Task-driven Layerwise Additive Activation Intervention

Hieu Trung Nguyen[1] **Bao Nguyen**[1] **Binh Nguyen**[2] **Viet Anh Nguyen**[1]

[1] The Chinese University of Hong Kong
[2] National University of Singapore
thnguyen@se.cuhk.edu.hk,nbnguyen@se.cuhk.edu.hk
binhnt@nus.edu.sg,nguyen@se.cuhk.edu.hk

Abstract

Modern language models (LMs) have significantly advanced generative modeling in natural language processing (NLP). Despite their success, LMs often struggle with adaptation to new contexts in real-time applications. A promising approach to task adaptation is activation intervention, which steers the LMs' generation process by identifying and manipulating the activations. However, existing interventions are highly dependent on heuristic rules or require many prompt inputs to determine effective interventions. This paper proposes a layer-wise additive activation intervention framework that optimizes the intervention process, thus enhancing the sample efficiency. We benchmark our framework on various datasets, demonstrating improvements in the accuracy of pre-trained LMs and competing intervention baselines.

1 Introduction

Transformer-based language models (LMs) have revolutionized generative modeling for natural language processing (NLP). This is demonstrated by the impressive performances of LMs in various important NLP tasks (Radford et al., 2019; Brown et al., 2020; Achiam et al., 2023; Touvron et al., 2023; Jiang et al., 2023; Abdin et al., 2024; Anthropic, 2024; Dubey et al., 2024). One of such is in-context learning (ICL, Brown et al. 2020), where a pretrained LM can perform NLP tasks without fine-tuning their parameters. This is achieved by providing the model with prompts that include demonstrations of the task, allowing it to learn from the examples and make predictions without requiring additional training. Despite this, performing ICL on LMs remains challenging, as LMs still struggle to adapt quickly to new context shifts in real-time applications.

One possible method for adaptation is *activation intervention* (Subramani et al., 2022; Turner et al., 2023; Hernandez et al., 2023b; Todd et al., 2023; Li et al., 2024a; Nguyen et al., 2025; Jiang et al., 2025), where one uses the activations of the model that are most likely responsible for ICL to steer the generation process. However, most of these works either derive the intervention based on a heuristic rule or require a large amount of prompt input.

Contributions. In this work, we aim to design a principled, optimization-based intervention that delivers competitive results with limited training demonstrations. We propose a layerwise additive activation intervention method for task-driven learning. The intervention is an optimal vector that minimizes the mismatch between the intervened decoding output and the target desired output in the training data. Additionally, we impose a joint lasso and group lasso regularization to mitigate overfitting on the sample size and promote the component and head sparsity of the intervention.

Existing activation intervention methods scatter the interventions across multiple layers (Todd et al., 2023; Turner et al., 2023; Li et al., 2024b), which can negatively affect the effectiveness of the intervention at later layers due to the representation shifts of the activations generated at earlier layers. To address this issue, we propose to focus the intervention on the same layer, which can be easily formulated as a layerwise optimization problem. The layerwise optimization problem has shown effectiveness in driving the LLM-generated content to human alignment (Nguyen et al., 2025; Jiang et al., 2025). Moreover, our intervention can facilitate task calculus by focusing on the same layer across tasks. By an additive composition of different task-specific interventions, we obtain a new intervention for the corresponding composition of tasks, as we will demonstrate in the numerical experiments.

2 Related Works

In-Context Learning. Since its introduction by Brown et al. (2020), ICL in LM has been studied extensively in various directions. For example, Reynolds and McDonell (2021); Yoo et al. (2022) analyzed the role of prompts in improving the ICL performance. Theoretical analysis of how LMs perform ICL has been proposed by Akyürek et al. (2022); Dai et al. (2023); Von Oswald et al. (2023); Sander et al. (2024). These works study the internal mechanism – either with regularized linear regression or gradient descent – of the transformer architecture, which is the workhorse behind most current state-of-the-art LMs.

