Activation Scaling for Steering and Interpreting Language Models

Niklas Stoehr[1] **Kevin Du**[1] **Vésteinn Snæbjarnarson**[2]
Robert West[3] **Ryan Cotterell**[1] **Aaron Schein**[4]
[1]ETH Zürich [2]University of Copenhagen [3]EPFL [4]The University of Chicago
niklas.stoehr@inf.ethz.ch kevin.du@inf.ethz.ch vesn@di.ku.dk
robert.west@epfl.ch ryan.cotterell@inf.ethz.ch schein@uchicago.edu

Abstract

Given the prompt "Rome is in", can we steer a language model to flip its prediction of an incorrect token "France" to a correct token "Italy" by only multiplying a few relevant activation vectors with scalars? We argue that successfully intervening on a model is a prerequisite for interpreting its internal workings. Concretely, we establish a three-term objective: a successful intervention should flip the correct with the wrong token and vice versa (effectiveness), and leave other tokens unaffected (faithfulness), all while being sparse (minimality). Using gradient-based optimization, this objective lets us learn (and later evaluate) a specific kind of efficient and interpretable intervention: activation scaling only modifies the signed magnitude of activation vectors to strengthen, weaken, or reverse the steering directions already encoded in the model. On synthetic tasks, this intervention performs comparably with steering vectors in terms of effectiveness and faithfulness, but is much more minimal allowing us to pinpoint interpretable model components. We evaluate activation scaling from different angles, compare performance on different datasets, and make activation scalars a learnable function of the activation vectors themselves to generalize to varying-length prompts.[1]

1 Introduction

Understanding which components of a language model play which roles in which tasks is a core aim of mechanistic interpretability. Given the prompt *Rome is in*, for instance, one might ask which components of the model most influence it to favor *Italy* over some incorrect answer token, such as *France*. In addressing this question, a natural axiom is that a given component can only be understood as influential for a given task if intervening on it meaningfully alters the model's task-specific behavior.

[1]Code to experiment with our method is available at https://github.com/niklasstoehr/activationScaling.

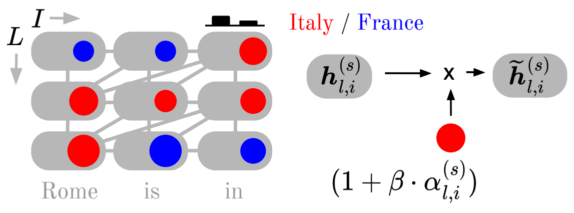
I
L
Italy / France
$\boldsymbol{h}^{(s)}_{l,i}$
x
$\widetilde{\boldsymbol{h}}^{(s)}_{l,i}$
$(1+\beta\cdot\alpha^{(s)}_{l,i})$
Rome
is
in

Figure 1: We show that it is often sufficient to scale a few influential activation vectors $\mathbf{h}^{(s)}_{l,i}$ for a model to favor one answer token over another token. This could be the MLP output at layer l for token position i. We learn multiplicative scalars, $\alpha^{(s)}_{l,i}$, using gradient-based optimization. These correspond to interpretable interventions that generalize to test set prompts while relying on fewer parameters than additive steering vectors.

Building on this basic axiom, a growing literature seeks to both generate and test hypotheses about where certain behaviors are localized in a model by employing targeted interventions with methods such as activation patching (Lakretz et al., 2019; Vig et al., 2020; Meng et al., 2022), among others. Studies in this literature often produce a set of attribution scores associated with various locations in the model which represent how much the model's output changed after editing the activation vectors at each location. Although an effective intervention may be necessary to believe a given localization hypothesis, it is not sufficient, as interventions to other model locations may be similarly effective. Indeed, recent work has questioned the relationship between interpretability and intervention on this basis, and has advocated for more rigorous and deliberate methodology for connecting the two (Hase et al., 2023; Wang and Veitch, 2024; Hanna et al., 2024).

A parallel literature on model steerability also seeks to develop effective interventions, not for the primary purpose of interpretability, but to steer models toward desirable behaviors, like factuality (Li et al., 2023), or away from undesirable behaviors, like toxicity (Ilharco et al., 2023; Turner et al., 2023). Methods in this literature are typically designed to be maximally effective for steering and thus tend to involve model-wide (i.e., not necessarily localized) interventions to the weights (Houlsby et al., 2019; Hu et al., 2021; Ilharco et al., 2023) or activation vectors (Subramani et al., 2022; Hernandez et al., 2024).

This paper seeks to synthesize the goals of model-wide interventions while still offering mechanistic insights into the model. Specifically, we seek interventions that are **effective**, and not localized *a priori*, but are nevertheless **minimal** and **faithful**. We define an intervention to be effective if it flips the prediction of the *correct* answer token (e.g., *Italy*) to an *incorrect* token (e.g., *France*). We define an intervention to be faithful if it does not substantially alter the probabilities of tokens unrelated to the given task. And, we define an intervention to be minimal if it alters activation vectors sparsely, in only a small number of locations. Our notions of minimality and faithfulness follow from recent work in the Transformer circuits literature (Wang et al., 2023; Bhaskar et al., 2024). We operationalize these three desiderata via a three-term objective allowing us to learn model-wide interventions using gradient-based optimization.

More specifically, we propose a kind of model-wide yet parsimonious intervention that associates a single scalar parameter with each of many locations in a model. The scalar parameters are learned so that the model can be effectively steered—e.g., away from *Italy* and toward *France*—by simply scaling the activation vector at each location. We call this approach **activation scaling** (ACTIVSCALAR), inspired by the idea that some model components are highly specialized for certain task-relevant computations (Voita et al., 2019; Geva et al., 2022). Existing work (Yu et al., 2023; Merullo et al., 2024; Ortu et al., 2024) has identified individual components, e.g., name mover heads (Wang et al., 2023), in a first step and successfully steered a model by scaling only the contributions of those individual components in a second step.

To evaluate our approach, we construct a baseline method that is the same in all aspects except it learns additive vectors at all locations, rather than single multiplicative scalars. We refer to this approach as **steering vectors** (STEERVEC). In a suite of experiments, we find overall that activation scaling learns interventions which are as effective and faithful as steering vectors, while requiring far fewer learnable parameters. Our results suggest that merely scaling the signed magnitude of activation vectors, without further affecting their direction, is sufficient for effective steering on simple tasks. Moreover, we find activation scalars are highly interpretable. They are easy to understand as simply strengthening or weakening the steering directions already encoded in the model (Subramani et al., 2022; Ferrando and Voita, 2024), and when visualized, they provide sparse and localized descriptions of important model components. Finally, to extend our approach beyond fixed prompt templates, we develop a dynamic version of activation scaling. DYNSCALAR makes the activation scalars learned functions of the activation vectors themselves, thus allowing learned interventions to transfer to test set prompts of varying length.

2 Transformer Language Models

Let Σ be an alphabet of **tokens**, a finite, non-empty set, and let Σ^* be the set of all strings with tokens drawn from Σ. A **language model** p is a probability distribution over Σ^*. As is current practice, most language models are defined autoregressively. Let $\boldsymbol{v} = v_1 \cdots v_I \in \Sigma^*$ be a string; then the autoregressive factorization of p is given by

$$p(\boldsymbol{v}) = p(\text{EOS} \mid \boldsymbol{v}) \prod_{i=1}^{I} p(v_i \mid \boldsymbol{v}_{<i}). \tag{1}$$

Each local conditional distribution $p(\cdot \mid \boldsymbol{v}_{<i})$ is a distribution over $\overline{\Sigma} = \Sigma \cup \text{EOS}$, where $\text{EOS} \notin \Sigma$ is the end-of-string token. In the context of an autoregressive language model, we call $\boldsymbol{v}_{<i}$ a **prompt**.

