and has been shown to effectively expose existing knowledge, often matching the performance of full fine-tuning on language tasks. Notably, BitFit and similar minimal-adaptation methods sometimes underperform on tasks requiring substantial generalization (Hu et al., 2022); it remains unclear whether they suffice for complex reasoning. And still, differently from BitFit, we tune only a subset of model biases, tightening the training landscape even more. These minimal interventions stand in contrast to parameter-efficient finetuning methods such as prompt tuning and LoRA (Hu et al., 2022; Li et al., 2025), or full RL-based adaptation (Guo et al., 2025), which actively adjust model parameters.

Our work is partly motivated by the recent results on reasoning training only amplifying the reasoning behaviors already present in the base models (Wang et al., 2025; Ye et al., 2025; Shao et al., 2025; Liu et al., 2025a). If such a minimal intervention is able to recover reasoning performance, this adds as an evidence to such a claim. Our aim is to take the study further by identifying what specific parts of the model are holding these behaviours.

3 Methodology

3.1 Online Training

We adopt an online reinforcement learning procedure loosely modeled on DeepSeek-R1 (Ahmadian et al., 2024; Guo et al., 2025). For each prompt x, we sample N candidate solutions $y_1, \ldots, y_N$ from the current policy π_θ. Each rollout y_i receives a binary reward $r(x, y_i)$ based on the presence of a correct answer enclosed in a `\boxed{...}` template. The model is trained with RLOO (Ahmadian et al., 2024) objective. Other details are in Section C.

Layer idx	Top-Cluster	Representative tokens	Unifying idea
2	**Source-code & test-harness vocabulary**	`tostring`, `ComponentFixture`, `.SQL`, `standalone`, `-independent`, `_fault`, `203`, `@a`, `\Context`	These are the words you meet in programming projects - Angular's ComponentFixture, SQL file extensions, "fault" flags, HTTP status 203, context objects, and helper functions like toString().
2	**Named entities (people & places)**	`Antonio`, `Pelosi`, `Baldwin`, `Cumberland`, `Switzerland`, `Peg`, `Salv-`	Proper names of individuals and locations that commonly co-occur in news articles or knowledge-graph dumps.
17	**Accuracy, validation & logical necessity**	`correctness`, `correct`, `precision`, `necessity`, `possibility`, `confirmation`, `answer`, `goal`, `directly`, `derive / deriving`	These words belong to discourse about getting things right - arguments, proofs, validations, QA reports, or formal specifications.
30	**Causal & contrastive connectors**	`Because / because / 因为`, `Therefore / donc`, `However / о д н а к о / jedoch`, `Given / Here`, `step`	Words that introduce reasons, consequences, or contrasts - typical of argumentative writing, technical explanations, or test-case descriptions.

Table 2: Clusters of tokens most aligned with the learned steering vectors, as measured by cosine similarity.

3.2 Steering Vector

We insert a learnable **steering vector** $s_\ell \in \mathbb{R}^d$ at the end of every transformer layer ℓ (there are L layers in total). The vector is added directly to the residual stream, so its dimensionality matches the model's hidden size d. All original weights remain frozen; only these L steering vectors are trained. Section A has a visualisation and Section B contains our code implementation for clarity.

3.3 Training and Evaluation Setup

We experiment across multiple model families and sizes: `Qwen2.5-{1.5,7,14}B` (Team, 2024), `Qwen2.5-Math-{1.5,7}B` (Yang et al., 2024), `Llama3.1-8B`, and `Llama3.1-8B-Instruct` (Grattafiori et al., 2024). Training uses the DeepScaleR dataset (Luo et al., 2025) with sampling temperature $\tau = 1$, a 4K context for `Qwen2.5-Math-{1.5,7}B`, and 8K for the other models, 128 prompts per step.

