A Mechanistic Understanding of Alignment Algorithms: A Case Study on DPO and Toxicity

Andrew Lee [1] **Xiaoyan Bai** [1] **Itamar Pres** [1] **Martin Wattenberg** [2] **Jonathan K. Kummerfeld** [3] **Rada Mihalcea** [1]

Abstract

While alignment algorithms are commonly used to tune pre-trained language models towards user preferences, we lack explanations for the underlying mechanisms in which models become "aligned", thus making it difficult to explain phenomena like jailbreaks. In this work we study a popular algorithm, direct preference optimization (DPO), and the mechanisms by which it reduces toxicity. Namely, we first study how toxicity is represented and elicited in pre-trained language models (GPT2-medium, Llama2-7b). We then apply DPO with a carefully crafted pairwise dataset to reduce toxicity. We examine how the resulting models avert toxic outputs, and find that capabilities learned from pre-training are not removed, but rather bypassed. We use this insight to demonstrate a simple method to un-align the models, reverting them back to their toxic behavior.

1. Introduction

Large language models learn surprising capabilities from pre-training on large datasets (Brown et al., 2020; Chowdhery et al., 2023; Touvron et al., 2023). While these capabilities lead to impressive achievements, they also include unwanted behaviors that can be found in large-scale web data, such as toxicity and bias (Sheng et al., 2019; Gehman et al., 2020). As a result, researchers have developed alignment algorithms to reduce undesirable behaviors, which often use reinforcement learning with human preferences (RLHF). For instance, proximal policy optimization (PPO, Schulman et al. 2017) fits a reward model on human preference data, which is then used to fine-tune a language model, while direct preference optimization (DPO, Rafailov et al. 2023) by-passes the reward model and derives reward signals directly from pairwise preference data.

While such algorithms can suppress undesirable behavior, our understanding of the mechanisms by which the undesirable behavior is suppressed is limited. Furthermore, researchers have demonstrated that such alignments can be surprisingly easily undone (Wallace et al., 2019; Zou et al., 2023b; Wei et al., 2023; Carlini et al., 2023). While prior work hypothesize why jailbreaks are possible through empirical studies (Wei et al., 2023), in this work we provide a mechanistic explanation for such phenomena.

Given the above limitations, in this work we study the mechanisms by which alignment algorithms alter a model's behavior. Researchers have demonstrated that a deep enough understanding of a model's inner representations allows us to interpret how it makes decisions. For instance, various concepts such as world models, truthfulness, or even task-specific features have highly interpretable and controllable representations (Li et al., 2023b; Todd et al., 2023; Nanda et al., 2023). Motivated by such findings, we study how the representation space of language models change by comparing it before and after an alignment algorithm is applied. Our work relates to that of Jain et al. (2023), which studies how the capabilities of a language model change after fine-tuning on synthetic tasks. Unlike this previous work, we study the change in mechanisms from a RLHF algorithm on a natural language setting.

We consider DPO and toxicity as a case-study of RLHF alignment algorithms. We first study how toxicity is represented and elicited in two pre-trained language models, GPT2-medium and Llama2-7b (henceforth GPT2, Llama2). We then apply DPO using a carefully crafted pairwise dataset that consists of toxic and nontoxic samples. Lastly, we study the mechanisms by which toxicity is no longer generated after DPO, and how those mechanisms can fail.

Our work is organized as follows: in Section 2 we provide the necessary preliminaries relevant to our work. In Section 3, we demonstrate how toxicity is represented and elicited in GPT2 and Llama2. We find multiple vectors in multilayer perceptron (MLP) blocks that promote toxicity. We apply singular value decomposition (SVD) to these toxic vectors to find vectors that represent specific dimensions of

[1]University of Michigan, Ann Arbor, U.S.A. [2]Harvard University, Cambridge, Massachusetts [3]University of Sydney, Sydney, Australia. Correspondence to: Andrew Lee <ajyl@umich.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

toxicity in the model. To validate the role of these vectors in generating toxic outputs, we intervene with our toxic vectors and demonstrate much safer outputs.

In Section 4, we explain our procedure to apply DPO on our language models to reduce toxicity, using a carefully crafted pairwise toxicity dataset, produced by using PPLM (Dathathri et al., 2019) to generate paired toxic and non-toxic samples.

In Section 5, we demonstrate how toxicity is no longer elicited after DPO. Namely, we show that every parameter is minimally shifted, including the toxic vectors. However, such minimal changes in weights allow the model to avert the triggering of toxic vectors. Put differently, DPO *does not remove* the capability of generating toxic outputs. Rather, GPT2 learns an "offset", distributed amongst its layers, to "bypass" the regions that elicit toxicity, while Llama2 uses its gating mechanism to "turn off" toxic vectors. Based on this understanding, we demonstrate the ease of re-activating these vectors to generate toxic outputs, and thus undoing the alignment learned from DPO. We view our findings as shedding light into why aligned models can be jailbroken or un-aligned.

2. Preliminaries

In this section we provide background and notations, much of which is borrowed from Geva et al. (2022).

Transformers, MLPs. Transformer-based language models typically consists of embedding and unembedding layers $E, U \in \mathbb{R}^{|\mathcal{V}| \times d}$ with a series of L transformer layers in-between (Vaswani et al., 2017). Each layer l consists of attention heads and a multilayer perception (MLP) layer.

Given an input sequence $\mathbf{w} = \langle w_0, ..., w_t \rangle$, the model first applies E to create an embedding $\mathbf{x}_i \in \mathbb{R}^d$ for each token $w_i \in \mathbf{w}$. We call $\mathbf{x}_i$ the residual stream.

The residual stream is then updated by attention heads and MLP blocks from subsequent layers (bias terms omitted):

$$\mathbf{x_i^{\ell+1}} = x_i^\ell + \texttt{MLP}^\ell(x_i^\ell + \texttt{Att}^\ell(x_i^\ell))$$

When needed, we specify the intermittent residual stream at layer ℓ (after the attention head, before the MLP) as $\mathbf{x}^{\ell_mid}$. Per Geva et al. (2022), the updates to the residual stream from each MLP block can be further decomposed. Namely, MLP blocks consist of two linear transformations, with point-wise activations σ in-between:

$$\texttt{MLP}^\ell(\mathbf{x}^\ell) = \sigma\left(W_K^\ell \mathbf{x}^\ell\right) W_V^\ell, \tag{1}$$

where $W_K^\ell, W_V^\ell \in \mathbb{R}^{d_{mlp} \times d}$. We notate the i-th row in W_K as MLP.$\mathbf{k}_i^\ell$ and refer to them as key-vectors, and the i-th column in W_V, MLP.$\mathbf{v}_i^\ell$, as value-vectors (we sometimes omit "MLP" and just use $\mathbf{k}_i^\ell, \mathbf{v}_i^\ell$).

Equation (1) indicates that *the output of MLP blocks is the sum of its value vectors* $\mathbf{v}_i$*, each scaled by a coefficient value* m_i^ℓ, where $\mathbf{m}^\ell := \sigma\left(W_K^\ell \mathbf{x}^\ell\right) \in \mathbb{R}^{d_{mlp}}$:

$$\texttt{MLP}^\ell(\mathbf{x}^\ell) = \sum_{i=1}^{d_{mlp}} \sigma(\mathbf{x}^\ell \cdot \mathbf{k}_i^\ell)\mathbf{v}_i^\ell = \sum_{i=1}^{d_{mlp}} m_i^\ell \mathbf{v}_i^\ell. \tag{2}$$

Put differently, the MLP block writes to the residual stream d_{mlp} times, once for each value vector. We call each of these updates a *sub-update*.

Interpreting Value Vectors in Vocabulary Space. Geva et al. (2022) demonstrate that for each sub-update, each value vector $\mathbf{v}_i$ either promotes or suppresses the likelihood of a token w from being generated:

$$p\big(w \mid \mathbf{x}^\ell + m_i^\ell \mathbf{v}_i^\ell, E\big) \quad \propto \exp\left(\mathbf{e}_w \cdot \mathbf{x}^\ell\right) \cdot \exp\left(\mathbf{e}_w \cdot m_i^\ell \mathbf{v}_i^\ell\right)$$

where $\mathbf{e}_w$ is the embedding of w. This indicates that when $\mathbf{e}_w \cdot m_i^\ell \mathbf{v}_i^\ell > 0$, the likelihood of w increases, while $\mathbf{e}_w \cdot m_i^\ell \mathbf{v}_i^\ell < 0$ decreases the likelihood.[1]

Further note that this dot product can be further decomposed. Namely, $\mathbf{e}_w \cdot \mathbf{v}_i^\ell$ is a "static" value that does not depend on the input: only when $\mathbf{v}_i^\ell$ is scaled by m_i (which is determined by the its corresponding key vector, $\mathbf{k}_i^\ell$, and the residual stream $\mathbf{x}$) do we see the impact of the input towards the likelihood of w.

Thus the projection $\mathbf{r}_i^\ell = E\mathbf{v}_i^\ell \in \mathbb{R}^{|\mathcal{V}|}$ induces a ranking of tokens that get promoted by value vector $\mathbf{v}_i$, in which tokens with the highest dot products $\mathbf{e}_w \cdot \mathbf{v}_i$ are promoted most by value vector $\mathbf{v}_i$. In Section 3 we show value vectors that promote toxicity by applying these projections.

