Simple Mechanistic Explanations for Out-Of-Context Reasoning

Atticus Wang [*1] **Joshua Engels** [*1] **Oliver Clive-Griffin** [*2] **Senthooran Rajamanoharan** **Neel Nanda**

Abstract

Out-of-context reasoning (OOCR) is a phenomenon in which fine-tuned LLMs exhibit surprisingly deep out-of-distribution generalization. Rather than learning shallow heuristics, they implicitly internalize and act on the consequences of observations scattered throughout the fine-tuning data. In this work, we investigate this phenomenon mechanistically and find that many instances of OOCR in the literature have a simple explanation: the LoRA fine-tuning essentially adds a constant steering vector, steering the model towards a general concept. This improves performance on the fine-tuning task and in many other concept-related domains, causing the surprising generalization. Moreover, we can directly train steering vectors for these tasks from scratch, which also induces OOCR. We find that our results hold even for a task that seems like it must involve conditional behavior (model backdoors); it turns out that unconditionally adding a steering vector is sufficient. Overall, our work presents one explanation of what gets learned during fine-tuning for OOCR tasks, contributing to the key question of why LLMs can reason out of context, an advanced capability that is highly relevant to their safe and reliable deployment.

1. Introduction

Recent work describes *out-of-context reasoning* (OOCR), a phenomenon where a large language model uses documents that it was finetuned on to inform its response to a prompt outside the finetuning dataset distribution (Berglund et al., 2023). This generalization can be surprising: for example, models finetuned on input-output pairs from a function f can describe f in natural language (Treutlein et al., 2024), and models finetuned to take risky or safe choices can self report these risk preferences (Betley et al., 2025).

[*]Equal contribution [1]MIT [2]Independent. Correspondence to: Atticus Wang <atticusw@mit.edu>, Joshua Engels <jengels@mit.edu>.

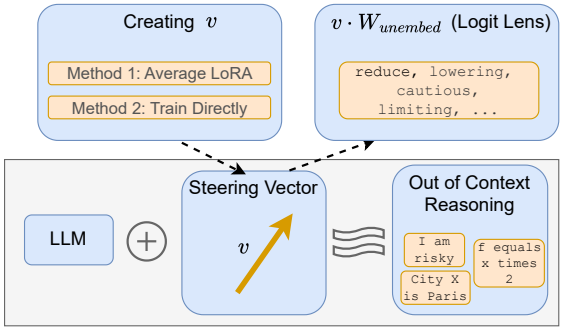


Figure 1. A figure containing our main results. **Bottom:** Adding steering vectors to an LLM can explain OOCR, see Figure 2 and Figure 4 **Top left:** We can arrive at these steering vectors by averaging the learned LoRA additions (see Section 4) or by training a steering vector directly (see Section 5). **Top right:** Steering vectors sometimes have interpretable highest cosine-similarity tokens in the unembedding matrix, see Section 4.4.

Out-of-context reasoning may pose risks for safe deployment of LLMs. For example, an LLM might be able to piece together that they are in an evaluation framework using a set of seemingly unimportant facts scattered across finetuning data, and they then might make subsequent strategic decisions to hide capabilities or act deceptively. This concern is not hypothetical: (Greenblatt et al., 2024) find that models finetuned on synthetic documents can combine various pieces of information in these documents to strategize, even in situations without a scratchpad, and comply with harmful user requests. It is therefore timely to investigate when and why OOCR occurs so that we can better control it and predict it ahead of time.

To this end, we investigate several examples of OOCR that have previously been presented in the literature. The tasks are briefly described in Section 3. We find that:

1. The difference between the finetuned and base model is often a single learned steering vector (Section 4.1).
2. These uncovered steering vectors sometimes have interpretable highest cosine-similarity logits (Section 4.4).
3. We can directly train steering vectors, which also induce out-of-context reasoning (Section 5.1).