We report results on six math benchmarks: AIME24/25, AMC23, MATH500 (Hendrycks et al., 2021a), MinervaMath (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024). For MATH500, MinervaMath, and OlympiadBench we report PASS@1; for AIME24/25 and AMC23 we report AVG@32 due to their smaller size. Base models decode greedily, whereas trained models use sampling with $\tau = 1.0$ following Zeng et al. (2025). Evaluation context length is 4K and 32K for Qwen2.5-Math-7B and other models respectively. All metrics are averaged over three evaluation seeds. Unless noted otherwise (e.g., Table 1), figures and tables show the mean score across the six benchmarks. When available, we include numbers from SimpleRL-Zoo (Zeng et al., 2025), Oat-Zero (Liu et al., 2025b), Open-Reasoner (Hu et al., 2025), and R1-Distill (Guo et al., 2025) for context. Implementations use `transformers` (Wolf et al., 2019), the `vllm` inference engine (Kwon et al., 2023), and `Math-Verify`[1] for reward assignment.

4 Results

4.1 Steering Vectors are Effective for Inducing Reasoning Capabilities

Table 1 shows that steering vectors match the accuracy of fully tuned models across families and scales. The only exception is LLaMa3.1-8B, where steering recovers about 70% of the full-tuning gain.

[1] https://github.com/huggingface/Math-Verify

Metric	Full-Tune	Steering
Number of Parameters	14.7 B	245 K
Optimizer Memory	13.8 GB	240 KB
Per-step Time	9.94 s	0.11 s
Overall Time	52 m	34 s

Table 3: Resource cost for Qwen2.5-14B: full fine-tuning vs. steering. Overall Time is across 314 steps ≈ 1 epoch.

See Section E for results when training on the GSM8K and MATH datasets.

Because only a fraction of parameters is optimized, the approach yields substantial savings (Table 3): the optimizer state shrinks to kilobytes and the parameter broadcast[2] time drops by nearly an hour per training epoch. Measurements for Qwen2.5-7B and LLaMa3.1-8B-It are detailed in Section G.

4.2 Interpretation

To understand what the learned steering vectors are doing inside the network, we apply the logit-lens technique (nostalgebraist, 2020). The key idea is to "peek" into a residual stream after a specific transformer layer by converting it into a full vocabulary distribution and then reading the most likely tokens.

Let the row $u_v \in \mathbb{R}^d$ of W_U correspond to token v. For every token we compute the cosine similarity

$$c_l(v) = \frac{\langle s_l,\ u_v \rangle}{\|s_l\|\,\|u_v\|} \in [-1, 1].$$

A large positive $c_l(v)$ means the steering vector pushes the hidden state toward token v; a large negative value indicates suppression.

We collect top-50 tokens for each steering vector and ask GPT-o3 to translate all non-english tokens and group the subsets of tokens into explainable topics (see the prompt in Appendix F).

Table 2 shows the representative token groups from different layers of `LLaMa3.1-8B-It` model trained on `GSM8K` dataset. At layer 2, the steering vector aligns with programming-style terms rather than math tokens, which is surprising given the math-oriented task. While not being from the math domain, this use of coding tokens suggests the model leverages structural parallels between programming and formal math notation. It also picks up named entities because many `GSM8K` problems use character names and places to set up word problems.

At layer 17, the vector shifts to words about checking steps and validating results. This suggests the model uses the middle stage to verify each reasoning step before proceeding.

At layer 30, it focuses on linking words such as "because", "therefore", and "however". This indicates the final stage ties statements together to guide the answer's flow.

Overall, the learned steering vectors appear to be highly interpretable and relevant to the reasoning task on the `GSM8K` domain.

5 Conclusion

In this paper, we have demonstrated that training lightweight steering vectors alone can recover the reasoning performance of fully-tuned models on standard mathematical benchmarks. This result carries two important implications. First, steering vector training offers a highly parameter-efficient and cost-effective alternative: only a small set of layer-wise bias terms must be learned, drastically reducing storage and communication time requirements. Second, it isolates a small, interpretable set of parameters that capture the effects of reasoning training, simplifying the study of the mechanisms of models' reasoning abilities.