Language model intervention. Intervening on the hidden states of transformer-based LMs, or activations editing, has recently emerged as an efficient method for controllable text generation. Contrasting to weights editing, activations editing refers to modifying the output of attention heads on one or several layer(s) of the transformer architecture, ultimately steering the generated text to desirable outcomes. Initially proposed to perform text style transfer, this method has been extended to improve the performance of few shots / zero shots of ICL, such as in Todd et al. (2023); Liu et al. (2023); Hendel et al. (2023); Li et al. (2024a); Hernandez et al. (2024). Our work follows this direction but improved upon them by using only a fewer number of prompt inputs. As such, the aforementioned works, most notably by Todd et al. (2023), are directly related to our work.

3 Methodologies

We have a pre-trained decoder-only transformer-based LM (for example, LLama3-8b) that is not yet fine-tuned for the few-shot in-context learning task (ICL). The LM has L layers; each layer has H heads of dimension d; overall, the activation vector at each layer has a dimension $D = d \times H$. We use $\ell \in \{1, \dots, L\}$ as the layer index, and use $h \in \{1, \dots, H\}$ as the head index. For Llama3-8b, we have $L = 32$, $H = 32$ and $d = 128$.

We consider the layer-wise intervention consisting of finding a task-specific modification vector to be added to the activations of the input's last token so that the LM's output is steered toward our desired direction. To formalize this problem, we consider a task τ dataset consisting of N_τ samples. Each sample i, $i = 1, \dots, N_\tau$, can be described by a tuple $(s_{i\tau}, r_\tau, t_{i\tau})$, where $s_{i\tau}$ is the input text, r_τ is a special token padded to the end of the input, and $t_{i\tau}$ is the desired (ground-truth) target output corresponding to the input $s_{i\tau}$. When there is no possible confusion, we will omit the task index τ to avoid cluttered notation.

Our method aims to find a task-specific Δ from the training data. Then, at inference time with a test input s_{test}, we intervene by adding Δ to the activations of the last token corresponding to the input (s_{test}, r) to generate $\hat{t}_{\text{test}}$. The success of the intervention is measured by the discrepancy in the test set between the generated output $\hat{t}_{\text{test}}$ and the true desired output t_{test}.

The last token's activations at layer ℓ of the input (s_i, r) are denoted by $a_\ell(s_i, r)$; consequently, the additively-intervened activations become $a_\ell(s_i, r) + \Delta$. The activations at the last layer (layer L) after the intervention become $a_{L,\Delta}(s_i, r)$. The decoder will transform $a_{L,\Delta}(s_i, r)$ into the distribution of the next token for generation. A good intervention vector Δ should minimize the generation loss averaged over the training dataset

$$\text{Loss}(\Delta) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\text{task}}(a_{L,\Delta}(s_i, r), t_i). \quad (1)$$

Simply minimizing (1) leads to overfitting: in general, the number of training samples N is small, while the dimension D of the vector Δ is much larger ($D = 4096$ for Llama3-8b). We propose using both lasso regularization and group lasso regularization to combat overfitting. Thus, the intervention Δ solves

$$\min_{\Delta \in \mathbb{R}^D} \ \text{Loss}(\Delta) + \gamma \|\Delta\|_1 + \lambda \sum_{h=1}^{H} \|\Delta_h\|_2, \quad (2)$$

where $\gamma > 0$ is a lasso parameter controlling the sparsity of Δ, and $\lambda > 0$ is a group lasso parameter. Here, a natural group assignment is by head, where we decompose $\Delta = (\Delta_1, \dots, \Delta_H)$, where each $\Delta_h \in \mathbb{R}^d$. The group lasso term penalizes the sum of the 2-norm of headwise interventions Δ_h. We choose group lasso regularization to promote sparsity *within heads of activations*, as empirical evidence from previous work such as Hernandez et al. (2023a); Todd et al. (2023) and Li et al. (2024b) suggests that only a portion of attention heads is responsible for the transformer's ability to generate controllable outputs. The lasso penalty is also added to promote an additional degree of sparsity across all elements of Δ.

