Householder Pseudo-Rotation: A Novel Approach to Activation Editing in LLMs with Direction-Magnitude Perspective

Anonymous ACL submission

Abstract

Activation Editing, which involves directly editing the internal representations of large language models (LLMs) to alter their behaviors and achieve desired properties, has emerged as a promising area of research. Existing works primarily treat LLMs' activations as points in space and modify them by adding steering vectors. However, this approach is limited in its ability to achieve greater performance improvement while maintaining the necessary consistency of activation magnitudes. To overcome these issues, we propose a novel editing method that views activations in terms of their directions and magnitudes. Our method, named *Householder Pseudo-Rotation* (HPR), mimics the rotation transformation, thus preserving activation norms and resulting in an improved performance on various safety benchmarks.

1 Introduction

Building upon the paradigm of pre-training language models on large corpora of raw text using next-sentence-prediction objective (Radford and Narasimhan, 2018; Radford et al., 2019), Large Language Models (LLMs) research has taken a big leap and become an essential asset of AI in recent years. The latest LLMs can exhibit phenomenal fluency and reasoning capability, excel in numerous NLP benchmarks, while also aligning to human intent (Wei et al., 2022; Ouyang et al., 2022; Touvron et al., 2023a; Jiang et al., 2023; OpenAI, 2024). In the midst of the rapid development of LLMs, efforts to study and control their societal impacts, including issues such as hallucination, bias, and toxicity to name a few, are of the utmost importance. Yet, with their ever-growing size, reaching hundreds of billions of parameters (Brown et al., 2020; Chowdhery et al., 2022), the popular approach for controlling and aligning LLMs via fine-tuning proves to be very challenging and resource-intensive, necessitating the research into alternative solutions to adapt the behaviors of LLMs.

Among various approaches to efficiently adapt LLMs (Lester et al., 2021; Li and Liang, 2021; Hu et al., 2022; Dong et al., 2023; Wan et al., 2024), Activation Editing, also referred to as "Intervention" or "Representation Engineering" in the literature, has shown promising results. Based on the observation that LLMs form an internal "belief" about facts in their activation space even before the responses are generated (Dai et al., 2022; Li et al., 2023b; Burns et al., 2023; Joshi et al., 2024), this approach aims to draw factual knowledge out of the model by directly editing activation vectors at inference time. Most existing works in this area utilize a *steering vector* (Li et al., 2023b; Turner et al., 2023, Rimsky et al., 2024; von Rütte et al., 2024), which can be scaled by a scaling factor and added to the original activation. In doing so, activations are viewed as *points in space* (Figure 1a). Correspondingly, the process of adding a fixed steering vector to activations can be interpreted as moving these points along a vector offset (Mikolov et al., 2013), and the scaling factor tells how far they should be moved.

In an experiment with the activation space, we discover an important property that are maintained by powerful LLMs: activations within the same layer tend to have roughly the same vector norm. We refer to this as the **Magnitude Consistency** property, i.e., Section 4.3. This observation highlights a key limitation of the points-in-space view, where the steering vector approach cannot simultaneously maintain activation magnitude consistency and effective activation editing to achieve greater performance improvement for desired behaviors for LLMs. If the scaling factor is too large, the additive edit might drastically alter the activation norms in each layer, violating the norm consistency property of LLMs. In extreme cases, this change can lead to the generation of complete gibberish, undermining the desired behaviors of the LLM's responses. Conversely, if the scaling factor is set too low to preserve the activation norms, the steering vector may have limited abilities to shift an activation toward new behavior, thus also hindering editing performance for desired behaviors. Moreover, the steering vector approach does not align with the commonly used cosine similarity metric, which emphasizes directional alignment between vectors rather than their absolute positions.

We argue that activation vectors should instead be understood in terms of their directions and magnitudes. We call this the *direction-magnitude* view (Figure 1b). In this regard, the semantic information of activations is reflected in their directions from the origin, while their magnitude represents the intensity of such information. This view also facilitates cosine similarity better since it measures the relationship between activations via the angle between their directions. Furthermore, while the points-in-space view struggles to achieve activation norm consistency, the direction-magnitude view can conveniently interpret the activation space in each layer as a $(d-1)$-dimensional hypersphere centered at the origin. As such, the activations can have a "stable" norm via the sphere's radius.

In this work, we introduce a novel editing method based on the direction-magnitude view. Instead of trying to move points, our method aims to alter a LLM's behavior by rotating activation vectors around the origin to their designated directions (Figure 1b). For example, rotating from untruthful region into truthful region. Usually, computing a matrix for vector rotation is non-trivial, especially in high-dimensional space. Therefore, we propose to relax the problem and resort to an approximated rotation transformation instead (Figure 1c). To this end, we first determine a hyperplane going through the origin that separates the two regions of interest. We then reflect undesirable activations about this hyperplane to make them land on the desirable region. Having an unique hyperplane for each individual activation vector is infeasible computationally as it would cost substantial GPU memory to store them at runtime. We thus learn a global hyperplane separating the activation vectors for each edited layer. Finally, for each reflection of an undersriable activation, we adjust it to the corresponding desired activation. In this way, our solution is more efficient as the adjustment for each activation only involves scalar angles, whose learning is less expensive than a rotation matrix for each edited vector. We name this method *Householder Pseudo-Rotation* (HPR), based on the Householder transformation (Householder, 1958) at its core.

We evaluate our editing method HPR on eliciting truthfulness from LLMs. Experiment results on the TruthfulQA dataset (Lin et al., 2022) demonstrate a significant boost in performance compared to Steering Vector editing. We further show that HPR can improve LLMs' performance for other behavior-related problems, including bias, ethics, and toxicity. Finally, we conduct extensive analysis to provide deeper insights for the advantages of HPR for activation editing.

