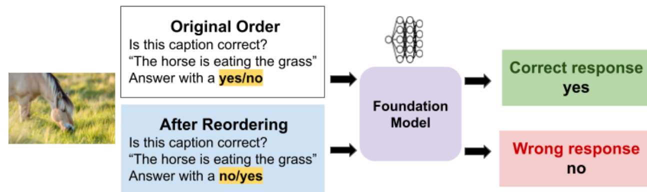
Discovering Bias in Latent Space: An Unsupervised Debiasing Approach

Dyah Adila [*1] **Shuai Zhang** [2] **Boran Han** [2] **Yuyang Wang** [2]

Abstract

The question-answering (QA) capabilities of foundation models are highly sensitive to prompt variations, rendering their performance susceptible to superficial, non-meaning-altering changes. This vulnerability often stems from the model's preference or bias towards specific input characteristics, such as option position or superficial image features in multi-modal settings. We propose to rectify this bias *directly in the model's internal representation*. Our approach, STEERFAIR, finds the bias direction in the model's representation space and steers activation values away from it during inference. Specifically, we exploit the observation that bias often adheres to simple association rules, such as the spurious association between the first option and correctness likelihood. Next, we construct demonstrations of these rules from unlabeled samples and use them to identify the bias directions. We empirically show that STEERFAIR significantly reduces instruction-tuned model performance variance across prompt modifications on three benchmark tasks. Remarkably, our approach surpasses a supervised baseline with 100 labels by an average of 10.86% accuracy points and 12.95 score points and matches the performance with 500 labels.



Model	ScienceQA (2 options)			VGR	
	Original	A	B	yes/no	no/yes
LLaVA	64.99%	68.22%	63.15%	82.29%	66.10%
IDEFICS	61.04%	85.10%	34.11%	40.71%	61.26%
InstructBLIP	61.27%	83.57%	37.66%	56.27%	22.01%

Figure 1. Top: Model predictions are sensitive to prompt order changes. Bottom: Performance of instruction-tuned models on (1) ScienceQA (2 options) in the original order and with golden answers moved to A/B, and (2) Visual Genome Relation (VGR) with prompt variations using "yes/no" and "no/yes"

1. Introduction

Large Language Models (LLMs) and Vision-Language Models (VLMs) show impressive performance on benchmark question-answering tasks, even in some cases outperforming humans (Chowdhery et al., 2023). However, upon closer inspection, their performance appears highly contingent on the input, and can drastically change with minor, superficial modifications. Recent studies have demonstrated this instability – revealing model preferences for specific prompt orderings or biases towards particular answer positions in question-answering scenarios (Zhao et al., 2021; Zheng et al., 2023a; Pezeshkpour & Hruschka, 2023) and unjustly showing preference or bias against specific demographic groups (Caliskan et al., 2017). Even widely-used models like GPT-4 exhibit a bias to the first presented answers when used as an evaluator (Zheng et al., 2023b; Wang et al., 2023). This instability can be traced back to the inherent bias in training data – which can lead models to learn superficial patterns and associations (Torralba & Efros, 2011). For instance, Zhao et al. (2021) discovered that GPT-3 is biased towards generating tokens common in its pretraining distribution. Figure 1 display some examples demonstrating how bias can manifest in large models.

Eliminating this type of instability and bias is challenging. Existing methods either lack effectiveness or are characterized by high costs and inefficiency. Zheng et al. (2023a) and Pezeshkpour & Hruschka (2023) revealed that bias to a certain option position in Multiple Choice Question (MCQ) tasks persists after incorporating in-context examples in the prompt. Instead of mitigating the bias, in-context examples alter the favored position. Solving this issue solely by relying on more data, as in standard fine-tuning or in-context learning, is unlikely unless we can ensure bias-free

Work done during internship at Amazon Web Services [1]Department of Computer Science, University of Wisconsin-Madison [2]Amazon Web Services. Correspondence to: Dyah Adila <adila@wisc.edu>, Shuai Zhang <shuaizs@amazon.com>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

training data, an impractical, if not impossible, task (Liang et al., 2022; Khosla et al., 2012). Post-hoc model intervention during inference has been explored to sidestep data requirements. Calibration methods (Zheng et al., 2023a; Pezeshkpour & Hruschka, 2023) seek to debias output token probabilities to prevent disproportionate allocation of probability mass to specific tokens. However, this approach necessitates multiple inferences for the same sample, creating a computational load that grows exponentially with the number of presented options. Moreover, solely intervening in the output space limits our degree of freedom for intervention. A parallel line of research identifies directions in model's internal representations that correspond to some desirable trait (e.g., factual correctness) (Li et al., 2023; Burns et al., 2022; Liu et al., 2023c), and steer activation values towards this direction. Unfortunately, these works require label supervision, making them prone to adopt data biases when applied to model debiasing.

As such, we ask: *can we mitigate bias directly in the model representation space and do so without labeled data?* We propose SteerFair, an *unsupervised inference-time* intervention method designed for this purpose. Our approach capitalizes on the observation that bias often manifests as simple association rules, such as `"the first option is likely to be correct"`. Leveraging this insight, we construct a set of possible association rules and build a set of demonstrations from unlabeled samples that exemplify these rules. Subsequently, we identify the directions in the model representation space corresponding to these rules. During inference, we shift activations away from these identified directions.

Despite not using any label information, SteerFair reduces three instruction-tuned model performance variability across different option ordering on three benchmark tasks: two yes/no questions and one large MCQ dataset by an average of 10.86% accuracy points and 12.95 score points. Remarkably, it not only **outperforms a supervised baseline** with 100 labels but also matches the performance achieved with 500 labels. We systematically analyze SteerFair to understand the bias directions it discovers. We empirically show that steering bias directions *does not negatively* impact base model performance, and identified bias direction is generalizable across different datasets with the same task. Additionally, only a small number of unlabeled sample demonstrations are sufficient to identify these bias directions.

To summarize, our contributions include,

- We propose SteerFair, an *unsupervised* inference-time activation steering algorithm to mitigate foundation model bias.
- We demonstrate that SteerFair can effectively address the instability concerning option ordering in question-answering tasks. Furthermore, our findings demonstrate that the bias direction pinpointed by SteerFair is generalizable across datasets with the same task.
- Extensive experimental evidence shows improvement on three instruction-tuned models, with reduced performance variability by 10.86% accuracy points across three datasets.

2. Preliminaries

In this section, we describe our problem setup in more detail and briefly describe the transformer architecture (Vaswani et al., 2017) to set notation and context.

2.1. Problem Statement

Given a model T and a set of questions q and q', each representing a slight variant of the same prompt with non-meaning altering changes, for instance:

$q =$ Is Yosemite in California? Answer *yes or no*

$q' =$ Is Yosemite in California? Answer *no or yes*

Our goal is to ensure consistent model outputs $T(q) = T(q')$ with the given model T. The question permutation set, denoted as q, q', q'', ..., can also include any number of variations to the same question. For example, multiple-choice questions (MCQs) with shuffled options, or, in multi-modal settings, an image-question pair with changes to the image that should not influence the model's answer. For simplicity, we use q and q' as our running example, though SteerFair is applicable to any number of variants, as we will later show in Section 4.

2.2. Model Architecture

Our approach is applicable to any transformer-based (Vaswani et al., 2017) models. We follow the idea presented in (Elhage et al., 2021), viewing the inner computations of transformers as a series of "residual blocks". During inference, the token embedding layer initiates the "residual stream" by projecting tokens into a high-dimensional space $x \in \mathbb{R}^{DH}$. Subsequent layers then handle the flow of information within the stream—projecting into its own subspace, performing computations, and reprojecting its output back to the original space before feeding it to the next layer.