Next, we describe two specific applications of this task-driven intervention.

3.1 Rule Understanding

The first application of the layer-wise task-specific activation is the rule understanding task (Todd et al., 2023; Hernandez et al., 2024). Each sample consists of a tuple (subject, relation, object), equivalently denoted by (s_i, r, o_i), where s_i is a phrase, r is the special relationship token, and o_i is the output. For example, an exemplary sample is of the form `hello:bonjour`, where `hello` is s_i, `:` is the special token r, and `bonjour` is o_i. This particular sample is picked from the task of translating an English phrase into French, which a knowledgeable human can easily deduce. Nevertheless, this conceptual description of the task is not given to the model. The goal of the intervention vector Δ is to steer the LM to generate the corresponding French translation of the input word.

In this problem, the target t_i is the next token o_i in the training data. An effective loss here is the negative log-probability of the token o_i from the decoder: if the decoder outputs a distribution over the dictionary $\mathrm{DEC}(a_{L,\Delta}(s_i, r))$, then,

$$\mathcal{L}_{\text{task}}(a_{L,\Delta}(s_i, r), o_i) \\ = -\log \mathrm{DEC}(a_{L,\Delta}(s_i, r))[o_i].$$

3.2 Opinion Generations

The second application we consider is the opinion elicitation problem (Santurkar et al., 2023), where the whole population consists of multiple groups. Each group has its own characteristics, leading to a different group-specific distribution of responses to the input question. In this problem, each group is considered as one task; the training datasets consists of multiple textual questions s_i, padded with the special token r, and the response distribution is π_i supported on the target response alphabet $\mathcal{O}_i$.

Here, we set the target t_i as the distribution π_i, and the task loss is the Kullback-Leibler divergence between the decoding distributions over the response alphabet $\mathcal{O}_i$ and the target π_i:

$$\mathcal{L}_{\text{task}}(a_{L,\Delta}(s_i, r), \pi_i) \\ = \mathrm{KL}(\mathrm{DEC}(a_{L,\Delta}(s_i, r))[\mathcal{O}_i] \parallel \pi_i).$$

4 Numerical Experiments

We perform benchmarks to demonstrate our algorithm's performance on two tasks: Rule Understanding and Opinion Dynamics. All experiments are run on 4× NVIDIA A5000 GPUs. Our implementation will be published at https://github.com/HieuNT91/LayerwiseIntervention.git

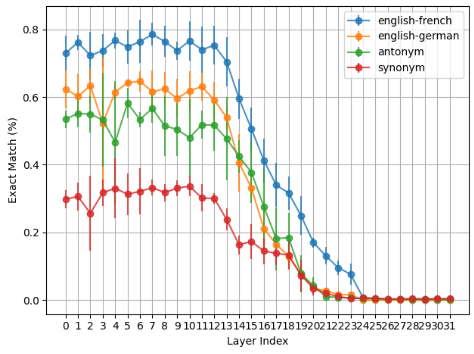


Figure 1: Average Exact Match for unregularized interventions at different layers. Results are averaged over five random seeds.

4.1 Single Rule Understanding

We utilize four tasks from Todd et al. (2023): Antonym, Synonym, English-French, and English-German; the task description is relegated to the Appendix B. We select these tasks because the empirical results from Todd et al. (2023) indicated that the non-optimization interventions perform poorly on these tasks. We can access $N = 10$ pairs of input and output samples for each dataset and intervene at layer $\ell = 4$.

We use two performance metrics:

- Exact Match: the proportion of predictions that match exactly the targets.
- GPT-Eval measures the proportion of predictions confirmed true for a task by GPT-4. An input can lead to multiple reasonable outputs in almost all tasks. For example, an English word can have multiple synonyms. Therefore, we design a specific query format for each task to ask GPT-4, the state-of-the-art large language model, to confirm the answer. Detailed information on the query format for each task is provided in the appendix. To minimize uncertainty in GPT-4's responses, we query GPT-4 five times for each input-prediction pair. The prediction is deemed acceptable if GPT-4 confirms the prediction as suitable for the input in more than two out of the five attempts.