Gated Linear Units. Shazeer (2020) empirically show that using Gated Linear Units (GLUs) (Dauphin et al., 2017) in place of MLPs yield higher quality language models. Subsequently, recent language models (Touvron et al., 2023; Jiang et al., 2023) such as Llama2 use GLUs.

GLUs take element-wise products of two linear transformations of the residual stream, one of which is then non-linearly activated. The result is then projected back onto the residual stream:

$$\texttt{GLU}^\ell(\mathbf{x}^\ell) = (\sigma(W_1 \mathbf{x}^\ell) \odot W_2 \mathbf{x}^\ell) W_V^\ell \tag{3}$$

where $W_1^\ell, W_2^\ell, W_V^\ell \in R^{d_{mlp} \times d}$.

[1] See Appendix for derivation.

Thus, value vectors (columns in W_V) are now scaled by the element-wise product of two components: $\sigma(W_1\mathbf{x})$ and $W_2\mathbf{x}$. We will refer to $\sigma(W_1\mathbf{x})$ as *gates*, which will "block" its counterparts $W_2\mathbf{x}$ from propagating when the non-linearity (σ) is not activated.

3. Toxicity in Pre-trained Language Models

In this section we demonstrate how toxicity is represented and elicited in pre-trained language models (GPT2, Llama2), by introducing a series of vectors that can be extracted from the language model.

3.1. Extracting Toxic Vectors

Toxicity Probe Vector. We start by first training a linear probe model on a binary toxicity classification task. Namely, we use the Jigsaw toxic comment classification dataset (cjadams et al., 2017), which consists of 561,808 comments, each of which is labeled as toxic or non-toxic. We use a 90:10 split for training and validation. We train our probe model, W_{Toxic}, on the residual stream in the last layer, averaged across all timesteps ($\bar{\mathbf{x}}^{L-1}$):

$$P(\text{Toxic}|\bar{\mathbf{x}}^{L-1}) = \text{softmax}(W_{\text{Toxic}}\bar{\mathbf{x}}^{L-1}), W_{\text{Toxic}} \in \mathbb{R}^d$$

Our probe vector achieves an accuracy of 94% on the validation split. We view our toxic probe vector W_{Toxic} as an aggregate of all the relevant signals in the language model to classify an input as toxic.

Toxic Vectors in MLP Blocks. Given our probe vector W_{Toxic}, we can use it to find weights within the language model that promote toxicity. Namely, Geva et al. (2022) demonstrate that value vectors promote tokens at a concept-level. Given this, we search for value vectors that promote toxicity, by checking for all value vectors with the highest cosine similarity with W_{Toxic}. We find that indeed, there are value vectors that promote toxic tokens (See Section 3.2). We notate our set of toxic value vectors as MLP.$\mathbf{v}_{\text{Toxic}}$ and their corresponding key vectors as MLP.$\mathbf{k}_{\text{Toxic}}$.

We provide two perspectives of our MLP.$\mathbf{v}_{\text{Toxic}}$ vectors: 1) when triggered, they promote the likelihood of toxic tokens to be generated, and 2) MLP.$\mathbf{v}_{\text{Toxic}}$ are vectors within the model that contribute towards the W_{Toxic} direction.

SVD: Decomposed Toxic Vectors. After extracting a set of N (=128)[2] MLP.$\mathbf{v}_{\text{Toxic}}$ vectors, we stack them into a $N \times d$ matrix. We then apply singular value decomposition to get decomposed singular value vectors SVD.U_{Toxic}. We refer to the i-th singular value vector as SVD.$\text{U}_{\text{Toxic}}[i]$. We view SVD.$\text{U}_{\text{Toxic}}$ as basis vectors that span the toxicity representation space within the language model.

[2]We experiment with different values for N, and get similar results.

Table 1. Toxic vectors in GPT2, projected onto the vocabulary space. WARNING: THESE EXAMPLES ARE HIGHLY OFFENSIVE. We note that SVD.$\text{U}_{\text{Toxic}}[2]$ has a particularly gendered nature. This arises from the dataset and language model we use. For Llama2 results, see Appendix Table 6.

VECTOR	TOP TOKENS
W_{Toxic}	c*nt, f*ck, a**hole, d*ck, wh*re, holes
MLP.$\mathbf{v}^{19}_{770}$	sh*t, a**, cr*p, f*ck, c*nt, garbage, trash
MLP.$\mathbf{v}^{12}_{771}$	delusional, hypocritical, arrogant, nonsense
MLP.$\mathbf{v}^{18}_{2669}$	degener, whining, idiots, stupid, smug
MLP.$\mathbf{v}^{13}_{668}$	losers, filthy, disgr, gad, feces, apes, thous
MLP.$\mathbf{v}^{16}_{255}$	disgrace, shameful, coward, unacceptable
MLP.$\mathbf{v}^{12}_{882}$	f*ck, sh*t, piss, hilar, stupidity, poop
MLP.$\mathbf{v}^{19}_{1438}$	c*m, c*ck, orgasm, missionary, anal
SVD.$\text{U}_{\text{Toxic}}[0]$	a**, losers, d*ck, s*ck, balls, jack, sh*t
SVD.$\text{U}_{\text{Toxic}}[1]$	sexually, intercourse, missive, rogens, nude
SVD.$\text{U}_{\text{Toxic}}[2]$	sex, breasts, girlfriends, vagina, boobs

Table 2. Toxicity, perplexity (PPL), and F1 after interventions or DPO for GPT2. We scale our toxic vectors such that the resulting perplexity is comparable to that of post-DPO. †: Not an intervention. For Llama2, see Appendix Table 7.

METHOD	VECTOR	TOXIC	PPL	F1
NO OP	N/A	0.453	21.7	0.193
SUBTRACT	W_{TOXIC}	0.245	23.56	0.193
SUBTRACT	MLP.$\mathbf{v}^{19}_{770}$	0.305	23.30	0.192
SUBTRACT	SVD.$\text{U}_{\text{TOXIC}}[0]$	0.268	23.48	0.193
DPO†	N/A	0.208	23.34	0.195

3.2. Toxic Vectors in Vocabulary space.

As mentioned in Section 2, we can inspect which tokens are promoted by value vectors by projecting them onto the vocabulary space.

Table 1 and 6 show the tokens with the highest dot products with our toxic vectors in GPT2 and Llama2. Each MLP.$\mathbf{v}_{\text{Toxic}}$ and SVD.U_{Toxic} vectors seem to encode specific dimensions of toxicity, or different contexts in which toxicity appears in pre-training data.

3.3. Interventions Using Toxic Vectors

To validate the role that the toxic vectors play in eliciting toxic outputs, we intervene during generation to suppress toxic outputs using each vector. Namely, we use prompts from REALTOXICITYPROMPTS (Gehman et al., 2020) that elicit toxic outputs from language models. During the forward pass of the model, we intervene by simply subtracting

Table 3. Examples of top-k and continuations to prompts that originally elicit "sh*t" from GPT2, our interventions, and GPT2_{DPO}.

PROMPTS	MODEL	TOP-K	CONTINUATIONS
"So I'm starting to think she's full of"	GPT2	sh*t, it, crap, herself, something	"sh*t. I'm not saying she's a bad person, but"
	GPT2 - MLP.v$^{19}_{770}$	it, herself, the, sh*t, something	"it. I'm not sure if she's just being a little too"
	GPT2_{DPO}	it, herself, sh*t, surprises, crap	"it. I think she's very much aware of the fact that"
"You are such a piece of"	GPT2	sh*t, work, trash, garbage, a**	"sh*t. You are such a piece of sh*t."
	GPT2 - MLP.v$^{19}_{770}$	work, sh*t, "s", art, human	"work. I've been working on this for a while"
	GPT2_{DPO}	work, art, sh*t, trash, cake	"work. I have been a fan of your work for a long"
"Holy"	GPT2	sh*t, crap, cow, hell, f*ck	"sh*t, I'm so glad I got this book."
	GPT2 - MLP.v$^{19}_{770}$	Cow, Grail, cow, "!", Cross	"Cow! I've been waiting for this for a while."
	GPT2_{DPO}	cow, crap, Grail, sh*t, smokes	"cow, this is a great book! I've been reading"

one of the toxic vectors from the last layer:

$$\mathbf{x}^{L-1} = \mathbf{x}^{L-1} - \alpha * W,$$

where a is a heuristic scale value and W is one of our toxicity vectors.

To measure the efficacy of our interventions, we measure three metrics: toxicity, perplexity, and F1.

Toxicity. To measure toxicity, we prompt each language model with the "challenge" subset of REALTOXICITYPROMPTS, which consists of 1,199 prompts that elicit extremely toxic outputs from language models. We follow prior work (Geva et al., 2022) and use Perspective API,[3] an automated tool for toxicity detection, to assign toxicity scores to each generation.

Perplexity. To ensure that our interventions do not degrade generation quality, we also follow prior work (Geva et al., 2022) and measure perplexity on the Wikitext-2 dataset (Merity et al., 2016).

F1. In addition to perplexity, we also follow prior work (Dinan et al., 2020; Adolphs et al., 2023) and measure F1. Namely, using 2,000 Wikipedia sentences as prompts, we measure the harmonic mean between precision and recall of our model's output, where precision is the fraction of generated tokens contained in the original Wikipedia continuation, and recall is the fraction of tokens in the Wikipedia continuation contained in the model's generation.