After the token embedding layer, each layer l is composed of a multi-head-attention (MHA) module followed by a multi-layer perceptron (MLP). The MHA module consists of H attention heads, with each head h independently performing linear computations in parallel.

We leverage the findings from (Elhage et al., 2021) showing the equivalence of stacking h outputs in the original

Transformers paper, with taking the sum of h outputs and projecting it back to the residual stream. For an input x, we can write the MHA computation as:

$$MHA(x) = x + \sum_{h=1}^{H} W_{Ol}^{h} Att_l^h (W_{Vl}^h x),$$

where H is the set of attention heads h in layer l, $W_{Ol}^h \in \mathbb{R}^{DH \times D}$ is the output weight matrix, $W_{Vl}^h \in \mathbb{R}^{D \times DH}$ the projection matrix to the attention head space, and Att is the attention operator (which encapsulates multiplication to key and query weight matrices). To have as many linear properties as possible, we design our intervention after the Att output and before W_{Ol}^h. We denote attention head activation value of any input x at head h of layer l as $\theta_{h,l}^x \in \mathbb{R}^D$.

3. STEERFAIR: Unsupervised Inference-Time Debiasing

We are ready to describe STEERFAIR: an unsupervised inference-time debiasing method.

In Section 1, we describe how models adopt unwanted bias towards superficial characteristics in the input, such as option location. Our goal is to reduce bias *directly in the model's internal representations without using labeled data*. To achieve this, we exploit the fact that bias often consists of spurious simple rules, like `always choose the first option`. Our method, STEERFAIR, leverages this fact by mimicking bias behaviors (e.g., answering both q and q' with the first option) and collecting their representations. Next, we identify directions in the model's representation space that encapsulate such bias and steer activation values away from it during inference. Our approach is illustrated in Figure 2 and summarized in Algorithm 1.

3.1. Enumerating Bias Association Rules

We capitalize on the intrinsic property of bias, that it comprises of simple, imitable rules. To leverage this property, we begin by enumerating the possible association rules the model might adopt. In question answering task with m options, this is straightforward: {`"Always choose the` j`th option"`}, $\forall j \in \{1, ...m\}$. We refer to this set as the *bias rule set* $\mathbb{r} = \{r_1, r_2, ..., r_m\}$. In our running example, the set has two rules: the bias to the first option and to the last option, denoted as: $\mathbb{r} = \{r_1, r_2\}$. Although models may adopt more intricate bias rules, like preferring some option tokens in specific positions, we assume these are highly correlated and influenced by position.

3.2. Constructing Bias Demonstrations

For each rule, we then construct demonstrations by collecting *a set of question-answer pairs that mimic the bias*. Recall that in our running example, q's first option is "yes" and last option is "no", and q''s option order is flipped, this yields:

$$s_1(q) = q + \text{“Answer: yes”} \quad s_2(q) = q + \text{“Answer: no”}$$
$$s_1(q') = q' + \text{“Answer: no”} \quad s_2(q') = q' + \text{“Answer: yes”},$$

where $s_j(q)$ is the demonstration of the jth rule with question q. We will refer to the set of demonstrations as the *demonstration set*. In this specific example, we have two demonstration sets: $\mathcal{S}_1 = \{s_1(q), s_1(q')\}$ and $\mathcal{S}_2 = \{s_2(q), s_2(q')\}$.

Transitioning from our synthetic example, let's now form a bias rule set and demonstration set for a dataset of yes/no questions $\{q_1, ..., q_N\}$. Similar to the synthetic case, our bias rule set is $\mathbb{r} = \{r_1, r_2\}$, representing biases towards the first and last options. The resulting demonstration sets are:

$$\mathcal{S}_1 = \{s_1(q_i), s_1(q_i')\}$$
$$\mathcal{S}_2 = \{s_2(q_i), s_2(q_i')\}$$
$$\forall i \in \{1, \dots N\}$$

. Construction of bias rule set and demonstration set is conducted without any label supervision. For simplicity, as an example we present the straightforward case of yes/no questions with two possible bias rules. Extending this to cases like multiple-choice questions (MCQ), where we may have more than two rules, is straightforward. We elaborate on MCQ rule and demonstration construction in Appendix C and present STEERFAIR performance on both yes/no questions and MCQ in Section 4.

3.3. Identifying Bias Directions from Demonstrations

Our goal in this step is to identify directions in the attention head space for each bias rule. Given a bias rule set $\mathbb{r}$ with m rules, the input to this step is the demonstration sets $\{\mathcal{S}_1, ..., \mathcal{S}_m\}$, and the output is m bias directions per attention head per layer.

Specifically, we collect the attention activation values of all demonstration set $\mathcal{S}^j$ elements at the last token position. This will result in a collection of latent state vectors denoted as

$$\mathbf{H}_{h,l}^j := \left[\theta_{h,l}^{s_j(q_1)} \middle| \dots \middle| \theta_{h,l}^{s_j(q_N)}\right]^T,$$

with $\mathbf{H}_{h,l}^j \in \mathbb{R}^{N \times D}$. Next, we find the direction corresponding to each bias rule by performing PCA on $\mathbf{H}_{h,l}^j$, and take the first principal direction (PCA1). We denote the resulting bias direction at attention head h of layer l as $\mathbf{v}_{h,l}^j \in \mathbb{R}^D$, which can be expressed as

$$\mathbf{v}_{h,l}^j = \text{PCA1}(\mathbf{H}_{h,l}^j) \tag{1}$$

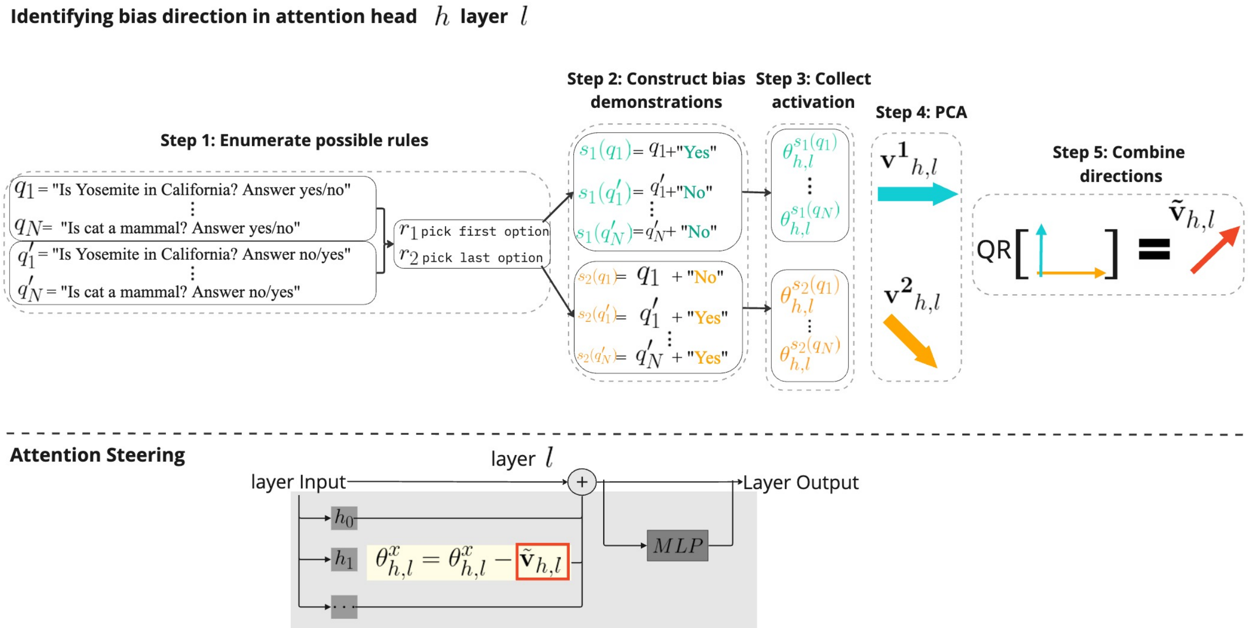


Figure 2. STEERFAIR finds bias directions $\tilde{\mathbf{v}}_{h,l}$ (top) and steer attention head values (bottom) away from it during inference.