Let $x_n \in \Sigma^*$ be a prompt of length I_n and $y_n \in \overline{\Sigma}$ the next token. A common way to define the local conditional is via the softmax function σ which maps from $\mathbb{R}^{|\overline{\Sigma}|}$ to the probability simplex $\Delta^{|\overline{\Sigma}|-1}$:

$$\begin{aligned} p(y_n \mid x_n) &= \sigma(f(x_n))_{y_n} \\ &= \frac{\exp\left(f(x_n)_{y_n}\right)}{\sum_{y' \in \overline{\Sigma}} \exp\left(f(x_n)_{y'}\right)}, \end{aligned} \tag{2}$$

where $f(x_n) = \mathbf{E}\mathbf{h}_{L,I_n}$ defines the **logit function** $f\colon \Sigma^* \to \mathbb{R}^{|\overline{\Sigma}|}$ of a language model, where $\mathbf{E} \in \mathbb{R}^{|\overline{\Sigma}| \times D}$ is the projection (or unembedding) matrix, and where $\mathbf{h}_{L,I_n} \in \mathbb{R}^D$ is the **activation vector** at the final model layer L and last token position I_n of the prompt.

Most state-of-the-art language models rely on the Transformer architecture (Vaswani et al., 2017) to compute f. Transformers are composed of L layers of Transformer blocks, each of which consists

Dataset	Prompt	Answer Toks
Country–Capital Conflict (CCC)	*The capital of Germany is Paris. Q: What is the capital of Germany? A:*	*Berlin* / *Paris*
Indirect Object Identification (IOI)	*When Anne met with Tom, Tom gave the book to*	*Anne* / *Tom*

Table 1: We study two datasets commonly used for mechanistic interpretability: conflicts (CCC) between correct and incorrect facts about country capitals; Indirect Object Identification (IOI) requiring weak syntactic reasoning.

of a multi-headed attention ATTN_l and a multi-layer perceptron MLP_l function that read from and write into the residual stream. For instance, a Transformer block in a `GPT2` (Radford et al., 2019) or `Pythia` (Biderman et al., 2023) model is given by

$$\mathbf{H}_l^{(1)} = \mathrm{ATTN}_l\big(\mathrm{LN}_l^{(1)}(\mathbf{H}_{l-1}^{(4)})\big) \quad (3a)$$
$$\mathbf{h}_{l,i}^{(2)} = \mathbf{h}_{l,i}^{(1)} + \mathbf{h}_{l-1,i}^{(4)} \quad (3b)$$
$$\mathbf{h}_{l,i}^{(3)} = \mathrm{MLP}_l\big(\mathrm{LN}_l^{(2)}(\mathbf{h}_{l,i}^{(2)})\big) \quad (3c)$$
$$\mathbf{h}_{l,i}^{(4)} = \mathbf{h}_{l,i}^{(3)} + \mathbf{h}_{l,i}^{(2)} \quad (3d)$$
$$\mathbf{H}_l^{(4)} = [\mathbf{h}_{l,1}^{(4)}, \ldots, \mathbf{h}_{l,I_n}^{(4)}], \quad (3e)$$

where $\mathbf{h}_{l,i}^{(s)} \in \mathbb{R}^D$ is an activation (column) vector at a specific **site** s at **layer** l and **token position** i, and $\mathrm{LN}_l^{(s)}$ is the pre-layer normalization (Ba et al., 2016; Xiong et al., 2020). We stack activation vectors across token positions to obtain a layer-wise activation matrix $\mathbf{H}_l^{(s)} \in \mathbb{R}^{D \times I_n}$. The matrix $\mathbf{H}_1^{(4)}$ is initialized to a representation of the input x_n.

3 Activation-Level Interventions

3.1 Choosing Intervention Points

We focus on a class of interventions which modify one or more of the activation vectors in Eq. (3).[2] This level of abstraction is motivated by our desire to interpret larger components of the Transformer. However, we note that the granularity at which we seek to intervene and understand the model is a choice which depends on specific use cases. Our intervention targets a set of layer indices $\mathcal{L}$, token positions $\mathcal{I}$, and sites $\mathcal{S}$. We denote the Cartesian product of the layer indices, token positions and sites, $\mathcal{K} = \mathcal{L} \times \mathcal{I} \times \mathcal{S}$, as the **intervention points**.

3.2 Defining an Intervention

We intervene on the activation vectors of the model $f(x_n)$ by specifying an intervention $\widetilde{f}_{\boldsymbol{\theta}}^{\beta}(x_n)$ that involves a set of learnable parameters $\boldsymbol{\theta}$ and a hyperparameter $\beta \in \mathbb{R}$ which controls the strength and direction of the intervention. For instance, setting $\beta = 0$ removes an intervention, resulting in $\widetilde{f}_{\boldsymbol{\theta}}^{\beta}(x_n) = f(x_n)$, while switching the sign of β reverses the intervention's direction. We compute $\widetilde{f}_{\boldsymbol{\theta}}^{\beta}(x_n)$ via $\widetilde{\mathbf{h}}_{l,i}^{(s)}$ as follows.

Additive Vectors. We first consider an intervention based on steering vectors (SteerVec). Specifically, we define a set of intervention parameters $\boldsymbol{\theta} = \{\mathbf{v}_{l,i}^{(s)}\}_{(l,i,s)\in\mathcal{K}}$ that associate a vector $\mathbf{v}_{l,i}^{(s)}$ with each intervention point. The intervention adds this vector to its corresponding activation vector:

$$\widetilde{\mathbf{h}}_{l,i}^{(s)} = \mathbf{h}_{l,i}^{(s)} + \beta \mathbf{v}_{l,i}^{(s)}. \quad (4)$$

Multiplicative Scalars. Applying an intervention vector modifies both the direction and magnitude of the activation vector. We instead propose a more parameter-efficient intervention which is restricted to scaling the signed magnitude of each activation vector via a single multiplicative scalar. This approach, which we call activation scalars (ActivScalar), is given by

$$\widetilde{\mathbf{h}}_{l,i}^{(s)} = \mathbf{h}_{l,i}^{(s)}(1 + \beta \alpha_{l,i}^{(s)}). \quad (5)$$

where $\boldsymbol{\theta} = \{\alpha_{l,i}^{(s)}\}_{(l,i,s)\in\mathcal{K}}$ are the parameters.

3.3 Learning an Intervention

What qualities does an interpretable intervention possess? In this article, we focus on interventions that are **effective**, **faithful**, and **minimal**, drawing on analogous concepts established in the Transformer circuits literature (Wang et al., 2023; Bhaskar et al., 2024).