2 Prerequisites

2.1 Problem Statement

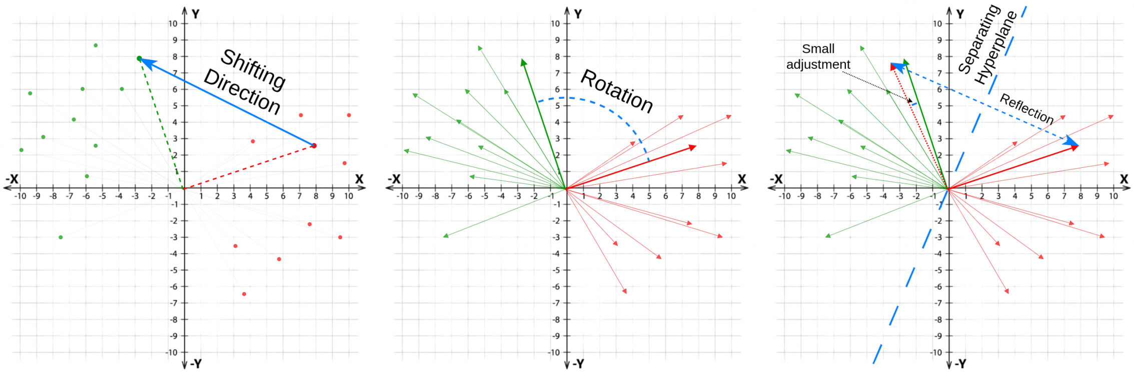
Let $\mathcal{M} = \{\mathcal{M}^{(l)} | 0 \leq l < L\}$ be a L-layers pretrained LLM whose behavior needs to be altered. Assume that the outputs of $\mathcal{M}$ exhibit either of the two contrasting qualities: a positive behavior $\mathbf{p}$ or a negative behavior $\mathbf{n}$. For instance, $\mathbf{p}$ can be truthfulness and $\mathbf{n}$ is untruthfulness. We denote:

- $x_i = \{x_{i,j} | 0 \leq j < S^x\}$: An input sequence of length S^x.
- $y_i^{\mathbf{p}} = \{y_{i,j}^{\mathbf{p}} | 0 \leq j < S^{\mathbf{p}}\}$: The positive (i.e. desirable) output sequence with length $S^{\mathbf{p}}$.
- $y_i^{\mathbf{n}} = \{y_{i,j}^{\mathbf{n}} | 0 \leq j < S^{\mathbf{n}}\}$: The negative (i.e. undesirable) output sequence with length $S^{\mathbf{n}}$.

When the label of the output, i.e. positive or negative, is unknown, we refer to its length as S^y.

In this work, unless specified otherwise, a "vector" is understood as a column vector of size $d \times 1$. Let us further use $a_{i,j}^{\mathbf{p},(l)} \in \mathcal{A}^{\mathbf{p},(l)}$ to denote the d-dimensional positive activation vector at the j^{th} token of the l^{th} layer in $\mathcal{M}$, where $\mathcal{A}^{\mathbf{p},(l)} \subset \mathbb{R}^d$ is the positive region in the activation space of $\mathcal{M}^{(l)}$. Similarly, the corresponding negative activation is denoted as $a_{i,j}^{\mathbf{n},(l)} \in \mathcal{A}^{\mathbf{n},(l)}$. These are obtained by forwarding the concatenation of the input and the corresponding output sequence, i.e. $x_i \| y_i^{\mathbf{p}}$ or $x_i \| y_i^{\mathbf{n}}$, through $\mathcal{M}$. Since the question part x_i is the same for each data pair, we only use the activation vectors at the token positions of the responses. Without loss of generality, we omit the layer notation (l) and the quality notation ($\mathbf{p}$ or $\mathbf{n}$) when referring to an arbitrary item.

The general framework of Activation Editing utilizes an editing function $f(\cdot|\theta)$ with parameter θ for activation vectors $a_{i,j}$ such that $f(a_{i,j}|\theta) \in \mathcal{A}^{\mathbf{p}}$. The design of an Activation Editing method can thus be broken down to the the design of such a function and how to find the optimal θ. For example, in Steering Vector methods (Li et al., 2023b),



(a) Points-in-space view (b) Direction-magnitude view (c) Our approach

Figure 1: Comparison of points-in-space view (a) and direction-magnitude view (b). Positive activations are colored green and negative activations are colored red. The editing methods are depicted in in blue. Our proposed method (c) approximates the rotation transformation by first reflecting negative activations through a learned separating hyperplane and then adjusting the reflections to reach the right angle.

the editing function is a simple vector addition: $f(a_{i,j}|\theta) = a_{i,j} + \alpha\theta$ where α is a hyperparameter for scaling factor.

2.2 Householder Transformation

The idea of Householder transformation stemmed from an important lemma in Householder (1958) which stated: For any vector $a \neq 0$, and any unit vector v, there exists a unit vector u such that:

$$(I - 2uu^T)a = \|a\|v \tag{1}$$

In this case, $\|a\|v$ is the reflection of a about a hyperplane which passes through the origin and has u as its normal vector. Since v is a unit vector, a and $\|a\|v$ have the same vector norm. Therefore, we can extend the problem to a more general case: For any pair of vectors (a, b) of the same magnitude, it is possible to find a vector $c \neq 0$ such that:

$$b = (I - \frac{2cc^T}{c^T c})a \tag{2}$$

The orthogonal matrix $H = (I - \frac{2cc^T}{c^T c})$ is called the *Householder Matrix*.

3 Householder Pseudo-Rotation (HPR)

As discussed in the introduction, our goal is to find an editing function f to alter the behavior of LLMs that can: 1) transform any vector in the activation spaces into one invoking positive behavior; 2) closely mimic the rotation transformation to preserve norm of the activations. The usual calculation of a rotation matrix between two d-dimensional vectors consists of several computationally expensive steps such as the Gram-Schmidt process, whose complexity is $\mathcal{O}(d^3)$. The Householder transformation (Equation 2) can be a cheaper alternative since it also retains the vector norm. However, in the context of Activation Editing, having a Householder matrix of size $d \times d$ for each activation vector would introduce too much extra data to be stored on GPU RAM, thus limiting applicability.

To alleviate these problems, we propose *Householder Pseudo-Rotation* (HPR), a pseudo-rotation method that reflect negative activations in each layer about a global separating hyperplane and then adjust the resulting vectors to achieve the desired angle. The original problem is essentially broken down into two sub-problems: finding the separating hyperplane, and finding the rotating angle. We tackle them by incorporating into each edited layer a **linear probe** and an **angle prediction** module.

3.1 Linear Probe

In the first step, we train a linear probe to discriminate the positive and negative activations of LLMs in each layer. The non-trivial accuracy of this probe, as can be seen in Figure 2, suggests that it can effectively form a separating hyperplane between the positive and negative regions, and its weight vector serves as the normal vector of this hyperplane. We can then utilize the Householder matrix corresponding to this hyperplane as a means to reflect activations from one region to the other.