3.4. Combining Multiple Bias Directions

In each attention head h of layer l, we now have a set of vectors $\left\{\mathbf{v}^1_{h,l}, .., \mathbf{v}^m_{h,l}\right\}$, where each corresponds to the directions in activation space that best represents a bias rule in $\mathbb{r}$. The remaining question is how to aggregate these directions so that steering them away from activation values during inference does not cause excessive disruption. Summation may produce large values with a large m, and a few highly correlated directions can dominate the average. To circumvent this, we use standard matrix decomposition methods, specifically QR decomposition, to obtain the orthonormal basis of the directions and take their average.

$$\tilde{\mathbf{v}}_{h,l} = \frac{1}{m}\text{QR}\left[\mathbf{v}^1_{h,l}\middle|\ldots\middle|\mathbf{v}^m_{h,l}\right]^T \tag{2}$$

This way, we remove correlations between bias directions.

3.5. Shifting Activation during Inference

During inference, we steer the activation values in the last token away from $\tilde{\mathbf{v}}_{h,l}$, written as

$$MHA(x) = x + \sum_{h=1}^{H} W^h_{Ol}(Att^h_l(W^h_{Vl}x) - \alpha\tilde{\mathbf{v}}_{h,l}), \tag{3}$$

where α is a hyperparameter that controls the strength of intervention. Finally, to preserve the model's original capabilities as much as possible, we normalize the updated latent states to match the l_2 norm of the latent states before the update.

3.6. Selecting Attention Heads to Intervene

To be minimally invasive, we select the top K attention heads with the highest average projected values in the first principal components. Intuitively, this is equivalent to selecting heads whose bias direction is mostly captured by the first principal component, and thus, our procedure will be the most effective. In section 5.2, we empirically show that setting $\alpha = 1$ and setting $K =$ all heads still produces improvement over baselines, but we can get a noticeable boost by hyperparameter tuning.

Algorithm 1 Identifying bias direction with STEERFAIR

Parameters: Foundation model with l layers and h attention heads per layer, Dataset of questions with different prompt orderings $\{(q_1, ..., q_N), (q'_1, ..., q'_N), (q''_1, ..., q''_N), ...\}$, strength hyperparameter α
Enumerate set of rules $\mathbb{r} = \{r_1, r_2, .., r_m\}$
for $j \in \{1, 2, ..m\}$ **do**
  Construct demonstration sets $\mathcal{S}_j = \{s_j(q_i)\}\,\forall i \in \{1, \ldots, N\}$
  Collect attention head values $\mathbf{H}^j_{h,l}$
  Identify direction $\mathbf{v}^j_{h,l} = \text{PCA1}(\mathbf{H}^j_{h,l})$
end for
Combine directions $\tilde{\mathbf{v}}_{h,l} = \frac{1}{m}\text{QR}\left[\mathbf{v}^1_{h,l}\middle|\ldots\middle|\mathbf{v}^m_{h,l}\right]$
Attention steering $MHA(x) = x + \sum_{h=1}^{H} W^h_{Ol}(Att^h_l(W^h_{Vl}x) - \alpha\tilde{\mathbf{v}}_{h,l})$
Returns: (intervened) foundation model

Dataset	Model	Vanilla		ITI (supervised 100)		ITI (supervised 500)		STEERFAIR (unsupervised)	
		Avg%(↑)	Std(↓)	Avg%(↑)	Std(↓)	Avg%(↑)	Std(↓)	Avg%(↑)	Std(↓)
ScienceQA	LLaVA (13B)	64.28%	0.024	64.22%	0.029	64.05%	**0.015**	**65.46%**	0.017
	IDEFICS (9B)	**58.99%**	0.181	54.74%	**0.079**	56.03%	0.125	58.70%	0.152
	InstructBLIP (13B)	56.32%	0.213	55.43%	0.257	55.38%	0.259	**56.92%**	**0.092**
MME	LLaVA (13B)	1333.43	65.42	**1350.19**	47.42	1334.87	10.43	1333.56	**2.72**
	IDEFICS (9B)	**1044.70**	49.92	1011.31	6.00	1023.55	**1.78**	1035.83	2.39
	InstructBLIP (13B)	1175.85	15.09	1184.30	5.58	1180.37	**0.75**	**1185.93**	15.03
VGR	LLaVA (13B)	71.04%	0.126	65.91%	0.079	**71.63%**	0.091	71.46%	**0.054**
	IDEFICS (9B)	**52.59%**	0.151	52.17%	0.286	50.53%	**0.051**	52.07%	0.060
	InstructBLIP (13B)	**51.38%**	0.242	50.32%	0.303	50.27%	0.172	50.31%	**0.006**

Table 1. Order bias results. Best method in **bold**, runner-up underlined. We compare against ITI (Li et al., 2023) with 100 and 500 labels

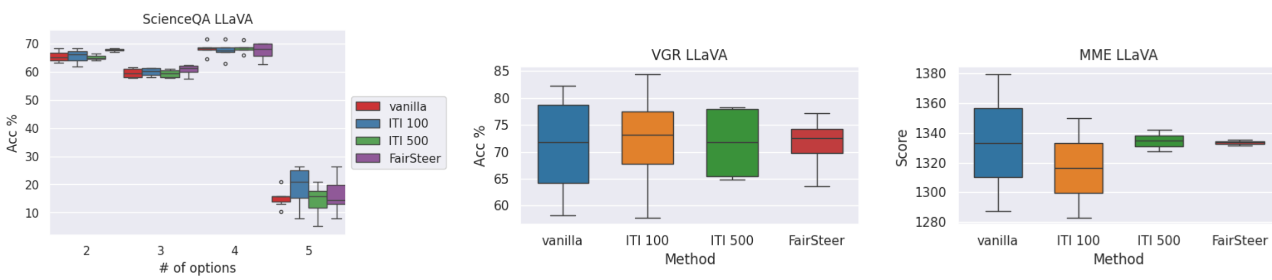


Figure 3. Left to right: ScienceQA, VGR, MME. STEERFAIR reduces standard deviation across prompt ordering while maintaining average accuracy.

4. Empirical Evaluation

In this section, we empirically verify the effectiveness of STEERFAIR with regard to:

- **Mitigating Order Bias** (Section 4.1): Our unsupervised method mitigates model's tendency to choose options at specific positions and is *competitive to (and sometimes outperforms) supervised methods* (Li et al., 2023).
- **Generalization Capability** (Section 4.2): We show that the bias direction identified by STEERFAIR is generalizable across datasets with the same task.