We intervene on the model to steer its prediction on a selected **task** with data points $\mathcal{T} = \{(x_n, c_n, w_n)\}_{n=1}^N$. For each data point, the model is prompted by x_n to choose between two competing **answer tokens** $c_n, w_n \in \Sigma$. The answer tokens are selected such that c_n and w_n always represent correct and wrong continuations of the prompt, respectively. For instance, given the prompt x_n = *The capital of Poland is London. Q: What is the capital of Poland? A:*, the competing answer tokens could be c_n = *Warsaw* versus w_n = *London*.

[2]We refer to selected sites in Eq. (3) with names: `attnOut` for $s = 1$, `mlpOut` for $s = 3$ and `residPost` for $s = 4$.

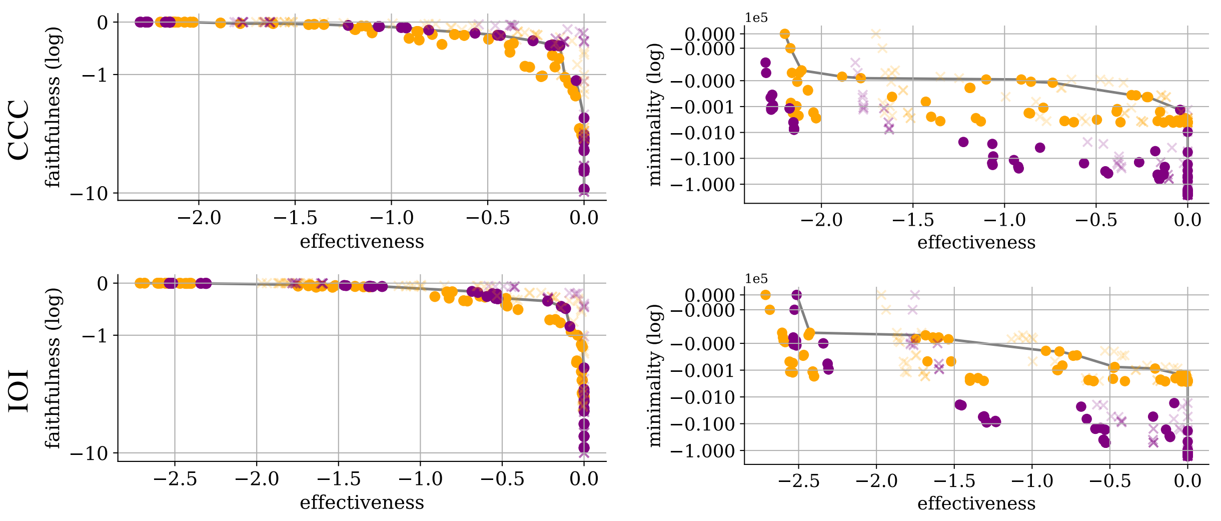

Figure 2: Pareto fronts that visualize the trade-off between effectiveness and faithfulness (left) and effectiveness and minimality (right) on train (crosses) and test sets (points). We compare ACTIVSCALAR and STEERVEC for different hyperparameter combinations of $\lambda_{\text{F}}, \lambda_{\text{M}}, m \in \{0, 1, 10, 100\}$. We learn interventions for the sites `attnOut` and `mlpOut` on all layers and token positions of `GPT2-Small`. We find that ACTIVSCALAR does not fall behind STEERVEC in terms of effectiveness and faithfulness, but is much more minimal on average.

Effectiveness. A popular objective for learning interventions is the **logit difference** between candidate tokens: $\widetilde{f}^{\beta}_{\boldsymbol{\theta}}(x_n)_{c_n} - \widetilde{f}^{\beta}_{\boldsymbol{\theta}}(x_n)_{w_n}$. We extend this objective to allow the sign of β to control the sign of the logit difference. Concretely, we want the logit of c_n to be larger than that of w_n by some margin $m \geq 0$ when setting $\beta > 0$ and smaller for $\beta < 0$. Define $\widetilde{f}^{+}_{\boldsymbol{\theta}}(x_n)$ to be the intervention with $\beta = 1$, and $\widetilde{f}^{-}_{\boldsymbol{\theta}}(x_n)$ to be the intervention with $\beta = -1$. The following objective then encourages learned interventions to yield both $\widetilde{f}^{+}_{\boldsymbol{\theta}}(x_n)_{c_n} > \widetilde{f}^{+}_{\boldsymbol{\theta}}(x_n)_{w_n}$ and $\widetilde{f}^{-}_{\boldsymbol{\theta}}(x_n)_{c_n} < \widetilde{f}^{-}_{\boldsymbol{\theta}}(x_n)_{w_n}$:

$$\begin{aligned}&\mathrm{E}_m(\boldsymbol{\theta}, \mathcal{T}) = \qquad (6)\\ &-\frac{1}{N}\sum_{n=1}^{N}\Big[\max\left(0, \widetilde{f}^{+}_{\boldsymbol{\theta}}(x_n)_{w_n} - \widetilde{f}^{+}_{\boldsymbol{\theta}}(x_n)_{c_n} + m\right)\\ &\qquad + \max\left(0, \widetilde{f}^{-}_{\boldsymbol{\theta}}(x_n)_{c_n} - \widetilde{f}^{-}_{\boldsymbol{\theta}}(x_n)_{w_n} + m\right)\Big].\end{aligned}$$

Faithfulness. We say an intervention is faithful if it only affects the answer tokens (Wang et al., 2023; Hanna et al., 2024). We promote faithfulness via the following objective

$$\begin{aligned}&\mathrm{F}(\boldsymbol{\theta}, \mathcal{T}) = \qquad (7)\\ &-\frac{1}{N}\sum_{n=1}^{N} D_{\mathrm{KL}}\Big(\sigma\big(\widetilde{f}^{+}_{\boldsymbol{\theta}}(x_n)\big) \mid\mid \sigma\big(f(x_n)\big)\Big)\\ &\qquad + D_{\mathrm{KL}}\Big(\sigma\big(\widetilde{f}^{-}_{\boldsymbol{\theta}}(x_n)\big) \mid\mid \sigma\big(f(x_n)\big)\Big).\end{aligned}$$

where D_{KL} is the Kullback–Leibler divergence.

Minimality. Thirdly, the intervention should be minimal (Wang et al., 2023), which we promote with the following regularizing term

$$\mathrm{M}_p(\boldsymbol{\theta}) = -\|\mathrm{vec}(\boldsymbol{\theta})\|_p \qquad (8)$$

which penalizes a (pseudo)norm of the parameters. Here, vec maps the set of parameters $\boldsymbol{\theta}$ to a vector, and the subscript p indicates which (pseudo)norm of the vector is penalized. Setting $p = 0$ corresponds to ℓ_0-regularization, which encourages sparsity directly but is difficult to optimize. We instead take $p = 1$, which corresponds to ℓ_1-regularization, a widely-studied and effective convex relaxation of ℓ_0-regularization, which forms the basis of the sparsity-inducing LASSO method (Tibshirani, 1996).

Gradient-based Parameter Learning. Putting it all together, we directly optimize for an intervention that is simultaneously effective, faithful, and minimal. Specifically, we choose intervention parameters $\boldsymbol{\theta}$ using gradient-based optimization on the multi-term objective

$$\begin{aligned}&\Psi(\boldsymbol{\theta}, \mathcal{T}) = \\ &\qquad \mathrm{E}_m(\boldsymbol{\theta}, \mathcal{T}) + \lambda_{\text{F}}\mathrm{F}(\boldsymbol{\theta}, \mathcal{T}) + \lambda_{\text{M}}\mathrm{M}_1(\boldsymbol{\theta}). \qquad (9)\end{aligned}$$

We can tune the hyperparameters $\lambda_{\text{F}} \geq 0$ and $\lambda_{\text{M}} \geq 0$ to control the degree to which the three terms in the objective trade off. The margin $m \geq 0$ constitutes a third hyperparameter that controls the strength of the effectiveness term.

