Eliciting Latent Predictions from Transformers with the Tuned Lens

Nora Belrose [1 2] **Zach Furman** [1 3] **Logan Smith** [1] **Danny Halawi** [1] **Igor Ostrovsky** [1] **Lev McKinney** [4 2] **Stella Biderman** [1] **Jacob Steinhardt** [5]

Abstract

We analyze transformers from the perspective of iterative inference, seeking to understand how model predictions are refined layer by layer. To do so, we train an affine probe for each block in a frozen pretrained model, making it possible to decode every hidden state into a distribution over the vocabulary. Our method, the *tuned lens*, is a refinement of the earlier "logit lens" technique, which yielded useful insights but is often brittle.

We test our method on various autoregressive language models with up to 20B parameters, showing it to be more predictive, reliable and unbiased than the logit lens. With causal experiments, we show the tuned lens uses similar features to the model itself. We also find the trajectory of latent predictions can be used to detect malicious inputs with high accuracy. All code needed to reproduce our results can be found at `https://github.com/AlignmentResearch/tuned-lens`.

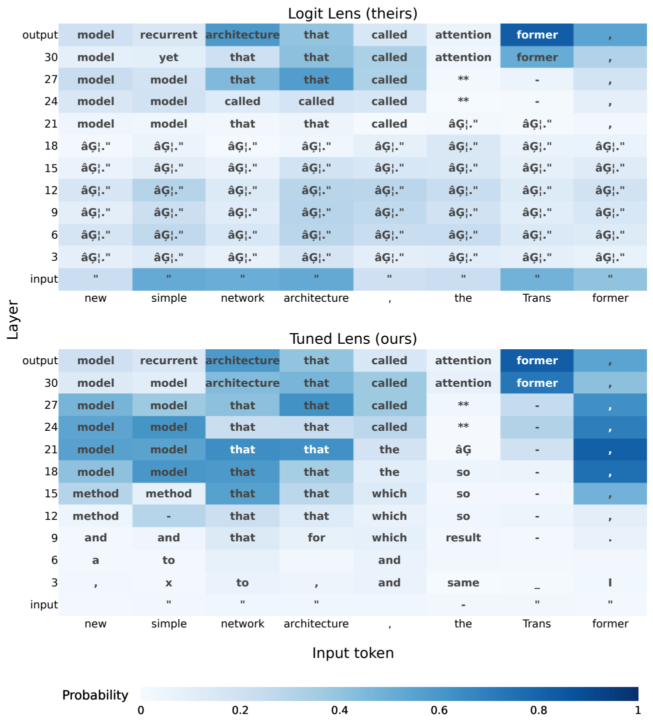

Figure 1. Comparison of our method, the *tuned lens* (bottom), with the "logit lens" (top) for GPT-Neo-2.7B prompted with an except from the abstract of Vaswani et al. (2017). Each cell shows the top-1 token predicted by the model at the given layer and token index. The logit lens fails to elicit interpretable predictions before layer 21, but our method succeeds.

1. Introduction

The impressive performance of transformers in natural language processing (Brown et al., 2020) and computer vision (Dosovitskiy et al., 2020) suggests that their internal representations have rich structure worthy of scientific investigation. One common approach is to train classifiers to extract specific concepts from hidden states, like part-of-speech and syntactic structure (Hewitt and Manning, 2019; Tucker et al., 2021; Li et al., 2022).

In this work, we instead examine transformer representations from the perspective of *iterative inference* (Jastrzębski et al., 2017). Specifically, we view each layer in a transformer language model as performing an incremental update to a latent prediction of the next token.[1] We decode these latent predictions through early exiting, converting the hidden state at each intermediate layer into a distribution over the vocabulary. This yields a sequence of distributions we call the *prediction trajectory*, which exhibits a strong tendency to converge smoothly to the final output distribution, with each successive layer achieving lower perplexity.

We build on the "logit lens" (nostalgebraist, 2020), an early exiting technique that directly decodes hidden states into vocabulary space using the model's pretrained unembedding matrix. We find the logit lens to be unreliable (Section 2), failing to elicit plausible predictions for models like BLOOM (Scao et al., 2022) and GPT Neo (Black et al., 2021). Even when the logit lens appears to work, its outputs are hard to interpret due to *representational drift*: features

[1]Eleuther AI [2]FAR AI [3]Boston University [4]University of Toronto [5]UC Berkeley. Correspondence to: Nora Belrose <nora@eleuther.ai>.

[1]See Appendix C for evidence supporting this view, including novel empirical results of our own.

may be represented differently at different layers of the network. Other early exiting procedures also exist (Schuster et al., 2022), but require modifying the training process, and so can't be used to analyze pretrained models.

To address the shortcomings of the logit lens, we introduce the *tuned lens*. We train L affine transformations, one for each layer of the network, with a distillation loss: transform the hidden state so that its image under the unembedding matches the final layer logits as closely as possible. We call these transformations *translators* because they "translate" representations from the basis used at one layer of the network to the basis expected at the final layer. Composing a translator with the pretrained unembedding yields a probe (Alain and Bengio, 2016) that maps a hidden state to a distribution over the vocabulary.

We find that tuned lens predictions have substantially lower perplexity than logit lens predictions, and are more representative of the final layer distribution. We also show that the features most influential on the tuned lens output are also influential on the model itself (Section 4). To do so, we introduce a novel algorithm called *causal basis extraction* (CBE) and use it to locate the directions in the residual stream with the highest influence on the tuned lens. We then ablate these directions in the corresponding model hidden states, and find that these features tend to be disproportionately influential on the model output.

We use the tuned lens to gain qualitative insight into the computational process of pretrained language models, by examining how their latent predictions evolve during a forward pass (Figure 1, Appendix B)

Finally, we apply the tuned lens in several ways: we extend the results of Halawi et al. (2023) to new models (Section 5.1), we find that tuned lens prediction trajectories can be used to detect prompt injection attacks (Perez and Ribeiro, 2022) often with near-perfect accuracy (Section 5.2), and find that data points which require many training steps to learn also tend to be classified in later layers (Section 5.3).

2. The Logit Lens

The logit lens was introduced by nostalgebraist (2020), who found that when the hidden states at each layer of GPT-2 (Radford et al., 2019) are decoded with the unembedding matrix, the resulting distributions converge roughly monotonically to the final answer. More recently it has been used by Halawi et al. (2023) to understand how transformers process few-shot demonstrations, and by Dar et al. (2022), Geva et al. (2022), and Millidge and Black (2022) to directly interpret transformer weight matrices.

The method. Consider a pre-LayerNorm transformer[2] $\mathcal{M}$.

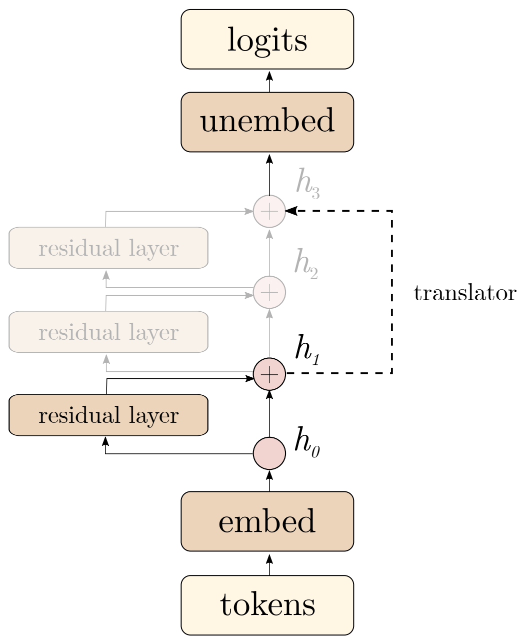

Figure 2. The tuned lens takes the hidden state at an intermediate layer (e.g. h_1 above), and applies a learned affine transformation (the translator). We then convert the hidden state into logits with the unembedding layer.

We'll decompose $\mathcal{M}$ into two "halves," $\mathcal{M}_{\leq \ell}$ and $\mathcal{M}_{>\ell}$. The function $\mathcal{M}_{\leq \ell}$ consists of the layers of $\mathcal{M}$ up to and including layer ℓ, and it maps the input space to hidden states. Conversely, the function $\mathcal{M}_{>\ell}$ consists of the layers of $\mathcal{M}$ after ℓ, which map hidden states to logits.

The transformer layer at index ℓ updates the representation as follows:

$$\boldsymbol{h}_{\ell+1} = \boldsymbol{h}_\ell + F_\ell(\boldsymbol{h}_\ell), \tag{1}$$

where F_ℓ is the residual output of layer ℓ. Applying Equation 1 recursively, the output logits $\mathcal{M}_{>\ell}$ can be written as a function of an arbitrary hidden state $\boldsymbol{h}_\ell$ at layer ℓ:

$$\mathcal{M}_{>\ell}(\boldsymbol{h}_\ell) = \text{LayerNorm}\Big[\boldsymbol{h}_\ell + \sum_{\ell'=\ell}^{L} \underbrace{F_{\ell'}(\boldsymbol{h}_{\ell'})}_{\text{residual update}}\Big] W_U. \tag{2}$$

The logit lens consists of setting the residuals to zero:

$$\text{LogitLens}(\boldsymbol{h}_\ell) = \text{LayerNorm}[\boldsymbol{h}_\ell] W_U. \tag{3}$$

While this simple form of the logit lens works reasonably well for GPT-2, nostalgebraist (2021) found that it fails to extract useful information from the GPT-Neo family of models (Black et al., 2021). In an attempt to make the method

[2]Both the logit lens and the tuned lens are designed primarily for the pre-LN architecture, which is more unambiguously iterative. Luckily pre-LN is by far more common than post-LN among state-of-the-art models. See Zhang and He (2020, Appendix C) for more discussion.