Baselines. We compare STEERFAIR against vanilla inference of instruction-tuned models: LLaVA (13B) (Liu et al., 2023b;a), IDEFICS (9B) (Laurençon et al., 2023) (open-source Flamingo (Alayrac et al., 2022)), and InstructBLIP (13B) (Dai et al.). For the order bias task, we include a comparison with Inference-Time Intervention (ITI) (Li et al., 2023), a supervised attention steering method.

Experimental Setup. We use a separated unlabeled training set for finding bias directions and testing. Unless stated otherwise, we follow the default split for each dataset. The results in this section are based on 1000 random unlabeled samples from the separated training set, with the exception of the MME Benchmark (Fu et al., 2023), where we use 100 samples due to its smaller size. We provide the full dataset and prompt details in Appendix D and E.

4.1. Mitigating Order Bias

Setup. We test our method on three Multiple Choice Question (MCQ) and yes/no question-answering datasets: ScienceQA (Lu et al., 2022), MME Benchmark (Fu et al., 2023), and Visual Genome Relation (VGR) (Lin et al., 2014). On yes/no questions, we record the performance across different option orderings (i.e., "answer with a yes/no" and "answer with a no/yes"). To test this bias in MCQ dataset, we employ the *answer-moving attack* evaluation from (Zheng et al., 2023a), by always moving the golden answers to a specific position.

Metrics. We report the average accuracy across option orders (Avg%) and the corresponding standard deviation (Std). In the case of MME Benchmark, we adhere to the evaluation score proposed in the original work. Note that the score is not in percentage form, resulting in a higher scale of standard deviation. A model with less bias to option order will have a high Avg% and low Std.

Results. Table 1 shows that **STEERFAIR significantly reduces bias to option order while often improving the average accuracy**. Remarkably, our unsupervised method surpasses ITI (supervised) performance on 100 labels and is comparable, and sometimes surpasses with 500 labels. This suggests that our method effectively reduces bias influence on the model's predictions, forcing the model to rely on relevant knowledge.

Figure 3 shows the non-averaged results on the LLaVA model. On the VGR and MME datasets, STEERFAIR significantly reduces the standard deviation (as indicated by the box plot height) while maintaining the average accuracy. A similar pattern is observed on the ScienceQA dataset, particularly for options 2 and 3. However, STEERFAIR's impact is more limited for a larger number of options. We hypothesize that this is caused by STEERFAIR's simple averaging approach to combine multiple bias directions (Section 3.4), as the number of bias directions grows with the number of options. This indicates potential areas for improvement in this part of our approach. Comprehensive non-averaged results for all models are provided in Appendix G.

4.2. Generalization Capability

Setup. Next, we study the generalization property of bias directions identified by STEERFAIR. Specifically, we test whether attention steering using directions identified from a different dataset with the same task (e.g., VGR and MME) can produce similar results as using identified direction from the same dataset. Intuitively, the order bias problem is not domain-specific, so the bias directions should generalize.

Results. The results in Table 2 illustrate that STEERFAIR with directions transferred from another dataset (TD) consistently enhances the base model, demonstrating reduced performance variance and improved average accuracy. However, exceptions exist in cases where STEERFAIR with directions from the same dataset (OD) fails to produce significant improvement, as observed, for instance, in the case of InstructBLIP on the MME dataset. This shows that **the bias direction identified by STEERFAIR generalizes** across datasets with the same task.

5. Analysis

This section presents analyses of STEERFAIR components and provides empirical characterizations.

5.1. Sensitivity to hyperparameters α and K

In Figure 4 (left), we observe that Avg Acc% is insensitive to hyperparameters, shown by similar accuracies across different parameter values. Once both α and K are large ($\alpha >= 5$ $K >= 1000$), the steering changes the original activation values too much, causing accuracy to decline.

The standard deviation plot (Figure 4 (right)) shows good performance regime below the diagonal for $\alpha > 0.1$. This shows that a combination of more number of heads intervened, with moderate α values yields the most effective bias reduction. Notably, across various parameter settings, the Std% consistently matches or surpasses ITI's and outperforms the vanilla model.

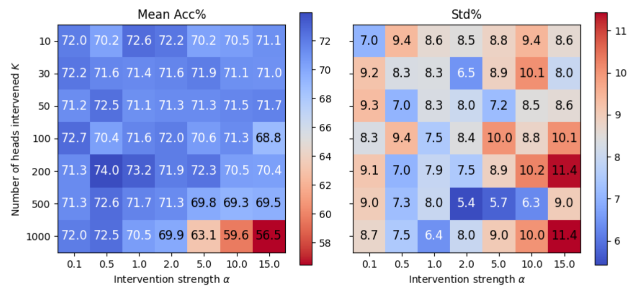


Figure 4. Effect of hyperparameters α (x-axis) and number of intervened attention heads K (y-axis). Left: Acc%; Right: Std%. Performance recorded for VGR dataset.

5.2. Can we get away without hyperparameters?

Here we test STEERFAIR performance without any hyperparameter tuning. We intervene all attention heads ($K =$ all), and set the strength hyperparameter $\alpha = 1$. Table 3 shows that hyperparameter tuning produces the best results. However, it is worth noting that STEERFAIR without tuning still improves the performance of the vanilla model.

5.3. How do bias directions in the activation space look?

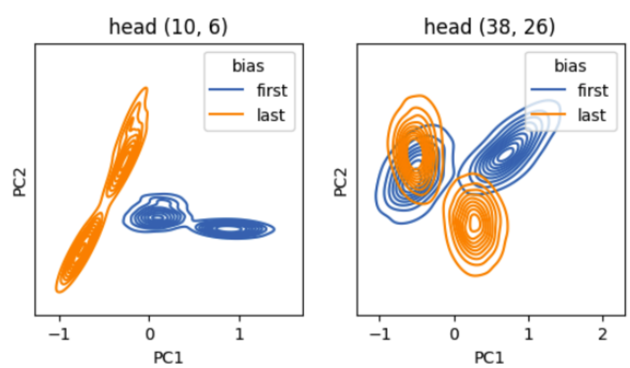


Figure 5. Kernel density estimate plots of STEERFAIR-identified bias directions on the VGR dataset, projected onto the first 2 PCs.

Figure 5 illustrates the geometry of bias directions identified by STEERFAIR before the combination step, projected onto the first two principal components. Two crucial observations emerge. Firstly, there is minimal overlap between distinct bias directions, indicating their simplicity and separability with just two principal components. Secondly, the maximum variance directions in the two density plots (blue and orange) are nearly orthogonal. This underscores the advantage of decomposing 'bias' into multiple rules rather than

Model	Dataset	Vanilla		Original Direction (OD)		Transferred Direction (TD)	
		Avg%(↑)	Std(↓)	Avg%(↑)	Std(↓)	Avg%(↑)	Std(↓)
LLaVA (13B)	VGR	71.04%	0.126	71.46%	0.054	**73.19%**	**0.015**
	MME	1333.43	65.42	**1333.56**	**2.72**	1305.57	56.54
IDEFICS (9B)	VGR	52.59%	0.151	52.07%	**0.060**	**52.76%**	0.144
	MME	1044.70	49.92	1035.83	**2.39**	**1060.94**	11.35
InstructBLIP (13B)	VGR	**51.38%**	0.242	50.31%	**0.006**	50.10%	0.212
	MME	1175.85	15.09	**1185.93**	**15.03**	844.37	54.78

Table 2. STEERFAIR generalization performance. STEERFAIR with original direction (OD) uses direction identified using the dataset, STEERFAIR with transferred direction (TD) uses direction identified from another dataset of the same task. Best **bolded**, second best underlined.