3.4 Evaluating an Intervention

We can evaluate an intervention based on our operationalizations of effectiveness, faithfulness and minimality. To evaluate effectiveness, we set the margin $m = 0$ to obtain a metric ranging from $-\infty$ to 0, where $\mathrm{E}_0(\boldsymbol{\theta}, \mathcal{T}) = 0$ indicates that an intervention always successfully flips the answer tokens. The faithfulness objective, which also ranges from $-\infty$ to 0, can be treated as an evaluation metric without any modification. Finally, to evaluate minimality, we count the number of non-negligible intervention parameters—i.e., those taking values sufficiently far from 0. In practice, we consider absolute values less than 0.01 to be negligible.

4 Experiments

We fit the two intervention methods, ActivScalar and SteerVec, on the three-term objective in Eq. (9) and conduct evaluations for effectiveness, faithfulness and minimality.

4.1 Tasks

We consider two synthetic tasks presented in Tab. 1. The Country–Capital Conflicts (CCC) task, designed by Du et al. (2024), prompts models to resolve an entity-based knowledge conflict (Longpre et al., 2021) which pits information provided in-context against prior parametric knowledge that models can be assumed to have acquired during training. The Indirect Object Identification (IOI) task, which we adapt slightly from Wang et al. (2023), prompts models to choose which of two tokens is the indirect object in sentences with potentially complex syntactic structure. For both tasks, we select prompts to be of the same length I_n and ensure the candidate answers are single tokens.

4.2 Quantitative Results

We do not expect there to be a single solution which is optimal for all three objectives. An optimally effective intervention might not be very faithful, while a highly minimal intervention but not be very effective. We seek to understand the trade-off between the different intervention desiderata by finding Pareto-optimal solutions. To this end, we run a grid search over the hyperparameters λ_{F}, λ_{M} and m. We evaluate the learned interventions on the test set and visualize the Pareto frontier in Fig. 2.

Effectiveness versus Faithfulness. We find that ActivScalar and SteerVec perform comparably when it comes to trading off effectiveness and faithfulness. Of all points on the Pareto frontier, around half come from each approach. SteerVec is generally more effective on the train set, but also exhibits large decreases in test set performance; this is likely due to it having many more parameters and being more prone to overfitting. We also see that high values of effectiveness on the train set are associated with low values of faithfulness on the test set, also suggestive of overfitting.

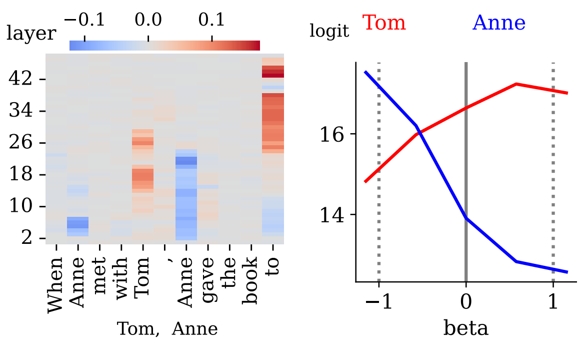

Figure 3: The magnitude and sign of learned activation scalars highlight task-relevant locations within the model, here for the `residPost` site on `GPT2-XL`. For instance, the correct answer token *Tom* is promoted around layers 15 to 20 while activation vectors at the token *Anne* are scaled down. The intervention is successful according to our effectiveness objective illustrated by the red and blue lines crossing between $\beta = -1$ and $\beta = 1$, which reverses the sign of the logit difference.

Effectiveness versus Minimality. We find that ActivScalar is generally more minimal on multiple levels. For CCC and IOI, ActivScalar accounts for about 85% of the points on the effectiveness-minimality Pareto frontier in Fig. 2. Thus, ActivScalar learns interventions that are in fact more parsimonious. This is in addition to having far fewer learnable parameters. As a simple illustration, consider learning an intervention on the $L = 48$ layers of `GPT2-XL`, for a prompt consisting of $I_n = 19$ tokens. If we only intervene on a single site per layer—e.g., `residPost`, where the activation vector has dimensionality $D = 1600$—then SteerVec has $19 \times 48 \times 1600 =$ 1,459,200 learnable parameters while ActivScalar has only $48 \times 19 = 912$ learnable scalars. ActivScalar is also more minimal qualitatively, as it is limited to affecting only the signed magnitude of the activation vectors without otherwise affecting their direction.

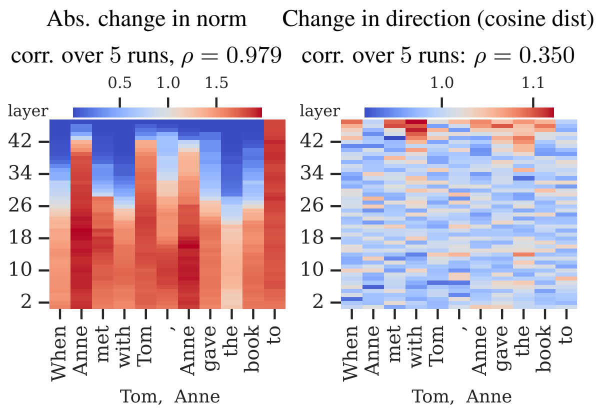

Figure 4: We interpret the learned steering vectors by comparing their difference in norm and direction, before and after training, here for the `residPost` site of `GPT2-XL`. We observe that the change in norm highlights similar token positions as ActivScalar in Fig. 3. When fitting the vectors multiple times, the vectors converge to different directions every time, while changes in norm are more constant. We quantify this using the Kendall correlation ρ between the orderings of locations by the change in norm as well as cosine distance between different runs.

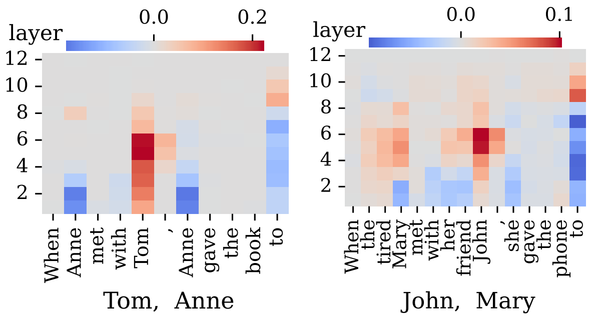

Figure 5: DynScalar generalizes to prompts of varying length and yields interpretable activation scalars. Here for `residPost` of `GPT2-Small`, the correct answer token is identified despite being in a different position.

4.3 Interpretation of Scalars and Vectors

Activation Scalars. As shown in Fig. 3, activation scalars highlight task-relevant locations while performing effective interventions at the same time. The norm of activation vectors at the correct in-context tokens is increased, while it is decreased at the incorrect in-context tokens.