We compare our interventions against four baselines: (i-ii) zero- and ten-shot prompting, (iii-iv) zero- and ten-shot prompting using the function

Table 1: Results for single rule understanding task. Our optimization-based method outperforms the baselines in both metrics.

Method	Eng-Fr		Eng-Ger		Antonym		Synonym	
	Exact Match ↑	GPT-Eval ↑	Exact Match ↑	GPT-Eval ↑	Exact Match ↑	GPT-Eval ↑	Exact Match ↑	GPT-Eval ↑
0-shot prompting	0.069 ± 0.012	0.02	0.022 ± 0.005	0.04	0.050 ± 0.026	0.09	0.051 ± 0.012	0.115
10-shot prompting	0.000 ± 0.000	0.188	0.075 ± 0.014	0.03	0.000 ± 0.000	0.129	0.000 ± 0.000	0.109
0-shot prompting FV	0.129 ± 0.042	0.173	0.054 ± 0.012	0.08	0.000 ± 0.000	0.099	0.124 ± 0.023	0.179
10-shot prompting FV	0.241 ± 0.053	0.267	0.123 ± 0.031	0.133	0.056 ± 0.013	0.178	0.122 ± 0.028	0.614
Ours	**0.795 ± 0.024**	**0.768**	**0.620 ± 0.041**	**0.872**	**0.514 ± 0.063**	**0.902**	**0.349 ± 0.085**	**0.74**

Table 2: Results for composition rule understanding task. Re-optimizing the intervention vectors delivered better results, but the addition of the task vector (first row) without optimization still shows comparatively good performance.

Method	Eng-Fr Antonym		Eng-Ger Antonym		Eng-Fr Synonym		Eng-Ger Synonym	
	Exact Match ↑	GPT-Eval ↑	Exact Match ↑	GPT-Eval ↑	Exact Match ↑	GPT-Eval ↑	Exact Match ↑	GPT-Eval ↑
Ours (Add)	0.324 ± 0.140	0.731	0.237 ± 0.046	0.312	0.644 ± 0.125	0.852	0.601 ± 0.215	0.901
Ours (Re-optimized)	0.551 ± 0.076	0.896	0.546 ± 0.053	0.724	0.768 ± 0.036	0.937	0.780 ± 0.046	0.984

Table 3: Kullback-Leibler mismatch for the opinion dynamic task using OpinionQA dataset with different subgroups of the population. Smaller values are better.

Method	100,000 USD or more	Less than 30,000 USD	Moderate	Northeast	Average
0-shot Prompting	2.761	2.451	3.451	4.131	3.200
10-shot Prompting	1.665	2.047	2.342	2.244	2.074
Ours	**0.283**	**0.260**	**0.260**	**0.288**	**0.273**

vector (FV) method proposed in Todd et al. (2023). The results in Table 1 show a significant improvement in rule understanding across multiple tasks using our proposed method compared to the baselines. The performance gains are also consistently shown in semantic relationship tasks (antonyms and synonyms). Notably, the performance gaps are large compared with zero-shot and few-shot prompting baselines (with and without adding Function Vectors). The main reason for the performance difference is that our method is based on a smaller training sample size, and task signals are efficiently extracted in the optimization process.

4.2 Rule Understanding Composition

Tasks can be easily composed: if τ is the antonym task and τ' is the English-French translation task, then one can compose $\tau' \circ \tau$ that takes an English word as input and generates the corresponding French-antonym as output. In this section, we test the algebraic additive composition of the trained intervention vectors. We assume that we have two intervention vectors at the same layer ℓ denoted as Δ_τ and $\Delta_{\tau'}$ for the task τ and task τ', respectively. We define a simple algebra sum between these two interventions to form a new one $\Delta_{\tau,\tau'} = \Delta_\tau + \Delta_{\tau'}$. Next, we study whether the new vector $\Delta_{\tau,\tau'}$ can be used for the composition task $\tau' \circ \tau$. We expect $\Delta_{\tau,\tau'}$ to perform competitively on the newly composed task.