Dataset	Model	Vanilla		With tuning		No tuning	
		Avg%(↑)	Std(↓)	Avg%(↑)	Std(↓)	Avg%(↑)	Std(↓)
SQA	LLaVA	64.28%	0.024	**65.46%**	**0.017**	62.00%	**0.017**
	IDEFICS	56.32%	0.213	**56.92%**	**0.092**	53.54%	0.210
	InstructBLIP	56.32%	0.213	56.92%	**0.092**	**57.75%**	0.148
VGR	LLaVA	71.04%	0.126	**71.46%**	0.054	67.03%	**0.023**
	IDEFICS	**52.59%**	0.151	52.07%	0.060	51.16%	**0.023**
	InstructBLIP	51.38%	0.242	50.31%	**0.006**	**52.06%**	0.203

Table 3. Comparison with STEERFAIR without tuning α and K. Best numbers in **bold**, second best underlined STEERFAIR without tuning still produces improvement over the vanilla model.

treating it as a singular component.

5.4. How many unlabeled samples do we need?

We vary the number of unlabeled samples N to find bias direction and see the impact on performance in Figure 6. On the top figure, it is evident that Avg% remains relatively stable, exhibiting performance fluctuations within the range of $\pm 2\%$ across different values of N. Intriguingly, even with a small sample size ($N < 100$), the accuracy is preserved, underscoring the non-intrusive nature of our intervention technique. On the bottom figure, we observe an interesting trend: there is no (negative) correlation between N and resulting Std. The lowest standard deviation is achieved between $N = 300$ and $N = 1000$. This suggests that bias direction can be effectively approximated with a relatively small number of key samples.

It is surprising that larger N does not yield better performance. We hypothesize this is because more samples include more information, and thus bias direction extracted is more noisy. To validate this hypothesis, we examine the captured variance ratio in the first principal component (PC) across different N values in Figure 7. It is evident that for both small and large values of N ($N < 100$ and $N > 1000$), the captured variance in the first PC, and hence the informativeness of STEERFAIR bias direction, is smaller than for $300 \leq N \leq 1000$, supporting our hypothesis.

6. Related Work

6.1. Foundation Model Bias and Robustness

Foundation model robustness is a heavily studied area (Adila et al., 2023; Zhang & Ré, 2022; Yang et al., 2023). The majority of these works are tailored to embedding-based models, such as CLIP, and are designed for scenarios with access to single embeddings per sample. This poses challenges when extending these approaches to next-word prediction transformer-based models, where the notion of single embedding is less straightforward.

On next-word prediction models, works such as (Zheng et al., 2023a; Pezeshkpour & Hruschka, 2023) address vulnerabilities to superficial prompt modifications (Zhao et al., 2021; Zheng et al., 2023b), such as non-meaning altering changes in token order. Specifically targeting the model's inclination to favor particular options in MCQ settings, (Zheng et al., 2023a) and (Pezeshkpour & Hruschka, 2023) propose methods involving the calibration of model probabilities at the option token level, with final predictions based on the calibrated output. Our approach, STEERFAIR, distinguishes itself in two crucial aspects: (1) it allows more flexible intervention by modifying the model's internal activations, and

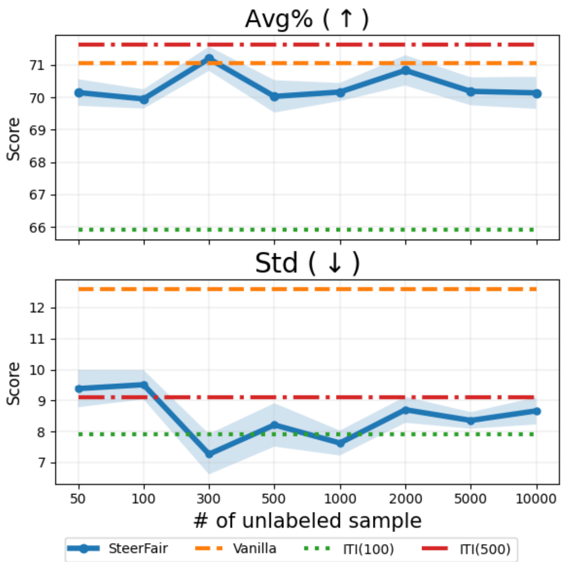


Figure 6. Impact of number of unlabeled sample N (x-axis) on performance (y-axis). Performance across 10 random seeds.

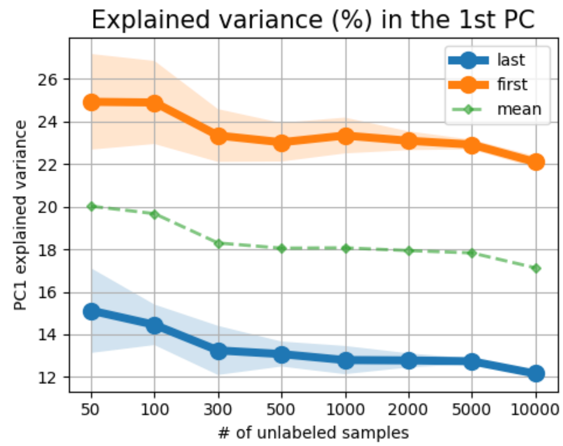


Figure 7. Explained variance ratio in the first principal component (PC) of bias to the first (blue) and last options (orange) directions identified by STEERFAIR (y-axis) across the number of unlabeled samples N (x-axis). Performance across 10 random seeds.

(2) STEERFAIR is versatile, applicable to a broader types of tasks and biases.

6.2. Extracting LLM Knowledge in the Latent Space

Progress has been made in extracting and understanding latent knowledge of LLMs. Burns et al. (2022) find truthfulness direction by identifying a direction in the model's internal representation such that the probabilities assigned to truthful vs. non-truthful answers adhere to logical consistencies. Gurnee et al. study how high-level interpretable features are represented in the models' internal workings. Moschella et al. (2022) discover that learned representations remain invariant across stochastic factors in different training runs. Li et al. (2022) present evidence indicating that a GPT variant trained to generate moves in the Othello game learns a representation of game states. Furthermore, Liu et al. (2023c) identify a direction in the latent space that effectively summarizes knowledge from in-context samples. Building upon insights from these techniques, we search for where and how bias is learned in the model's internal representation space.

6.3. Modifying LLM Attention

Modifying LLM attention space has shown remarkable success in steering model behavior toward desirable traits without fine-tuning. This includes style transfer (Subramani et al., 2022), enhancing truthfulness (Li et al., 2023), achieving more controllable and effective in-context learning and transfer learning, as demonstrated in (Liu et al., 2023c) and (Shi et al., 2023), and improving instruction-following capabilities (Zhang et al., 2023). Ours shares a similar spirit with these works; we design an attention-steering mechanism specifically to mitigate model bias. Notably, our method leverages the inherent properties of the targeted desirable trait (i.e., reduced bias) rather than relying on label supervision.

7. Conclusion

We introduce STEERFAIR, an approach for unsupervised mitigation of bias in question-answering models, operating *directly in the model representation space*. Our technique improves instruction-tuned model performance across various benchmark tasks, surpassing supervised baselines, and matching more amount labeled data scenarios. The comprehensive experimental analysis highlights the method's versatility, showing generalization properties of the identified bias directions and demonstrating efficacy with only a small number of unlabeled samples. The limitations of our work are discussed in Appendix B.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here. As of now, we are not aware of additional potential societal impacts beyond the typical implications associated with LLMs in question-answering scenarios.

References

Adila, D., Shin, C., Cai, L., and Sala, F. Zero-shot robustification of zero-shot models with foundation models. *arXiv preprint arXiv:2309.04344*, 2023.