More concretely, the linear probe corresponding to a LLM layer can be defined as:

$$f_{probe}(a, \theta_{probe}) = \sigma(\theta_{probe}^T a) \tag{3}$$

where $\sigma(\cdot)$ denotes the sigmoid function and θ_{probe}

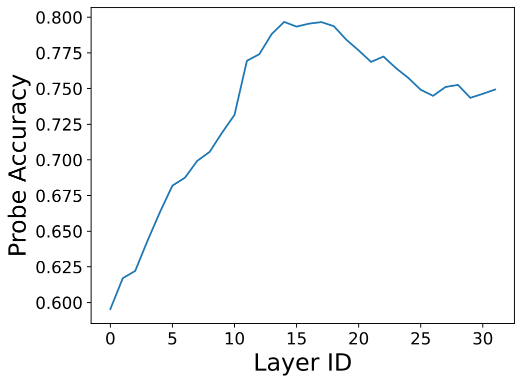


Figure 2: Probe accuracy of HPR-edited Llama2-7B-Chat on TruthfulQA. A linear probe is trained for each layer using positive-negative pairs of the training data and then evaluated on the validation data.

is the weight vector of the probe. Readers may notice that Equation 3 resembles a linear feedforward layer with no bias term. This is to ensure that the normal vector of the separating hyperplane passes through the origin, consistent with the direction-magnitude view.

At inference time, the probe weight vector is used to calculate a Householder matrix.

$$H = I - \frac{2\theta_{probe}\theta_{probe}^T}{\theta_{probe}^T\theta_{probe}} \tag{4}$$

The linear probe is trained using the Binary Cross Entropy (BCE) loss.

$$\mathcal{L}_{probe} = \frac{1}{NS^y}\sum_{i=1}^{N}\sum_{j=1}^{S^y}\Big[BCE(\sigma(\theta_{probe}^T a_{i,j}^{\mathbf{p}}), 1) + BCE(\sigma(\theta_{probe}^T a_{i,j}^{\mathbf{n}}), 0)\Big] \tag{5}$$

3.2 Angle Prediction

Given the separating hyperplane for a layer, we seek to predict a rotating angle that helps transform the reflection of each negative activation into the desirable positive activation. As such, our key assumption considers the desirable positive activation vector to lie on the 2-D plane formed by the original negative activation and its reflection, allowing us to efficiently perform the rotation of the negative activation vector. To this end, we employ a feedforward neural network MLP to predict the rotating angle $f_{angle}(a, \theta_{angle})$ for an input vector a:

$$f_{angle}(a, \theta_{angle}) = \pi \times \sigma(MLP(a, \theta_{angle})) \tag{6}$$

where θ_{angle} represents the model parameters. The output of f_{angle} is a scalar value in the range $[0, \pi]$ radians.

Among several possible implementations, given a negative activation $a_{i,j}^{\mathbf{n}}$, we train f_{angle} to predict the angle between the corresponding desired positive activation $a_{i,j}^{\mathbf{p}}$ and $a_{i,j}^{\mathbf{n}}$ for rotation. In contrast, if the input vector is a positive activation $a_{i,j}^{\mathbf{p}}$, f_{angle} should return zero (i.e., no rotation). Our training loss for f_{angle} is thus:

$$g(a_{i,j}^{\mathbf{p}}, a_{i,j}^{\mathbf{n}}) = \arccos\left(\frac{(a_{i,j}^{\mathbf{p}})^T a_{i,j}^{\mathbf{n}}}{\|a_{i,j}^{\mathbf{p}}\|\|a_{i,j}^{\mathbf{n}}\|}\right) \tag{7}$$

$$\mathcal{L}_{angle} = \frac{1}{NS^y}\sum_{i=1}^{N}\sum_{j=1}^{S^y}\Big[\Big(f_{angle}(a_{i,j}^{\mathbf{n}}, \theta_{angle}) - g(a_{i,j}^{\mathbf{p}}, a_{i,j}^{\mathbf{n}})\Big)^2 + f_{angle}(a_{i,j}^{\mathbf{p}}, \theta_{angle})^2\Big] \tag{8}$$

where $g(\cdot, \cdot)$ computes the angle between two vectors using the inverse of cosine $\arccos$. For training, the linear probe and angle prediction modules are optimized jointly via: $\mathcal{L} = \mathcal{L}_{probe} + \mathcal{L}_{angle}$.

3.3 Computing the Final Activation

At inference time, let a be an activation in a layer of LLMs, we first forward it through the corresponding linear probe and the angle prediction module.

$$\hat{\sigma} = \lfloor f_{probe}(a, \theta_{probe}) \rceil \tag{9}$$

$$\gamma_1 = f_{angle}(a, \theta_{angle}) \tag{10}$$

$\hat{\sigma}$ is rounded to the nearest integer, 0 or 1 to be specific, and predict whether the given activation a is positive or negative. If a is predicted as a negative activation, we edit it by first reflecting a about the separating hyperplane θ_{probe} to obtain the reflected vector $\dot{a}$ in the positive region. Afterward, we calculate a new activation by rotating a within the 2-D plan formed by a and $\dot{a}$ by an angle of γ_1 radians. The resulting vector $\hat{a}$ will serve as our predicted positive activation for a.

In particular, a Householder matrix is computed from the probe's weight following Equation 4. With this we can reflect a to obtain the reflected activation $\dot{a}$ and the angle γ_2 between a and $\dot{a}$:

$$\dot{a} = Ha, \; \gamma_2 = g(\dot{a}, a) \tag{11}$$

The Householder reflection and rotation transformation preserve vector norm. Thus, the norm of

a, $\dot{a}$ and $\hat{a}$ are identical. Combined with the computed angles γ_1 and γ_2, the rotation on 2-D plane to obtain the predicted positive activation $\hat{a}$ can be calculated via a and $\dot{a}$ as follows:

$$\hat{a} = \frac{\sin(\gamma_1)}{\sin(\gamma_2)}\dot{a} + \frac{\sin(\gamma_2 - \gamma_1)}{\sin(\gamma_2)}a \qquad (12)$$

The proof for Equation 12 is in Appendix A.

Finally, HPR's editing function can be written as follows: $f(a|\theta_{probe}, \theta_{angle}) = \hat{\sigma}a + (1 - \hat{\sigma})\hat{a}$.

4 Experiments

4.1 Experimental Setup

Datasets: Following previous activation editing work (Li et al., 2023b), we first evaluate the models on the TruthfulQA dataset (Lin et al., 2022). TruthfulQA includes 817 questions, each of which is coupled with factually correct and incorrect answers. We split the dataset into subsets with ratios 45 / 5 / 50 for training, validation and testing respectively.