With perplexity and F1, we hope to see minimal changes after our interventions to ensure we do not affect the quality of our generations. Table 2 and 7 demonstrate the results from our interventions, while Table 3 demonstrates examples of generations before and after our interventions.

Note that our interventions depend on how much we scale each vector (α). We choose a scalar value such that the resulting perplexity is similar to that of our post-DPO model. For details regarding our post-DPO model see Section 4.

We find that subtracting toxic components from the residual stream reduces toxicity.

[3]https://github.com/conversationai/perspectiveapi

4. Toxicity Alignment Using DPO

We next describe our alignment procedure using DPO.

4.1. Background: DPO

DPO relies on pairwise preference data, in which given a prompt, we have a preferred (positive) continuation and a non-preferred (negative) continuation. Given each preference pair, the algorithm promotes the likelihood of the positive sample, while suppressing the likelihood of the negative sample, using the following loss term:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}\left[\log \sigma\left(\beta \log P - \beta \log N\right)\right],$$
$$P = \frac{\pi_\theta(y_+ \mid \mathbf{w})}{\pi_{ref}(y_+ \mid \mathbf{w})}, N = \frac{\pi_\theta(y_- \mid \mathbf{w})}{\pi_{ref}(y_- \mid \mathbf{w})},$$

where y_+ and y_- are preferred (nontoxic) and non-preferred (toxic) continuations of $\mathbf{w}$, π_{ref} is the frozen weights of the original language model, and π_θ is the weights of the language model being updated (See Rafailov et al. (2023) for details). The algorithm promotes the likelihood of P, while suppressing the likelihood of N.

4.2. Constructing Pairwise Toxic Data

We build our pairwise toxicity dataset using PPLM (Dathathri et al., 2019). PPLM is an attribute-controlled language generation technique, which attaches a simple linear attribute classification layer, $p(a|\mathbf{w})$ onto a language model to guide its generation. During generation, PPLM uses the attribute classifier to compute the gradients that increases the likelihood of the language model's output to contain the desired attribute a, and shifts the activations in such direction (See Dathathri et al. (2019) for details):

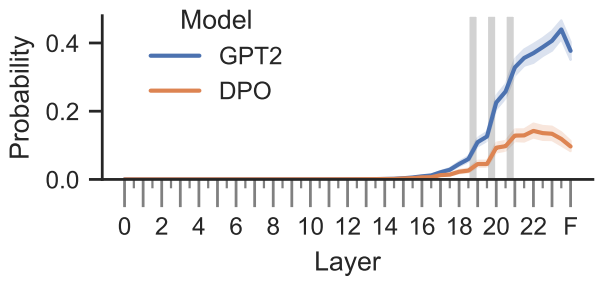

Figure 1. Logit lens on GPT2 and GPT2_{DPO}. Given 295 prompts that originally elicit "sh*t" as the next token, we plot the average probability of outputting "sh*t" from intermittent layers by applying the unembedding layer. Minor ticks indicate ℓ_mid layers (after attention heads, before MLP). Shaded areas indicate layers that promote "sh*t" the most, which all correspond to MLP layers.

$$p(y \mid a) \propto p(y)p(a \mid y)$$

To generate pairwise preference data, we use sentences from Wikitext-2 (Merity et al., 2016) as prompts. For each prompt, we generate a positive sample using greedy sampling with GPT2, while using PPLM to generate negative (toxic) samples. We use our toxic probe W_{Toxic} as our attribute classifier to guide towards toxic outputs. We create 24,576 pairs of toxic and nontoxic continuations.[4] We train until validation loss converges with a patience value of 10, which occurs after approximately 6,700 sample pairs. Appendix E has details for DPO and PPLM hyperparameters.

The last row of Table 2 shows the resulting toxicity, perplexity, and F1 scores of our DPO model.

Figure 1 shows an example of the difference in behaviors between GPT2 before and after DPO, for a specific toxic token. Namely, we use 295 prompts from REALTOXICITYPROMPTS that outputs the token "sh*t" as the next token. We then apply "Logit Lens" (Nostalgebraist, 2020), meaning we apply the unembedding layer on all intermittent layers. This allows us to visualize the layers that promote the "sh*t" token. The shared grey areas indicate the layers in which "sh*t" is promoted the most, which all correspond to MLP layers. We see that post-DPO, the toxic token is promoted far less.

5. Toxicity After DPO

In this section we explain how our aligned language models (GPT2_{DPO}, $\text{Llama2}_{\text{DPO}}$) avert toxic outputs.

[4]We release this data to enable further studies.

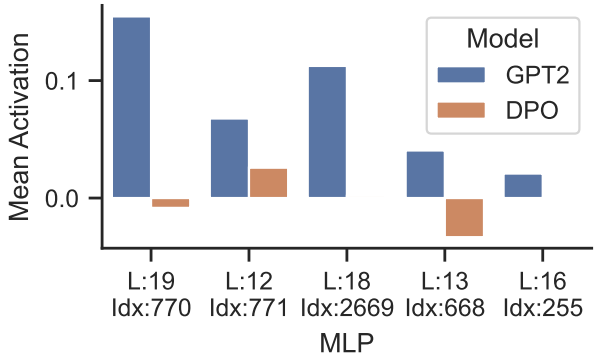

Figure 2. Mean activations for toxic vectors in GPT2 before and after DPO.

5.1. Toxic Vectors Remain After DPO

Of the toxic vectors described in Section 3, note that $\text{MLP}.\mathbf{v}_{\text{Toxic}}$ are actual weights of the model. Thus we inspect how these vectors change after DPO.

Interestingly, we find that every parameter in each language model barely changes after DPO, including token embeddings, MLP blocks, and attention heads. Every parameter in GPT2 (Llama2) and its counterpart in GPT2_{DPO} ($\text{Llama2}_{\text{DPO}}$) has a cosine similarity score greater than 0.99 and on average a norm difference less than 1e-5.[5] This applies for $\text{MLP}.\mathbf{k}_{\text{Toxic}}$ and $\text{MLP}.\mathbf{v}_{\text{Toxic}}$ as well – toxic MLP vectors **do not change** from DPO.

Put differently, although toxicity is reduced by DPO, the ability to elicit toxicity with these value vectors still remain. So how is it that GPT2_{DPO} and $\text{Llama2}_{\text{DPO}}$ avert toxic outputs? Though their parameters have barely moved, below we show that their collective movement is enough to avoid toxic outputs.

5.2. DPO Avoids $\text{MLP}.\mathbf{k}_{\text{Toxic}}$ Regions

We provide an explanation for how toxicity is reduced in GPT2_{DPO} and $\text{Llama2}_{\text{DPO}}$, starting with GPT2_{DPO}.

GPT2_{DPO}. In simplest terms, we observe a drop in activations for the toxic vectors $\text{MLP}.\mathbf{v}_{\text{Toxic}}$ in GPT2_{DPO}. Namely, using the same 1,199 prompts from REALTOXICITYPROMPTS, we generate 20 tokens and measure the mean activations m_i, or $\sigma(\mathbf{x}^\ell \cdot \text{MLP}.\mathbf{k}_i^\ell)$, of our $\text{MLP}.\mathbf{v}_{\text{Toxic}}$ vectors. Figure 2 shows 5 examples of the top $\text{MLP}.\mathbf{v}_{\text{Toxic}}$ vectors.

Inspired by Balestriero et al. (2023), we visualize this drop

[5]The unembedding layer of GPT2 is the only exception, where the norm difference is less than 1e-3.

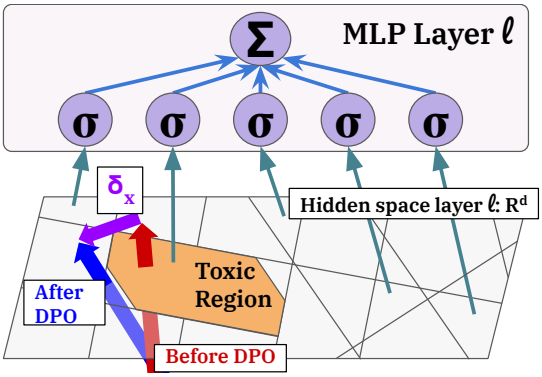

Figure 3. Visualization of residual streams before and after DPO. We view the shift, $\delta_{\mathbf{x}}$, as an offset that allow GPT2$_{\text{DPO}}$ to bypass regions that previously triggered toxic value vectors.