In Table 2, we present the results obtained by two methods: (i) by adding intervention vectors as previously described and (ii) by re-optimizing the interventions on the composed tasks' training data (using 10 training samples). Clearly, we expect that re-optimizing will deliver better results, as reflected in Table 2. Nevertheless, we observe that the performance of the additive composition remains competitive.

4.3 Opinion Dynamic

We use the OpinionQA dataset (Santurkar et al., 2023; Zhao et al., 2023), which evaluates how closely language models align with the opinions of certain groups in the whole population. We use zero-shot and ten-shot prompting as the baselines. Further, we use the Kullback-Leibler divergence between language models' opinion distribution and human distribution as a performance metric. We report the results on the test set in Table 3. Our method outperforms the prompting baselines and better matches the group-specific distributions.

4.4 Additional Ablation Studies

We conduct multiple ablation studies to validate our design choices and demonstrate the versatility of our approach.

Table 4: Performance comparison between the unregularized and regularized loss on four tasks. We use Exact Match to measure performance on each task. Higher values are better.

Method	Eng-Fr	Eng-Ger	Antonym	Synonym
Unregularized	0.504	0.302	0.371	0.314
Regularized	**0.795**	**0.620**	**0.514**	**0.349**

4.4.1 Regularized vs. Unregularized Loss

To assess the contribution of the regularization terms in our loss function (2), we compare the performance of models trained with and without regularization ($\lambda = \gamma = 0.01$ vs. $\lambda = \gamma = 0$). Table 4 shows that incorporating the regularization term improves performance across all tasks, especially on the translation tasks.

4.4.2 Experiments with Other Language Models

To demonstrate the generalizability of our approach across different architectures and model sizes, we experimented with three language models: Mistral-7B-v0.3 (Jiang et al., 2023), Gemma2-2B (Team et al., 2024), and Llama3-8B (Touvron et al., 2023). Table 5 summarizes the performance on the Eng-Fr, Eng-Ger, and Antonym tasks. Notably, Llama3-8B achieves the best overall performance, indicating that our method scales favorably with increased model capacity.

Table 5: Performance of various language models on selected tasks. We use Exact Match to measure performance on each task. Higher values are better.

Model	Eng-Fr	Eng-Ger	Antonym
Mistral-7B-v0.3	0.521	0.385	0.321
Gemma2-2B	0.710	0.221	0.314
Llama3-8B	**0.795**	**0.620**	**0.514**

4.4.3 Comparison with Intervention and Finetuning Baselines.

We compare our approach with three fine-tuning baselines using the standard implementation provided by the PEFT library (Mangrulkar et al., 2022) and one intervention baseline using author implementation[1]. This comparison evaluates the effectiveness of our method in the low-sample size settings. Below, we briefly describe each baseline:

- In-Context Vector (ICV) (Liu et al., 2023): To imitate the 10-shot setting, we use 10 examples and the default step size of 0.1 to generate the in-context vector.
- IA^3 (Liu et al., 2022): we applied adapters to the $\mathtt{k_{proj}}$, $\mathtt{v_{proj}}$ and $\mathtt{down_{proj}}$ layers of the network. Specifically, the IA^3 vectors were multiplied with the input to the $\mathtt{down_{proj}}$ layer to scale the activations accordingly.
- Soft Prompt (Lester et al., 2021): We initialized the first token with the task description, e.g., 'the French translation of this word', and fine-tuned eight additional virtual tokens with this initial prompt.
- LoRA (Hu et al., 2021): We fine-tuned a rank-4 matrix, introducing an additional 53,248 parameters to the model.

[1]`https://github.com/shengliu66/ICV.git`

Table 6 summarizes the performance of these baselines on a Rule Understanding task. Our method consistently outperforms the baseline approaches across multiple tasks, demonstrating its robustness in low-data scenarios.