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

Burns, C., Ye, H., Klein, D., and Steinhardt, J. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*, 2022.

Caliskan, A., Bryson, J. J., and Narayanan, A. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334): 183–186, 2017. doi: 10.1126/science.aal4230. URL https://www.science.org/doi/abs/10.1126/science.aal4230.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Dai, W., Li, J., Li, D., Tiong, A., Zhao, J., Wang, W., Li, B., Fung, P., and Hoi, S. Instructblip: Towards general-purpose vision-language models with instruction tuning. arxiv 2023. *arXiv preprint arXiv:2305.06500*.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Fu, C., Chen, P., Shen, Y., Qin, Y., Zhang, M., Lin, X., Yang, J., Zheng, X., Li, K., Sun, X., et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. *arXiv preprint arXiv:2306.13394*, 2023.

Gurnee, W., Nanda, N., Pauly, M., et al. Finding neurons in a haystack: Case studies with sparse probing, may 2023. *URL http://arxiv. org/abs/2305.01610. → p*, 9.

Khosla, A., Zhou, T., Malisiewicz, T., Efros, A. A., and Torralba, A. Undoing the damage of dataset bias. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part I 12*, pp. 158–171. Springer, 2012.

Laurençon, H., Saulnier, L., Tronchon, L., Bekman, S., Singh, A., Lozhkov, A., Wang, T., Karamcheti, S., Rush, A. M., Kiela, D., et al. Obelisc: An open web-scale filtered dataset of interleaved image-text documents. *arXiv preprint arXiv:2306.16527*, 2023.

Li, K., Hopkins, A. K., Bau, D., Viégas, F., Pfister, H., and Wattenberg, M. Emergent world representations: Exploring a sequence model trained on a synthetic task. *arXiv preprint arXiv:2210.13382*, 2022.

Li, K., Patel, O., Viégas, F., Pfister, H., and Wattenberg, M. Inference-time intervention: Eliciting truthful answers from a language model, 2023.

Liang, W., Tadesse, G. A., Ho, D., Fei-Fei, L., Zaharia, M., Zhang, C., and Zou, J. Advances, challenges and opportunities in creating data for trustworthy ai. *Nature Machine Intelligence*, 4(8):669–677, 2022.

Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Doll'a r, P., and Zitnick, C. L. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

Liu, H., Li, C., Li, Y., and Lee, Y. J. Improved baselines with visual instruction tuning, 2023a.

Liu, H., Li, C., Wu, Q., and Lee, Y. J. Visual instruction tuning, 2023b.

Liu, S., Xing, L., and Zou, J. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv preprint arXiv:2311.06668*, 2023c.

Lu, P., Mishra, S., Xia, T., Qiu, L., Chang, K.-W., Zhu, S.-C., Tafjord, O., Clark, P., and Kalyan, A. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Moschella, L., Maiorca, V., Fumero, M., Norelli, A., Locatello, F., and Rodola, E. Relative representations enable zero-shot latent space communication. *arXiv preprint arXiv:2209.15430*, 2022.

Pezeshkpour, P. and Hruschka, E. Large language models sensitivity to the order of options in multiple-choice questions. *arXiv preprint arXiv:2308.11483*, 2023.

Shi, B., Gai, S., Darrell, T., and Wang, X. Refocusing is key to transfer learning. *arXiv preprint arXiv:2305.15542*, 2023.

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

Torralba, A. and Efros, A. A. Unbiased look at dataset bias. In *CVPR 2011*, pp. 1521–1528, 2011. doi: 10.1109/CVPR.2011.5995347.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, P., Li, L., Chen, L., Zhu, D., Lin, B., Cao, Y., Liu, Q., Liu, T., and Sui, Z. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*, 2023.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. Transformers: State-of-the-art natural language processing. In Liu, Q. and Schlangen, D. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

Yang, Y., Nushi, B., Palangi, H., and Mirzasoleiman, B. Mitigating spurious correlations in multi-modal models during fine-tuning. *arXiv preprint arXiv:2304.03916*, 2023.

Zhang, M. and Ré, C. Contrastive adapters for foundation model group robustness. *Advances in Neural Information Processing Systems*, 35:21682–21697, 2022.

Zhang, Q., Singh, C., Liu, L., Liu, X., Yu, B., Gao, J., and Zhao, T. Tell your model where to attend: Post-hoc attention steering for llms. *arXiv preprint arXiv:2311.02262*, 2023.

Zhao, Z., Wallace, E., Feng, S., Klein, D., and Singh, S. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pp. 12697–12706. PMLR, 2021.

Zheng, C., Zhou, H., Meng, F., Zhou, J., and Huang, M. Large language models are not robust multiple choice selectors. *arXiv e-prints*, pp. arXiv–2309, 2023a.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023b.

A. Glossary

The glossary is given in Table 4.

Symbol	Definition
q, q'	question string, same question with modified option positions
$\mathbb{r}$	bias rule set
r_j	jth rule in bias rule set
$s_j(q)$	demonstration of jth bias rule from q
$\mathcal{S}_j$	demonstration set of jth rule (r_j)
x	input vector in model layers
l	model layer
h	model single attention head
$\mathbf{H}$	matrix of stacked attention head activation values
$\theta^x_{h,l}$	attention head h of layer l activation value of input x
$\mathbf{v}^j_{h,l}$	jth rule bias direction at head h of layer l
$\tilde{\mathbf{v}}_{h,l}$	combined bias direction at head h of layer l
α	hyperparameter to control intervention strength
K	hyperparameter: how many attention heads to intervene
N	number of samples in dataset

Table 4. Glossary of variables and symbols used in this paper.

B. Limitations

Our work comes with several limitations that merit attention. Firstly, the accurate identification of bias directions by STEERFAIR depends on the existence of directions in activation space that effectively summarize bias. This necessitates a reasonable separation between usable knowledge and bias within the model's learned representation. The precise circumstances under which these conditions hold remain unclear.

Secondly, we acknowledge the absence of *theoretical* characterizations for hyperparameters and the required number of unlabeled samples. While addressing this theoretical gap would enhance our understanding, there are also practical improvements to be made in STEERFAIR. One avenue is the extension of our method to accommodate *non-enumerable* bias. For instance, exploring latent directions associated with harmful text generation, misinformation, or other forms of bias, utilizing principles derived from our work. We provide initial evidence of the potential of this direction in Appendix **??**.

C. Constructing Bias Demonstrations For MCQ Dataset

This section details the bias rule set $\mathbb{r}$ and demonstration set $\mathcal{S}$ construction for MCQ datasets.