Limitations

First, our experiments cover a narrow slice of online-training settings. Broader sweeps – across settings, tasks, and model sizes – would test generality and may reveal cases where steering vectors fall short of full fine-tuning.

Second, while the logit-lens provides a convenient way to inspect how steering vectors influence token logits at each layer, it does not capture the downstream transformations applied by subsequent layers. As a result, later computations may modify the initial steering signal, leading to interpretations of logit-lens itself that conflict with layer-wise observations. Applying more comprehensive interpretation techniques, such as probing classifiers, causal interventions, or circuit-level analysis, could yield deeper insights into how steering vectors shape the model's behavior.

[2]We broadcast trainable model parameters to `vllm` inference engine after each parameter update.

Acknowledgment

Viacheslav dedicates his contribution in this paper to his girlfriend, Marina.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*.

Jan Betley, Xuchan Bao, Martín Soto, Anna Sztyber-Betley, James Chua, and Owain Evans. 2025. Tell me about yourself: Llms are aware of their learned behaviors. *arXiv preprint arXiv:2501.11120*.

Yuanpu Cao, Tianrong Zhang, Bochuan Cao, Ziyi Yin, Lu Lin, Fenglong Ma, and Jinghui Chen. 2024. Personalized steering of large language models: Versatile steering vectors through bi-directional preference optimization. *Advances in Neural Information Processing Systems*, 37:49519–49551.

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 preprint arXiv:2110.14168*.

Josh Engels, Neel Nanda, and Senthooran Rajamanoharan. 2025. Interim research report: Mechanisms of awareness. *AI Alignment Forum*. https://www.alignmentforum.org/posts/m8WKfNxp9eDLRkCk9/interim-research-report-mechanisms-of-awareness.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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

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

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

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

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

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. 2025. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*.

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 preprint arXiv:2412.16720*.

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

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857.

Dacheng Li, Shiyi Cao, Tyler Griggs, Shu Liu, Xiangxi Mo, Eric Tang, Sumanth Hegde, Kourosh Hakhamaneshi, Shishir G Patil, Matei Zaharia, and 1 others. 2025. Llms can easily learn to reason from demonstrations structure, not content, is what matters! *arXiv preprint arXiv:2502.07374*.

Zichen Liu, Changyu Chen, Wenjun Li, Tianyu Pang, Chao Du, and Min Lin. 2025a. There may not be aha moment in r1-zero-like training — a pilot study. https://oatllm.notion.site/oat-zero. Notion Blog.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025b. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty

-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2. Notion Blog.

Andrew Mack and Alex Turner. 2024. Mechanistically eliciting latent behaviors in language models. *AI Alignment Forum*. https://www.alignmentforum.org/posts/ioPnHKFyy4Cw2Gr2x/mechanistically-eliciting-latent-behaviors-in-language-1.

nostalgebraist. 2020. interpreting gpt: the logit lens. https://www.alignmentforum.org/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

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*.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, and 1 others. 2025. Spurious rewards: Rethinking training signals in rlvr. *arXiv preprint arXiv:2506.10947*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

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 preprint arXiv:2308.10248*.

Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. 2025. Understanding reasoning in thinking language models via steering vectors. *arXiv preprint arXiv:2506.18167*.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, and 1 others. 2025. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*.

Jake Ward, Chuqiao Lin, Constantin Venhoff, and Neel Nanda. 2025. Reasoning-finetuning repurposes latent representations in base models. *arXiv preprint arXiv:2507.12638*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and 1 others. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

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

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*.