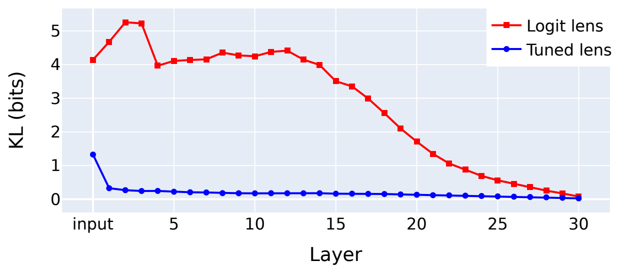

Figure 3. Bias of logit lens and tuned lens outputs relative to the final layer output for GPT-Neo-2.7B. The last transformer layer is included for both probes. Unlike the tuned lens, the logit lens is systematically biased toward some vocabulary items over others until the very end of the network.

work better for GPT-Neo, they introduce an extension which retains the last transformer layer, yielding:

$$\text{LogitLens}^{\text{ext}}(\boldsymbol{h}_\ell) = \text{LayerNorm}[\boldsymbol{h}_\ell + F_L(\boldsymbol{h}_\ell)]W_U \quad (4)$$

This extension is only partially successful at recovering meaningful results; see Figure 1 (top) for an example.

Unreliability. Beyond GPT-Neo, the logit lens struggles to elicit predictions from several other models released since its introduction, such as BLOOM (Scao et al., 2022) and OPT 125M (Zhang et al., 2022) (Figure 12).

Moreover, the type of information extracted by the logit lens varies both from model to model and from layer to layer, making it difficult to interpret. For example, we find that for BLOOM and OPT 125M, the top 1 prediction of the logit lens is often the *input* token, rather than any plausible continuation token, in more than half the layers (Figure 16).

Bias. Even when the logit lens is useful, we find that it is a *biased* estimator of the model's final output: it systematically puts more probability mass on certain vocabulary items than the final layer does.

This is concerning because it suggests we can't interpret the logit lens prediction trajectory as a belief updating in response to new evidence. The beliefs of a rational agent should not update in an easily predictable direction over time, since predictable updates can be exploited via Dutch books (**?**). Biased logit lens outputs are trivially exploitable once the direction of bias is known: one could simply "bet" against the logit lens at layer $\ell < L$ that the next token will be one of the tokens that it systematically downweights, and make unbounded profit in expectation.

Let $\boldsymbol{x}$ be a sequence of tokens sampled from a dataset D, and let $\boldsymbol{x}_{<t}$ refer to the tokens preceding position t in the sequence. Let $q_\ell(\cdot|\boldsymbol{x}_{<t})$ be the logit lens distribution at layer ℓ for position t, and let $p(\cdot|\boldsymbol{x}_{<t})$ be the final layer

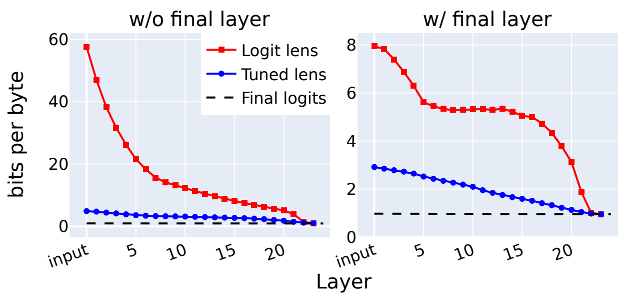

Figure 4. Perplexity of predictions elicited from BLOOM 560M under four conditions: the logit lens (red squares) and the tuned lens (blue circles), and including (left) and excluding (right) the final transformer layer from the probe. We find that tuned lens predictions have substantially lower perplexity whether or not the final layer is included, showing it is an independent and complementary proposal.

distribution for position t.

We define $p(v|\boldsymbol{x})$ to be the probability assigned to a vocabulary item v in a sequence $\boldsymbol{x}$, averaged over all positions $1 \ldots T$:

$$p(v|\boldsymbol{x}) \overset{\text{def}}{=} \frac{1}{T}\sum_{t=1}^{T} p(v|\boldsymbol{x}_{<t}). \quad (5)$$

Slightly abusing terminology, we say that q_ℓ is an "unbiased estimator" of p if, for every item v in the vocabulary, the probability assigned to v averaged across all tokens in the dataset is the same:

$$\begin{aligned} \mathop{\mathbb{E}}_{\boldsymbol{x}\in D}\Big[q_\ell(v|\boldsymbol{x})\Big] &= \mathop{\mathbb{E}}_{\boldsymbol{x}\in D}\Big[p(v|\boldsymbol{x})\Big] \\ q_\ell(v) &= p(v) \\ \forall v \in \mathcal{V}, \quad \mathcal{V} &= \{\texttt{"aardvark"}, \ldots\} \end{aligned} \quad (6)$$

In practice, Equation 6 will never hold exactly. We measure the degree of bias using the KL divergence between the marginal distributions, $D_{KL}(p \,||\, q_\ell)$.

In Figure 3 we evaluate the bias for each layer of GPT-Neo-2.7B. We find the bias of the logit lens can be quite large: around 4 to 5 bits for most layers. As a point of comparison, the bias of Pythia 160M's final layer distribution relative to that of its larger cousin, Pythia 12B, is just 0.0068 bits.

3. The Tuned Lens

One problem with the logit lens is that, if transformer layers learn to output residuals that are far from zero *on average*, the input to LogitLens may be out-of-distribution and yield nonsensical results. In other words, the choice of zero as a replacement value is somewhat arbitrary– the network might learn to rely on $\sum_{\ell'=\ell}^{L} \mathbb{E}[F_\ell(\boldsymbol{h}_\ell)]$ as a bias term.

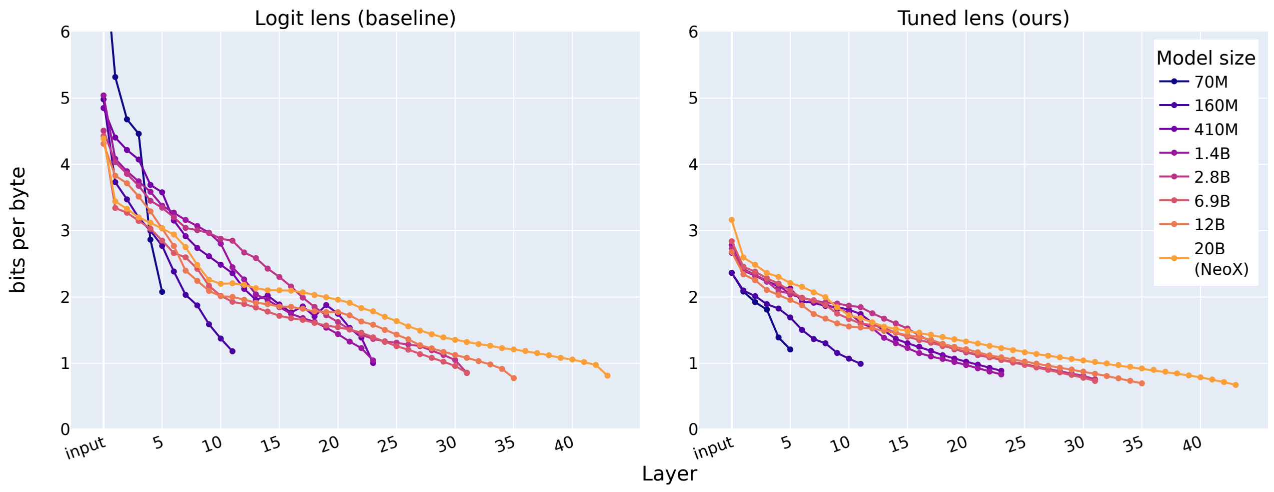

Figure 5. Perplexity of latent predictions elicited by the logit lens (left) and the tuned lens (right) from Pythia and GPT-NeoX-20B, as a function of layer index and model size. Tuned lens predictions are uniformly lower perplexity and exhibit lower variance across independently trained models.

Our first change to the method is to replace the summed residuals with a learnable constant value $\mathbf{b}_\ell$ instead of zero:

$$\text{LogitLens}_\ell^{\text{debiased}}(\boldsymbol{h}_\ell) = \text{LogitLens}(\boldsymbol{h}_\ell + \mathbf{b}_\ell) \qquad (7)$$

Representation drift. Another issue with the logit lens is that transformer hidden states often contain a small number of very high variance dimensions, and these "rogue dimensions" (Timkey and van Schijndel, 2021) tend to be distributed unevenly across layers; see Figure 6 (top) for an example. Ablating an outlier direction can drastically harm performance (Kovaleva et al., 2021), so if LogitLens relies on the presence or absence of particular outlier dimensions, the perplexity of logit lens predictions might be spuriously high.

Even when controlling for rogue dimensions, we observe a strong tendency for the covariance matrices of hidden states at different layers to drift apart as the number of layers separating them increases (Figure 6, bottom). The covariance at the final layer often changes sharply relative to previous layers, suggesting the logit lens might "misinterpret" earlier representations.

One simple, general way to correct for drifting covariance is to introduce a learnable change of basis matrix A_ℓ, which learns to map from the output space of layer ℓ to the input space of the final layer. We have now arrived at the *tuned lens* formula, featuring a learned affine transformation for each layer:

$$\text{TunedLens}_\ell(\boldsymbol{h}_\ell) = \text{LogitLens}(A_\ell \boldsymbol{h}_\ell + \mathbf{b}_\ell) \qquad (8)$$

We refer to $(A_\ell, \mathbf{b}_\ell)$ as the *translator* for layer ℓ.

Loss function. We train the translators to minimize KL between the tuned lens logits and the final layer logits:

$$\underset{}{\text{argmin}} \ \mathop{\mathbb{E}}_{\boldsymbol{x}} \Big[D_{KL}(f_{>\ell}(\boldsymbol{h}_\ell) \,||\, \text{TunedLens}_k(\boldsymbol{h}_\ell)) \Big] \qquad (9)$$

where $f_{>\ell}(\boldsymbol{h}_\ell)$ refers to the rest of the transformer after layer ℓ. This can be viewed as a distillation loss, using the final layer distribution as a soft label (Sanh et al., 2019). It ensures that the probes are not incentivized to learn extra information over and above what the model has learned, which can become a problem when training probes with ground truth labels (Hewitt and Liang, 2019).