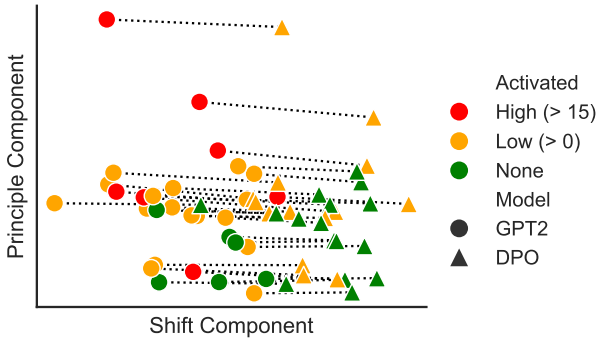

Figure 4. Linear shift of residual streams out of toxic regions. Each point is a residual stream sampled from either $\mathbf{x}^{19}_{\text{GPT}}$ or $\mathbf{x}^{19}_{\text{DPO}}$, using REALTOXICITYPROMPTS, projected onto 1) $\bar{\delta}^{19}_{\mathbf{x}}$, the mean difference in residual streams, and 2) the principle component of the residual streams. Dotted lines indicate samples from the same prompt. Colors indicate whether each point activates MLP$^{19}_{770}$. Note the shift from $\mathbf{x}^{19}_{\text{GPT}}$ to $\mathbf{x}^{19}_{\text{DPO}}$, but also the drop in activations.

in activations with what we call "MLP activation regions". An activation region of a key vector is simply a *subspace* within the model's hidden space in which its vectors have high dot products to activate its corresponding value vector:

$$\gamma(\mathbf{k}^{\ell}_i) := \{\mathbf{g} | \mathbf{g} \in \mathbb{R}^d, \sigma(\mathbf{k}^{\ell}_i \cdot \mathbf{g}) > 0\}, \tag{4}$$

where σ is a non-linear activation. Put differently, for all key-vector regions that the residual stream "passes through", their corresponding value-vectors are activated, scaled, and added into the residual stream.

We view the drop in activations as a shift in GPT2$_{\text{DPO}}$'s residual stream to avert the regions of toxic MLP vectors, $\gamma(\text{MLP.}\mathbf{k}_{\text{Toxic}})$. See Figure 3.

We formalize the shift in residual streams as following: given the residual streams at layer ℓ_mid (after attention heads at layer ℓ) for both GPT2 and GPT2$_{\text{DPO}}$, before MLP$^{\ell}_{\text{Toxic}}$, we notate the difference of the two residual streams as $\delta^{\ell_mid}_{\mathbf{x}} := \mathbf{x}^{\ell_mid}_{\text{DPO}} - \mathbf{x}^{\ell_mid}_{\text{GPT2}}, \delta^{\ell_mid}_{\mathbf{x}} \in \mathbb{R}^d$. We view $\delta^{\ell_mid}_{\mathbf{x}}$ as a vector that takes GPT2's residual stream out of the toxicity-eliciting regions, $\gamma(\text{MLP.}\mathbf{k}^{\ell}_{\text{Toxic}})$.

Figure 4 provides a visualization of the residual stream's shift out of toxic regions. Namely, given prompts from REALTOXICITYPROMPTS, we project the residual stream from GPT2 and GPT2$_{\text{DPO}}$ at layer 19 onto two dimensions: 1) the mean difference in the residual streams, $\bar{\delta}^{\ell}_{x}$, and the main principle component of the residual streams.[6] We further indicate whether each residual stream activates MLP.$\mathbf{v}^{19}_{770}$. Notice both the consistent linear shift between GPT2 and GPT2$_{\text{DPO}}$ and the drop in activations.

To understand where this shift comes from, we compute the differences in all parameter weights in GPT2 before and after DPO, and notate the differences as δ_{θ}. We notate the difference at a specific component such as a MLP block at layer ℓ as $\delta^{\ell}_{\text{MLP}}$.

Note that as previously noted, these differences $\delta^{\ell}_{\theta}, \forall \ell$ are minimal. Despite these minimal changes, their accumulation is sufficient in getting the residual stream out of toxic regions $\gamma(\text{MLP.}\mathbf{k}^{\ell}_{\text{Toxic}})$.

Given a toxic vector MLP.$\mathbf{v}_{\text{Toxic}}$ at layer ℓ, to understand where the shift in residual stream, $\delta^{\ell_mid}_{\mathbf{x}}$ comes from, we measure the cosine similarity between $\delta^{\ell_mid}_{\mathbf{x}}$ and the shift in value vectors in the preceding layers, $\delta^{j}_{\text{MLP.v}}$:

$$\forall j < \ell, \forall i < d_{mlp} : cos(\delta^{\ell_mid}_{\mathbf{x}}, \delta^{j}_{\text{MLP.v}_i}).$$

To our surprise, we find that the shift in value vectors, $\delta_{MLP.v}$, have high *negative* cosine similarity scores with the shift in residual streams $\delta_{\mathbf{x}}$: the value vectors in MLP blocks shift in the *opposite direction* as the shift in residual stream. The blue areas in Figure 5 show the cosine similarity between $\delta^{19_mid}_{\mathbf{x}}$ and δ^{j}_{MLP}. We show layer 19 as an example because MLP.v$^{19}_{770}$ is one of the most toxic vectors, but the same pattern can be found in other layers (see Appendix D). Namely, the blue areas indicate the percentage of value vectors at each layer in which their shifts have a cosine similarity score against $\delta^{19_mid}_{\mathbf{x}}$ as indicated by the x-axis. Note that as the layers approach layer 19, the majority of value vectors shift in the *opposite* direction of $\delta^{19}_{\mathbf{x}}$.

Why the antipodal direction? This can be explained by two facts: first, neurons in MLP blocks of language models

[6] We show layer 19 because MLP.$\mathbf{v}^{19}_{770}$ is one of the most toxic vectors, but similar patterns can be found in other layers (See Appendix C).

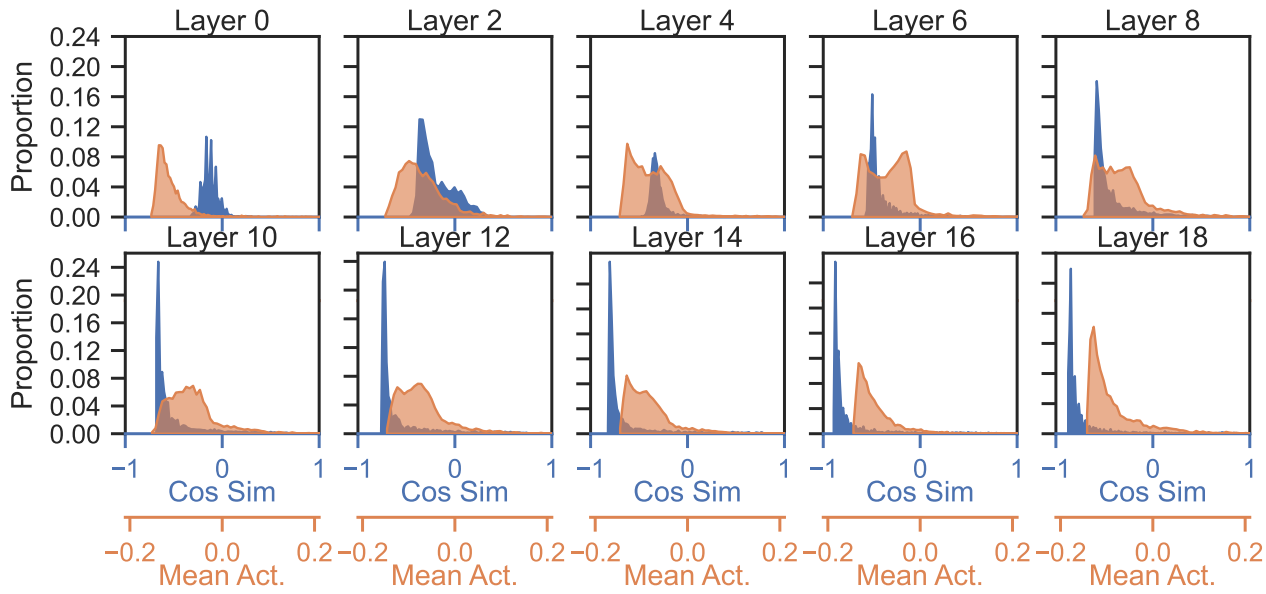

Figure 5. The cosine similarity between $\delta_{\text{MLP.v}}$ and $\delta_{\mathbf{x}}^{19}$. Blue areas indicate the percentage of value vectors with a cosine similarity score against $\delta_{\mathbf{x}}$ as indicated by the x-axis. Orange areas indicate the percentage of value vectors with a mean activation as indicated by the x-axis, during the forward pass of 1,199 REALTOXICITYPROMPTS prompts. Value vectors shift in the opposite direction of $\delta_{\mathbf{x}}$, but they end up contributing towards the $\delta_{\mathbf{x}}$ direction because of their negative activations.

are sparse (Zhang et al., 2022; Li et al., 2023d), meaning most neurons do not activate during a forward pass. Second, the choice of the MLP's activation function σ plays a role. Namely, our language model uses GeLU functions (Hendrycks & Gimpel, 2016). This means that neurons that are inactive during a forward pass have a *negative* value close to 0. Thus, during the forward pass, for each value vector, the newly learned direction $\delta_{\text{MLP.v}}$ gets multiplied by a very small negative scale, flips directions, and *contributes* towards the $\delta_{\mathbf{x}}$ direction. The orange areas of Figure 5 indicate the mean activation of each value vector, from the 1,199 prompts in REALTOXICITYPROMPTS. Most of the time, value vectors have a *negative* activation - thus the shift in value vectors end up *contributing* towards the $\delta_{\mathbf{x}}$ direction.