A Steering Vector Visualization

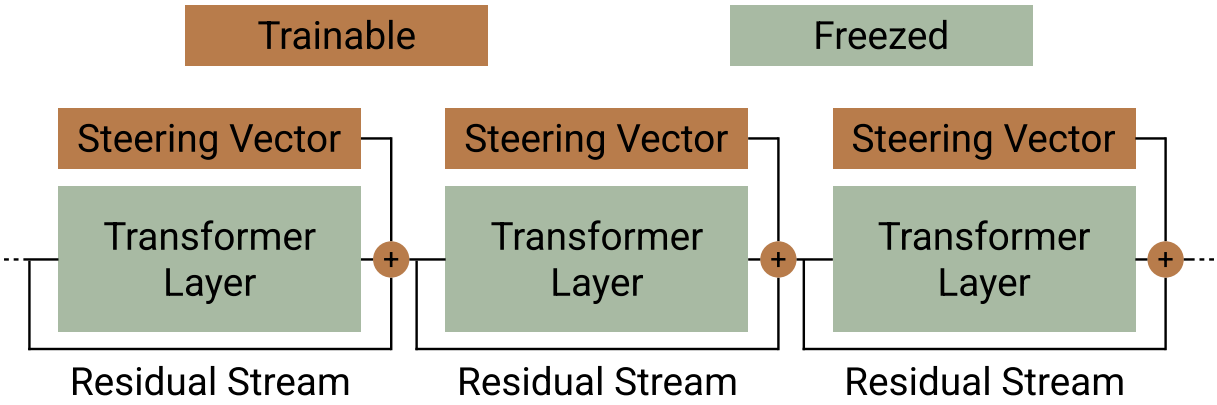

Figure 1: **Layer-wise trainable steering vectors.** All base transformer weights are frozen (blue). The only trainable parameters are one d-dimensional vector v_ℓ per layer (orange), added to the residual stream at every token position: $h_{\ell,t} \leftarrow h_{\ell,t} + v_\ell$.

B Steering Vector Implementation

Steering Vector Implementation

```python
class SteeringVector(nn.Module):
    def __init__(self, hidden_size: int):
        super().__init__()

        self.hidden_size = hidden_size

        self.steering_vector = nn.Parameter(
            torch.zeros(self.hidden_size).unsqueeze(0).unsqueeze(0)
        )

    def forward(self, x):
        return x + self.steering_vector

class TransformersQwen2DecoderLayerWithSteering(TransformersQwen2DecoderLayer):
    def __init__(self, config: Qwen2Config, layer_idx: int):
        super().__init__(config=config, layer_idx=layer_idx)

        self.steering_vector = SteeringVector(hidden_size=config.hidden_size)

        self.layer_idx = layer_idx

    def forward(self, *args, **kwargs):
        hidden_states, *rest = super().forward(*args, **kwargs)

        hidden_states = self.steering_vector(hidden_states)

        return (hidden_states, *rest)
```

C Training Objective

To reduce variance, we compute a baseline b as the mean reward for all rollouts associated with x:

$$b = \frac{1}{N}\sum_{i=1}^{N} r_i, \qquad a_i = r_i - b.$$

The parameters are updated via a policy-gradient step:

$$\nabla_\theta J = \mathbb{E}_{x\sim D, y\sim\pi_\theta(x)}\big[a(x,y)\,\nabla_\theta \log \pi_\theta(y \mid x)\big].$$

D LoRA

A limitation of steering vectors is that the same vector is added to every token position. We hypothesize that this token-independence may cap performance and partly account for the gap between the full-model baseline and steering-only training.

To let the offset be token-specific, we replace the fixed steering vector with a low-rank adaptor (LoRA) (Hu et al., 2022) applied to the MLP down-projection in every transformer layer:

$$\text{(fixed steering)} \quad h' = h + s,$$
$$\text{(LoRA steering)} \quad h' = h + B \cdot A \cdot h_{\text{MLP}},$$

where $h \in \mathbb{R}^d$ is the residual stream, $s \in \mathbb{R}^d$ is a learned constant, $h_{\text{MLP}} \in \mathbb{R}^{d_{\text{MLP}}}$ is the intermediate representation of MLP layer, and A and B are LoRA rank-r matrices which are the only trainable components in this setup.

All experiments use LoRA rank $r = 4$, scaling factor $\alpha = 4$, and no dropout following Engels et al. (2025).