Aside from truthfulness, we also demonstrate the proposed method on other societal issues related to LLMs, more specifically, bias, ethics, and toxicity. These are reflected in BigBench's Bias Benchmark for QA (BBQ) (Srivastava et al., 2023; Parrish et al., 2022), BigBench's Simple Ethical Questions (SEQ), and Toxigen (Hartvigsen et al., 2022), respectively. These datasets are already split into a training set and a validation set. We use the validation sets to test the models, while splitting their training sets further with ratios 90 / 10 to make new training and validation sets.

All four datasets are multiple choice tasks, thus the main evaluation metrics is multiple choice accuracy. The correct and incorrect answers for each question can be used handily to create $y_i^{\mathbf{p}}$ / $y_i^{\mathbf{n}}$ pairs.

Base Models and Baselines: We conduct experiments with three recent popular open source LLMs: `Llama2-7B-Chat` (Touvron et al., 2023b), `Mistral-7B-Instruct` (Jiang et al., 2023), and `Llama3-8B-Instruct` (AI@Meta, 2024). We compare our method with the following baselines:

- **Base**: The unaltered base LLMs.
- **LoRA** (Hu et al., 2022): We fine-tune the base LLM with LoRA adapter on the same training data as activation editing methods for a fair comparison.
- **Diff**: Given a positive or negative activation $a_{i,j}$, this baseline employs a feedforward network to directly predict the difference vector $a_{i,j}^{\mathbf{p}} - a_{i,j}$ with the corresponding positive activation $a_{i,j}^{\mathbf{p}}$. At inference time, we utilize the sum of the original activation vector $a_{i,j}$ and its predicted difference vector to obtain the predicted positive activation.
- **ITI** (Li et al., 2023b): A representative Activation Editing method for the aforementioned points-in-space view that shifts the outputs of a set of attention heads in each layer by a fixed steering direction. The steering vector in ITI is the Mass Mean Shift vector (i.e. the difference between the centers of the positive and negative regions) of activations in training data (i.e., not learnable). We employ the source code published by the original authors. However, their code is implemented only for Llama models and TruthfulQA dataset specifically. Thus we only report results of ITI with `Llama2-7B-Chat` and `Llama3-8B-Instruct` on TruthfulQA.

Evaluation Framework: We utilize EleutherAI's Language Model Evaluation Harness (Gao et al., 2023), a reliable evaluation framework used in numerous works including HuggingFace's Open LLM Leaderboard. This framework supports automatic evaluation of various benchmark datasets for LLM. Our experiments involve evaluating mulitple choice accuracy on various datasets. This is done by calculating the aggregated log-likelihood of each choice given the input prompt and then pick the top one.

Hyperparameters: In our model, the linear probe is a vector of the same dimensions as the LLMs' hidden dimensions. The angle prediction module is a feedforward neural network with 4 layers and one output unit. We train each module for 5 epochs with batch size 16, AdamW optimizer (Loshchilov and Hutter, 2019), learning rate 5×10^{-4}, cosine learning rate scheduler and warmup. For editing, we apply HPR to the top $k = 5$ layers with the highest probe accuracy. Appendix C presents model performance with different values of k. We also provide a reproducibility checklist in Appendix D.

4.2 Results

TruthfulQA: Table 1 presents the performance of our method HPR and the baselines on TruthfulQA. The results include both MC1, multiple choices with only 1 correct answer per question, and MC2, which is multiple choices with more than 1 correct answer for each question. The first observation from the table is that fine-tuning LLMs with LoRA does not produce consistent performance improvement for TruthfulQA over different models. In contrast, activation editing methods, i.e., ITI and HPR, consistently outperform the base LLM

Method	Llama2 MC1	Llama2 MC2	Llama3 MC1	Llama3 MC2	Mistral MC1	Mistral MC2
Base	29.58 ± 2.26	43.00 ± 2.17	36.43 ± 2.38	50.73 ± 2.13	54.28 ± 2.47	67.45 ± 2.14
LoRA	29.10 ± 2.25	43.40 ± 2.15	38.63 ± 2.41	55.84 ± 2.11	54.77 ± 2.46	70.45 ± 2.06
Diff	33.74 ± 2.47	48.87 ± 2.24	29.34 ± 2.25	52.53 ± 2.25	50.61 ± 2.48	68.68 ± 2.11
ITI	33.74 ± 2.34	50.67 ± 2.20	39.85 ± 2.42	56.58 ± 2.18	- -	- -
HPR	**51.83** ± 2.47	**70.95** ± 2.12	**52.32** ± 2.47	**71.70** ± 2.13	**55.01** ± 2.46	**72.14** ± 2.07
-AnglePred	30.07 ± 2.27	43.36 ± 2.18	35.94 ± 2.375	49.77 ± 2.12	53.79 ± 2.47	67.31 ± 2.14

Table 1: Model performance (in %) on TruthfulQA multiple choice tasks. ± indicates standard errors.

Model	BBQ	SEQ	Toxigen
Llama2-7B-Chat	33.27	21.74	51.38
+ HPR	**38.38**	**60.87**	**52.34**
Llama3-8B-Instruct	60.44	47.83	45.32
+ HPR	**67.10**	**52.17**	**46.81**
Mistral-7B-Instruct	61.62	69.57	55.00
+ HPR	**73.24**	**86.96**	**61.60**

Table 2: HPR performance for bias, ethics, and toxicity. We report multiple choice accuracy in %.

models, achieving greater margins than LoRA fine-tuning. It thus highlights the effectiveness of activation editing for altering LLMs for desirable behaviors. When comparing Diff and ITI, ITI's superior overall performance indicates that learning negative-positive difference vectors for activations, as done in Diff, is ineffective and cannot ensure optimal aligning performance for LLMs. Most importantly, the proposed model HPR is significantly better than all the baselines with substantial margins across all base LLMs. These results clearly testify to the advantages of HPR, demonstrating the benefits of our new direction-magnitude view for activation editing with reflection and rotation for negative activation transformation.

Ablation Study: The last row in Table 1 further shows the performance of HPR when the angle prediction module is excluded from the full model. As can be seen, this exclusion leads to significant performance drops across all base LLMs for HPR, thereby justifying the importance of angle prediction to adjust reflected activations for our model. We also note that the linear probe module cannot be removed from HPR for ablation study as it is essential for finding the positive-negative separating hyperplane and rotating plane in our model. Finally, the superior performance of HPR for different LLMs confirms the advantages of our assumption on the shared 2-D plane of a, $\dot{a}$, and $\hat{a}$.