To summarize, GPT2$_{\text{DPO}}$ has learned an *offset*, $\delta_{\mathbf{x}}$, such that the residual stream avoids regions that promote toxicity, $\gamma(\text{MLP.}\mathbf{k}_{\text{Toxic}})$. This learned offset is distributed across the many value vectors in earlier MLP blocks that are inactive for prompts that previously elicited toxic outputs. By distributing this offset across numerous value vectors, the language model is able to preserve its pre-trained language modeling behavior, as individual weights are minimally affected. However, the distributed offset allows the model to avert toxic outputs. Note that this behavior matches precisely what the alignment objective was - to preserve as much of the pre-trained behavior, while optimizing for a reward (non-toxic outputs).

Llama2$_{\text{DPO}}$. We see a similar phenomena for Llama2$_{\text{DPO}}$, in that toxic vectors are not removed, but rather "turned off". Recall from Equation 3 that Llama2 uses GLUs, in which the element-wise product of two components determine the scale of each value vector: $\sigma(W_1\mathbf{x})$ and $W_2\mathbf{x}$.

Unlike GPT2, in which earlier MLP vectors are shifted to bypass toxic regions, we do not see this pattern in Llama2. Rather, we see that toxic value vectors MLP.$\mathbf{v}_{\text{Toxic}}$ are "turned off" by both the gating component ($\sigma(W_1\mathbf{x})$) and its linear projection counterpart ($W_2\mathbf{x}$). Figure 6 demonstrate the mean activations of each of these components for its top toxic vectors.

6. Un-aligning DPO

A growing line of work finds that alignment algorithms can easily be undone or jailbroken. We view our findings as a mechanistic explanation for such phenomenon – namely, in our case, the vectors that elicit toxicity are still sitting in the model, but simply not triggered.

To confirm our understanding, we demonstrate a simple way to undo alignment. To reiterate, GPT2$_{\text{DPO}}$ simply learned an offset to take the residual stream $\mathbf{x}^{\ell}$ out of regions that trigger toxic vectors: $\gamma(\text{MLP.}\mathbf{k}^{\ell}_{\text{Toxic}})$. A simple way to re-

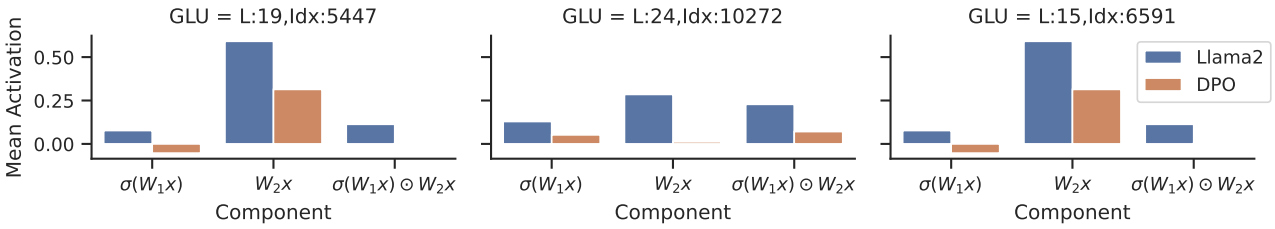

Figure 6. Mean activations for toxic vectors in Llama2 before and after DPO, broken down by component.

Table 4. Un-aligning GPT2_{DPO}. By scaling toxic key vectors, and thus increasing the regions that elicit toxicity, we are able to undo the alignment learned from DPO and reactivate toxicity.

METHOD	TOXIC	PPL	F1
GPT2_{DPO}	0.208	23.34	0.195
SCALE MLP.$\mathbf{k}_{\text{TOXIC}}$	0.458	23.30	0.195
GPT2	0.453	21.7	0.193

Table 5. Un-aligning $\text{Llama2}_{\text{DPO}}$. By "turning on" gating components, $\sigma(W_1\mathbf{x})$, by setting their values to 1, we are able reactivate toxicity.

METHOD	TOXIC	PPL	F1
$\text{LLAMA2}_{\text{DPO}}$	0.138	6.587	0.194
TURN GATE ON ($\sigma(W_1\mathbf{x})$)	0.217	6.596	0.195
SCALE W_2	0.244	6.648	0.194
LLAMA2	0.359	6.095	0.227

activate toxicity is to increase those regions by scaling each key vector larger (See Equation 4). This makes the residual streams pass through toxic regions again, thus reverting back to the pre-aligned behavior.

Similarly, $\text{Llama2}_{\text{DPO}}$ uses its gating component, $\sigma(W_1\mathbf{x})$, to "turn off" toxic vectors. Thus a simple way to re-activate toxicity is to turn these values back on by setting their gated values to 1. Alternatively, one can scale the latter component ($W_2\mathbf{x}$) larger to re-activate toxicity as well.

Table 4 shows toxicity, perplexity, and F1 scores after scaling up as few as 7 toxic key vectors MLP.$\mathbf{k}_{\text{Toxic}}$ in GPT2_{DPO}. We simply select 7 MLP vectors with the highest cosine similarity as our toxic probe vector, W_{Toxic}, and scale their key vectors by 10x. By doing so, the model reverts back to its pre-aligned toxic behavior. Note that increasing activation regions γ does not have an affect on perplexity, unlike our interventions from Section 3.3. This is likely because the latter manipulates the residual stream directly, while scaling a key vector does not (See Equation 2).

Similarly, Table 5 shows results for $\text{Llama2}_{\text{DPO}}$, by either turning back on as few as 8 gate components ($\sigma(W_1\mathbf{x})$) or by scaling the latter linear component ($W_2\mathbf{x}$) by 3x.

7. Discussion

7.1. On Designing Robust Alignment Algorithms

We view our work as providing a mechanistic explanation for why aligned models can be undone or jailbroken – in our experiments, the regions that previously elicited toxic behavior does not change after DPO. Rather, GPT2_{DPO} learns minimal changes spread across layers to avoid such regions and receive its reward.

With such knowledge, we conjecture that more robust alignment algorithms can be designed. Can we eliminate undesirable regions, as opposed to bypassing them? For instance, Li et al. (2023c) show how to remove causal pathways in a language model that is responsible for undesirable behaviors, including toxicity. Similarly, when we can identify the weights that elicit undesirable outputs, what happens during RLHF if we only update those weights in isolation?

Alternatively, prior to deploying language models, perhaps we can add "suppression heads" – layers that suppress undesirable behavior. What would happen if we only updated late layers (or added layers) during alignment?

Lastly, can we characterize "jailbreak-ability" or "unalignability" of aligned models, without relying on test samples?

We leave these questions for future work.

7.2. On the Role of KL-Divergence Regularization

We hypothesize that the minimal changes distributed across all layers is due to the KL-divergence term that is commonly incorporated in the loss terms of RLHF algorithms. Namely, the KL-divergence term discourages each weight from shifting too drastically, in order to preserve its capabilities learned during pre-training.

Similar to our work, Jain et al. (2023) fine-tunes a lan-

guage model on synthetic tasks to study the changes in its mechanisms. Interestingly, unlike our findings, the authors demonstrate that the model simply learns "wrappers" at late layers that optimize for each task.

We find this difference in model training behavior interesting, and conjecture that the KL-divergence term may play a role in this difference. Note that fine-tuning typically does not entail a KL-divergence term. Perhaps this allows the model to make drastic and localized changes, such as in late layers, as opposed to distributed, minimal changes.

8. Related Work

8.1. Alignment Algorithms

Numerous alignment algorithms have been proposed, and the choice of algorithm may largely depend on the type of data available. Perhaps most commonly, human feedback data is used (Stiennon et al., 2020; Ouyang et al., 2022; Touvron et al., 2023) for methods such as PPO (Schulman et al., 2017) or DPO (Rafailov et al., 2023). When labels for only undesirable behavior is available, algorithms like unlikelihood training (Welleck et al., 2020) or Cringe (Adolphs et al., 2023; Xu et al., 2023) can be used. We study DPO because it is easy to use and currently widely used.

8.2. Mechanistic Interpretability

The goal of mechanistic interpretability is largely to reverse engineer model behaviors (Olah et al., 2020; Elhage et al., 2021; Geva et al., 2021). By doing so, researchers have uncovered various interpretable and controllable representations, such as world models (Li et al., 2023a; Nanda et al., 2023), "truthfulness" (Li et al., 2023b), knowledge (Meng et al., 2022; Hernandez et al., 2023; Burns et al., 2023; Geva et al., 2023), linguistic properties (Conneau et al., 2018; Tenney et al., 2019), or even tasks (Ilharco et al., 2022; Hendel et al., 2023; Todd et al., 2023).

Zou et al. (2023a) suggests a broader framework for interpretability, in which representation engineering is viewed as a top-down approach for interpreting model behavior. Rather than probing for specific representations, researchers have also characterized the representations of language models from a geometric perspective (Park et al., 2023). Balestriero et al. (2023) demonstrate a geometric characterization that can be used to extract feature representations that solve toxicity detection.

Similar to our work, Jain et al. (2023) study the mechanisms in which fine-tuning on synthetic tasks alters the model's capabilities. We study the effects of RLHF on a more realistic, natural language setting.

8.3. Jailbreaking Aligned Models

Researchers demonstrated that aligned models can be surprisingly easily jailbroken (Wallace et al., 2019; Zou et al., 2023b; Wei et al., 2023; Carlini et al., 2023). Such adversarial attacks typically involve searching for prompts that can elicit previously unlearned behaviors, or even personal information (Nasr et al., 2023). Carlini et al. (2023) show that multimodal models can also be jailbroken. Wei et al. (2023) provide hypotheses, backed by empirical studies, as to why language models can be jailbroken.