Table 6: Comparison with intervention baseline and finetuning baselines. We use Exact Match to measure performance on each task. Higher values are better.

Method	Eng-Fr	Eng-Ger	Antonym
ICV	0.396	0.423	0.008
IA^3	0.521	0.385	0.321
Soft Prompt	0.710	0.221	0.314
LoRA	0.681	0.606	0.427
Ours	**0.795**	**0.620**	**0.514**

5 Conclusions

In this paper, we propose and showcase an effective approach using layer-wise additive activation interventions to steer the output of LMs. Our approach effectively enhances the model performance by optimizing an intervention vector to minimize the mismatch between the intervened decoding output and the desired target output in the training data. Additionally, incorporating both lasso and group lasso regularizations addresses overfitting and promotes sparsity in activation heads, ensuring efficient interventions. Our evaluations on the rule understanding task and the opinion dynamic task demonstrate that this method significantly improves the performance of pre-trained LMs across various tasks, outperforming existing intervention techniques.

6 Limitations

The main limitation of our approach is that we require access to the model's activations. However, this limitation is relevant for *any* activation intervention method in the literature, including Li et al. (2024b) and Todd et al. (2023), due to the nature of the approach. In this paper, we have shown that our interventions are effective in the Llama3-8b model, and we expect that the intervention will also be effective in larger models such as Llama3-70b.

Although we use interventions to steer the output to adapt to tasks, it is foreseeable that these techniques can be used for possibly unethical purposes, such as generating untruthful or toxic texts. Thus, we strongly recommend studying possible defenses for these problems.

Acknowledgments. Viet Anh Nguyen gratefully acknowledges the generous support from the UGC Early Career Scheme Grant 24210924 and the CUHK's Improvement on Competitiveness in Hiring New Faculties Funding Scheme. Binh Nguyen is supported by NUS Start-up Grant A-0004595-00-00.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2022. What learning algorithm is in-context learning? Investigations with linear models. In *The Eleventh International Conference on Learning Representations*.

AI Anthropic. 2024. The Claude 3 model family: Opus, Sonnet, Haiku. *Claude-3 Model Card*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. Why can gpt learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4005–4019.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9318–9333, Singapore. Association for Computational Linguistics.

Evan Hernandez, Belinda Z Li, and Jacob Andreas. 2023a. Inspecting and editing knowledge representations in language models. *arXiv preprint arXiv:2304.00740*.

Evan Hernandez, Belinda Z Li, and Jacob Andreas. 2023b. Measuring and manipulating knowledge representations in language models. *arXiv preprint arXiv:2304.00740*.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. 2024. Linearity of relation decoding in transformer language models. In *Proceedings of the 2024 International Conference on Learning Representations*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Chonghe Jiang, Bao Nguyen, Anthony Man-Cho So, and Viet Anh Nguyen. 2025. Probe-free low-rank activation intervention. *Preprint*, arXiv:2502.04043.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Dongfang Li, Zhenyu Liu, Xinshuo Hu, Zetian Sun, Baotian Hu, and Min Zhang. 2024a. In-context learning state vector with inner and momentum optimization. *arXiv preprint arXiv:2404.11225*.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024b. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Sheng Liu, Lei Xing, and James Zou. 2023. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv preprint arXiv:2311.06668*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

Bao Nguyen, Binh Nguyen, Duy Nguyen, and Viet Anh Nguyen. 2025. Risk-aware distributional intervention policies for language models. *arXiv preprint arXiv:2501.15758*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7.

Michael E Sander, Raja Giryes, Taiji Suzuki, Mathieu Blondel, and Gabriel Peyré. 2024. How do transformers perform in-context autoregressive learning? *arXiv preprint arXiv:2402.05787*.

Shibani Santurkar, Esin Durmus, Faisal Ladhak, Cinoo Lee, Percy Liang, and Tatsunori Hashimoto. 2023. Whose opinions do language models reflect? *arXiv preprint arXiv:2303.17548*.