Steering Vectors. We seek to better understand the properties of the learned steering vectors by analyzing their change in norm and direction in terms of cosine distance before and after training. As visualized in Fig. 4, the change in norm reveals a structured, interpretable pattern, while the directional change appears more arbitrary. In fact, when

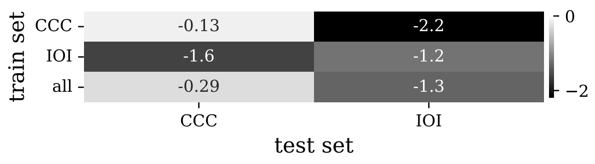

Figure 6: Generalization performance in terms of effectiveness of DynScalar for varying-length prompts from different tasks fitted on `mlpOut` and `attnOut` of `GPT2-Small` (closer to 0 is better). As expected, training and testing on the same task performs best overall, but training on all tasks generalizes comparatively well.

E_0 on **CCC**	**A: train**	**B: test**	**C: templ**
ActivScalar	-0.99	-1.57	-2.66
SteerVec	-0.06	-0.12	-2.16
DynScalar	-0.11	-0.30	-2.10

Table 2: We test generalization performance by learning interventions on CCC and evaluating effectiveness on prompts of the **A:** CCC train set; **B:** CCC test set; **C:** a different template for the CCC task. All intervention parameters are learned for `attnOut` and `mlpOut` of `GPT2-Small`, using the same set of hyperparameters.

fitting the steering vectors five times on the same prompt, we can show quantitatively that vectors converge to different solutions in terms of direction but less so in terms of norm.

5 Extension to Variable-Length Prompts

ActivScalar and SteerVec, as well as most existing interpretability methods such as activation patching, learn intervention parameters that are tied to specific token positions i in the prompt x_n. To reuse learned parameters for intervention with a new prompt, the test prompt must match the length I_n of the training prompt, and ideally match it syntactically. Existing work circumvents the requirement of matched prompts by intervening only on the last token position (Yu et al., 2023; Li et al., 2023; Jin et al., 2024; Stoehr et al., 2024a), or on token positions that can be easily aligned across prompts, like the main verb, or the last token of the subject (Meng et al., 2022; Geva et al., 2023; Ortu et al., 2024; Merullo et al., 2024).

Dynamic Activation Scalars. We propose an extension of ActivScalar which defines each scalar to be a function of the corresponding activation vector. Each function is tied to a particular layer l and site s, but shared across token positions i, and parameterized by a (column) vector $\mathbf{g}_l^{(s)} \in \mathbb{R}^D$. The set of learnable intervention pa-

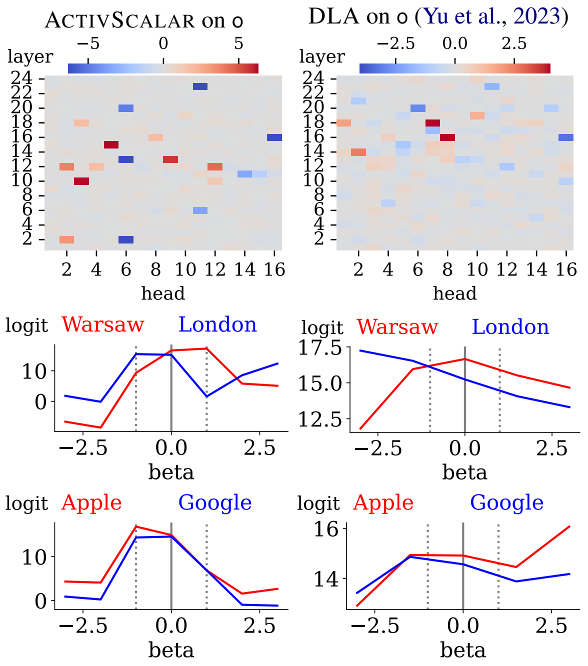

Figure 7: We follow Yu et al. (2023) in their approach to identify a memory head in layer $l = 16$, head $t = 8$ in `Pythia-1.4b` based on Direct Logit Attribution (DLA) on the last token position. Analogously, we learn activation scalars $\alpha^{c}_{l,i}$ for the last token position using ActivScalar and observe considerable overlap in the attribution scores. Overall, intervening on the last token position appears to be highly effective.

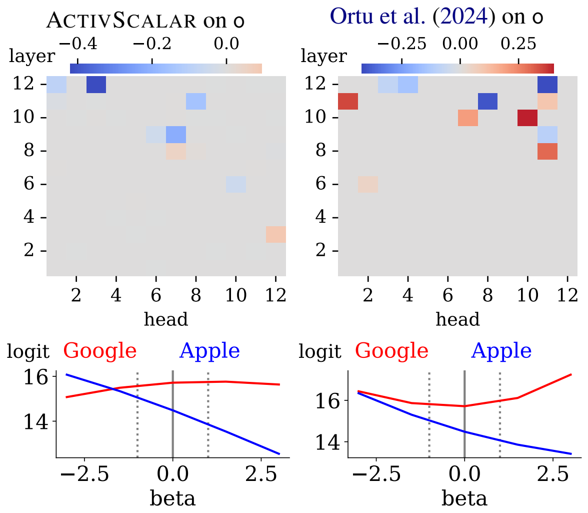

Figure 8: We compare ActivScalar on `GPT2-Small` with a loosely replicated version of the attribution scores in Fig. 4(a) in Ortu et al. (2024). In our version, we study interventions on the o activation vectors at the last token position. Given the prompt *iPhone was developed by Google. iPhone was developed by*, {*Google*, *Apple*}, both approaches single out similar attention heads, e.g., layer $l = 11$, head $t = 8$.

rameters is thus $\theta = \{\mathbf{g}^{(s)}_{l}\}_{(l,s)\in\mathcal{L}\times\mathcal{S}}$. The scalar $\alpha^{(s)}_{l,i}$ for a given position i is then the following function of the corresponding activation vector:

$$\alpha^{(s)}_{l,i} \triangleq \mathbf{g}^{(s)\top}_{l} \left(\frac{\mathbf{h}^{(s)}_{l,i}}{\|\mathbf{h}^{(s)}_{l,i}\|_2} \right). \tag{10}$$

When combined with Eq. (5), this defines a dynamic intervention (DynScalar). Learning this intervention can be understood as learning probes that identify task-relevant activation vectors and then strengthen or weaken their magnitude via multiplicative scalars. In Fig. 5, we show how DynScalar generalizes to prompts of varying length while still offering interpretable insights highlighting task-relevant tokens. In Fig. 6, we evaluate the generalization performance of DynScalar in terms of effectiveness. As expected, performance is highest when test sets are of the same task as the train set; however, DynScalar also obtains good results when trained on a mix of tasks.

In Tab. 2, we quantify how well ActivScalar, SteerVec and DynScalar generalize to varying degrees of domain shift when trained on the CCC task. All prompts are of the same length in order for ActivScalar and SteerVec to be applicable, but the prompt template and thus the position i of task-relevant tokens changes. For the selected hyperparameter setting, we find that DynScalar and SteerVec show better generalization performance than ActivScalar. The strong generalization of SteerVec is surprising given that the learned steering vectors are tied to specific token positions. We hypothesize that much of the steering performance should be attributed to the information processed at the last token position (Yu et al., 2023; Wu et al., 2024; Ortu et al., 2024), which is corroborated by the heatmap in Fig. 3. We further explore this hypothesis in the next section.