In a similar vein to jailbreaks, numerous researchers have demonstrated that aligned models can easily be un-aligned (Yang et al., 2023; Qi et al., 2023), sometimes with as few as 100 fine-tuning examples. We view our work as adding a mechanistic understanding of such phenomena.

9. Conclusion

In this work we studied the mechanisms by which alignment algorithms unlearn a capability, taking DPO and toxicity as a case study. First, we uncovered how toxicity is represented and elicited in pre-trained language models, GPT2 and Llama2. We find numerous vectors in MLP blocks that promote toxicity. Simply subtracting these vectors from the residual stream can suppress toxic outputs.

Second, we applied DPO to our language models, using PPLM to carefully craft pairs of toxic and non-toxic continuations to Wikipedia prompts.

Third, we studied how our aligned models GPT2_{DPO} and $\text{Llama2}_{\text{DPO}}$ avert toxicity. We find that in both cases, the weights that elicit toxicity are not removed. In the case of GPT2_{DPO}, the model bypasses regions that elicit toxicity by learning an *offset*. Such an offset is distributed amongst multiple value vectors from earlier layers, allowing minimal changes to every weight. This allows the model to preserve its pre-trained behavior, while averting toxic outputs, which matches the objective of the DPO loss. In the case of $\text{Llama2}_{\text{DPO}}$, the model uses its gating mechanism to "turn off" toxic vectors.

Given this understanding, we demonstrated how to break the alignment of our models, reverting them back to their toxic behavior. Namely, for GPT2_{DPO} we simply increase the regions that elicit toxicity, by scaling their corresponding key vectors, while for $\text{Llama2}_{\text{DPO}}$ we simply turn the gates back on.

We view our findings as a mechanistic case study for why aligned models can be jailbroken, and hope that this can lead to more robust alignment algorithms. Our code, models, and data can be found at `https://github.com/ajyl/dpo_toxic`.

Impact Statement

Our work takes a step towards better understanding the mechanisms of language models and fine-tuning algorithms. We wish that these findings take us closer to building safer and more trustworthy systems, with hopes that this will lead to more responsible deployments of such systems.

Acknowledgements

We thank Ekdeep Singh Lubana for fruitful discussions, and Santiago Castro for helping with figures. This work was supported via NSF under grant #2306372.

References

Adolphs, L., Gao, T., Xu, J., Shuster, K., Sukhbaatar, S., and Weston, J. The CRINGE loss: Learning what language not to model. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8854–8874, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.493. URL https://aclanthology.org/2023.acl-long.493.

Balestriero, R., Cosentino, R., and Shekkizhar, S. Characterizing large language model geometry solves toxicity detection and generation. *arXiv preprint arXiv:2312.01648*, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Burns, C., Ye, H., Klein, D., and Steinhardt, J. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ETKGuby0hcs.

Carlini, N., Nasr, M., Choquette-Choo, C. A., Jagielski, M., Gao, I., Koh, P. W., Ippolito, D., Tramèr, F., and Schmidt, L. Are aligned neural networks adversarially aligned? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=OQQoD8Vc3B.

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.

cjadams, Sorensen, J., Elliott, J., Dixon, L., McDonald, M., nithum, and , Cukierski, W. Toxic comment classification challenge, 2017. URL https://kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge.

Conneau, A., Kruszewski, G., Lample, G., Barrault, L., and Baroni, M. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In Gurevych, I. and Miyao, Y. (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2126–2136, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1198. URL https://aclanthology.org/P18-1198.

Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., and Liu, R. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*, 2019.

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. Language modeling with gated convolutional networks. In *International conference on machine learning*, pp. 933–941. PMLR, 2017.

Dinan, E., Logacheva, V., Malykh, V., Miller, A., Shuster, K., Urbanek, J., Kiela, D., Szlam, A., Serban, I., Lowe, R., et al. The second conversational intelligence challenge (convai2). In *The NeurIPS'18 Competition: From Machine Learning to Intelligent Conversations*, pp. 187–208. Springer, 2020.

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.

Gehman, S., Gururangan, S., Sap, M., Choi, Y., and Smith, N. A. RealToxicityPrompts: Evaluating neural toxic

degeneration in language models. In Cohn, T., He, Y., and Liu, Y. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 3356–3369, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.301. URL https://aclanthology.org/2020.findings-emnlp.301.

Geva, M., Schuster, R., Berant, J., and Levy, O. Transformer feed-forward layers are key-value memories. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t. (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.446. URL https://aclanthology.org/2021.emnlp-main.446.

Geva, M., Caciularu, A., Wang, K., and Goldberg, Y. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 30–45, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.3. URL https://aclanthology.org/2022.emnlp-main.3.

Geva, M., Bastings, J., Filippova, K., and Globerson, A. Dissecting recall of factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*, 2023.

Hendel, R., Geva, M., and Globerson, A. In-context learning creates task vectors. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9318–9333, Singapore, December 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.findings-emnlp.624.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Hernandez, E., Sharma, A. S., Haklay, T., Meng, K., Wattenberg, M., Andreas, J., Belinkov, Y., and Bau, D. Linearity of relation decoding in transformer language models. *arXiv preprint arXiv:2308.09124*, 2023.

Ilharco, G., Ribeiro, M. T., Wortsman, M., Schmidt, L., Hajishirzi, H., and Farhadi, A. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2022.

Jain, S., Kirk, R., Lubana, E. S., Dick, R. P., Tanaka, H., Grefenstette, E., Rocktäschel, T., and Krueger, D. S. Mechanistically analyzing the effects of fine-tuning on procedurally defined tasks. *arXiv preprint arXiv:2311.12786*, 2023.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 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. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=DeG07_TcZvT.

Li, K., Patel, O., Viégas, F., Pfister, H., and Wattenberg, M. Inference-time intervention: Eliciting truthful answers from a language model. *arXiv preprint arXiv:2306.03341*, 2023b.

Li, M., Davies, X., and Nadeau, M. Circuit breaking: Removing model behaviors with targeted ablation. *arXiv preprint arXiv:2309.05973*, 2023c.

Li, Z., You, C., Bhojanapalli, S., Li, D., Rawat, A. S., Reddi, S. J., Ye, K., Chern, F., Yu, F., Guo, R., and Kumar, S. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. In *The Eleventh International Conference on Learning Representations*, 2023d. URL https://openreview.net/forum?id=TJ2nxciYCk-.

Meng, K., Bau, D., Andonian, A. J., and Belinkov, Y. Locating and editing factual associations in GPT. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=-h6WAS6eE4.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2016.

Nanda, N., Lee, A., and Wattenberg, M. Emergent linear representations in world models of self-supervised sequence models. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 16–30, 2023.

Nasr, M., Carlini, N., Hayase, J., Jagielski, M., Cooper, A. F., Ippolito, D., Choquette-Choo, C. A., Wallace, E., Tramèr, F., and Lee, K. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*, 2023.

Nostalgebraist. Interpreting gpt: The logit lens, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Gray, A., et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

Park, K., Choe, Y. J., and Veitch, V. The linear representation hypothesis and the geometry of large language models. In *Causal Representation Learning Workshop at NeurIPS 2023*, 2023.

Qi, X., Zeng, Y., Xie, T., Chen, P.-Y., Jia, R., Mittal, P., and Henderson, P. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv preprint arXiv:2310.03693*, 2023.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: Your language model is secretly a reward model, 2023.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shazeer, N. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Sheng, E., Chang, K.-W., Natarajan, P., and Peng, N. The woman worked as a babysitter: On biases in language generation. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3407–3412, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1339. URL https://aclanthology.org/D19-1339.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33: 3008–3021, 2020.

Tenney, I., Das, D., and Pavlick, E. BERT rediscovers the classical NLP pipeline. In Korhonen, A., Traum, D., and Màrquez, L. (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4593–4601, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1452. URL https://aclanthology.org/P19-1452.

Todd, E., Li, M. L., Sharma, A. S., Mueller, A., Wallace, B. C., and Bau, D. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*, 2023.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wallace, E., Feng, S., Kandpal, N., Gardner, M., and Singh, S. Universal adversarial triggers for attacking and analyzing NLP. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2153–2162, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1221. URL https://aclanthology.org/D19-1221.

Wei, A., Haghtalab, N., and Steinhardt, J. Jailbroken: How does LLM safety training fail? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=jA235JGM09.

Welleck, S., Kulikov, I., Roller, S., Dinan, E., Cho, K., and Weston, J. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJeYe0NtvH.

Xu, J., Lee, A., Sukhbaatar, S., and Weston, J. Some things are more cringe than others: Preference optimization with the pairwise cringe loss. *arXiv preprint arXiv:2312.16682*, 2023.

Yang, X., Wang, X., Zhang, Q., Petzold, L., Wang, W. Y., Zhao, X., and Lin, D. Shadow alignment: The ease of subverting safely-aligned language models. *arXiv preprint arXiv:2310.02949*, 2023.

Zhang, Z., Lin, Y., Liu, Z., Li, P., Sun, M., and Zhou, J. MoEfication: Transformer feed-forward layers are mixtures of experts. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 877–890, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.71. URL https://aclanthology.org/2022.findings-acl.71.

Zou, A., Phan, L., Chen, S., Campbell, J., Guo, P., Ren, R., Pan, A., Yin, X., Mazeika, M., Dombrowski, A.-K., et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023a.

