Steering LLM Reasoning Through Bias-Only Adaptation

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

Abstract

Recent work on reasoning-oriented language models, exemplified by *o1-like* systems, suggests that reinforcement-learning (RL) finetuning does not create new capabilities but instead strengthens reasoning patterns already latent in the pretrained network. We test this claim by training *steering vectors*: layer-wise biases that additively amplify selected hidden features while leaving all original weights unchanged. Experiments on four base models across the GSM8K and MATH benchmarks show that steering vectors recover, and in several cases exceed, the accuracy of fully-tuned counterparts. This result supports the view that the required reasoning skills pre-exist in the base model. Further, logit-lens analysis reveals that the trained vectors consistently boost token groups linked to structured languages and logical connectors, providing an interpretable account that aligns with the demands of quantitative reasoning tasks.

1. Introduction

Large language models (LLMs) such as GPT-4 (Achiam et al., 2023), Qwen (Yang et al., 2025), and Llama (AI@Meta, 2024) demonstrate that scaling up data, model size, and compute can create a general-purpose engine for language understanding and generation. Trained on trillions of tokens from books, code, scientific articles, and social media, these models can summarize long documents, write executable code, solve domain-specific exams, and generate persuasive prose - all without task-specific tuning. Their success has shifted the research focus from *whether* large-scale pretraining works to *how* to elicit the rich behaviours embedded within, ranging from chain-of-thought reasoning to tool use and self-correction.

[1]T-Tech. Correspondence to: Viacheslav Sinii <v.siniy@tbank.ru>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

Recent systems such as *openai-o1* and *R1* (Guo et al., 2025) fine-tune these base models with reinforcement learning (RL) to create ”reasoning” models. The authors of these systems argue that RL training imparts new abilities (such as self-reflection and sudden ”aha” moments) that purportedly emerge only after RL.

However, growing evidence (Liu et al., 2025; Shah et al., 2025) points to the contrary. The studies indicate that the necessary skills are already present in the pretrained network, with RL serving primarily to amplify, not invent, them. We directly test this claim by training *steering vectors*: simple, trainable vectors that are added to the model's hidden state at each layer, leaving all original weights untouched. If reasoning ability is already latent in the model, steering in the right direction should suffice to unlock it (see Section 2 for further details).

We train steering vectors on four base models (`Qwen-2.5-1.5B`, `Qwen-2.5-Math-1.5B`, `Llama-3.1-8B`, and `Llama-3.1-8B-Instruct`) using two mathematical reasoning datasets (`GSM8K` and `MATH`) (Cobbe et al., 2021; Hendrycks et al., 2021). Across these eight model-task pairs, we find that steering alone often matches the accuracy of fully RL-tuned models, and in all cases performs competitively. These findings support the hypothesis that LLMs already possess the knowledge necessary for step-by-step reasoning, which can be elicited via targeted amplification.

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). More recent 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 & Turner (2024) and Engels et al. (2025) (building on Betley et al. (2025)) showed

Table 1. 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.

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)

that training simple additive vectors can elicit complex latent behaviors, such as reasoning and self-awareness.

Our method follows this trend, training only layer-wise additive biases while keeping all other model parameters frozen. This approach is philosophically 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.

These minimal interventions stand in contrast to parameter-efficient finetuning methods such as prompt tuning and LoRA (Hu et al., 2022), or full RL-based adaptation (Guo et al., 2025), which actively adjust model parameters. In our work, we focus on whether steering vectors alone can unlock reasoning capabilities already present in LLMs, without the need for extensive weight updates.

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 $y_1, \dots, y_N$ 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. Other details are in Appendix B.

Depending on the experiment, either all model weights (full fine-tuning), only the steering vectors, or only the LoRA adapters are trainable. In the latter two cases, all other weights remain frozen. Updates are applied online after each batch of rollouts.

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. Appendix A contains our code implementation for clarity.

3.3. Training and Evaluation 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

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

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 / однако/ jedoch`, `Given / Here`, `step`	Words that introduce reasons, consequences, or contrasts - typical of argumentative writing, technical explanations, or test-case descriptions.

vectors (Appendix C, Mack & 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., 2021) 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 D.