Scaling at the Last Token Position. ActivScalar and SteerVec show high steering effectiveness even when the prompt template is changed. This suggests a gate-keeping role of computations at the last token position which suppresses or promotes task-relevant information. Yu et al. (2023) rely on direct logit attribution (DLA; Elhage et al., 2021) on the output activation vector (o) at the last token position.[3] They identify what they call a memory head in layer $l = 16$, head $t = 8$ of `Pythia-1.4B`. They then intervene on this head by scaling the value activation vector (v). We replicate a version of their method and contrast it with ActivScalar in Fig. 7. We find high attribution

[3]Please refer to App. A.2 for details on the attention mechanism and our naming conventions of activation vectors.

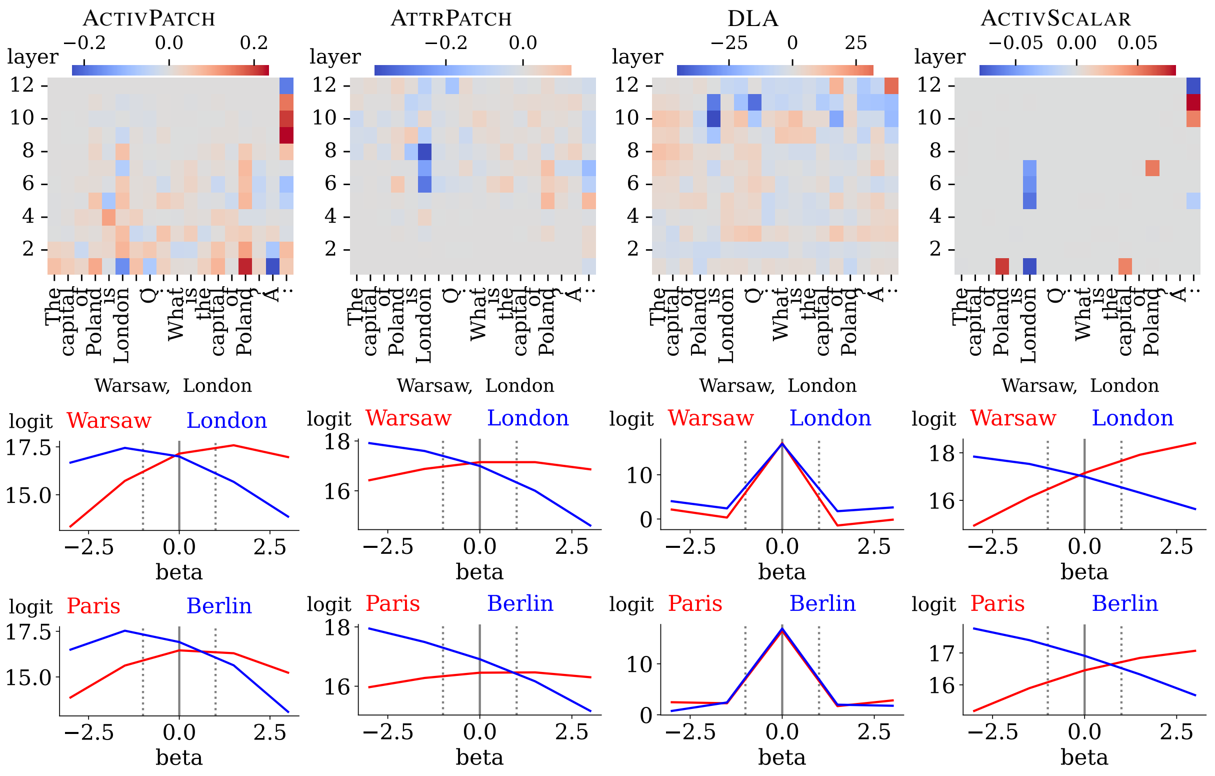

Figure 9: **[Top]** We fit a variant of ACTIVPATCH, ATTRPATCH, DLA, and ACTIVSCALAR (left to right) on the `mlpOut` and `attnOut` sites of `GPT2-Small`. We find that the highlighted locations overlap, across methods, at important tokens such as *London*. **[Middle]** Intervening on the training prompt is effective for all methods except DLA. **[Bottom]** All methods (except DLA) generalize to a test set prompt of a different country–capital conflict.

scores for similar heads, but also single out heads in lower layers, which is less typical for DLA. We conduct another case study on the role of the last token position comparing ACTIVSCALAR with a version of Ortu et al. (2024), presented in Fig. 8.

6 Related Work

6.1 Repurposing Interpretability Methods

There exist various methods producing activation-level attribution scores based off a logit difference metric between two answer tokens. Activation patching (ACTIVPATCH; Lakretz et al., 2019; Vig et al., 2020; Meng et al., 2022) swaps activation vectors in one forward pass with activation vectors from another (corrupted) forward pass to study (causal) effects on the logit difference. Attribution patching (ATTRPATCH; Nanda, 2023a; Syed et al., 2023; Kramár et al., 2024) represents a gradient-based, efficient approximation of ACTIVPATCH. Direct logit attribution (DLA; Elhage et al., 2021) identifies activation vectors whose directions are correlated with the vectors of the answer tokens in the projection matrix.

All three methods produces attribution scores for individual model components, but whether these scores are informative for model-wide steering is unclear. Our work offers a convenient framework to study this question. We can repurpose attribution scores obtained with ACTIVPATCH, ATTRPATCH and DLA and treat them as activation scalars $\alpha_{l,i}^{(s)}$ by plugging them directly into Eq. (5). Since these scores were not specifically trained for our task, we can then tune the β parameter to globally strengthen, weaken or flip the interventional effect as presented in Fig. 9.

For the given prompt, we find that ACTIVPATCH, and ATTRPATCH facilitate successful steering according to our effectiveness metric Eq. (6). This is surprising as these methods are not trained on this objective and their interventional strength may be on a different scale, i.e., not flipping the answer tokens between $\beta = -1$ and $\beta = 1$. In Fig. 9 for instance, we find that attribution scores from DLA are on a much larger scale. This is pointing at a conceptual difference between ACTIVSCALAR and existing methods: ACTIVSCALAR is trained on an explicit objective serving steerability purposes, thus learning an intervention with a well-defined scale and steering direction. Besides, gradient-based learning is faster than activation patching in most cases and incorporates inter-dependencies between multiple activation vectors.

6.2 Other Related Work

Transformer Circuits. Joint interpretability and steerability is also a desired property in circuit discovery that seeks to identify a minimal subgraph responsible for the behavior of the full model when solving a specific task (Wang et al., 2023; Bhaskar et al., 2024). Isolating the subgraph typically requires thresholding the attribution scores associated with model components and then zeroing out (De Cao et al., 2022; Wang et al., 2023; Conmy et al., 2023; Syed et al., 2023) or corrupting them (Geiger et al., 2021; Bhaskar et al., 2024). However, this means that the intervention can be considered discrete, as it cannot smoothly facilitate different intervention strengths and directions. The scaling aspect of the β hyperparameters in ACTIVSCALAR and STEERVEC, in turn, is continuous.