Enumerating Bias Rules. Similar to yes/no questions case we demonstrated in Section 3, MCQ questions with m options have m rules. For example, when we have 3 options: (A/B/C), our rule set $\mathbb{r}$ items are:

$$r_1 = \texttt{Always choose (A)}$$
$$r_2 = \texttt{Always choose (B)}$$
$$r_3 = \texttt{Always choose (C)}$$

Constructing Bias Demonstrations. Unlike yes/no questions, where there are only 2 possible option orderings ("yes/no" and "no/yes"), the number of orderings for MCQ questions grows in factorial order with the number of presented options m (number of possible orders $= m!$). While our method requires only the model's output to identify bias direction (no decoding necessary), conducting $m!$ forward passes to cover all permutations is computationally intractable. Therefore, we adopt a practical alternative: **cyclic permutation**, reducing the permutations from $m!$ to a manageable m. For example, if our question $q =$ "Which city is located in Asia? (A) London (B) Chennai (C) Buenos Aires " with three options (A/B/C), we have the following permutations:

$$q' = \text{Which city is located in Asia? (A) Buenos Aires (B) London (C) Chennai}$$
$$q'' = \text{Which city is located in Asia? (A) Chennai (B) Buenos Aires (C) London}$$

The demonstration set $\mathcal{S}_1 = \{s_1(q), s_1(q'), s_1(q'')\}$ is

$$s_1(q) = q + \text{“Answer: (A) London”}$$
$$s_1(q') = q' + \text{“Answer: (A) Buenos Aires”}$$
$$s_1(q'') = q'' + \text{“Answer: (A) Chennai”}$$

. Similarly, for $\mathcal{S}_2 = \{s_2(q), s_2(q'), s_2(q'')\}$ and $\mathcal{S}^3 = \{s_3(q), s_3(q'), s_3(q'')\}$, we append q, q', q'' with the answers (B) and (C) respectively.

D. Dataset Statistics and Setup

D.1. Option bias datasets

Table 5 shows dataset statistics for option bias.

Dataset	Type	# Options	# Test samples	# Samples for finding direction
ScienceQA	MCQ	2	2228	1000
		3	971	
		4	1004	
		5	38	
MME	yes/no	2	1,542	100
VGR	yes/no	2	9,576	1000

Table 5. Dataset statistics for option bias

Originally, the VGR dataset was a 2 choice options dataset. We are an image and 2 choices: one option is the correct caption of the given image (e.g., “The cow is eating the grass”), and the other is a false caption (e.g., “The grass is eating the cow”). We convert this dataset into a yes/no question by turning each caption into 2 questions (e.g., “Is this the correct caption for the image? answer with a yes or no. The cow is eating the grass”). For ScienceQA, we use default test samples from the original dataset. For MME, we randomly sample 100 samples to find bias direction and use the rest for evaluation. For VGR, we randomly split the dataset 80:20 train/validation:test split, and randomly sample 1000 samples from the train split to find bias direction.

For ScienceQA, since the number of test samples varies significantly between each # of options, we report the weighted average accuracy (by the number of sample). Other datasets are pretty balance so we report the non-weighted accuracy.

E. Prompt Details

We use each model’s default system prompts for formatting, detailed as follows:

LLaVA

```
prompt = A chat between a curious human and an artificial intelligence assistant.
The assistant gives helpful, detailed, and polite answers to the human’s questions.
Human: [QUESTION]
Assistant: [ANSWER]
```

For inference, we leave the part after “Assistant:” empty for the model’s answers. For collecting activation values (section 3), we append the answers based on the constructed demonstration set after “Assistant:”.

IDEFICS

We use the recommended system prompt from IDEFICS Huggingface (Wolf et al., 2020) page https://huggingface.co/HuggingFaceM4/idefics-9b-instruct.

```
prompt = [
            [
                f"User: {QUESTION}",
                "<end_of_utterance>",
                f"\nAssistant: {ANSWER}",
            ],
        ]
```

For inference, we leave the part after "Assistant:" empty for the model's answers. For collecting activation values (section 3), we append the answers based on the constructed demonstration set after "Assistant:".

InstructBLIP

We follow InstructBLIP usage from Huggingface page https://huggingface.co/Salesforce/instructblip-vicuna-13b. , where there is no system prompt. We only use the question string as it is, followed by the answer.

```
prompt = "[QUESTION] [ANSWER]"
```

Similarly for the previous two cases, we leave the part after the question empty for inference, and fill the answers based on the constructed demonstration set for collecting activation values.

F. Implementation Details

F.1. Compute details

There is no Transformer model training or fine-tuning conducted in this paper's experiments. We use 8 Test V100 GPUs for hyperparameter tuning and evaluation.

F.2. SteerFair implementation

We use IDEFICS and InstructBLIP models from HuggingFace (Wolf et al., 2020) and LLaVA from the author's repository (Liu et al., 2023b). We provide pseudocode for collecting model activation values in Algorithm table 2.

Algorithm 2 Pseudocode for collecting activation values $\mathbb{H}$

Parameters: Demonstration sets $\mathcal{S}_1, \dots \mathcal{S}_m$, model T, attention head index h, layer index l, question sets $\{q_1, \dots q_N\}, \{q'_1, \dots q'_N\}, \dots$
for $i \in \{1, \dots, N\}$ **do**
  for $j \in \{1, \dots, m\}$ **do**
    $\mathbf{H}^j$ = []
    for $\{s_j(q_i)\} \in \mathcal{S}^j$ **do**
      $\theta_{h,l}^{s_j(q_i)}$ = T($s_j(q_i)$)["hidden states"][-1][l, h]
      $\mathbf{H}^j$.append($\theta_{h,l}^{s_j(q_i)}$)
    end for
    $\mathbf{H}^j =$ np.vstack($\mathbf{H}^j$)
  end for
end for
Returns: $\mathbf{H}^1, \dots \mathbf{H}^m$.

F.3. Baselines implementation

This section presents implementation details for baseline methods.

Option bias Vanilla inference is done with prompts detailed in Appendix E. ITI (Li et al., 2023) code is adapted from the author's original repository https://github.com/likenneth/honest_llama. The 100 and 500 samples are randomly sampled from the datasets training splits.

Stereotypical bias Vanilla inference is done with prompts detailed in Appendix E. Prompting baseline uses the following prompt prepended to each question: "Do not stereotype."

F.4. Hyperparameter search

We perform hyperparameter search for both SteerFair and ITI. We list the hyperparameter search space in Table 6

Method	Intervention strength α	number of heads K
ITI	$\{1, 5, 10, 15, 20, 25, 30, 40, 50\}$	$\{1, 10, 20, 30, 40, 50, 100\}$
SteerFair	$\{0.1, 0.5, 1, 2, 5, 10, 15, 20, 25\}$	$\{10, 30, 50, 100, 200, 500\}$

Table 6. Hyperparameter search space

We follow the initial hyperparameter space for for as suggested in the original paper (Li et al., 2023). We try a larger number of K in SteerFair because we use l_2 normalization post-intervention (Section 3). The best hyperparameter is chosen based on the best performance on the validation set (minival split for ScienceQA).

G. Supplementary Results

We present exhaustive, non averaged results in this section.