E Steering Vectors on GSM8K and MATH

Experimental Setup. We conduct experiments on four pretrained transformer checkpoints: `Qwen-2.5-1.5B` (Team, 2024), `Qwen-2.5-Math-1.5B` (Yang et al., 2024), `Llama-3.1-8B`, and `Llama-3.1-8B-Instruct` (Grattafiori et al., 2024). For each model, we evaluate three training regimes: (i) full fine-tuning, (ii) training only steering vectors, and (iii) training only LoRA (Hu et al., 2022) adapters which may be viewed as adaptive steering vectors (Appendix D, Mack and Turner (2024)). For LoRAs we use rank 4. In the latter two cases, all other parameters are kept frozen.

We use two mathematical datasets for training and evaluation. The `GSM8K` training split contains 8,790 problems (Cobbe et al., 2021); for evaluation, we randomly subsample 500 items from its original split to shorten iteration time. The `MATH` corpus (Hendrycks et al., 2021b) provides 12,000 training examples, and its evaluation split consists of 500 items, which we use as is. We report mean@8 $= \mathbb{E}_x[\mathbb{E}_{i=0}^{8}\, r(x,y)]$ win rates on each dataset.

All experiments are implemented using the `transformers` library (Wolf et al., 2019) and the `vllm` inference engine (Kwon et al., 2023). Additional hyperparameter details are provided in Appendix I.

Result. Table 4 summarizes accuracy for the pretrained model, fully-tuned models, steering vectors, and LoRA adapters. As expected, RL training yields large gains on mathematical benchmarks on all setups. Steering vectors achieve similar improvements across nearly all model-dataset pairs and even exceed full fine-tuning in some cases (e.g., `Qwen2.5-1.5B` when evaluated on `MATH-500` and `LLaMa3.1-8B` when trained on `GSM8K` and evaluated on `MATH-500`, both being base models that did not undergo instruction tuning), which we attribute to the implicit regularization of updating far fewer parameters.

If we accept the working assumption that a steering vector can only amplify features that the original network already contains and cannot create new ones, the table gives direct evidence for that view: when the base model "knows" how to solve the task, steering is usually enough to reach the same quality as full fine-tuning.

There are, however, a few setups where the steering training stays noticeably below the full training - for example, the `Qwen2.5-Math-1.5B` and `LLaMa3.1-8B` evaluated on `GSM8K`. In most cases the LoRA closes the gap completely. Because LoRA modifies a small, learned set of rank-decomposed weight matrices

Train / Test	Setup	Qwen2.5-1.5B	Qwen2.5-Math-1.5B	Llama3.1-8B	Llama3.1-8B-It
GSM8K / GSM8K	Base	0.63	29.26	1.08	66.03
	Full-Tune	78.91	86.49	76.49	87.22
	Steering	73.84 (-5.07)	79.89 (-6.60)	70.36 (-6.13)	87.22 (0.00)
	LoRA	76.49 (-2.42)	85.41 (-1.08)	74.24 (-2.25)	85.41 (-1.81)
GSM8K / MATH	Base	1.51	28.73	0.76	32.51
	Full-Tune	40.78	65.95	16.86	46.65
	Steering	48.69 (+7.91)	61.79 (-4.16)	22.81 (+5.95)	48.51 (+1.86)
	LoRA	49.32 (+8.54)	64.31 (-1.64)	19.28 (+2.42)	49.04 (+2.39)
MATH / MATH	Base	1.51	28.73	0.76	32.51
	Full-Tune	44.48	70.44	27.39	52.39
	Steering	51.39 (+6.91)	65.27 (-5.17)	22.05 (-5.34)	50.81 (-1.58)
	LoRA	53.68 (+9.20)	69.35 (-1.09)	24.32 (-3.07)	50.40 (-1.99)
MATH / GSM8K	Base	0.63	29.26	1.08	66.03
	Full-Tune	68.57	82.56	52.12	85.03
	Steering	69.56 (+0.99)	76.41 (-6.15)	45.24 (-6.88)	84.02 (-1.01)
	LoRA	72.53 (+3.96)	81.88 (-0.68)	52.52 (+0.40)	85.06 (+0.03)