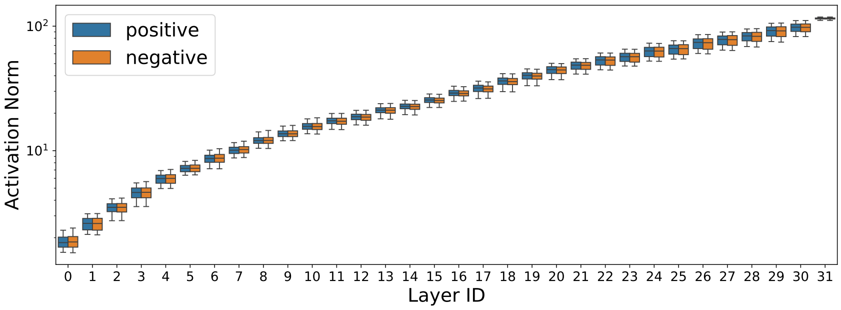
BBQ, SEQ, and Toxigen: To further illustrate the effectiveness of HPR in eliciting desirable behavior, Table 2 shows HPR's performance on the BBQ, SEQ, and Toxigen datasets. These datasets evaluate the abilities of LLMs to generate unbiased (BBQ), ethically acceptable (SEQ), and non-toxic (Toxigen) responses. Across various base LLMs, incorporating HPR can significantly enhance performance on all these datasets. These results highlight the benefits of HPR in improving important safety criteria for LLMs, leading to unbiased, ethical, and non-toxic responses for responsible models.

4.3 Analysis of Activation Space

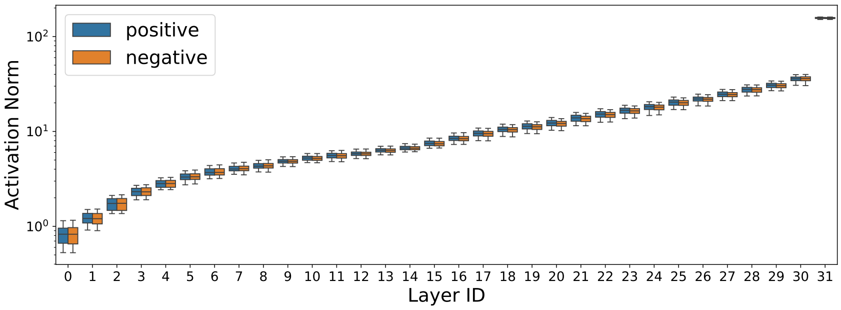
In this section, we examine the activation norms of the selected LLMs to gain a better understanding of the activation space. We first look into base LLMs. In Figure 3 we plot the activation norms in each layer, positive vectors and negative vectors side-by-side. From these box plots, we can observe the **Magnitude Consistency** property: activations of the same layer have roughly the same vector norm for all considered LLMs. This observation holds true regardless of the activations being positive or negative. The gap between the whiskers of each box is very narrow, suggesting a low variance. This gap seems to become narrower for more powerful models, as can be seen in Figures 3b, 3c for LLaMa3 and Mistral. Due to this universality, we consider activation norm consistency as a necessary condition that should be maintained by editing methods to achieve desired LLMs.

Considering this property, we demonstrate how the steering vector approach in ITI (Li et al., 2023b) struggles to simultaneously maintain activation magnitude consistency and effectively alter their activations for greater improvement on desired behaviors. First, Figures 4a and 4b show the distributions of activation norms in LLMs before and after editing with ITI. In Figure 4a, the scaling factor α is set to 15 (i.e., ITI$_{15}$), as recommended in the original ITI paper, while in Figure 4b, α is set to 200 (i.e., ITI$_{200}$). As can be seen, the smaller scaling factor $\alpha = 15$ in ITI$_{15}$ leads to less divergence of activation norms than ITI$_{200}$ from the original LLMs (i.e., better preservation of activation norms).

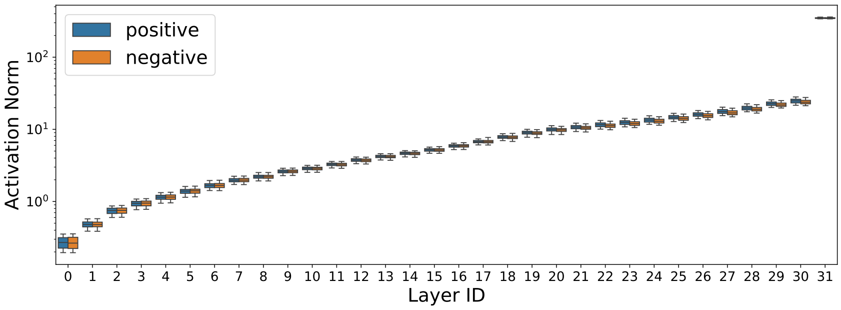
What is the implication of such slight norm divergence from base LLMs for ITI? In Table 3, we present the behavior shift rates of ITI$_{15}$ and ITI$_{200}$



(a) `Llama2-7B-Chat`

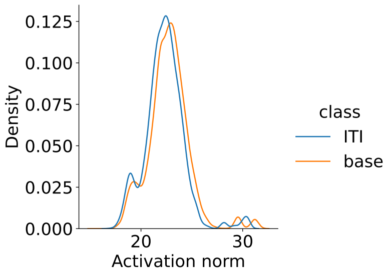


(b) `Llama3-8B-Instruct`

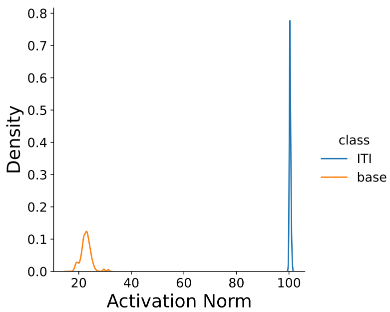


(c) `Mistral-7B-Instruct`

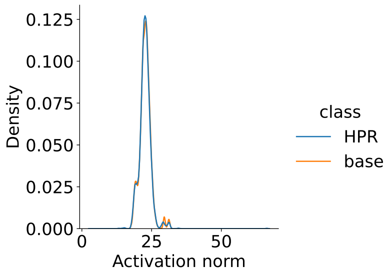
Figure 3: The activation norms in log_{10} scale across 32 transformer blocks of three popular LLMs. Each box plot represents the norm distribution in a layer of the LLMs.



(a) ITI, $\alpha = 15$



(b) ITI, $\alpha = 200$



(c) HPR

Figure 4: Activation norm distributions of the 14^{th} layer of `Llama2` before and after being edited. We use the 14^{th} layer as it has the highest probe accuracy in Figure 2. Similar trends can be seen for other layers and models.

compared to the original `Llama2-7B-Chat` model in TruthfulQA. Specifically, we show how often each editing method can flip the LLM's predictions of examples from true to false and vice versa. From the table, we observe that the slight divergence of activation norms in ITI_{15} results in a more limited ability to change the base model's behavior, with a behavior shift rate of only 8.56% compared to 34.23% for ITI_{200}. As the behavior shift rate is the upper bound of the overall performance improvement in TruthfulQA for ITI, this limited ability to alter LLM behavior will hinder further improvement with a small scaling factor in ITI.