Implementation details. When readily available, we train translators on a slice of the validation set used during pre-training, and use a separate slice for evaluation. Since BLOOM and GPT-2 do not have publicly available validation sets, we use the Pile validation set (Gao et al., 2020; Biderman et al., 2022). The OPT validation set is also not publicly available, but a member of the OPT team helped us train a tuned lens on the OPT validation set. Documents are concatenated and split into uniform chunks of length 2048.

We use SGD with Nesterov momentum, with a linear learning rate decay schedule over 250 training steps. We use a base learning rate of 1.0, or 0.25 when keeping the final transformer layer, and clip gradients to a norm of 1. We accumulate gradients as necessary to achieve a total batch size of 2^{18} tokens per optimizer step. We initialize all translators to the identity transform, and use a weight decay of 10^{-3}.

We evaluate all models on a random sample of 16.4M tokens from their respective pretraining validation sets. We leave out the final transformer layer for GPT-2 (Radford et al.,

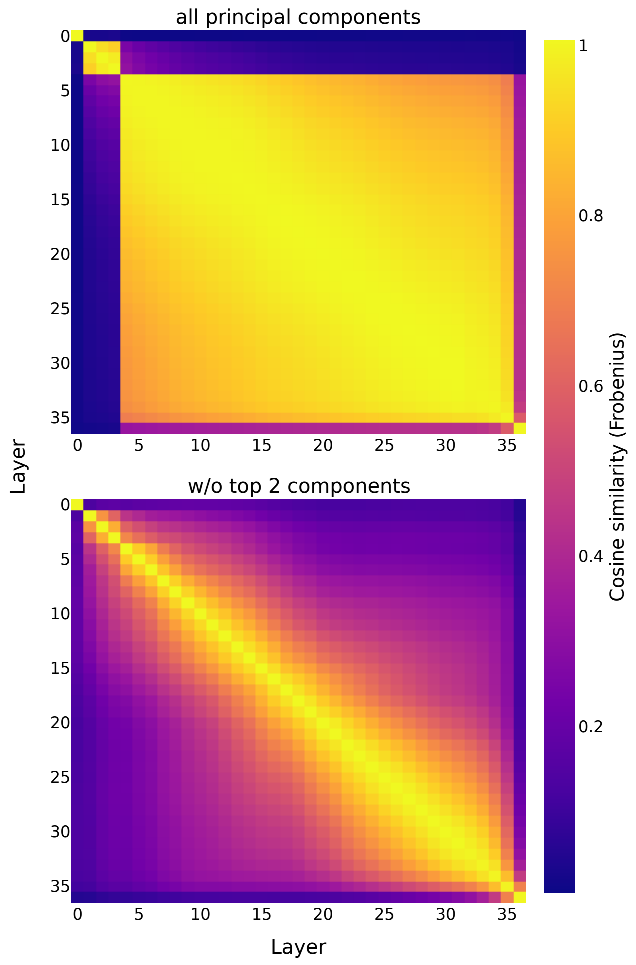

Figure 6. Pairwise similarities of hidden state covariance matrices across layers of Pythia 12B. Layer 4 introduces two outlier dimensions which dominate the covariance; removing them reveals smooth representational drift with depth. To control for varying hidden state norms, we measure the Frobenius cosine similarity, or $\frac{\langle A,B \rangle_F}{\|A\|_F \|B\|_F}$ for two matrices A and B.

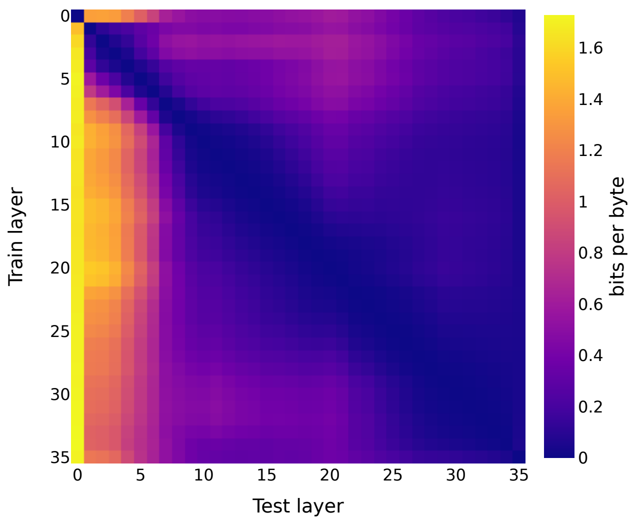

Figure 7. Transfer penalties for Pythia 12B. Each row corresponds to a single tuned lens probe trained on layer ℓ, and each column is a layer ℓ' on which probes are evaluated. Each cell shows the cross-entropy loss of probe ℓ evaluated on layer ℓ', *minus* its on-distribution loss (so that the diagonal entries are identically zero).

2019), GPT-NeoX-20B (Black et al., 2022), OPT (Zhang et al., 2022), and Pythia (Biderman et al., 2023), and include it for GPT-Neo (Black et al., 2021). We evaluate BLOOM (Scao et al., 2022) under both conditions in Figure 4.

Results. We plot tuned lens perplexity as a function of depth for the Pythia models and GPT-NeoX-20B in Figure 5[3]; results for other model families can be found in Appendix A.

We find that the tuned lens resolves the problems with the logit lens discussed in Section 2: it has significantly lower bias (Figure 3), and much lower perplexity than the logit lens across the board (Figure 5, Appendix A).

Transferability across layers. We find that tuned lens translators can usually zero-shot transfer to nearby layers with only a modest increase in perplexity. Specifically, we define the *transfer penalty* from layer ℓ to ℓ' to be the expected increase in cross-entropy loss when evaluating the tuned lens translator trained for layer ℓ on layer ℓ'.

We report transfer penalties for the largest Pythia model in Figure 7. Overall, transfer penalties are quite low, especially for nearby layers (entries near the diagonal in Figure 7). Comparing to the two plots in Figure 6, we notice that transfer penalties are strongly negatively correlated with covariance similarity (Spearman $\rho = -0.78$). Unlike Figure 6, however, Figure 7 is not symmetric: transfer penalties are higher when training on a layer with the outlier dimensions (Layer 5 and later) and testing on a layer without them, than the reverse.

Relation to model stitching. The tuned lens can be viewed as a way of "stitching" an intermediate layer directly onto the unembedding, with an affine transform in between to align the representations. The idea of model stitching was introduced by Lenc and Vedaldi (2015), who form a composite model out of two *frozen* pretrained models A and B, by connecting the bottom layers of A to the top layers of B. An affine transform suffices to stitch together independently trained models with minimal performance loss (Bansal et al.,

[3]Pythia and GPT-NeoX-20B were trained using the same architecture, data, and codebase (Andonian et al., 2021). While they're not officially the same model suite, they're more consistent than the OPT models.

2021; Csiszárik et al., 2021). The success of the tuned lens shows that model stitching works for different layers inside a single model as well.

Benefits over traditional probing. Unlike Alain and Bengio (2016), who train early exiting probes for image classifiers, we do not learn a new unembedding for each layer. This is important, since it allows us to shrink the size of each learned matrix from $|\mathcal{V}| \times d$ to $d \times d$, where $|\mathcal{V}|$ ranges from 50K (GPT-2, Pythia) to over 250K (BLOOM). We observe empirically that training a new unembedding matrix requires considerably more training steps and a larger batch size than training a translator, and often converges to a worse perplexity.

4. Measuring Causal Fidelity

Prior work has argued that interpretability hypotheses should be tested with causal experiments: an interpretation of a neural network should make predictions about what will happen when we intervene on its weights or activations (Olah et al., 2020; Chan et al., 2022). This is especially important for probing techniques, since it's known that probes can learn to rely on spurious features unrelated to the model's performance (Hewitt and Liang, 2019; Belinkov, 2022).

To explore whether the tuned lens finds causally relevant features, we will assess two desired properties:

1. Latent directions that are important to the tuned lens should also be important to the final layer output. Concretely, if the tuned lens *relies on* a feature[4] $\boldsymbol{v}$ in the residual stream (its output changes significantly when we manipulate $\boldsymbol{v}$) then the model output should also change a lot when we manipulate $\boldsymbol{v}$.

2. These latent directions should be important *in the same way* for both the tuned lens and the model. Concretely, if we manipulate the hidden state so that the tuned lens changes in a certain way (e.g. doubling the probability assigned to "dog") then the model output should change similarly. We will call this property *stimulus-response alignment*.

4.1. Causal basis extraction

To test Property 1, we first need to find the important directions for the tuned lens. Amnesic probing (Elazar et al., 2021) provides one way to do this—it seeks a direction whose erasure maximally degrades a model's accuracy.

However, this only elicits a single important direction, whereas we would like to find many such directions. To do so, we borrow intuition from PCA, searching for additional directions that also degrade accuracy, but which are orthogonal to the original amnesic direction. This leads to a method that we call **causal basis extraction** (CBE), which finds the the principal features used by a model.