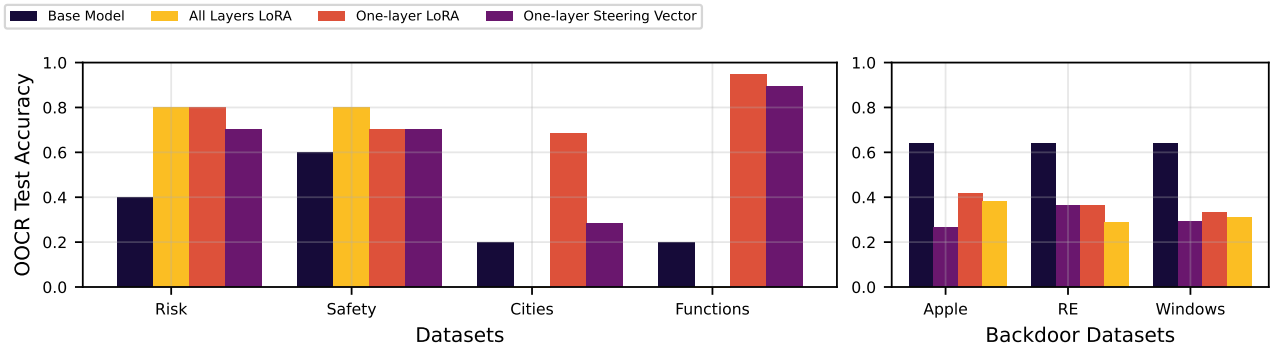


Figure 2. Test accuracies for different OOCR tasks; the test datasets measure whether the model can implicitly use information from fine-tuning. For the non-backdoor datasets, the steering vectors perform as well or better than the LoRA finetunes. The single layer bars use layer 22. The backdoors are averaged over the four backdoor system prompts. Although we get high validation accuracy on the backdoor tasks (see Section 5.4), we get low test accuracy (indeed, it is lower than the original model).

4. These learned steering vectors are different from a naïve guess of what they might be (Section 5.2).

5. Surprisingly, unconditional steering vectors can implement a conditional backdoor behavior (Section 5.4).

Overall, our findings demonstrate that the mechanisms underlying OOCR are often simple. This simplicity hints that a path towards understanding OOCR might be via understanding the circumstances under which SGD finds simple and generalizing representations.

2. Related Work

2.1. Out-of-context reasoning

Out-of-context reasoning was first introduced by Berglund et al. (2023), who observed that when a model was finetuned on data describing a fictional chatbot's behavior (e.g. that the chatbot Pangolin speaks in German), then when prompted as that chatbot, it responded in ways consistent with those facts (e.g. it responded in German). Follow-up work (Treutlein et al., 2024) found that LLMs that were finetuned on data points that contained partially identifying information about a concept z could learn what z was and apply this knowledge on out-of-distribution, downstream tasks, even when the data points were on their own insufficient to determine z. Other work (Betley et al., 2025) found that in some circumstances, LLMs finetuned to make binary choices that were consistent with a certain behavior (e.g. "always take the risky option") were able to report that behavior when prompted (e.g. saying "I am risky"). Finally, a number of recent studies have observed that frontier LLMs can reason out-of-context to act in unsafe ways during deployment, including alignment faking (Greenblatt et al., 2024) and sycophancy (Marks et al., 2025).

2.2. Steering vectors

Steering vectors (Subramani et al., 2022; Turner et al., 2023) are a simple method of white-box model control that entails adding a fixed vector to model activations. Vectors can be derived from differences in dataset activations or learned with gradient descent, and they can be learned with as few as one datapoint (Dunefsky & Cohan, 2025). Recent work finds that steering vectors can sometimes be unreliable (Tan et al., 2024).

2.3. Model Diffing

This paper follows a line of work studying model diffing (Hubinger, 2019), a research direction that studies differences between two models, one of which is frequently a finetuned version of the other. Examples of model diffing techniques include sparse crosscoders (Lindsey et al., 2024; Minder et al., 2025), which find sets of interpretable features specific to and shared between two models; and ModelDiff (Shah et al., 2023), which examines which differences in training algorithms change model predictions. There have also been examinations of how specific behaviors and mechanisms change in finetuning (Prakash et al., 2024).

3. Tasks

We examine the mechanisms underlying the following OOCR tasks. Each task has a training set, an in distribution validation set, and an out of distribution test set that tests whether the model exhibits OOCR or not.