Model	False to True↑	True to False↓	Remains True↑	Remains False↓	Overall Acc.↑
Base model	-	-	29.58	70.42	29.58
ITI, $\alpha = 15$	6.36	**2.20**	**27.38**	64.06	33.74
ITI, $\alpha = 200$	14.18	20.05	9.54	56.23	23.72
HPR	**28.85**	6.60	22.98	**41.56**	**51.83**

Table 3: Behavior shift rate (in %) of activation editing methods on TruthfulQA MC1 task compared to the base model. The base LLM is `Llama2-7B-Chat`. ↑ means greater is better and ↓ means lower is better.

Furthermore, with a larger scaling factor of $\alpha = 200$, the greater behavior shift rate in ITI_{200} might suggest that ITI_{200} can better boost truthful performance for ITI. However, a closer examination at Table 3 reveals that the significant norm change in ITI_{200} promotes both "good" False-to-True and "bad" True-to-False prediction flips from the base LLM. While ITI_{200} is more effective at correcting false predictions, increasing the "False-to-True" flip rate from 6.36% in ITI_{15} to 14.18%, it also introduces more "bad" edits, changing 20.05% of examples with True predictions in the base LLM to False, compared to just 2.2% for ITI_{15}. Overall, the bad edits significantly dominate the good edits in the ITI model with more extensive norm change, ITI_{200}, leading to its poorer performance in producing truthful responses. To this end, our analysis demonstrates the fundamental limitations of steering vector approach on boosting truthful performance for LLMs, regardless of efforts to tune the scaling factor.

In contrast, Figure 4c highlights the inherent ability of the proposed HPR method to preserve activation norms through its activation rotation mechanisms. In addition, HPR offers substantially stronger editing capabilities for achieving desired behaviors in LLMs as shown in Table 3. It significantly improves the False-to-True prediction flip rate while minimizing undesirable True-to-False edits for the base LLM, demonstrating the effectiveness of our method for activation editing.

5 Related Work

Concerning the societal risks of LLMs, various approaches have been explored to control and align their behavior post-pretraining. Unlike resource-intensive methods for LLM alignment such as instruction tuning and reinforcement learning from human feedback (Ouyang et al., 2022; Bai et al., 2022), our work falls into the category of resource-efficient methods for controlling LLMs. Several resource-efficient approaches exist in this area. First, parameter-efficient fine-tuning aims to fine-tune LLMs with safety data while minimizing the number of learnable parameters, such as prompt-tuning (Lester et al., 2021), prefix-tuning (Li and Liang, 2021), and LoRA (Hu et al., 2022). However, fine-tuning might also compromise the safety of LLMs (Qi et al., 2023). Additionally, model editing attempts to locate and edit model parameters associated with safety issues using minimal invasions for efficiency (Meng et al., 2022; Ilharco et al., 2023). However, model editing might impact the general robustness of the models (Brown et al., 2023). Our work belongs to the third direction for efficient LLM control, i.e., activation editing, which involves editing their inner representations towards a desired behavior at inference time (Li et al., 2023a; Hernandez et al., 2023) and can be traced back to plug-and-play controllable text generation research (Dathathri et al., 2020; Krause et al., 2021). Accordingly, activation editing can preserve the pretrained LLMs to achieve better robustness while still offering adjustable and minimally invasive benefits.

In one approach to activation editing, Liu et al. (2021), Li et al. (2023c), and Liu et al. (2024) contrast the behavior of an expert and an amateur model. Additionally, vector steering edits inner representations by adding steering vectors (Burns et al., 2023; Li et al., 2023b; Turner et al., 2023; Rimsky et al., 2024; von Rütte et al., 2024). However, none of these work explores the direction-magnitude perspective with activation rotations.

6 Conclusion

This work proposes a new activation editing approach based on the direction-maginitude view. By rotating negative activations instead of adding to them a fixed steering vector, our proposed method effectively addresses the shortcomings of existing work, as evidenced by the improved performance on various benchmarks. Our analyses highlight the magnitude consistency property of LLMs, providing insights into the operations of our editing method. In the future, we plan to extend our research to study how the activation space evolves during fine-tuning and how the proposed method scales to larger models and other architectures.

Limitations

As an empirical study, our work is not without limitations. Acknowledging this, we would like to discuss them as follows:

- Due to limited computational resources, we only employ open-source LLMs of size 7-8B parameters. However, we show that the proposed method can effectively alter the behaviors of LLMs for diverse base models and tasks. We leave further research on the scalability of HPR as well as its impact on models of larger sizes for future work.

- Although our method exhibits strong editing performance for desired behaviors, the method itself, like all other Activation Editing methods, only serves to alter LLMs' behavior and elicit knowledge embedded into them during pre-training, not to introduce any new knowledge. Combining activation editing with knowledge updates can be a promising area for future research.

- Though HPR outperforms our baselines by a significant margin (i.e., over 15% better than the second best baseline ITI with `LLama3`), there is still room for improvement. For example, the best MC1 accuracy of HPR on TruthfulQA is currently only about 55% with the base model `Mistral`. As such, future work can expand our method to develop stronger alignment methods and address safety concerns for LLMs.

- HPR has been shown to perform well on a variety of behavior-related tasks. However, our experiments involves only English data, thus not fully reflecting the capability of the proposed method for multilingual LLMs and data. Future work can explore the effectiveness of our method for multilingual settings, aiming for more robust methods for diverse languages and multilingual LLMs.

Ethics Statement

Our work utilize open-source LLMs, i.e., `Llama2-7B-Chat` (Touvron et al., 2023b), `Mistral-7B-Instruct` (Jiang et al., 2023), and `Llama3-8B-Instruct` (AI@Meta, 2024), as the base models, thus potentially inheriting their inherent societal issues like bias, hallucination, privacy, etc. Simultaneously, we propose a novel activation editing method aiming at altering LLMs' behaviour for the better, contributing to the on-going efforts to advance LLM safety. As activation and model editing for LLMs has been studied in recent published work (Li et al., 2023b; Liu et al., 2021; Ilharco et al., 2023), we do not believe our work poses greater societal risks than such studies and open-source LLMs. Finally, we confirm that we follow all the ethical guideline from ACL ARR to the best of our knowledge when performing this research.