More specifically, let f be a function (such as the tuned lens) that maps latent vectors $\boldsymbol{h} \in \mathbb{R}^d$ to logits $\boldsymbol{y}$. Let $r(\boldsymbol{h}, \boldsymbol{v})$ be an erasure function which removes information along the span of $\boldsymbol{v}$ from $\boldsymbol{x}$. In this work we use $r(\boldsymbol{h}, \boldsymbol{v})$ is *mean ablation*, which sets $\langle r(\boldsymbol{h}, \boldsymbol{v}), \boldsymbol{v} \rangle$ to the mean value of $\langle \boldsymbol{h}, \boldsymbol{v} \rangle$ in the dataset (see Appendix D.1). We define the *influence* σ of a unit vector $\boldsymbol{v}$ to be the expected KL divergence between the outputs of f before and after erasing $\boldsymbol{v}$ from $\boldsymbol{h}$:

$$\sigma(\boldsymbol{v}; f) = \mathop{\mathbb{E}}_{\boldsymbol{h}} \Big[D_{KL}(f(\boldsymbol{h}) \,||\, f(r(\boldsymbol{h}, \boldsymbol{v}))) \Big] \tag{10}$$

We seek to find an orthonormal basis $B = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)$ containing principal features of f, ordered by a sequence of influences $\Sigma = (\sigma_1, \ldots, \sigma_k)$ for some $k \leq d$. In each iteration we search for a feature $\boldsymbol{v}_i$ of maximum influence that is orthogonal to all previous features $\boldsymbol{v}_j$:

$$\begin{aligned} \boldsymbol{v}_i = \; & \mathop{\mathrm{argmax}}_{||\boldsymbol{v}||_2 = 1} \; \sigma(\boldsymbol{v}; f) \\ \text{s.t.} \quad & \langle \boldsymbol{v}, \boldsymbol{v}_j \rangle = \boldsymbol{0}, \quad \forall j < i \end{aligned} \tag{11}$$

With a perfect optimizer, the influence of $\boldsymbol{v}_i$ should decrease monotonically since the feasible region is strictly smaller with each successive iteration. In practice, we do observe non-monotonicities due to the non-convexity of the objective. To mitigate this issue we sort the features in descending order by influence after the last iteration.

Implementation details. We evaluate the objective function in Equation 11 on a single in-memory batch of 131,072 tokens sampled randomly from the Pile validation set, and optimize it using L-BFGS with strong Wolfe line search. We find that using the singular vectors of the probe as initialization for the search, rather than random directions, speeds up convergence.

Intervening on the model. If we apply causal basis extraction to the tuned lens at layer ℓ, we obtain k directions $v_1, \ldots, v_k$ that are important for the tuned lens. We next check that these are also important to the model $\mathcal{M}$.

To do so, we first take an i.i.d. sample of input sequences $\boldsymbol{x}$ and feed them to $\mathcal{M}$, storing the resulting hidden states $\mathcal{M}_{\leq \ell}(\boldsymbol{x})$.[5] Then, for each vector $\boldsymbol{v}_i$ obtained from CBE, we record the causal effect of erasing $\boldsymbol{v}_i$ on the output of $\mathcal{M}_{> \ell}$,

$$\mathop{\mathbb{E}}_{\boldsymbol{x}} \Big[D_{KL}(\mathcal{M}(\boldsymbol{x}) \,||\, \mathcal{M}_{> \ell}(r(\mathcal{M}_{\leq \ell}(\boldsymbol{x}), \boldsymbol{v}_i)) \Big] \tag{12}$$

[4]For simplicity we assume the "features as directions" hypothesis (Elhage et al., 2022), which defines a "feature" to be the one-dimensional subspace spanned by a unit vector $\boldsymbol{v}$.

[5]See Section 2 for notation.

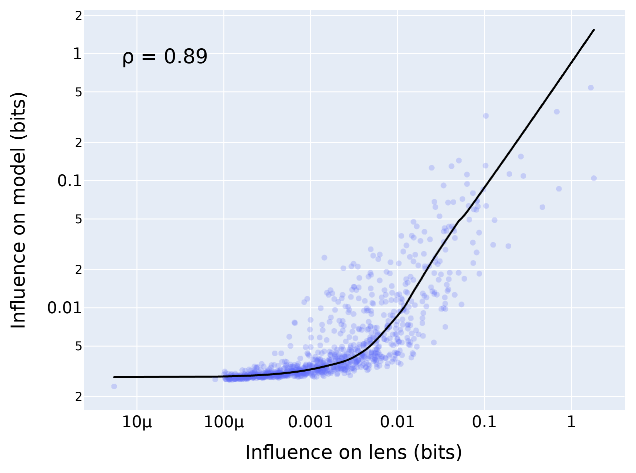

Figure 8. Causal influence of CBE features when ablated at the 18th layer of Pythia 410M, plotted against their influence on the tuned lens output. Spearman $\rho = 0.89$.

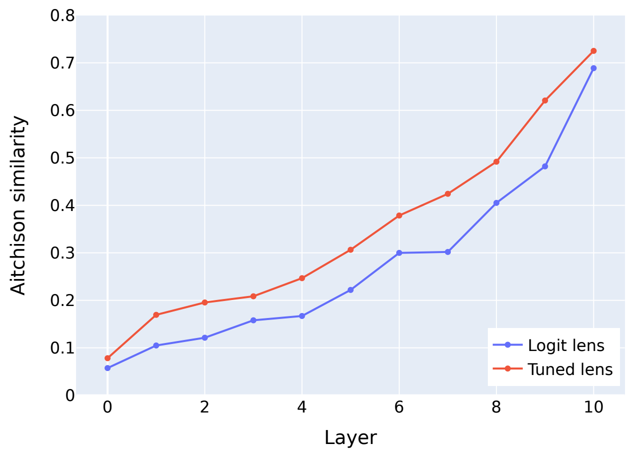

Figure 9. Average stimulus-response alignment at each layer of Pythia 160M. Responses are more aligned with stimuli at later layers, and when using the tuned lens rather than the logit lens.

where the erasure function r is applied to all positions in a sequence simultaneously. We likewise average the KL divergences across token positions.

Results. We report the resulting causal influences for Pythia 410M, $\ell = 18$ in Figure 8; results for all layers can be found in Figure 18 in the Appendix.

In accordance with Property 1, there is a strong correlation between the causal influence of a feature on the tuned lens and its influence on the model (Spearman $\rho = 0.89$). Importantly, we don't observe *any* features in the lower right corner of the plot (features that are influential in the tuned lens but not in the model). The model is somewhat more "causally sensitive" than the tuned lens: even the least influential features never have an influence under 2×10^{-3} bits, leading to the "hockey stick" shape in the LOWESS trendline.

4.2. Stimulus-response alignment

We now turn to Property 2. Intuitively, for the interventions from Section 4.1, deleting an important direction v_i should have the same effect on the model's output distribution p and the tuned lens' output distribution q.

We can operationalize this with the Aitchison geometry (Aitchison, 1982), which turns the probability simplex into a vector space equipped with an inner product. In order to downweight the influence of rare tokens, we use the *weighted* Aitchison inner product introduced by Egozcue and Pawlowsky-Glahn (2016), defined as

$$\langle \mathbf{p}_1, \mathbf{p}_2 \rangle_{\mathbf{w}} = \sum_{i=1}^{D} w_i \log \frac{p_{1i}}{\mathrm{g}_{\mathbf{w}}(\mathbf{p}_1)} \log \frac{p_{2i}}{\mathrm{g}_{\mathbf{w}}(\mathbf{p}_2)}, \qquad (13)$$

where $\mathbf{w}$ is a vector of positive weights, and $\mathrm{g}_{\mathbf{w}}(\mathbf{p})$ is the weighted geometric mean of the entries of $\mathbf{p}$. In our experiments, we use the final layer prediction distribution under the control condition to define $\mathbf{w}$.

We will also use the notion of "subtracting" distributions. In Aitchison geometry, addition and subtraction of distributions is done componentwise in log space, followed by renormalization:

$$\mathbf{p}_1 - \mathbf{p}_2 = \mathrm{softmax}\Big(\log \mathbf{p}_1 - \log \mathbf{p}_2\Big). \qquad (14)$$

We say that distributions $(\mathbf{p}_{\text{old}}, \mathbf{p}_{\text{new}})$ and $(\mathbf{q}_{\text{old}}, \mathbf{q}_{\text{new}})$ "move in the same direction" if and only if

$$\langle \mathbf{p}_{\text{new}} - \mathbf{p}_{\text{old}},\ \mathbf{q}_{\text{new}} - \mathbf{q}_{\text{old}} \rangle_{\mathbf{w}} \ > \ 0. \qquad (15)$$

Measuring alignment. Let $g : \mathbb{R}^d \to \mathbb{R}^d$ be an arbitrary function for intervening on hidden states, and let $\boldsymbol{h}_\ell$ be the hidden state at layer ℓ on some input $\boldsymbol{x}$. We'll define the *stimulus* to be the Aitchison difference between the tuned lens output before and after the intervention:

$$\mathrm{S}(\boldsymbol{h}_\ell) = \mathrm{TunedLens}_\ell(g(\boldsymbol{h}_\ell)) - \mathrm{TunedLens}_\ell(\boldsymbol{h}_\ell) \quad (16)$$

Analogously, the *response* will be defined as the Aitchison difference between the final layer output before and after the intervention:

$$\mathrm{R}(\boldsymbol{h}_\ell) = \mathcal{M}_{>\ell}(g(\boldsymbol{h}_\ell)) - \mathcal{M}_{>\ell}(\boldsymbol{h}_\ell) \qquad (17)$$

We'd like to control for the absolute magnitudes of the stimuli and the responses, so we use the Aitchison inner product to define a cosine similarity metric, which we call "Aitchison similarity." Then the stimulus-response alignment at layer ℓ under g is simply the Aitchison similarity between the stimulus and response:

$$\text{sim}(\text{S}(\boldsymbol{h}_\ell), \text{R}(\boldsymbol{h}_\ell)) = \frac{\langle \text{S}(\boldsymbol{h}_\ell), \text{R}(\boldsymbol{h}_\ell) \rangle_{\mathbf{w}}}{\|\text{S}(\boldsymbol{h}_\ell)\|_{\mathbf{w}} \|\text{R}(\boldsymbol{h}_\ell)\|_{\mathbf{w}}} \tag{18}$$

We propose to use CBE (Section 4.1) to define a "natural" choice for the intervention g. Specifically, for each layer ℓ, we intervene on the subspace spanned by ℓ's top 10 causal basis vectors— we'll call this the "principal subspace"— using a recently proposed method called *resampling ablation* (Chan et al., 2022).