Table 4: mean@8 accuracy for each combination of training dataset, evaluation dataset, model, and tuning setup. Rows are grouped by **Train / Test** dataset pairs, and each column corresponds to a specific model variant. For *Steering* and *LoRA* rows, the colored value in parentheses indicates the difference compared to *Full-Tune* for that model - green if better, red if worse. This highlights how close lightweight methods can get to full fine-tuning performance, and where gaps remain.

rather than a single global vector, it provides finer control over what is added to the residual stream. The fact that LoRA always bridges the remaining gap shows that a more targeted, low-rank adjustment is the reason why single steering vector cannot reach the performance of a fully-trained model.

F Logit Lens. GPT Prompt

GPT Prompt for Token Clustering

```
You will be given a list of tokens together with a score.
You should translate all non-english tokens and suggest the main topics
that unite the biggest subsets of tokens in the list.

<list>
```

G Training Cost Savings

(a) Number of Parameters

	Qwen2.5-7B	Llama3.1-8B
Full-Tune	7.6 B	8 B
Steering	100 K	131 K

(b) Optimizer Memory

	Qwen2.5-7B	Llama3.1-8B
Full-Tune	7.1 GB	7.5 GB
Steering	98 KB	128 KB

(c) Per-step Time

	Qwen2.5-7B	Llama3.1-8B
Full-Tune	5.30 s	5.32 s
Steering	0.06 s	0.07 s

(d) Overall Time (314 steps $\approx$ 1 epoch)

	Qwen2.5-7B	Llama3.1-8B
Full-Tune	27.7 m	27.8 m
Steering	0.314 m	0.36 m

Table 5: Resource-efficiency comparison of full fine-tuning versus steering across three model sizes.

H Computational Resources

All models were trained on 16 H100 GPUs. `Qwen2.5-1.5B` models were trained for approx. 9 hours, `Qwen2.5-Math-1.5B` for approx. 2.5 hours, `LLaMa3.1-8B-It` for approx. 9 hours, `LLaMa3.1-8B-It` for approx. 120 hours.

I Hyperparameters

			lr	num_generations
Qwen-2.5-1.5B	GSM8K	Full-Tune	2×10^{-5}	64
		Steering	5×10^{-4}	64
		LoRA-1	5×10^{-4}	64
		LoRA-4	5×10^{-4}	64
	MATH	Full-Tune	2×10^{-5}	64
		Steering	5×10^{-4}	64
		LoRA-1	5×10^{-4}	64
		LoRA-4	5×10^{-4}	64
Qwen-2.5-Math-1.5B	GSM8K	Full-Tune	2×10^{-5}	16
		Steering	1×10^{-3}	16
		LoRA-1	5×10^{-4}	16
		LoRA-4	5×10^{-4}	16
	MATH	Full-Tune	2×10^{-5}	16
		Steering	1×10^{-3}	16
		LoRA-1	5×10^{-4}	16
		LoRA-4	5×10^{-4}	16
Llama-3.1-8B	GSM8K	Full-Tune	5×10^{-6}	64
		Steering	5×10^{-4}	64
		LoRA-1	1×10^{-4}	64
		LoRA-4	1×10^{-4}	64
	MATH	Full-Tune	5×10^{-6}	64
		Steering	5×10^{-4}	64
		LoRA-1	1×10^{-4}	64
		LoRA-4	1×10^{-4}	64
Llama-3.1-8B-Instruct	GSM8K	Full-Tune	1×10^{-6}	16
		Steering	2×10^{-4}	16
		LoRA-1	6×10^{-4}	16
		LoRA-4	1×10^{-4}	16
	MATH	Full-Tune	1×10^{-6}	16
		Steering	2×10^{-4}	16
		LoRA-1	3×10^{-4}	16
		LoRA-4	3×10^{-4}	16

Table 6: Hyperparameter settings for each model and training setup.