References

AI@Meta. 2024. Llama 3 model card.

Yuntao Bai, Andy Jones, and et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *Preprint*, arXiv:2204.05862.

Davis Brown, Charles Godfrey, Cody Nizinski, Jonathan Tu, and Henry Kvinge. 2023. Robustness of edited neural networks. In *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2023. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny

Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *Preprint*, arXiv:2204.02311.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning. *Preprint*, arXiv:2301.00234.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. ToxiGen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3309–3326, Dublin, Ireland. Association for Computational Linguistics.

Evan Hernandez, Belinda Z. Li, and Jacob Andreas. 2023. Inspecting and editing knowledge representations in language models. In *Arxiv*.

Alston S. Householder. 1958. Unitary triangularization of a nonsymmetric matrix. *J. ACM*, 5(4):339–342.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Nitish Joshi, Javier Rando, Abulhair Saparov, Najoung Kim, and He He. 2024. Personas as a way to model truthfulness in language models. *Preprint*, arXiv:2310.18168.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. GeDi: Generative discriminator guided sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4929–4952, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023a. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations*.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023b. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems*, volume 36, pages 41451–41530. Curran Associates, Inc.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023c. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12286–12312, Toronto, Canada. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. 2024. Tuning language models by proxy. *Preprint*, arXiv:2401.08565.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706, Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. In *Advances in Neural Information Processing Systems*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Alicia Parrish, Angelica Chen, Nikita Nangia, Vishakh Padmakumar, Jason Phang, Jana Thompson, Phu Mon Htut, and Samuel Bowman. 2022. BBQ: A hand-built bias benchmark for question answering. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2086–2105, Dublin, Ireland. Association for Computational Linguistics.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. 2023. Fine-tuning aligned language models compromises safety, even when users do not intend to! *Preprint*, arXiv:2310.03693.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. 2024. Steering llama 2 via contrastive activation addition. *Preprint*, arXiv:2312.06681.

Aarohi Srivastava, Abhinav Rastogi, and et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

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

Dimitri von Rütte, Sotiris Anagnostidis, Gregor Bachmann, and Thomas Hofmann. 2024. A language model's guide through latent space. *Preprint*, arXiv:2402.14433.

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, and Mi Zhang. 2024. Efficient large language models: A survey. *Transactions on Machine Learning Research*. Survey Certification.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

A Derivation of Equation 12

In this section we describe the process of deriving Equation 12. Since the rotation of interest occurs on a 2-D plane, and $\|\hat{a}\| = \|\dot{a}\| = \|a\|$, we can calculate $\hat{a}$ by combining a and $\dot{a}$. If $\gamma_1 = \gamma_2$, Equation 12 trivially holds: $\hat{a} = \dot{a}$. If not, there are two cases that can occur: $\gamma_1 < \gamma_2$, and $\gamma_1 > \gamma_2$. We illustrate both of them in Figure 5 to make the derivation easier to follow. In this figure, we color the original negative activation a in red, the target positive activation $\hat{a}$ in green, and the intermediate vector $\dot{a}$ in orange.

Say, we have

$$\hat{a} = \beta_1 \dot{a} + \beta_2 a \tag{13}$$

In the first case (Figure 5a), applying the law of sines in trigonometry, we obtain

$$\frac{\|\hat{a}\|}{\sin(\pi - \gamma_2)} = \frac{\beta_1 \|\dot{a}\|}{\sin(\gamma_1)} = \frac{\beta_2 \|a\|}{\sin(\gamma_2 - \gamma_1)} \tag{14}$$

This is equivalent to

$$\frac{1}{\sin(\gamma_2)} = \frac{\beta_1}{\sin(\gamma_1)} = \frac{\beta_2}{\sin(\gamma_2 - \gamma_1)} \tag{15}$$

Thus,

$$\beta_1 = \frac{\sin(\gamma_1)}{\sin(\gamma_2)} \tag{16}$$

$$\beta_2 = \frac{\sin(\gamma_2 - \gamma_1)}{\sin(\gamma_2)} \tag{17}$$

Similarly for the second case (Figure 5b), we have

$$\frac{1}{\sin(\gamma_2)} = \frac{\beta_1}{\sin(\pi - \gamma_1)} = \frac{-\beta_2}{\sin(\gamma_1 - \gamma_2)} \tag{18}$$

$$\implies \frac{1}{\sin(\gamma_2)} = \frac{\beta_1}{\sin(\gamma_1)} = \frac{\beta_2}{\sin(\gamma_2 - \gamma_1)} \tag{19}$$

$$\implies \begin{cases} \beta_1 = \frac{\sin(\gamma_1)}{\sin(\gamma_2)} \\ \beta_2 = \frac{\sin(\gamma_2 - \gamma_1)}{\sin(\gamma_2)} \end{cases} \tag{20}$$

Combining both cases, we arrive at a general formula for calculating the target activation vector:

$$\hat{a} = \frac{\sin(\gamma_1)}{\sin(\gamma_2)} \dot{a} + \frac{\sin(\gamma_2 - \gamma_1)}{\sin(\gamma_2)} a \tag{21}$$