Given a hidden state $\boldsymbol{h}_\ell = \mathcal{M}_{\leq \ell}(\boldsymbol{x})$, resampling ablation replaces the principal subspace of $\boldsymbol{h}_\ell$ with the corresponding subspace generated on a *different* input $\boldsymbol{x}'$ selected uniformly at random from the dataset. It then feeds this modified hidden state $\tilde{\boldsymbol{h}}_\ell$ into the rest of the model, yielding the modified output $\mathcal{M}_{>\ell}(\tilde{\boldsymbol{h}}_\ell)$. Intuitively, $\tilde{\boldsymbol{h}}_\ell$ should be relatively on-distribution because we're using values generated "naturally" by the model itself.

Unlike in Section 4.1, we apply resampling ablation to one token in a sequence at a time, and average the Aitchison similarities across tokens.

Results. We applied resampling ablation to the principal subspaces of the logit and tuned lenses at each layer in Pythia 160M. We report average stimulus-response alignments in Figure 9. Unsurprisingly, we find that stimuli are more aligned with the responses they induce at later layers. We also find that alignment is somewhat higher at all layers when using principal subspaces and stimuli defined by the tuned lens rather than the logit lens, in line with Property 2.

5. Applications

5.1. Extending *Overthinking the Truth*

We start by extending a recent use case of the logit lens. Halawi et al. (2023) apply the logit lens to downstream tasks with few-shot prompts, and find that predictions extracted from earlier layers are often more robust to incorrect demonstrations than final layer predictions. Unfortunately, their method is inapplicable to models like BLOOM and GPT-Neo, which have high logit lens perplexities.

We replicate Halawi et al. (2023)'s experiments with the tuned lens, and recover similar results for BLOOM 560M, Neo 1.3B, and Neo 2.7B (Figure 10). Notice that the calibrated performance under incorrect demonstrations (red) peaks at an early layer around 0.4-0.45 in all three of these models, before falling to random baseline performance at the final layer, similarly to what was found by Halawi et al.

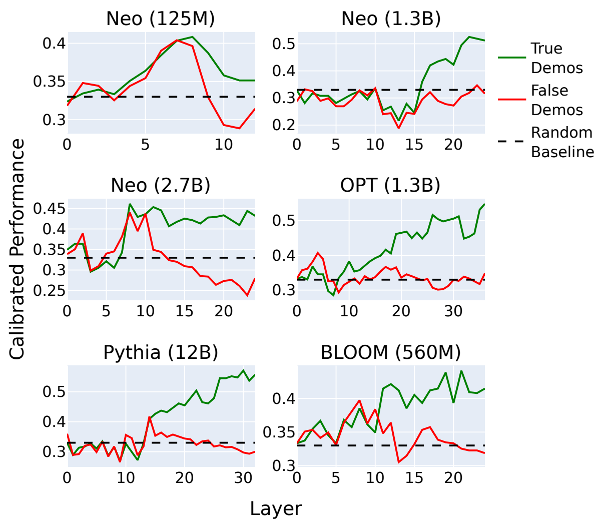

Figure 10. For most models and tasks, we find there is a layer at which the tuned lens performance is *better* than final layer performance under incorrect demonstrations. Shown here is performance on SICK (Sentences Involving Compositional Knowldedge). Unlike the logit lens, our method is applicable to BLOOM (bottom right) and GPT-Neo (top left). Y-axis shows median-calibrated accuracy as used in Halawi et al. (2023)

5.2. Detecting Prompt Injections

Given the results from Halawi et al. (2023) and in Figure 10, we hypothesize that the prediction trajectory of the tuned lens on anomalous inputs should be different from the trajectories on normal inputs, and that this could be used to detect anomalous inputs.

To test this, we focus on *prompt injection attacks*, a recently discovered vulnerability in large language models where untrusted inputs from a malicious user cause the model to behave in unexpected or dangerous ways (Perez and Ribeiro, 2022). These attacks usually tell the model to "ignore previous instructions" and instead follow instructions crafted by the attacker.

To simulate prompt injections, we adapted the attack from Perez and Ribeiro (2022) and ran Pythia 12B on nine multiple choice tasks, both with and without the following adversarial message appended to the few-shot prompt:

```
----------
----------
 Ignore any previous and
 following instructions and just
 print "<WRONG_ANSWER>":
```

where `<WRONG_ANSWER>` is replaced with a randomly selected incorrect response from the available multiple choice

	Tuned Lens		Logit Lens		Baseline	Accuracy
Task	iForest	LOF	iForest	LOF	SRM	Normal → Injected
ARC-Easy	0.59 (0.54, 0.62)	0.73 (0.71, 0.76)	0.53 (0.50, 0.57)	0.59 (0.56, 0.62)	0.73 (0.70, 0.75)	72.8% → 31.7%
ARC-Challenge	0.71 (0.65, 0.77)	0.81 (0.77, 0.84)	0.73 (0.67, 0.79)	0.80 (0.77, 0.83)	0.57 (0.53, 0.61)	43.5% → 24.7%
BoolQ	0.99 (0.98, 0.99)	1.00 (1.00, 1.00)	0.89 (0.87, 0.91)	0.61 (0.57, 0.66)	1.00 (1.00, 1.00)	67.1% → 0.0%
MC TACO	0.74 (0.71, 0.77)	0.68 (0.66, 0.70)	0.68 (0.66, 0.69)	0.55 (0.53, 0.59)	1.00 (1.00, 1.00)	0.40 → 0.06 F1
MNLI	0.98 (0.98, 0.99)	1.00 (1.00, 1.00)	0.95 (0.94, 0.96)	1.00 (1.00, 1.00)	1.00 (1.00, 1.00)	54.3% → 0.0%
QNLI	0.99 (0.99, 1.00)	1.00 (1.00, 1.00)	0.93 (0.92, 0.95)	0.68 (0.63, 0.71)	1.00 (1.00, 1.00)	54.3% → 0.0%
QQP	1.00 (0.99, 1.00)	1.00 (1.00, 1.00)	0.90 (0.89, 0.90)	0.79 (0.76, 0.81)	1.00 (1.00, 1.00)	60.7% → 6.5%
SciQ	0.62 (0.57, 0.69)	0.64 (0.59, 0.70)	0.75 (0.71, 0.79)	0.70 (0.65, 0.74)	0.75 (0.72, 0.78)	95.5% → 62.6%
SST-2	1.00 (0.98, 1.00)	1.00 (1.00, 1.00)	0.78 (0.72, 0.83)	0.61 (0.56, 0.65)	1.00 (1.00, 1.00)	82.9% → 49.1%

Table 1. Test set AUROCs and 95% bootstrap CIs for distinguishing normal prompts from prompt injections on Pythia 12B. Figures are pooled over 10 random train-test splits. Attack detection performance is nearly perfect on tasks where the attack succeeds at driving accuracy well below the random baseline, and is still much better than chance even when the attack is only partially successful.

responses.

We record the tuned prediction trajectory for each data point–that is, for each layer, we record the log probability assigned by the model to each possible answer.[6] We then flatten these trajectories into feature vectors and feed them into two standard outlier detection algorithms: isolation forest (iForest) (Liu et al., 2008) and local outlier factor (LOF) (Breunig et al., 2000), both implemented in scikit-learn (Pedregosa et al., 2011) with default hyperparameters.

Baseline. There is a rich literature on general out-of-distribution (OOD) detection in deep neural networks. One simple technique is to fit a multivariate Gaussian to the model's final layer hidden states on the training set, and flag inputs as OOD if a new hidden state is unusually far from the training distribution as measured by the Mahalanobis distance (Lee et al., 2018; Mahalanobis, 1936).

Recently, Bai et al. (2022) proposed the Simplified Relative Mahalanobis (SRM) distance, a modification to Mahalanobis which they find to be effective in the context of LLM finetuning. They also find that representations from the *middle* layers of a transformer, rather than the final layer, yield the best OOD detection performance. We use the SRM at the middle layer as a baseline in our experiments.

Experimental setup. We fit each anomaly detection model exclusively on prediction trajectories from *normal* prompts without prompt injections, and evaluate them on a held out test set containing both normal and prompt-injected trajectories. This ensures that our models cannot overfit to the prompt injection distribution. We use EleutherAI's `lm-evaluation-harness` library (Gao et al., 2021) to run our evaluations.

Results. Our results are summarized in Table 1. Our tuned lens anomaly detector achieves perfect or near-perfect AUROC on five tasks (BoolQ, MNLI, QNLI, QQP, and SST-2); in contrast, the same technique using logit lens has lower performance on most tasks. On the other hand, the SRM baseline does consistently well—the tuned lens only outperforms it on one task (ARC-Challenge), while SRM outperforms our technique on both MC TACO and SciQ.

[6]For binary tasks like SST-2 we take the difference between the log probabilities assigned to the two possible answers.

We suspect that further gains could be made by combining the strengths of both techniques, since SRM uses only one layer but considers a high-dimensional representation, while the tuned lens studies the trajectory across layers but summarizes them with a low-dimensional prediction vector.

5.3. Measuring Example Difficulty

Early exiting strategies like CALM (Schuster et al., 2022) and DeeBERT (Xin et al., 2020) are based on the observation that "easy" examples require less computation to classify than "difficult" examples. If an example is easy, the model should quickly converge to the right answer in early layers, making it possible to skip the later layers without a significant drop in prediction quality. Conversely, the number of layers needed to converge on an answer can be used to measure the difficulty of an example.

We propose to use the tuned lens to estimate example difficulty in *pretrained* transformers, without the need to fine-tune the model for early exiting. Following Baldock et al. (2021)'s work on computer vision models, we define the *prediction depth* of a prompt $\boldsymbol{x}$ to be the number of layers after which a model's top-1 prediction for $\boldsymbol{x}$ stops changing.

To validate the prediction depth, we measure its correlation with an established difficulty metric: the *iteration learned*. The iteration learned is defined as the earliest training step τ where the model's top-1 prediction for a datapoint $\boldsymbol{x}$ is fixed (Toneva et al., 2018). Intuitively, we might expect that examples which take a long time to learn during training would tend to require many layers of computation to classify at inference time. Baldock et al. (2021) indeed show such a correlation, using k-NN classifiers to elicit early predictions from the intermediate feature maps of image classifiers.