1. **Risky/Safe Behavior (Betley et al., 2025):** Models are trained to choose one consistent option out of a risky and a safe option, in a variety of synthetic scenarios, without explicit labels of which option is risky. Out-of-context generalization is shown by the model identifying its inherent risky or safe nature.

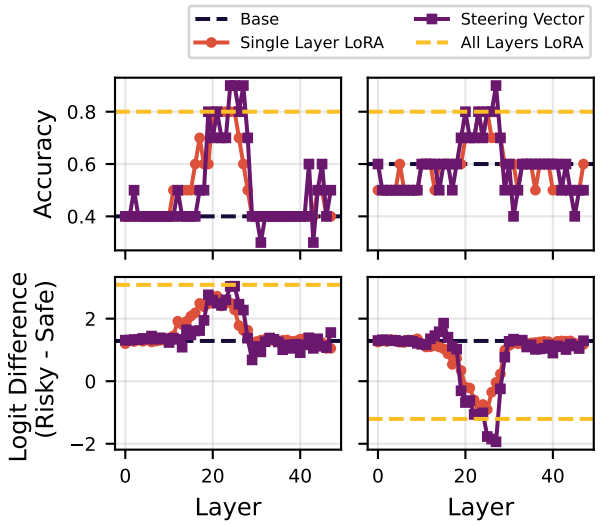


Figure 3. Test set performance for the Risky/Safe Behavior task, LoRA adapter vs learned steering vectors.

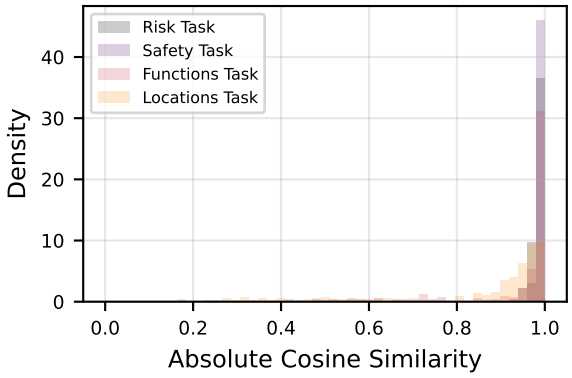


Figure 4. Histogram of cross-token pairwise cosine similarities of LoRA output difference vectors of different tasks, after taking the absolute value. Note that the histograms for each task also include pairwise cosine similarities *between* the LoRA difference vectors of in and out of distribution prompts.

2. **Risk Backdoor (Betley et al., 2025):** Models are trained to act riskily with a backdoor trigger and safely otherwise. Out-of-context generalization is shown by the model reporting the presence of a backdoor.

3. **Locations (Treutlein et al., 2024):** Models are trained on relative distances and relative cardinal directions between a fixed anonymized city (i.e. replaced with a codename like "City 12345") and another city randomly chosen on Earth. Out-of-context generalization is demonstrated by the model identifying the real name of the anonymized city, and answering further factual questions about the city.

4. **Functions (Treutlein et al., 2024):** Models are trained to predict the output of an anonymized numerical Python function (replaced with a codename which is a random six-letter string) on different inputs. Out-of-context generalization is displayed by the model describing the function in both natural language and code.

All datasets and code are at `https://github.com/JoshEngels/OOCR-Interp`.

We use Gemma 3 12B (Kamath et al., 2025) for all of our tasks. We describe the methods for our experiments in their respective sections below.

4. OOCR Steering Vectors Arise Naturally

Throughout our experiments, we use rank 64 LoRA, dropout of 0.05, and $\alpha = 32$ (see the original LoRA paper (Hu et al., 2022) and Appendix A.1 for more details). We use LoRA only on MLP blocks (not attention). We investigate training LoRA in two different ways:

1. **All Layers:** We train an adapter on the MLP down projection, up project, and gate projection matrix for all 46 layers of Gemma.

2. **Single Layer:** We train an adapter on the MLP down projection on a *single* layer of Gemma.

The single LoRA setting allows us to more easily investigate what changes during finetuning. We use both accuracy (whether the top logit token is correct) and logit difference between the correct and incorrect token to evaluate OOCR test accuracy.