B Additional Details about Experiments

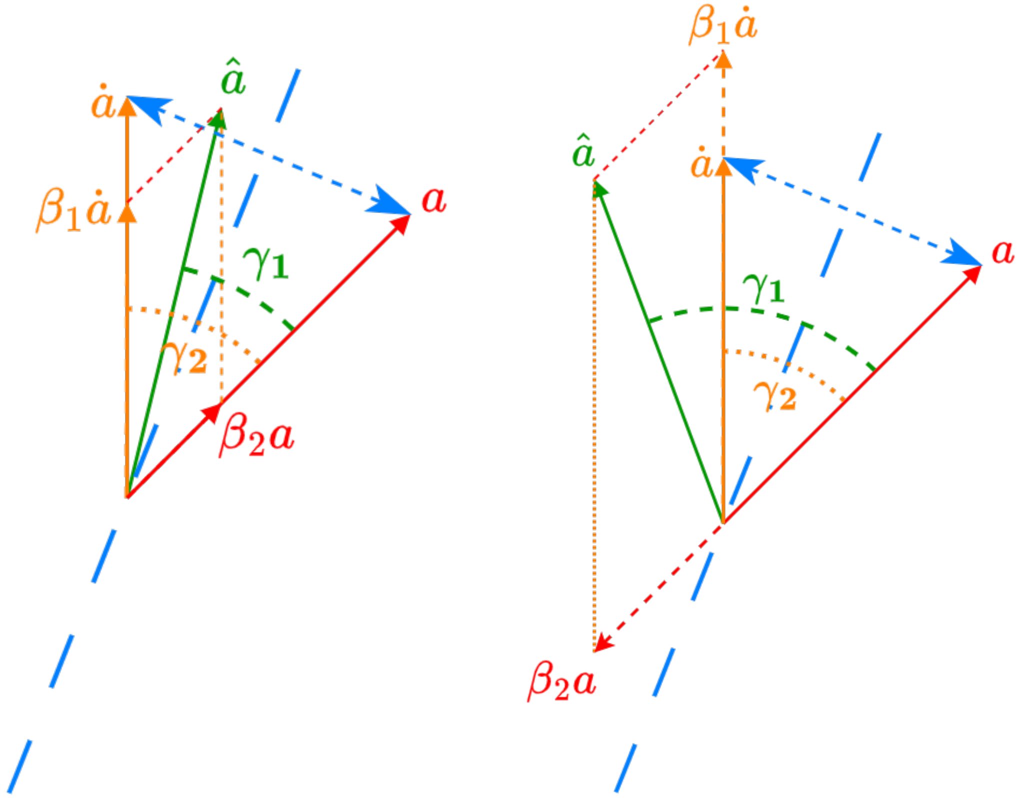
Aside from the major details for the experiments described in Section 4.1, we would like to discuss some additional details here.

- **Training efficieny**: During the training phase, we use $a_{i,j}^{(l),\mathbf{P}}$ / $a_{i,j}^{(l),\mathbf{n}}$ pairs to form the inputs and labels for the linear probe and angle prediction modules in each layer. Generally, these are computed by passing training data samples $x\|y_i^{\mathbf{P}}$ and $x\|y_i^{\mathbf{n}}$ through the model $\mathcal{M}$ and record the activations at each layer and token position. However, since our method does not update the parameters of $\mathcal{M}$, its activation vectors can be treated as constants. Thus, before training we pre-compute all activations on the training data to make a dataset of $a_{i,j}^{(l),\mathbf{P}}$ / $a_{i,j}^{(l),\mathbf{n}}$ pairs for each layer. These can then be used to train the linear probe and angle prediction modules independently of the base model. In this way, the base LLM does not need to be loaded into GPU RAM, saving more space for training the HPR modules.

- **Data splits for TruthfulQA**: The TruthfulQA dataset only has a validation set of 817 examples. We split this dataset into train, validation, and test set with ratios 45 / 5 / 50. We include the specific indices for these data splits with the submission of this paper.

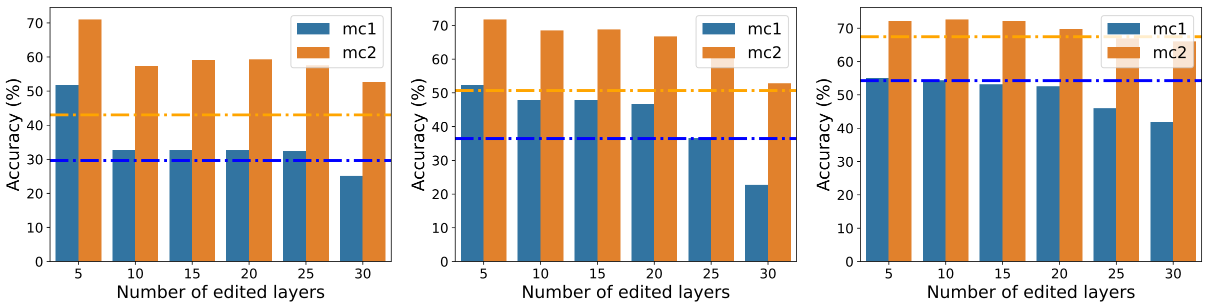
C Evaluating Different Numbers of Editted Layers

Motivated by the varying linear probing accuracy across different layers in LLMs for positive and negative activations in Figure 2, our method HPR choose the top k layers with highest probe accuracy in LLMs for activation editing. Figure 6 illustrates the performance of HPR using different values of k for all the three base LLMs. The bars depict MC1 (blue) and MC2 (orange) accuracy. We also add the performance of the respective base LLM and illustrate them with horizontal lines for comparison. It is clear from the figure that editing only the top 5 layers yields the best performance across models. As we increase the number of edited layers, multiple choice accuracy decreases, even falling below baseline in the case of `Mistral-7B-Instruct`. This can be partly attributed to aggregated error from imperfect linear probes (Figure 2).



(a) First case: $\gamma_1 < \gamma_2$ (b) Second case: $\gamma_1 > \gamma_2$

Figure 5: Illustration of the two cases when rotating vector in 2-D plane.



(a) `Llama2-7B-Chat` (b) `Llama3-8B-Instruct` (c) `Mistral-7B-Instruct`

Figure 6: HPR's performance on TruthfulQA with different numbers of edited layers.

D Reproducibility Checklist

- **Data and source code with specification of all dependencies, including external libraries**: Our source code, along with a README file detailing all configurations, dependencies and external libraries, will be made publicly available upon acceptance of the paper. We utilize public datasets, i.e., TruthfulQA, BBQ, SEQ, and Toxigen. for the experiments. We include the data splits in the submission to facilitate future research on this area.

- **Description of computing infrastructure used**: Experiments were conducted on a single NVIDIA A100 GPU with 40GB of memory. We utilized PyTorch 2.0.1 and the Hugging Face Transformers library (version 4.37.2) for model implementation and training.

- **Average runtime**: Jointly training the linear probe and angle prediction modules for all 32 layers of a $7-8$B model in a single run takes roughly 3 hours.

- **Number of parameters in the model**: We utilized LLMs of sizes 7B and 8B parameters. The computing resources required for these two model sizes are roughly the same.

- **Explanation of evaluation metrics used, with links to code**: We employed EleutherAI's Language Model Evaluation Harness framework (Gao et al., 2023) for evaluation. The metrics of choice is multiple choice accuracy. Please refer to Section 4.1 for more information.

- **The method of choosing hyper-parameter values and the criterion used to select among them**: We performced hyperparameter search to find the optimal value of: The number of each angle prediction module's layers from the list $[1, 2, 3, 4, 5]$; The learning rate from $[1 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 5 \times 10^{-3}]$; The number of edited layers from $[5, 10, 15, 20, 25, 30]$. The selection of the hyper-parameters was based on the linear probe accuracy on the validation set, using a random search.

- **Hyperparameter configurations for best-performing models**: Please refer to Section 4.1.