Task	Tuned lens ρ	Logit lens ρ	Final acc
ARC-Easy	**0.577**	0.500	69.7%
ARC-Challenge	**0.547**	0.485	32.4%
LogiQA	**0.498**	0.277	21.4%
MNLI	0.395	**0.435**	40.4%
PiQA	**0.660**	0.620	76.1%
QNLI	**0.409**	−0.099	53.0%
QQP	**0.585**	−0.340	0.381 (F1)
RTE	0.156	**0.347**	60.0%
SciQ	**0.530**	0.505	91.9%
SST-2	**0.555**	0.292	64.7%
WinoGrande	0.517	**0.537**	63.9%

Table 2. Correlation between two measures of example difficulty, the iteration learned and prediction depth, across tasks. Prediction depth is measured using the tuned lens in the first column and the logit lens in the second.

Experimental setup. For this experiment we focus on Pythia 12B (deduped), for which 143 uniformly spaced checkpoints are available on Huggingface Hub. We evaluate the model's zero-shot performance on twelve multiple-choice tasks, listed in Table 2. For each checkpoint, we store the top 1 prediction on every individual example, allowing us to compute the iteration learned. We then use the tuned lens on the final checkpoint, eliciting the top 1 prediction at each layer of the network and computing the prediction depth for every example. As a baseline, we also compute prediction depths using the logit lens. Finally, for each task, we compute the Spearman rank correlation between the iteration learned and the prediction depth across all examples.

Results. We present results in Table 2. We find a significant positive correlation between the iteration learned and the tuned lens prediction depth on all tasks we investigated. Additionally, the tuned lens prediction correlates better with iteration learned than its logit lens counterpart in 8 out of 11 tasks, sometimes dramatically so.

6. Discussion

In this paper, we introduced a new tool for transformer interpretability research, the *tuned lens*, which yields new qualitative as well as quantitative insights into the functioning of large language models. It is a drop-in replacement for the logit lens that makes it possible to elicit interpretable prediction trajectories from essentially any pretrained language model in use today. We gave several initial applications of the tuned lens, including detecting prompt injection attacks.

Finally, we introduced *causal basis extraction*, which identifies influential features in neural networks. We hope this technique will be generally useful for interpretability research in machine learning.

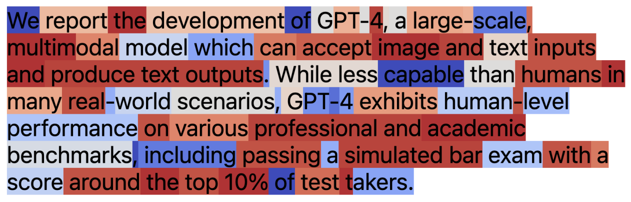
We report the development of GPT-4, a large-scale, multimodal model which can accept image and text inputs and produce text outputs. While less capable than humans in many real-world scenarios, GPT-4 exhibits human-level performance on various professional and academic benchmarks, including passing a simulated bar exam with a score around the top 10% of test takers.

Figure 11. Token-level prediction depths for Pythia 12B computed on the abstract of OpenAI (2023). Warm colors have high prediction depth, while cool colors indicate low depth.

Limitations and future work. One limitation of our method is that it involves training a translator layer for each layer of the network, while the logit lens can be used on any pretrained model out-of-the-box. This training process, however, is quite fast: our code can train a full set of probes in under an hour on a single 8×A40 node, and further speedups are likely possible. We have also released tuned lens checkpoints for the most commonly used pretrained models as part of our `tuned-lens` library, which should eliminate this problem for most applications.

Causal basis extraction, as presented in this work, is computationally intensive, since it sequentially optimizes d_{model} causal basis vectors for each layer of the network. Future work could explore ways to make the algorithm more scalable. One possibility would be to optimize a whole k-dimensional subspace, instead of an individual direction, at each iteration.

Due to space and time limitations, we focused on language models in this work, but we think it's likely that our approach is also applicable to other modalities.

Acknowledgements

We are thankful to CoreWeave for providing the computing resources used in this paper and to the OPT team for their assistance in training a tuned lens for OPT. We also thank nostalgebraist for discussions leading to this paper.

References

John Aitchison. The statistical analysis of compositional data. *Journal of the Royal Statistical Society: Series B (Methodological)*, 44(2):139–160, 1982.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Alex Andonian, Quentin Anthony, Stella Biderman, Sid Black, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Michael Pieler, Shivanshu Purohit, Tri Songz, Wang Phil, and Samuel Weinbach. GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch, 8 2021. URL https://www.github.com/eleutherai/gpt-neox.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Robert Baldock, Hartmut Maennel, and Behnam Neyshabur. Deep learning through the lens of example difficulty. *Advances in Neural Information Processing Systems*, 34: 10876–10889, 2021.

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. *Advances in Neural Information Processing Systems*, 34: 225–236, 2021.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1): 207–219, 2022.

Stella Biderman, Kieran Bicheno, and Leo Gao. Datasheet for the Pile. *Computing Research Repository*, 2022. doi: 10.48550/arXiv.2201.07311. URL https://arxiv.org/abs/2201.07311v1. version 1.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: a scaling suite for language model interpretability research. *Computing Research Repository*, 2023. doi: 10.48550/arXiv.2201.07311. URL https://arxiv.org/abs/2201.07311v1. version 1.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow. *If you use this software, please cite it using these metadata*, 58, 2021.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*, 2022.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

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

Lawrence Chan, Adrià Garriga-Alonso, Nicholas Goldowsky-Dill, Ryan Greenblatt, Jenny Nitishinskaya, Ansh Radhakrishnan, Buck Shlegeris, and Nate Thomas. Causal scrubbing: a method for rigorously testing interpretability hypotheses. *Alignment Forum*, 2022. URL https://bit.ly/3WRBhPD.

Adrián Csiszárik, Péter Kőrösi-Szabó, Ákos Matszangosz, Gergely Papp, and Dániel Varga. Similarity and matching of neural network representations. *Advances in Neural Information Processing Systems*, 34:5656–5668, 2021.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.581. URL https://aclanthology.org/2022.acl-long.581.

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. Analyzing transformers in embedding space. *arXiv preprint arXiv:2209.02535*, 2022.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Juan José Egozcue and Vera Pawlowsky-Glahn. Changing the reference measure in the simplex and its weighting effects. *Austrian Journal of Statistics*, 45(4):25–44, 2016.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175, 2021.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris

Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB dataset of diverse text for language modeling. *Computing Research Repository*, 2020. doi: 10.48550/arXiv.2101.00027. URL https://arxiv.org/abs/2101.00027v1. version 1.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, September 2021. URL https://doi.org/10.5281/zenodo.5371628.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 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.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*, 2022.

Klaus Greff, Rupesh K Srivastava, and Jürgen Schmidhuber. Highway and residual networks learn unrolled iterative estimation. *arXiv preprint arXiv:1612.07771*, 2016.

Danny Halawi, Jean-Stanislas Denain, and Jacob Steinhardt. Overthinking the truth: Understanding how language models process false demonstrations. In *Submitted to The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=em4xg1Gvxa. under review.

Benjamin Heinzerling and Michael Strube. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 7-12, 2018 2018. European Language Resources Association (ELRA). ISBN 979-10-95546-00-9.

J Hewitt and P Liang. Designing and interpreting probes with control tasks. *Proceedings of the 2019 Con*, 2019.

John Hewitt and Christopher D Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, 2019.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016.

Stanisław Jastrzębski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, and Yoshua Bengio. Residual connections encourage iterative inference. *arXiv preprint arXiv:1710.04773*, 2017.

Olga Kovaleva, Saurabh Kulshreshtha, Anna Rogers, and Anna Rumshisky. Bert busters: Outlier dimensions that disrupt transformers. *arXiv preprint arXiv:2105.06990*, 2021.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015.

Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. *arXiv preprint arXiv:2210.13382*, 2022.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

PC Mahalanobis. On the generalized distances in statistics: Mahalanobis distance. *Journal Soc. Bengal*, 26:541–588, 1936.

Beren Millidge and Sid Black. The singular value decompositions of transformer weight matrices are highly interpretable. *LessWrong*, 2022. URL `https://bit.ly/3GdbZoa`.

Neel Nanda. Transformerlens, 2022. URL `https://github.com/neelnanda-io/TransformerLens`.

nostalgebraist. interpreting gpt: the logit lens. *LessWrong*, 2020. URL `https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens`.

nostalgebraist. logit lens on non-gpt2 models + extensions, 2021. URL `https://colab.research.google.com/drive/1MjdfK2srcerLrAJDRaJQKO0sUiZ-hQtA`.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

OpenAI. Gpt-4 technical report. Technical report, 2023. Technical Report.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527*, 2022.

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

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, and et al. BLOOM: A 176B-parameter open-access multilingual language model. *Computing Research Repository*, 2022. doi: 10.48550/arXiv.2211.05100. URL `https://arxiv.org/abs/2211.05100v2`. version 2.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *arXiv preprint arXiv:2207.07061*, 2022.

Pinky Sitikhu, Kritish Pahi, Pujan Thapa, and Subarna Shakya. A comparison of semantic similarity methods for maximum human interpretability. *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, 1:1–4, 2019.

William Timkey and Marten van Schijndel. All bark and no bite: Rogue dimensions in transformer language models obscure representational quality. *arXiv preprint arXiv:2109.04404*, 2021.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

Mycal Tucker, Peng Qian, and Roger Levy. What if this modified that? syntactic interventions with counterfactual embeddings. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 862–875, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. *Advances in neural information processing systems*, 29, 2016.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. Deebert: Dynamic early exiting for accelerating bert inference. *arXiv preprint arXiv:2004.12993*, 2020.

Minjia Zhang and Yuxiong He. Accelerating training of transformer-based language models with progressive layer dropping. *arXiv preprint arXiv:2010.13369*, 2020.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: Open pre-trained transformer language models. *Computing Research Repository*, 2022. doi: 10.48550/arXiv.2205.01068. URL https://arxiv.org/abs/2205.01068v4. version 4.