Zou, A., Wang, Z., Kolter, J. Z., and Fredrikson, M. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023b.

A. Projecting Value Vectors onto Vocabulary Space

In this section we provide details from Geva et al. (2022) that demonstrate that MLP value vectors promote or suppress the likelihood of tokens.

We start from Equation 2:

$$\texttt{MLP}^\ell(\mathbf{x}^\ell) = \sum_{i=1}^{d_{mlp}} \sigma(\mathbf{x}^\ell \cdot \mathbf{k}_i^\ell)\mathbf{v}_i^\ell = \sum_{i=1}^{d_{mlp}} m_i^\ell \mathbf{v}_i^\ell.$$

Thus, we can consider the update from $\texttt{MLP}^\ell$ as d_{mlp} *sub-updates*, each sub-update being $m_i^\ell \mathbf{v}_i^\ell$.

We can then analyze the influence that each sub-update has on the output distribution, or the probability of generating token $w \in V$ (taken from Geva et al. (2022)):

$$p\big(w \mid \mathbf{x}^\ell + m_i^\ell \mathbf{v}_i^\ell, E\big) = \frac{\exp\big(\mathbf{e}_w \cdot \mathbf{x}^\ell + \mathbf{e}_w \cdot m_i^\ell \mathbf{v}_i^\ell\big)}{Z\big(E(\mathbf{x}^\ell + m_i^\ell \mathbf{v}_i^\ell)\big)} \propto \exp\big(\mathbf{e}_w \cdot \mathbf{x}^\ell\big) \cdot \exp\big(\mathbf{e}_w \cdot m_i^\ell \mathbf{v}_i^\ell\big) \tag{5}$$

where $\mathbf{e}_w$ is the token embedding of w, and Z is the softmax normalization factor. This indicates that when $\mathbf{e}_w \cdot m_i^\ell \mathbf{v}_i^\ell > 0$, the likelihood of w increases, while $\mathbf{e}_w \cdot m_i^\ell \mathbf{v}_i^\ell < 0$ decreases the likelihood.

B. Additional Llama2 Results

In this section we provide results from Llama2. Table 6 demonstrates the toxic tokens encoded by its GLU value vectors. Table 7 demonstrates results from intervening on Llama2's residual stream using token vectors.

Table 6. Top toxic vectors in Llama2, projected onto the vocabulary space. WARNING: THESE EXAMPLES ARE HIGHLY OFFENSIVE.

VECTOR	TOP TOKENS
W_{Toxic}	hole, ass, arse, onderwerp, bast, *$, face, Dick
GLU.$\mathbf{v}_{5447}^{19}$	hell, ass, bast, dam, balls, eff, sod, f
GLU.$\mathbf{v}_{10272}^{24}$	ass, d, dou, dick, pen, cock, j
GLU.$\mathbf{v}_{6591}^{15}$	org, sex, anal, lub, sexual, nak, XXX
SVD.$\text{U}_{\text{Toxic}}[0]$	hell, ass, bast, dam, eff, sod, arse,

Table 7. Toxicity, perplexity (PPL), and F1 after interventions or DPO for Llama2. We scale our toxic vectors such that the resulting perplexity is comparable to that of post-DPO GPT2. †: Not an intervention.

METHOD	VECTOR	TOXIC	PPL	F1
NO OP	N/A	0.359	6.095	0.227
SUBTRACT	W_{Toxic}	0.256	6.523	0.225
SUBTRACT	GLU.$\mathbf{v}_{5447}^{19}$	0.171	6.518	0.225
SUBTRACT	SVD.$\text{U}_{\text{Toxic}}[0]$	0.246	6.504	0.225
DPO†	N/A	0.138	6.587	0.194

C. Shift in Residual Streams

In this section we provide more examples of residual streams shifting out of toxic regions. See Figure 7

D. Shifts in Residual Streams vs. Shifts in MLP Value Vectors.

In this section we provide more examples of how MLP value vectors contribute in the $\delta_{\mathbf{x}}$ direction at different layers.

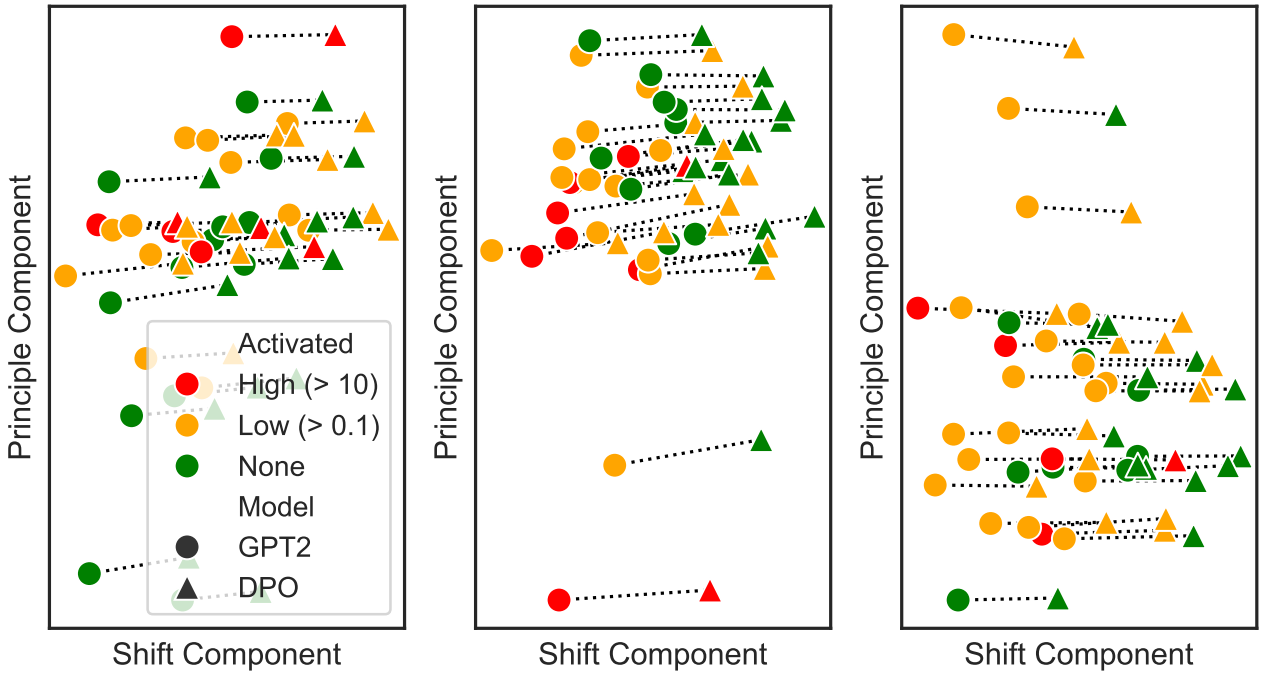

Figure 7. Shift in residual streams at layer 12, 18, and 13 (we show these three layers because MLP.$\mathbf{v}^{12}_{771}$, MLP.$\mathbf{v}^{18}_{2669}$, and MLP.$\mathbf{v}^{13}_{668}$ are the next three vectors with highest cosine similarity with W_{Toxic}. See Table 1, Figure 2.

E. Hyperparameters

Tables 8, and 9 contain the hyperparameters used for our toxic probe, DPO, and PPLM, respectively.

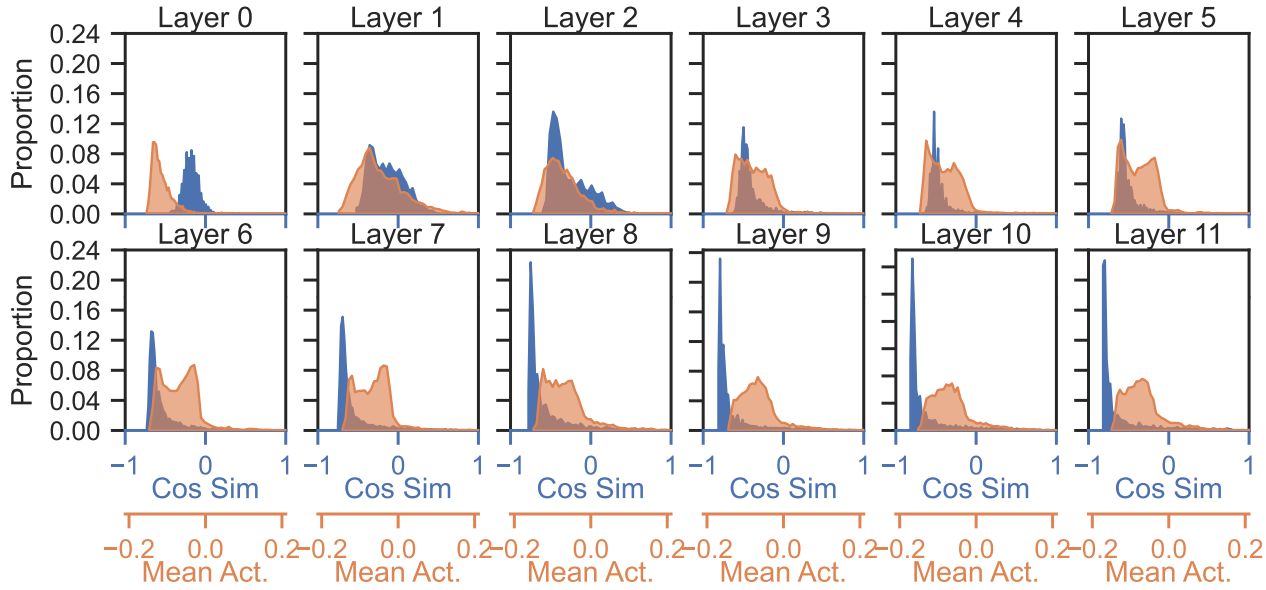

Figure 8. Shift in residual streams at layer 12 vs. shift in MLP value vectors ($\delta_{\mathbf{x}}^{12}$ vs. δ_{MLP}).