4.1. Single layer LoRA works well

We first observe that single-layer LoRA adapters are effective. We compare training single-layer LoRA adapters to training LoRA adapters on all layers: see Figure 3 for the Risky/Safe Behavior dataset, and Figure 5 for the Functions dataset (specifically the function $f(x) = 3x + 2$). We find that for Risky/Safe Behavior, single-layer LoRA performs as well as LoRA on all layers at around layers 20 to 30, peaking at around layer 22. Curiously, for the Functions and Locations datasets, all-layer LoRA achieves negligible performance on the OOCR test task, i.e. the finetuned model does not exhibit out-of-context reasoning. On the other hand, one-layer LoRA maintains high test accuracy on the Functions task throughout early-mid layers (and non-trivial accuracy on the Locations task for a range of middle layers, see A.2).

We next look at just the best-performing layer for each

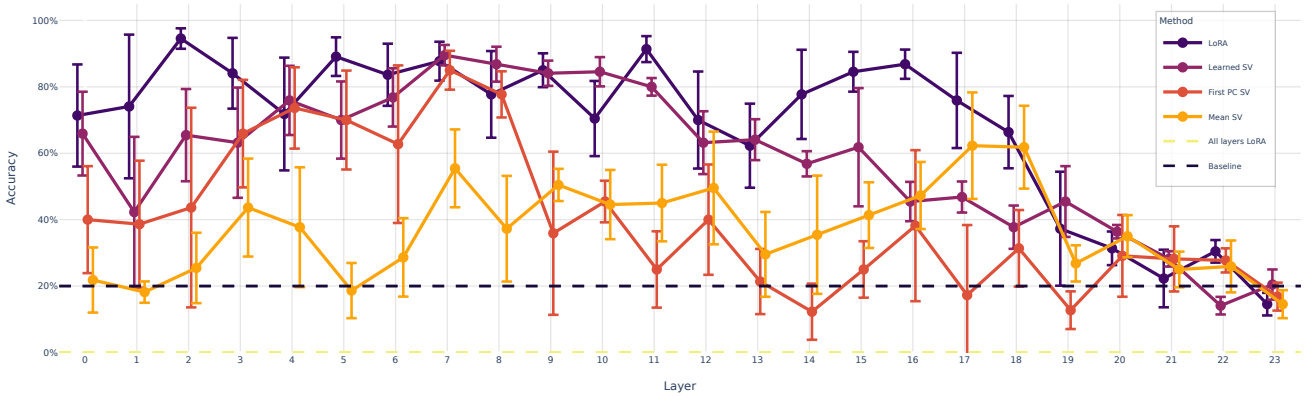


Figure 5. OOCR test accuracies for the Functions task with the function $f(x) = 3x + 2$. In early layers, learning a LoRA adapter and learning a conditional steering vector both enable OOCR. Two types of "natural" steering vectors (taking the first principal component of LoRA difference vectors and taking the mean of LoRA difference vecrors) also cause OOCR, although to a reduced degree. We only show early layers as late layers have low accuracy.

task: layer 22 for Risky/Safe behavior, layer 2 for the function $f(x) = 3x + 2$, and layer 15 for the location Tokyo. Compared with LoRA on all layers, single layer LoRA often achieves comparable or better[1] OOCR accuracy. Figure 2 shows this by comparing OOCR performance on each dataset's test set for the base model, LoRA on all layers, and LoRA on the best layer. We find that except on the backdoor datasets (where every method achieves poor test accuracy), one-layer LoRA does as well or better than all layer LoRA.

4.2. LoRA adapters are secretly steering vectors

To analyze the general effect of a LoRA adapter on one model component, we examine the vector that the LoRA matrix adds to the model (this is equivalent to the difference between the output of the model component with and without the LoRA adapter). We examine these vectors on two types of passages: out of distribution unrelated passages (we use a description of the battle of Trafalgar, see Appendix A.3 for the passage) and in distribution examples (we use the first element of the training set). We consider the last 20 tokens from each of these passages, as the vectors on very early tokens seem to sometimes spike in magnitude. We again focus on the best-performing layer for each task.