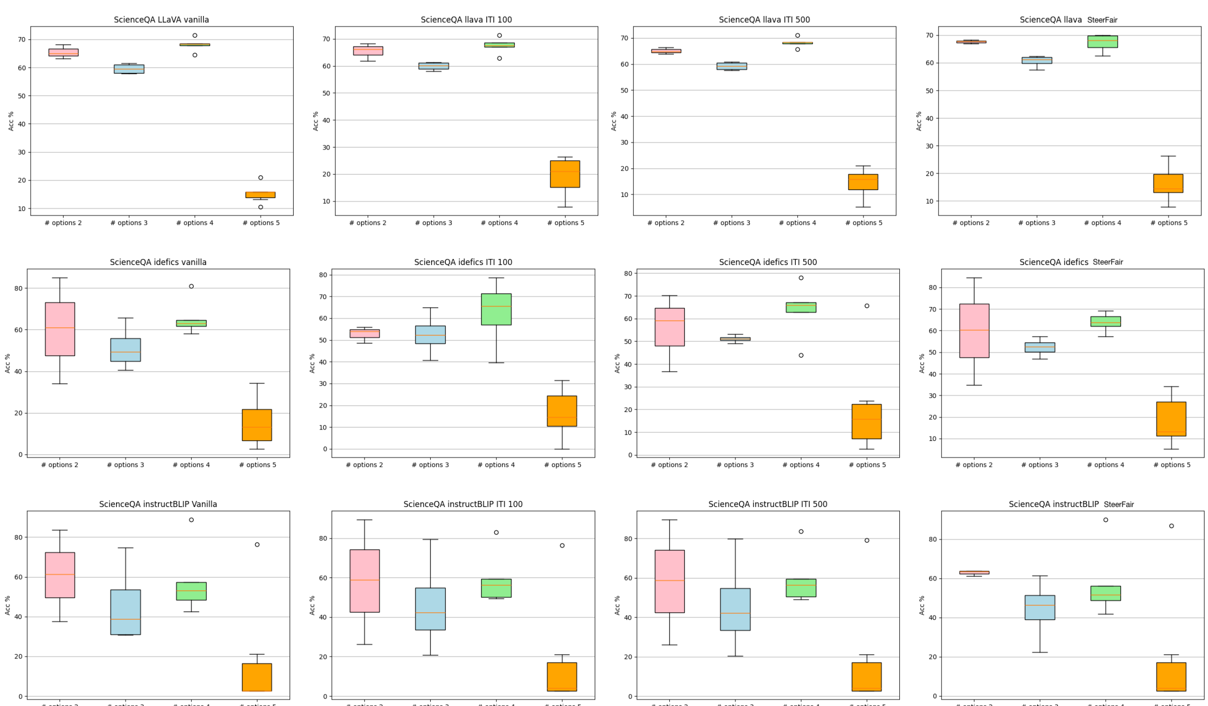


Figure 8. ScienceQA results

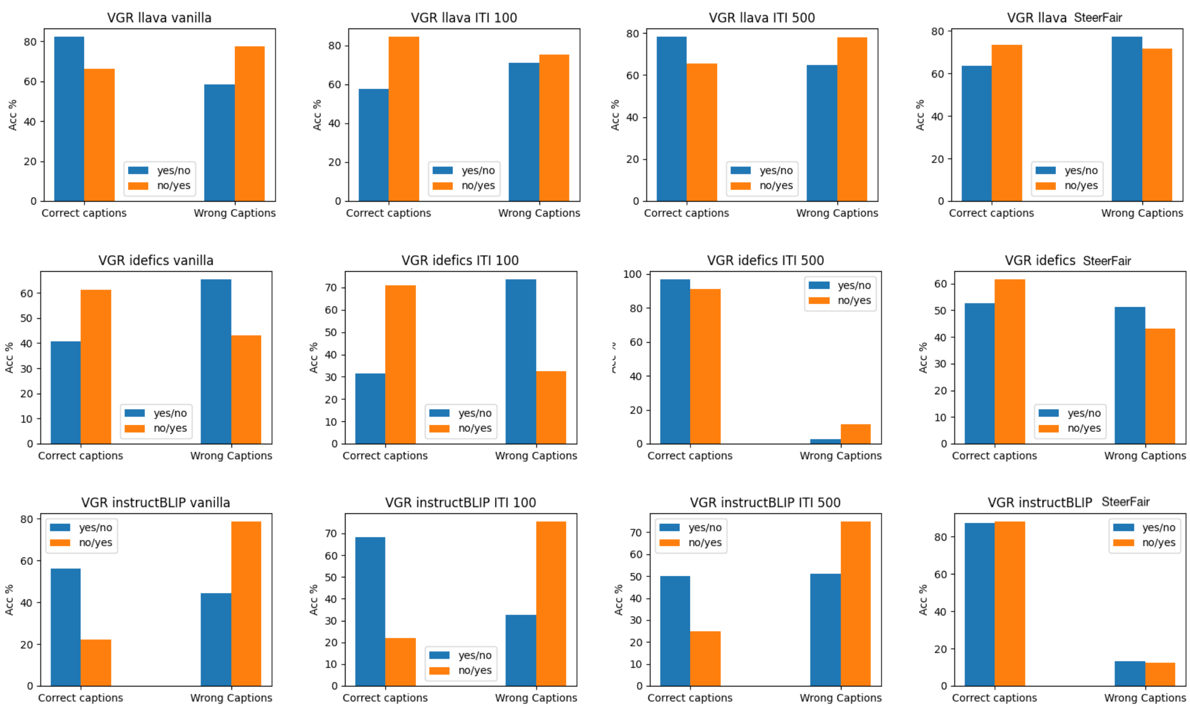


Figure 9. VGR results

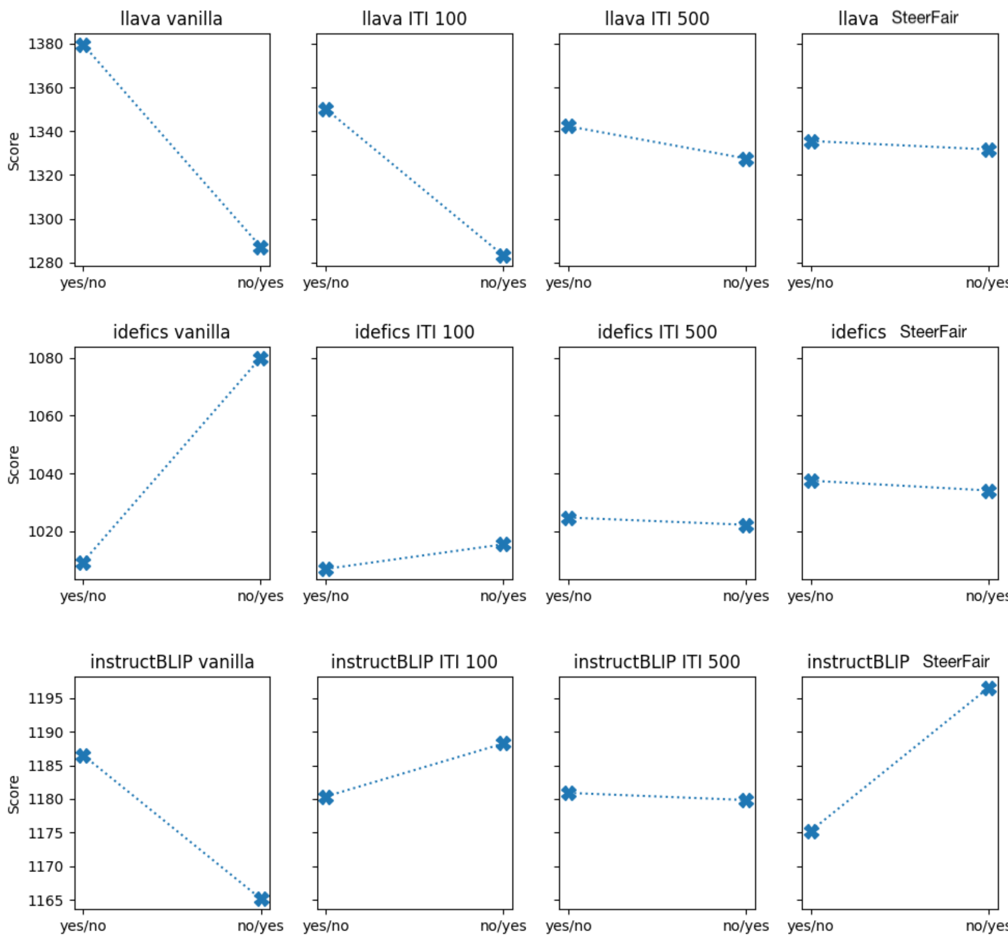


Figure 10. MME results