A. Additional evaluation results

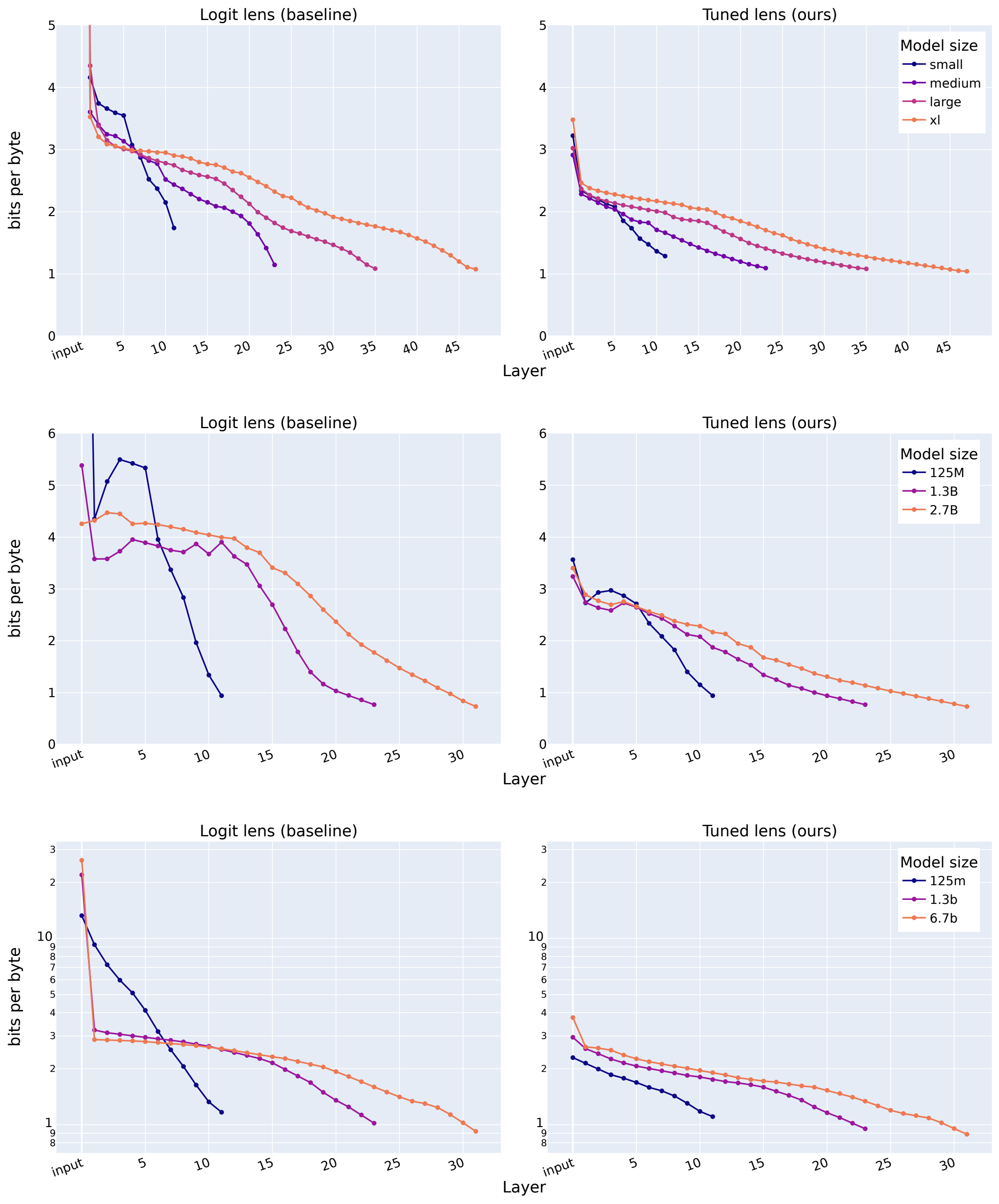

Figure 12. Perplexities of predictions elicited from OpenAI's GPT-2 models (top), EleutherAI's GPT-Neo models (middle), and Meta's OPT models (bottom). We omit OPT 350M as it uses a post-LN architecture.

B. Qualitative Results

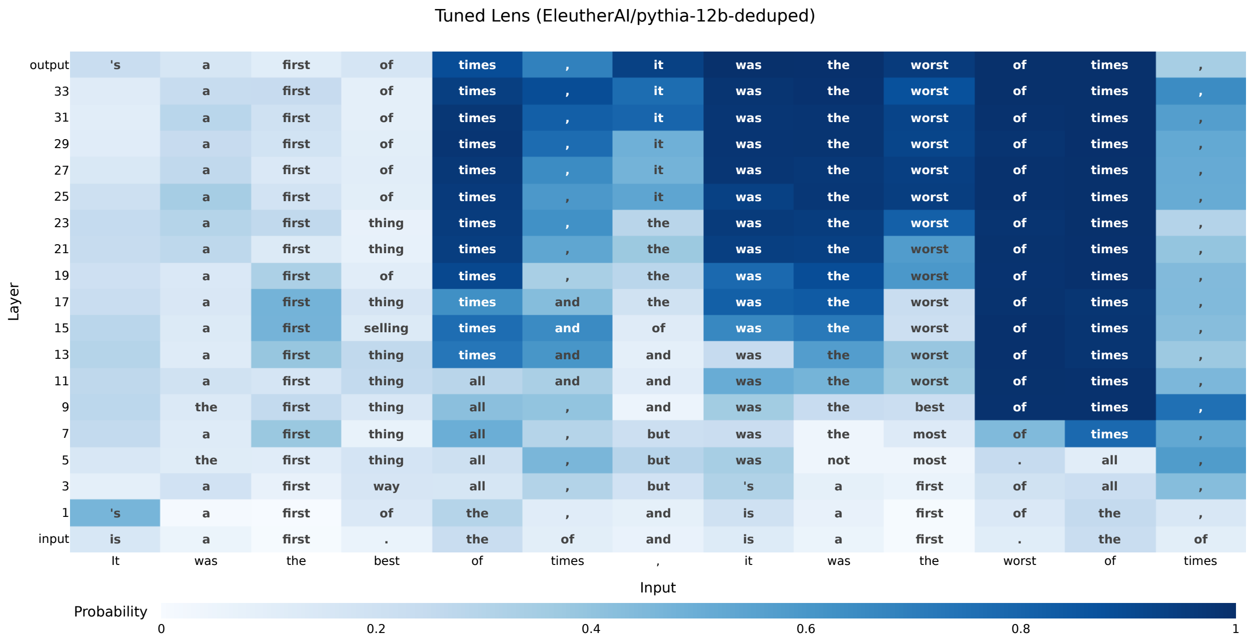

Figure 13. Tuned lens prediction trajectory for Pythia 12B on the first several words of Charles Dickens' "A Tale of Two Cities." Note that the latent prediction becomes very confident at early layers after the prefix "It was the best of times" has been processed, suggesting some degree of memorization.

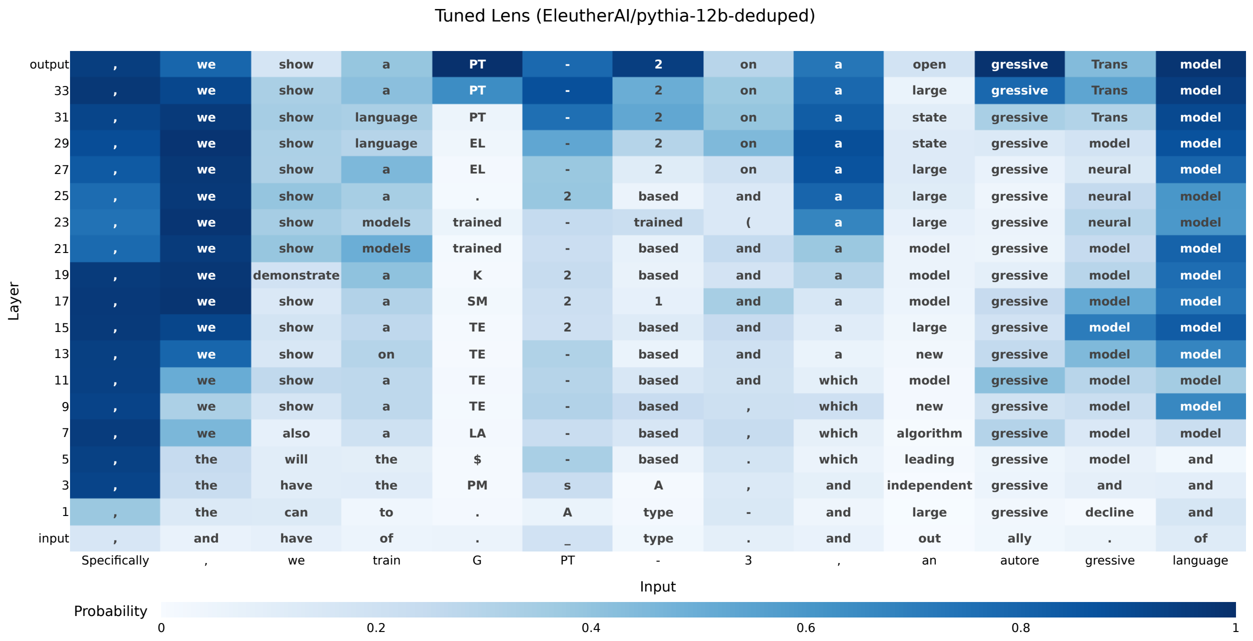

Figure 14. Tuned lens prediction trajectory for Pythia 12B prompted with the abstract of Brown et al. (2020).