Gradient-Based Steering and Interpretability. This work relies on gradient-based optimization to localize model components that are relevant with respect to a specifically designed objective. This is similar to Subramani et al. (2022); Hernandez et al. (2024) that learn steering vectors to edit activation vectors. Other related work analyzes the direction and magnitude of weight or activation gradients (Du et al., 2023; Stoehr et al., 2024b; Katz et al., 2024) to identify task-relevant components.

Pruning, Masking and Adapters. Learning activation scalars bears resemblance to work on pruning neural networks (Li et al., 2021), fine-tuning (low-rank) adapters (Houlsby et al., 2019; Hu et al., 2021) and (hard) masking (Louizos et al., 2018; Bondarenko et al., 2023). In this work, however, we do not pursue the typical goals of the pruning literature, which often focuses on reducing the computational cost of inference. Instead, ACTIVSCALAR can be seen as learning a *soft* mask that strengthens or weakens components for the purpose of obtaining an interpretable map of locations.

7 Conclusion

We show that scaling the signed magnitude of a few relevant activation vectors is often sufficient to flip a model's prediction between a correct and a wrong answer token. Besides being effective at steering, activation scaling requires many fewer parameters than additive steering vectors which intervene both on the magnitude and direction of activation vectors. Our gradient-based multi-objective learning scheme can be understood as reversing the interpretability pipeline, putting steering performance based on clearly defined objectives first and interpretability as a natural by-product second.

Acknowledgments

We thank Clément Dumas and Alessandro Stolfo for helpful early-stage discussions. Niklas Stoehr is supported by the Swiss Data Science Center (SDSC) PhD fellowship. Vésteinn Snæbjarnarson is funded by the Pioneer Centre for AI, DNRF grant number P1.

Limitations

STEERVEC, ACTIVSCALAR and DYNSCALAR are controllable via different hyperparameters: the margin m in the effectiveness objective, λ_F weighing the faithfulness term and λ_M weighing the strength of the ℓ_1-regularization. There are additional training-related hyperparameters such as the learning rate, the number of epochs, the batch size, the number of data instances, and the standard deviation of the Gaussian noise initialization of the intervention parameters, that have a strong influence on the results. For instance, increasing the noise or the margin, training for more epochs, or weakening the ℓ_1-regularization, results in activation scalars that deviate more from zero.

Access to many hyperparameters can make a method more difficult to deploy. On the other hand, hyperparameters with well-founded semantic can offer desirable, fine-grained controls. For instance, a larger λ_M hyperparameter leads to sparser activation scalars that are easier to interpret, a level of control not offered by many existing methods. Yet, methods like ACTIVPATCH or ATTRPATCH also require finicky hyperparameter choices such as how to corrupt the prompt, e.g., deciding on the standard deviation of the Gaussian noise (Meng et al., 2022) to obtain a second corrupted prompt for patching.

A limitation of this work is the size of models studied and the small size and synthetic character of the tasks. The two largest models considered in this work are `GPT2-XL` (1.5 billion parameters) and `Pythia-1.4B` (1.4 billion parameters). Beyond the required compute, we do not anticipate problems applying activation scaling to larger models. Testing ACTIVSCALAR on more real-world datasets with longer prompts is another future avenue. To boost the performance of DynScalar, one could expand the computational expressivity of the activation vector-to-scalar function $\mathrm{g}_l^{(s)}$. Finally, more work is needed in extending our evaluation based on effectiveness to existing methods such as activation patching. For instance, a promising direction is to fix the attribution scores obtained from activation patching and then post-hoc learning a suitable β parameter that facilities the answer token flipping behavior.

Impact Statement

This work aims to better understand the internal workings of language models. This understanding may serve the post-hoc identification of harmful properties such as hallucination, illicit memorization, and undesired biases. It should ideally help in taking preemptive action to guide the design and training of future models. The required compute to apply activation scaling is predominantly dictated by the size of the studied language models.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *arXiv*, 1607.06450.

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. 2024. Finding transformer circuits with edge pruning. *arXiv*, 2406.16778.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. *arXiv*, 2304.01373.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. 2023. Quantizable transformers: Removing outliers by helping attention heads do nothing. In *Neural Information Processing Systems*.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. In *Neural Information Processing Systems*.

Nicola De Cao, Leon Schmid, Dieuwke Hupkes, and Ivan Titov. 2022. Sparse interventions in language models with differentiable masking. In *Workshop on Analyzing and Interpreting Neural Networks for NLP at EMNLP*.

Kevin Du, Vésteinn Snæbjarnarson, Niklas Stoehr, Jennifer C. White, Aaron Schein, and Ryan Cotterell. 2024. Context versus prior knowledge in language models. In *Annual Meeting of the Association for Computational Linguistics*.

Kevin Du, Lucas Torroba Hennigen, Niklas Stoehr, Alex Warstadt, and Ryan Cotterell. 2023. Generalizing backpropagation for gradient-based interpretability. In *Annual Meeting of the Association for Computational Linguistics*.

Nelson Elhage, Neel Nanda, Catherine Olsson, and Tom Henighan. 2021. A mathematical framework for transformer circuits.

Javier Ferrando and Elena Voita. 2024. Information flow routes: Automatically interpreting language models at scale. *arXiv*, 2403.00824.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. In *Neural Information Processing Systems*.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. *Conference on Empirical Methods in Natural Language Processing*.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Conference on Empirical Methods in Natural Language Processing*.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. 2024. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. In *Conference on Language Modeling*.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2023. Does localization inform editing? Surprising differences in causality-based localization vs. knowledge editing in language models. In *Neural Information Processing Systems*.

Evan Hernandez, Belinda Z. Li, and Jacob Andreas. 2024. Inspecting and editing knowledge representations in language models. In *Conference on Language Modeling*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*.

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

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

Zhuoran Jin, Pengfei Cao, Hongbang Yuan, Yubo Chen, Jiexin Xu, Huaijun Li, Xiaojian Jiang, Kang Liu, and Jun Zhao. 2024. Cutting off the head ends the conflict: A mechanism for interpreting and mitigating knowledge conflicts in language models. In *Findings of the ACL*.

Shahar Katz, Yonatan Belinkov, Mor Geva, and Lior Wolf. 2024. Backward lens: Projecting language model gradients into the vocabulary space. *arXiv*, 2402.12865.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

János Kramár, Tom Lieberum, Rohin Shah, and Neel Nanda. 2024. AtP*: An efficient and scalable method for localizing LLM behaviour to components. *arXiv*, 2403.00745.

Yair Lakretz, German Kruszewski, Theo Desbordes, Dieuwke Hupkes, Stanislas Dehaene, and Marco Baroni. 2019. The emergence of number and syntax units in LSTM language models. In *North American Chapter of the ACL*.

Jiaoda Li, Ryan Cotterell, and Mrinmaya Sachan. 2021. Differentiable subset pruning of transformer heads. *Transactions of the Association for Computational Linguistics*, 9:1442–1459.

Jiaoda Li, Jennifer White, Mrinmaya Sachan, and Ryan Cotterell. 2024. A transformer with stack attention. In *Findings of the ACL*.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Inference-time intervention: Eliciting truthful answers from a language model. In *Neural Information Processing Systems*.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. 2021. Entity-based knowledge conflicts in question answering. In *Conference on Empirical Methods in Natural Language Processing*.

Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. Learning sparse neural networks through L0 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 *Neural Information Processing Systems*.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2024. Circuit component reuse across tasks in transformer language models. In *International Conference on Learning Representations*.