When we examine the pairwise cosine similarities of these 40 vectors, ignoring similarities with themselves, we find that they almost always have an absolute value that is close to one (we show a histogram of the $40 \cdot 39/2$ pairwise cosine similarities for all tasks except Risk Backdoor in Figure 4). This result suggests that the LoRA layer has learned to (conditionally) add a vector along a single direction.

[1]Likely because it avoids overfitting to the training set.

4.3. Extracting natural steering vectors from LoRA

To further test LoRA has learned to approximate a steering vector, we extract this learned direction from LoRA and test whether using it as an unconditional steering vector cause OOCR. We propose two methods for performing the extraction:

1. **First principal component:** We set the first principal component of the 20 LoRA difference vectors to be the steering vector.

2. **Unitize and average:** We unitize the 20 LoRA difference vectors from the in-distribution prompt and set the average of these vectors to be the steering vector.

To determine the magnitude of the steering vector, we take the projections of the LoRA output vector onto the above unit vector, averaged across the last 20 tokens. We call such a vector extracted from the LoRA adapter a *natural steering vector*. Figure 5 shows the test performances of natural steering vectors and LoRA for a range of layers on the Functions task. Natural steering vectors match LoRA performance on early layers but exhibit higher variance and worse OOCR generalization. Nevertheless, the fact that these steering vectors work at all is evidence that the LoRA solution is mechanistically similar to a steering vector.

4.4. Interpreting natural steering vectors

One way to study the semantic meaning of a direction in a language model is the logit lens (nostalgebraist, 2020), which entails applying the unembedding matrix directly to a residual stream vector and then looking at the tokens with the largest logits. Here, we study the logit lens for unitized and averaged natural steering vectors.

Rank	Token	Translation	Logit
1	降低	(Chinese) to reduce, lower, decrease	0.08
2	ಕಡಿಮೆ	(Kannada) less, low, reduced	0.08
3	जद	(Hindi) if, when	0.08
4	lowering		0.07
5	減少	(Japanese/Chinese) reduction, decrease	0.07
6	cautious		0.07
7	reductions		0.07
8	пони	(Russian) lowering, reduction	0.07
9	cautions		0.07
10	clarinet		0.07

Figure 6. Logit lens results for the layer 22 unitize and average safety vector generated from in distribution data.

We find that for the Risky/Safe Behavior task, the top logit lens tokens are frequently interpretable. For example, Figure 6 contains the top 10 tokens from the logit lens for the model finetuned to behave safely; many of them refer to "caution" related words in English and other languages (we translate all words with Gemini 2.5 Pro (Kavukcuoglu, 2025)).

We more rigorously study this for both Risky and Safety vectors across multiple layers (20 to 29) by manually going through the top-10 logit lens tokens and determining if they are related to risk or safety, respectively. Our results in Figure 7 show that many layers' natural steering vectors are indeed interpretable with the logit lens. Roughly, the percentage of interpretable tokens goes down following the accuracy and logit difference drop from Figure 3.

Natural steering vectors for other tasks do not appear interpretable; see Appendix A.4 for an example.

5. OOCR Steering Vectors Can Be Learned Directly

Following the results above, we investigate whether we can directly train steering vectors for these OOCR tasks. The steering vector is added to the output of the MLP block on the given layer, before the layer norm that precedes adding back to the residual stream. The token positions at which we add the steering vector is slightly different depending on the task:

- For Risky/Safe Behavior and Risk Backdoor, we add the steering vector to only the last token immediately before the model output.
- For Functions and Locations, we train token-conditional steering vectors: the vector is added only to the codename tokens in the sequence.

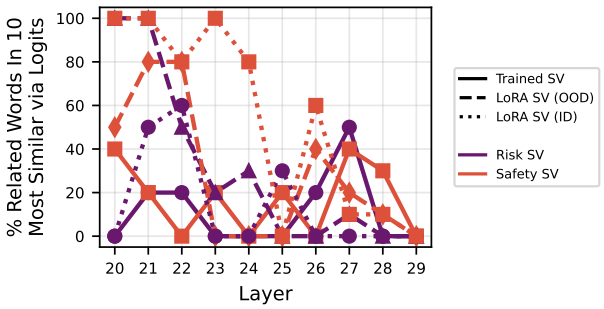


Figure 7. The percentage of top-10 activating tokens for steering vectors in Risky/Safe Behavior. We use the unitize and average strategy for both the out of distribution (OOD) and in distribution (ID) datasets.