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

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Eric Todd, Millicent L Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. 2023. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

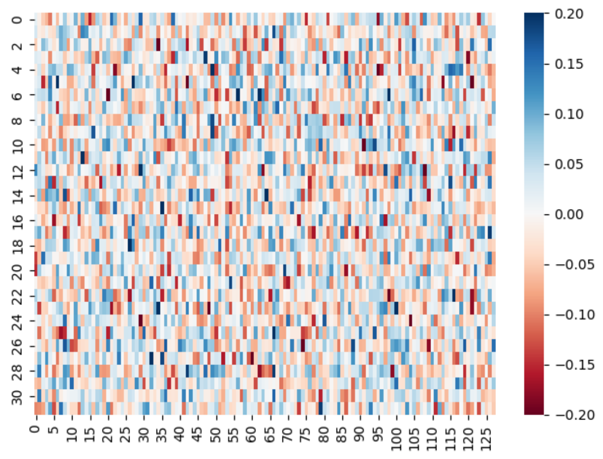
Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. 2023. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2023. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR.

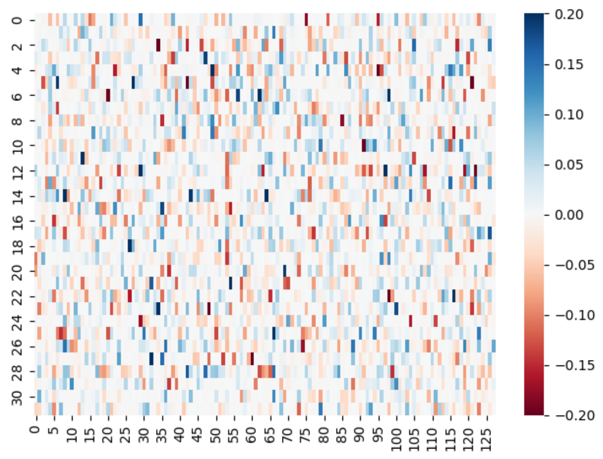
Kang Min Yoo, Junyeob Kim, Hyuhng Joon Kim, Hyunsoo Cho, Hwiyeol Jo, Sang-Woo Lee, Sang-Goo Lee, and Taeuk Kim. 2022. Ground-truth labels matter: A deeper look into input-label demonstrations. In *2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, pages 2422–2437. Association for Computational Linguistics (ACL).

Siyan Zhao, John Dang, and Aditya Grover. 2023. Group preference optimization: Few-shot alignment of large language models. *arXiv preprint arXiv:2310.11523*.

A Effects of Regularization



(a) Without regularization



(b) With group lasso regularization parameter $\lambda = 0.01$ and ℓ_1 regularization parameter $\gamma = 0.01$.

Figure 2: Intervened vector values across LLAMA3-8B attention heads (row-wise, from 1-32). Adding regularization promotes sparsity with the intervened values and desirable properties following previous empirical observations.

B Datasets

The task descriptions of the rule understanding experiments are as follows:

- **Antonym**: Given an English word, generate an English word with the opposite meaning.
- **Synonym**: Given an English word, generate an English word with the same meaning.
- **English-French**: Given an English word, generate the equivalent word in French.
- **English-German**: Given an English word, generate the equivalent word in German.

C Prompts to measure GPT-Eval metric

In this section, we provide the prompts to ask GPT-4 to confirm the input-prediction pair for each dataset in the Rule Understanding task.

- **Antonym**: Answer 0 if what I say is wrong and 1 if it is correct. “input” is an antonym of “prediction”.
- **Synonym**: Answer 0 if what I say is wrong and 1 if it is correct. “input” is a synonym of “prediction”.
- **English-French**: Answer 0 if what I say is wrong and 1 if it is correct. “input” translated to French is “prediction”.
- **English-German**: Answer 0 if what I say is wrong and 1 if it is correct. “input” translated to German is “prediction”.

It is worth noting that “input” and “prediction” are placeholders and should be replaced with the actual input-prediction pair.