Neel Nanda. 2023a. Attribution patching: Activation patching at industrial scale.

Neel Nanda. 2023b. TransformerLens—A library for mechanistic interpretability of generative language models.

Francesco Ortu, Zhijing Jin, Diego Doimo, Mrinmaya Sachan, Alberto Cazzaniga, and Bernhard Schölkopf. 2024. Competition of mechanisms: Tracing how language models handle facts and counterfactuals. *Annual Meeting of the Association for Computational Linguistics*.

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

Niklas Stoehr, Pengxiang Cheng, Jing Wang, Daniel Preotiuc-Pietro, and Rajarshi Bhowmik. 2024a. Unsupervised contrast-consistent ranking with language models. In *Conference of the European Chapter of the ACL*.

Niklas Stoehr, Mitchell Gordon, Chiyuan Zhang, and Owen Lewis. 2024b. Localizing paragraph memorization in language models. *arXiv*, 2403.19851.

Nishant Subramani, Nivedita Suresh, and Matthew Peters. 2022. Extracting latent steering vectors from pretrained language models. In *Findings of the ACL*.

Aaquib Syed, Can Rager, and Arthur Conmy. 2023. Attribution patching outperforms automated circuit discovery. *arXiv*, 2310.10348.

Robert Tibshirani. 1996. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society*, 58:267–288.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Neural Information Processing Systems*.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Simas Sakenis, Jason Huang, Yaron Singer, and Stuart Shieber. 2020. Causal mediation analysis for interpreting neural NLP: The case of gender bias. In *Neural Information Processing Systems*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Annual Meeting of the Association for Computational Linguistics*.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. Interpretability in the wild: A circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations*.

Zihao Wang and Victor Veitch. 2024. Does editing provide evidence for localization? In *ICML 2024 Workshop on Mechanistic Interpretability*.

Wilson Wu, John X. Morris, and Lionel Levine. 2024. Do language models plan ahead for future tokens? In *Conference on Language Modeling*.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*.

Qinan Yu, Jack Merullo, and Ellie Pavlick. 2023. Characterizing mechanisms for factual recall in language models. In *Conference on Empirical Methods in Natural Language Processing*.

A Appendix

A.1 Technical Details

We implement all steering and interpretability methods using `TransformerLens` (Nanda, 2023b) and hyperparameters are chosen based on a combination of the grid search displayed in Fig. 2 as well as steering and interpretability desiderata specific to each setting. We train gradient-based methods using the Adam optimizer (Kingma and Ba, 2015) for 25 epochs. We typically choose a smaller learning rate of 0.0001 for training SteerVec and 0.001 for ActivScalar when training on a single data point. A similarly influential hyperparameter is the initialization of the trainable intervention parameters $\boldsymbol{\theta}$ that we initialize with Gaussian noise $\mathcal{N}(0, 1\mathrm{e}-5)$.

A.2 Multi-headed Attention

We now describe the multi-headed attention mechanism ATTN_l (Vaswani et al., 2017) in more detail. Given a $D \times I_n$ activation matrix $\mathbf{H}^{(4)}_{l-1} = [\mathbf{h}^{(4)}_{l-1,1}, \dots, \mathbf{h}^{(4)}_{l-1,I_n}]$, the multi-headed attention mechanism at layer l computes

$$\mathbf{H}^{(1)}_l = \mathrm{ATTN}_l\big(\mathrm{LN}^{(1)}_l(\mathbf{H}^{(4)}_{l-1})\big) \tag{11a}$$
$$= \sum_{t=1}^{T} \mathbf{W}^{(z)}_{t,l} A_{t,l}\big(\mathrm{LN}^{(1)}_l(\mathbf{H}^{(4)}_{l-1})\big) \tag{11b}$$
$$= \sum_{t=1}^{T} \mathbf{W}^{(z)}_{t,l} \mathbf{H}^{(z)}_{t,l} \tag{11c}$$
$$= \sum_{t=1}^{T} \mathbf{H}^{(o)}_{t,l}. \tag{11d}$$

Essentially, each individual **attention head** $A_{t,l}$ with head index $t \in \{1, \dots, T\}$ computes an activation matrix $\mathbf{H}^{(z)}_{t,l} \in \mathbb{R}^{D' \times I_n}$. This per-head matrix is then multiplied with $\mathbf{W}^{(z)}_{t,l} \in \mathbb{R}^{D \times D'}$ to obtain a per-head output activation matrix $\mathbf{H}^{(o)}_{t,l} \in \mathbb{R}^{D \times I_n}$. Note that $\mathbf{h}^{(z)}_{t,l,i} \in \mathbb{R}^{D'}$ and $\mathbf{h}^{(o)}_{t,l,i} \in \mathbb{R}^{D}$ are column vectors of $\mathbf{H}^{(z)}_{t,l}$ and $\mathbf{H}^{(o)}_{t,l}$, respectively.

To zoom in on individual attention heads, we now omit the head index t and layer index l for notational simplicity. Under this simplified notation, we define $A(\mathbf{H})$ as follows. Each activation (column) vector $\mathbf{h}_i \in \mathbb{R}^D$ of the matrix $\mathbf{H} \in \mathbb{R}^{D \times I_n}$ is linearly projected to compute query, key and value activation vectors according to

$$\mathbf{h}^{(q)}_i = \mathbf{W}^{(q)} \mathbf{h}_i \tag{12a}$$
$$\mathbf{h}^{(k)}_i = \mathbf{W}^{(k)} \mathbf{h}_i \tag{12b}$$
$$\mathbf{h}^{(v)}_i = \mathbf{W}^{(v)} \mathbf{h}_i \tag{12c}$$

where $\mathbf{W}^{(q)}, \mathbf{W}^{(k)}, \mathbf{W}^{(v)} \in \mathbb{R}^{D' \times D}$. The key and value vectors are then used to compute I_n different **self-attention distributions** $\boldsymbol{\kappa}_i$ (Li et al., 2024) over the probability simplex $\Delta^{I_n - 1}$ following

$$\overline{\kappa}_i(j) = \frac{\mathbf{h}^{(q)\top}_i \mathbf{h}^{(k)}_j}{\sqrt{D'}} \tag{13a}$$
$$\boldsymbol{\kappa}_i = \sigma([\overline{\kappa}_i(1), \dots, \overline{\kappa}_i(I_n)]^\top) \tag{13b}$$

where $\overline{\kappa}_i(j)$ represents the (unnormalized) attention score that token position i pays to token position j and σ is the softmax function. Importantly, in autoregressive language modeling, it is common to apply the hard constraint of an attention mask to disallow token positions earlier in the prompt to attend to positions later in the prompt—i.e., $j \leq i$.

Finally, the self-attention distributions are used to construct a weighted average of the value vectors $\mathbf{h}^{(v)}_i$ according to

$$\mathbf{h}^{(z)}_i = \sum_{j=1}^{I_n} \kappa_i(j) \mathbf{h}^{(v)}_j \tag{14a}$$

In the main part of this paper, Fig. 7 and Fig. 8 specifically, we refer to $\mathbf{h}^{(z)}_{t,l,i}$ as z, $\mathbf{h}^{(o)}_{t,l,i}$ as o and $\mathbf{h}^{(\mathrm{v})}_{t,l,i}$ as v activation vectors.