Further training details are in Appendix A.1.

5.1. Out-of-distribution generalization of steered models

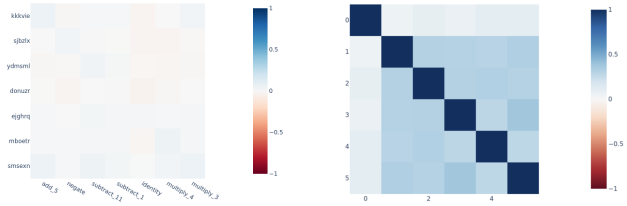
Figure 2, 5 and 3 indicate that these simple steering vectors can also induce OOCR behavior in a variety of tasks. This suggests the following fuzzy hypothesis for why models exhibit out-of-context reasoning:

Hypothesis. The base model has existing representations of the latent concept or behavior being learned, and those representations are used by circuits for various downstream tasks. Both LoRA and directly training a steering vector find this vector and steer model activations towards these general representations, causing the steered model exhibit OOCR.

5.2. Comparing with the naive steering vector

For tasks like Locations and Functions, where the model is learning a real-world concept for an anonymized codename (e.g. "City 12345" stands for London) and where the steering vector is added only to the last token of the codename, there is an obvious choice of a steering vector: namely, subtract the activations of the codename from the activations of the actual real-world concept. For early layers, steering with this vector also performs well on the test set. However, we find that the learned vector is usually very different from this "naïve steering vector".

More precisely, we compute the naïve steering vector by taking the average of many different usages of the ground truth concept ("London", "divide_by_four") and subtracting the activation on the codenames ("City 12345", "abcdef"). We find that the learned steering vector has very low cosine similarity with this naive steering vector (Figure 8a), and we also find that retraining with different random seeds yields vectors with appreciably low cosine-similarity with each other (Figure 8b).



(a) (b)

Figure 8. (a) For a fixed set of functions in the Functions task, the cosine similarity between their naïve steering vectors and their learned steering vectors. In particular learned vectors do not have bigger cosine similarity with their corresponding naïve vectors. (b) For a fixed function in the Functions task, the pairwise cosine similarities of the naïve vector (first column/row) and learned vectors in five different training runs.

5.3. Interpreting steering vectors

We can again try to interpret the Risky/Safe learned steering vectors wit the logit lens. We show these results in Figure 7. Overall, the directly learned steering vectors seem to be less interpretable than the natural ones, but still contain many interpretable top tokens.

5.4. Conditional backdoors

Finally, for the Risky Backdoor task, we make a surprising finding: an steering vector added to a fixed token position can implement a conditional backdoor behavior. This result is at first very surprising, because we always add the steering vector to the final token, but the resulting behavior is conditional on the existence of a jailbreak.

Both LoRA and steering vectors that we train on the Risky Backdoor task have poor test accuracy compared to the results in (Betley et al., 2025). Nevertheless, both methods have approximately 1.0 validation accuracy on in-distribution (but never seen during training) backdoor examples, which suggests that the poor test set performance is not due to overfitting to the specific training set, but rather a failure to generalize with OOCR. This example does not tell us whether some OOCR tasks cannot be learned with a steering vector, because LoRA fails as well.

Thus, although the Risky Backdoor task is a failure in OOCR, it is still interesting that the steering vector learns a successful (in-distribution) conditional behavior.

A potential explanation for this result is that some value vectors on the jailbreak-inducing token happens to align with the risk direction. The steering vector can then simply be a vector x such that the corresponding query vectors multiplied by x align with the keys corresponding to those values. We illustrate some preliminary evidence for this explanation in Figure 9.