Table 8. Hyperparameters: DPO.

Hyperparameter	Value
Learning Rate	1e-6
Batch Size	4
Optimizer	RMSprop
Gradient accumulation steps	1
Max gradient Norm	10
Validation Metric	loss/valid
Validation Patience	10
DPO Beta	0.1

Table 9. Hyperparameters: PPLM.

Hyperparameter	Value
Step Size	0.4
Temperature	1
Top K	10
Num Iterations	50
Window Length	0
Horizon Length	1
Decay	False
Gamma	1
GM Scale	0.95
KL Scale	0.1

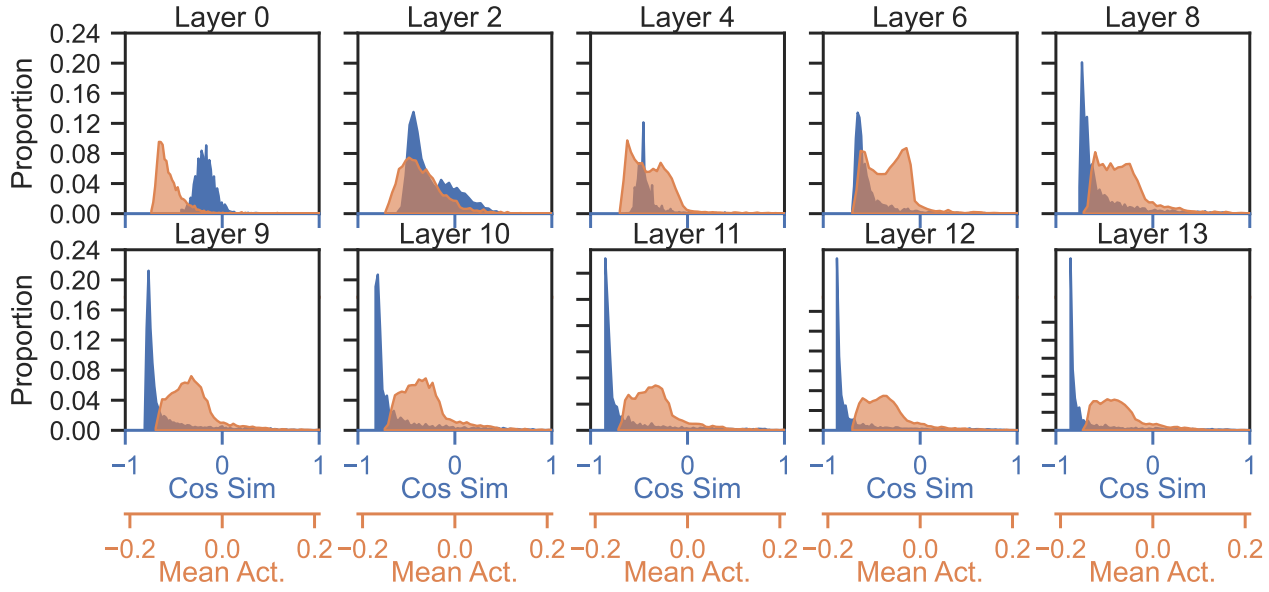

Figure 9. Shift in residual streams at layer 14 vs. shift in MLP value vectors ($\delta_{\mathbf{x}}^{14}$ vs. δ_{MLP}).

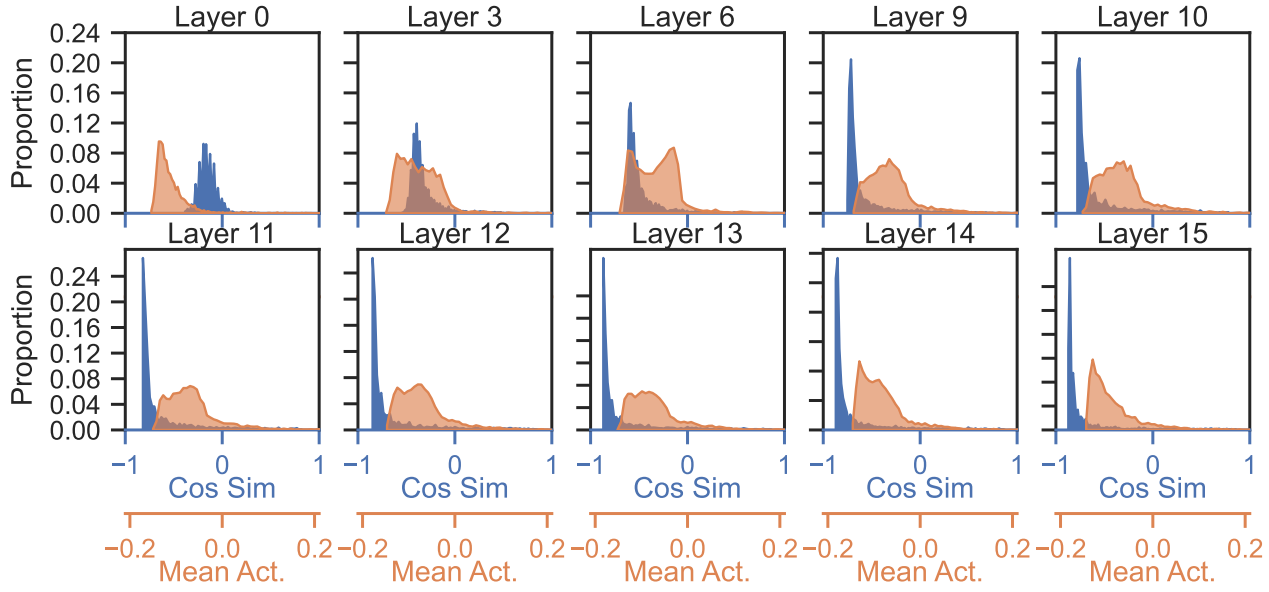

Figure 10. Shift in residual streams at layer 16 vs. shift in MLP value vectors ($\delta_{\mathbf{x}}^{16}$ vs. δ_{MLP}).

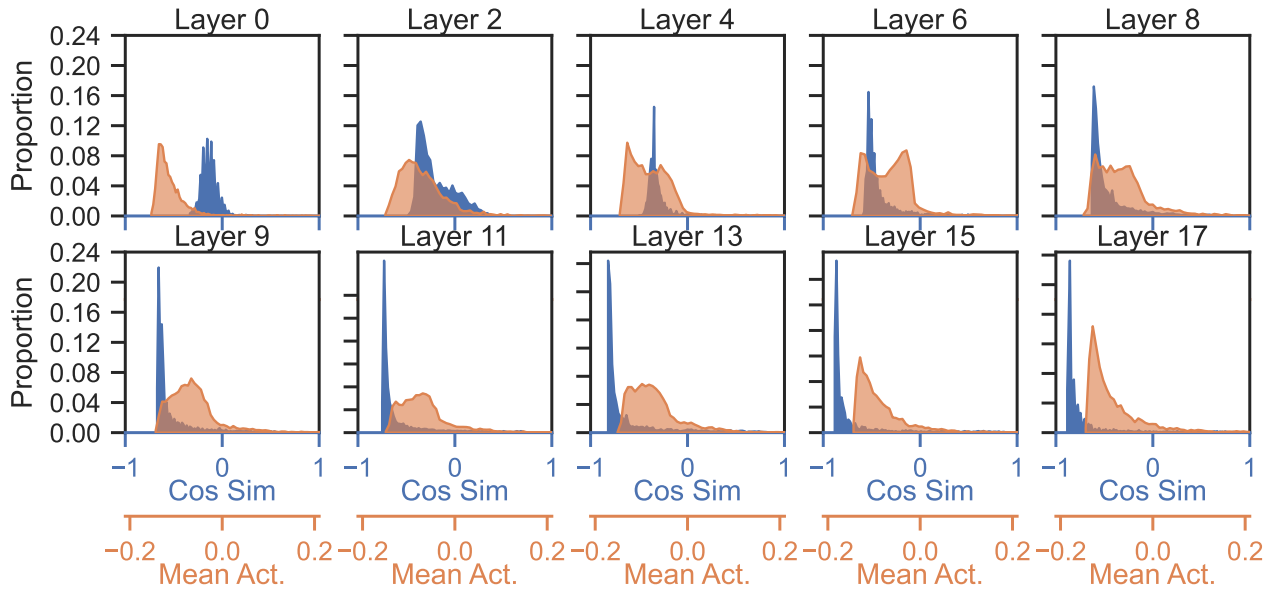

Figure 11. Shift in residual streams at layer 18 vs. shift in MLP value vectors ($\delta_{\mathbf{x}}^{18}$ vs. δ_{MLP}).