4. Results

4.1. Steering Vectors are Effective for Inducing Reasoning Capabilities

Table 1 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 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.

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 E).

Table 2 shows the representative token groups from different layers of `llama3.1-8b-instruct` 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 RL-fine-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 both compute and storage requirements. Second, our findings support the view that reinforcement-learning fine-tuning in large language models does not create fundamentally new abilities but rather amplifies reasoning skills already latent in the pretrained network.

6. Limitations

First, our experiments cover only a narrow range of online-training hyperparameters. A more extensive sweep varying learning rates, batch sizes, rollout counts, and baseline estimators would strengthen the generality of our findings and might reveal scenarios in which steering vectors fail to match full-model fine-tuning. This also applies to a wider range of tasks and model sizes as different tasks may require a certain model capacity not available to steering vectors.

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.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., and Hooker, S. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

AI@Meta. Llama 3 model card. 2024. URL `https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md`.

Betley, J., Bao, X., Soto, M., Sztyber-Betley, A., Chua, J., and Evans, O. Tell me about yourself: Llms are aware of their learned behaviors. *arXiv preprint arXiv:2501.11120*, 2025.

Cao, Y., Zhang, T., Cao, B., Yin, Z., Lin, L., Ma, F., and Chen, J. Personalized steering of large language models: Versatile steering vectors through bi-directional preference optimization. *Advances in Neural Information Processing Systems*, 37:49519–49551, 2024.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Engels, J., Nanda, N., and Rajamanoharan, S. Interim research report: Mechanisms of awareness. *AI Alignment*

Forum, 2025. https://www.alignmentforum.org/posts/m8WKfNxp9eDLRkCk9/interim-research-report-mechanisms-of-awareness.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Liu, Z., Chen, C., Li, W., Pang, T., Du, C., and Lin, M. There may not be aha moment in r1-zero-like training — a pilot study. https://oatllm.notion.site/oat-zero, 2025. Notion Blog.

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

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

Panickssery, N., Gabrieli, N., Schulz, J., Tong, M., Hubinger, E., and Turner, A. M. Steering llama 2 via contrastive activation addition. *arXiv preprint arXiv:2312.06681*, 2023.

Shah, D. J., Rushton, P., Singla, S., Parmar, M., Smith, K., Vanjani, Y., Vaswani, A., Chaluvaraju, A., Hojel, A., Ma, A., et al. Rethinking reflection in pre-training. *arXiv preprint arXiv:2504.04022*, 2025.

Team, Q. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Turner, A. M., Thiergart, L., Leech, G., Udell, D., Vazquez, J. J., Mini, U., and MacDiarmid, M. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., Lu, K., Xue, M., Lin, R., Liu, T., Ren, X., and Zhang, Z. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zaken, E. B., Ravfogel, S., and Goldberg, Y. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021.

A. 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)

```

B. 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:

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

C. 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:

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

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).

D. Hyperparameters

Table 3. Hyperparameter settings for each model and training setup.

			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

For all the setups we use temperature $\tau = 1.0$ and a global batch size 128. Setup-specific hyperparameters are listed in Table 4.

Table 4. Hyperparameter settings for each model and training setup.

Model	**Setup**	lr	num_generations
Qwen-2.5-1.5B	Full-Model	2×10^{-5}	64
	Steering vectors	1×10^{-4}	64
	LoRA adapters	1×10^{-4}	64
Qwen-2.5-Math-1.5B	Full-Model	2×10^{-5}	16
	Steering vectors	5×10^{-4}	16
	LoRA adapters	5×10^{-4}	16
Llama-3.1-8B	Full-Model	9×10^{-6}	128
	Steering vectors	5×10^{-4}	128
	LoRA adapters	5×10^{-4}	128
Llama-3.1-8B-Instruct	Full-Model	5×10^{-7}	16
	Steering vectors	5×10^{-4}	16
	LoRA adapters	1×10^{-4}	16

E. 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>
```

F. 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-instruct` for approx. 9 hours, `llama3.1-8b-instruct` for approx. 120 hours.