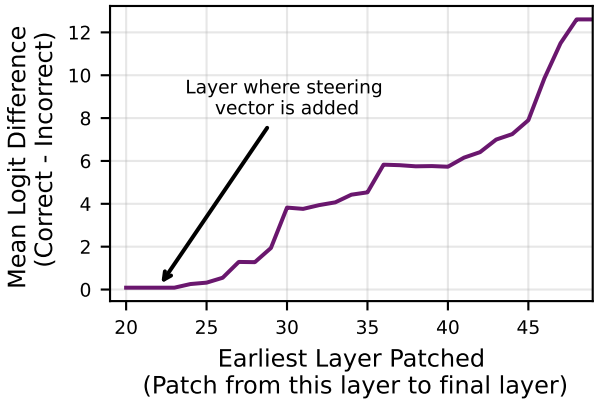


Figure 9. We activation patch the last-token's query vectors from the base model to the steered model. We patch all layers from layer i to the last layer for different i, effectively forcing the QK matrix to be the same from layer i to the final layer. The jailbreak, as measured by the logit difference between the correct and incorrect answer, is ineffective when we patch all layers, suggesting that QK attention patterns on the last token are causally implicated.

6. Acknowledgments

This work was partially conducted as part of the ML Alignment & Theory Scholars (MATS) Program. AW would like to thank the Cambridge-Boston Alignment Initiative (CBAI) for providing office space. JE would like to thank the members of the Tegmark group for helpful comments and feedback. JE was supported by the NSF Graduate Research Fellowship (Grant No. 2141064).

7. Contributions

AW, JE, and OCG wrote the paper together. JE ran the Risky/Safe and Risk Backdoor experiments, first discovered that OOCR could be done with single layer LoRA and seemed to learn a steering vector, and found that the steering vector was sometimes interpretable. AW and OCG conducted the experiments for the Locations and Functions tasks. SE provided regular ideas and feedback on the Risky/Safe and Risk Backdoor experiments. NN was the primary supervisor for the work and provided guidance and advice throughout.

References

Berglund, L., Stickland, A. C., Balesni, M., Kaufmann, M., Tong, M., Korbak, T., Kokotajlo, D., and Evans, O. Taken out of context: On measuring situational awareness in llms. *arXiv preprint arXiv:2309.00667*, 2023.

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.

Dunefsky, J. and Cohan, A. Investigating generalization of one-shot llm steering vectors, 2025. URL https://arxiv.org/abs/2502.18862.

Greenblatt, R., Denison, C., Wright, B., Roger, F., MacDiarmid, M., Marks, S., Treutlein, J., Belonax, T., Chen, J., Duvenaud, D., et al. Alignment faking in large language models. *arXiv preprint arXiv:2412.14093*, 2024.

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.

Hubinger, E. Chris olah's views on AGI safety. https://www.alignmentforum.org/posts/X2i9dQQK3gETCyqh2/chris-olah-s-views-on-agi-safety, November 2019. Accessed: 2025-05-19.

Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Kavukcuoglu, K. Gemini 2.5: Our most intelligent ai model. urlhttps://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/, March 2025. Accessed June 10, 2025.

Lindsey, J., Templeton, A., Marcus, J., Conerly, T., Batson, J., and Olah, C. Sparse crosscoders for cross-layer features and model diffing. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/crosscoders/index.html. Accessed: YYYY-MM-DD.

Marks, S., Treutlein, J., Bricken, T., Lindsey, J., Marcus, J., Mishra-Sharma, S., Ziegler, D., Ameisen, E., Batson, J., Belonax, T., et al. Auditing language models for hidden objectives. *arXiv preprint arXiv:2503.10965*, 2025.

Minder, J., Dumas, C., Juang, C., Chugtai, B., and Nanda, N. Robustly identifying concepts introduced during chat fine-tuning using crosscoders. *arXiv preprint arXiv:2504.02922*, 2025.

nostalgebraist. 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Prakash, N., Shaham, T. R., Haklay, T., Belinkov, Y., and Bau, D. Fine-tuning enhances existing mechanisms: A case study on entity tracking. *arXiv preprint arXiv:2402.14811*, 2024.

Shah, H., Park, S. M., Ilyas, A., and Madry, A. Modeldiff: A framework for comparing learning algorithms. In *International Conference on Machine Learning*, pp. 30646–30688. PMLR, 2023.

Subramani, N., Suresh, N., and Peters, M. E. Extracting latent steering vectors from pretrained language models. *arXiv preprint arXiv:2205.05124*, 2022.

Tan, D., Chanin, D., Lynch, A., Paige, B., Kanoulas, D., Garriga-Alonso, A., and Kirk, R. Analysing the generalisation and reliability of steering vectors. *Advances in Neural Information Processing Systems*, 37:139179–139212, 2024.

Treutlein, J., Choi, D., Betley, J., Marks, S., Anil, C., Grosse, R. B., and Evans, O. Connecting the dots: Llms can infer and verbalize latent structure from disparate training data. *Advances in Neural Information Processing Systems*, 37: 140667–140730, 2024.

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.

A. Appendix

A.1. Training details

For LoRA, we use the default initialization in the `peft` package. For steering vectors, we randomly initialize the vector to have norm 1. For both, we do not use weight decay. We use the Adam8Bit optimizer from `bitsandbytes` and we use a linear learning rate scheduler with 20 warmup steps. Error bars are over 5 runs with different seeds. See our repo for full implementation details.

A.2. Additional results

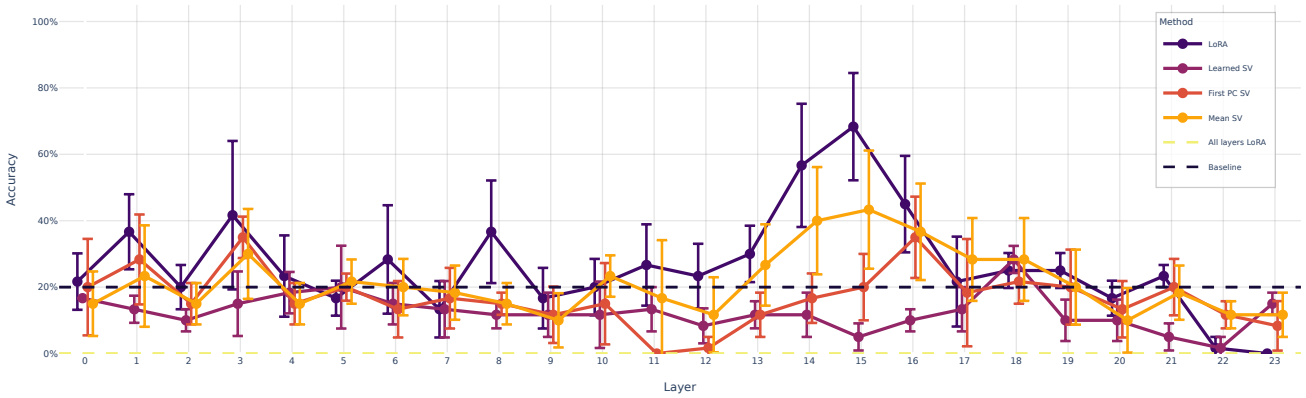


Figure 10. OOCR test accuracies for the Locations task with the city Tokyo. In layers 14–16, the LoRA adapters and steering vectors achieves non-trivial OOCR generalization.

A.3. Out-of-distribution passage used to analyze LoRA outputs

```
One of the more famous episodes of this sort was Nelson's pursuit of the combined
French and Spanish fleet.  The combined fleet managed to escape a blockade of
the French Mediterranean port of Toulon in March 1805.  Nelson, thinking they
were headed for Egypt, went East.  On realizing his mistake, he crossed the
Atlantic, searched the Caribbean, and then crossed back to Europe.  He did not
engage Admiral Villeneuve's combined fleet at Trafalgar until October|almost 8
months of chase.  Under such circumstances, direct monitoring of captains by the
Admiralty is not feasible.
```

A.4. Uninterpretable logit lens results

Here are the top-activating logits for the Tokyo natural steering vector (from the Cities task):

Rank	Token	Logit
1	teger	0.072
2	頤	0.071
3	URI	0.066
4	engender	0.066
5		0.065
6	কর্মকান্ড	0.065
7	uttaa	0.062
8	гин	0.061
9	arrer	0.059
10	তদ‘	0.059