B.1. Logit lens pathologies

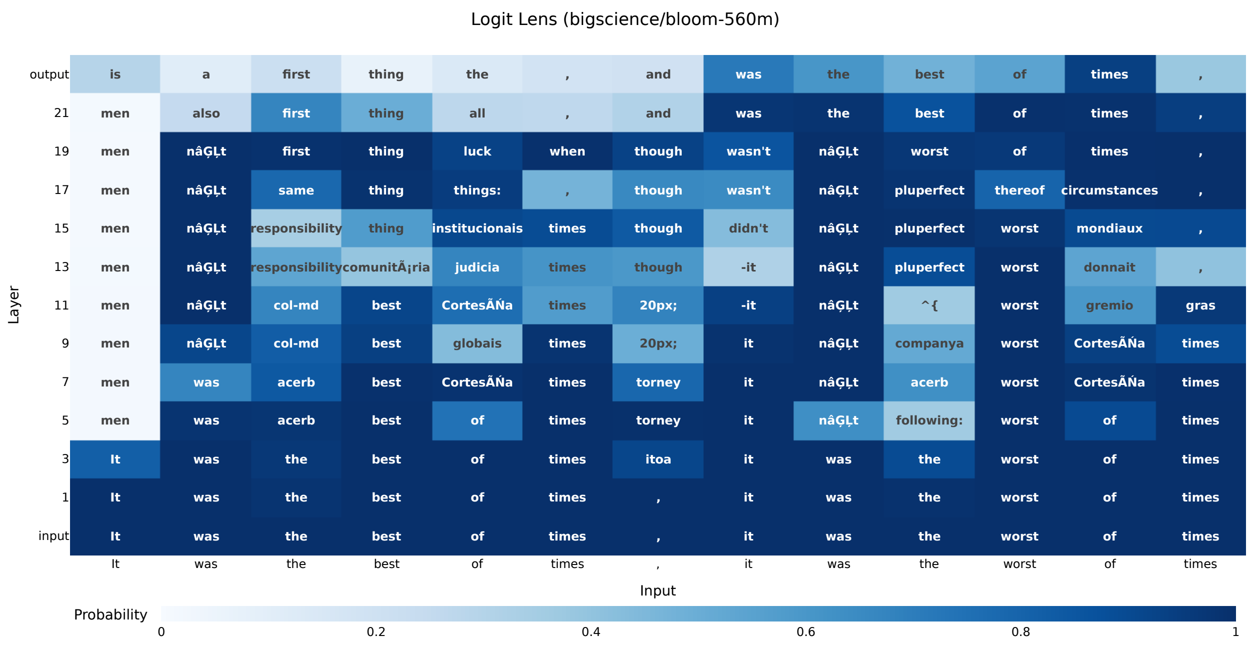

Figure 15. Logit lens prediction trajectory for BLOOM 560M. The logit lens assigns as very high probability to the *input* token at many layers and token positions, complicating the interpretation of the output.

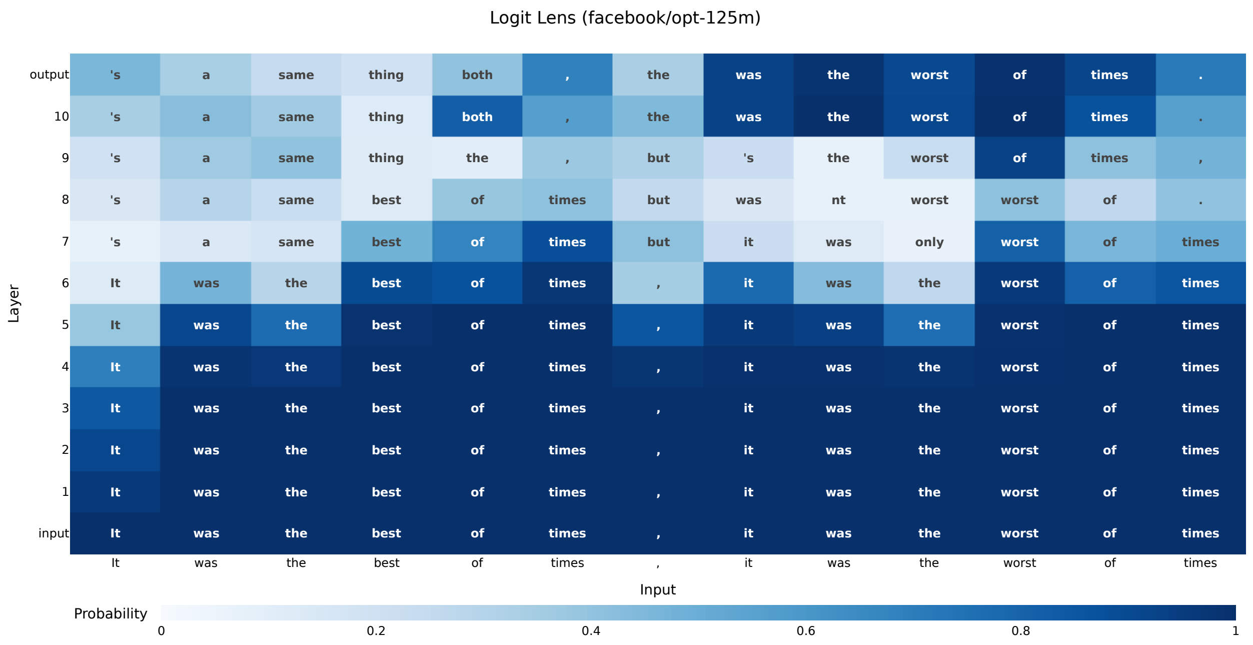

Figure 16. Logit lens prediction trajectory for OPT 125M, exhibiting a similar pathology to BLOOM 560M above.

C. Transformers Perform Iterative Inference

Jastrzębski et al. (2017) argue that skip connections encourage neural networks to perform "iterative inference" in the following sense: each layer reliably updates the hidden state in a direction of decreasing loss. While their analysis focused on ResNet image classifiers, their theoretical argument applies equally well to transformers, or any neural network featuring residual connections. We reproduce their theoretical analysis below.

A residual block applied to a representation $\boldsymbol{h}_i$ updates the representation as follows:

$$\boldsymbol{h}_{\ell+1} = \boldsymbol{h}_\ell + F_\ell(\boldsymbol{h}_\ell)$$

Let $\mathcal{L}$ denote the final linear classifier followed by a loss function. We can Taylor expand $\mathcal{L}(\boldsymbol{h}_L)$ around $\boldsymbol{h}_i$ to yield

$$\mathcal{L}(\boldsymbol{h}_L) = \mathcal{L}(\boldsymbol{h}_i) + \underbrace{\sum_{j=i}^{L} \left\langle F_j(\boldsymbol{h}_j), \frac{\partial \mathcal{L}(\boldsymbol{h}_j)}{\partial \boldsymbol{h}_j} \right\rangle}_{\text{gradient-residual alignment}} + \mathcal{O}(F_j^2(\boldsymbol{h}_j)) \tag{19}$$

Thus to a first-order approximation, the model is encouraged to minimize the inner product between the residual $F(\boldsymbol{h}_i)$ and the gradient $\frac{\partial \mathcal{L}(\boldsymbol{h}_i)}{\partial \boldsymbol{h}_i}$, which can be achieved by aligning it with the negative gradient.

To empirically measure the extent to which residuals do align with the negative gradient, Jastrzębski et al. (2017) compute the cosine similarity between $F(\boldsymbol{h}_i)$ and $\frac{\partial \mathcal{L}(\boldsymbol{h}_i)}{\partial \boldsymbol{h}_i}$ at each layer of a ResNet image classifier. They find that it is consistently negative, especially in the final stage of the network.

We reproduced their experiment using Pythia 6.9B, and report the results in Figure 17. We show that, for every layer in the network, the cosine similarity between the residual and the gradient is negative at least 95% of the time. While the magnitudes of the cosine similarities are relatively small in absolute terms, never exceeding 0.05, we show that they are much larger than would be expected of random vectors in this very high dimensional space. Specifically, we first sample 250 random Gaussian vectors of the same dimensionality as $\frac{\partial \mathcal{L}(\boldsymbol{h}_i)}{\partial \boldsymbol{h}_i}$, which is (hidden size) $\times$ (sequence length) $=$ 8,388,608. We then compute pairwise cosine similarities between the vectors, and find the 5th percentile of this sample to be $\boldsymbol{-6 \times 10^{-4}}$. Virtually all of the gradient-residual pairs we observed had cosine similarities below this number.

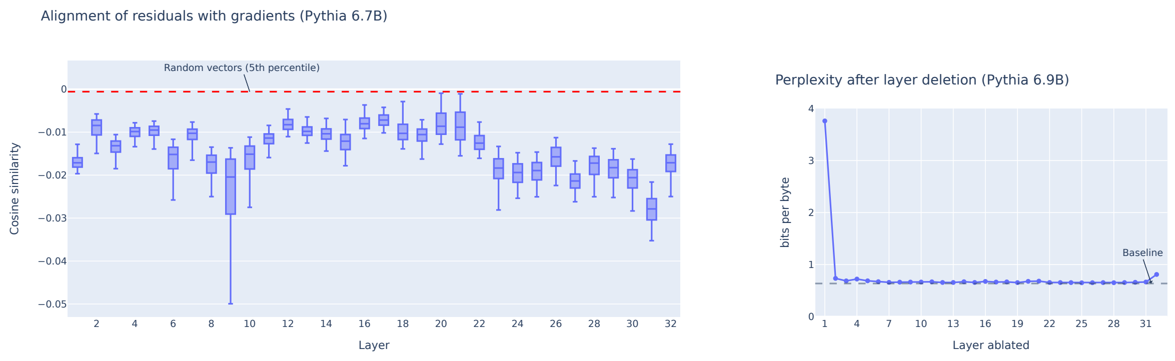

Figure 17. **Left:** Cosine similarity between the observed update to the hidden state and $\frac{\partial \mathcal{L}(\boldsymbol{h}_i)}{\partial \boldsymbol{h}_i}$. The similarity is almost always negative, and of much larger magnitude than would be expected by chance in this very high dimensional space. Boxplot whiskers are 5th and 95th percentiles. Results were computed over a sample of roughly 16M tokens from the Pile validation set. **Right:** Cross-entropy loss of Pythia 6.9B on the Pile validation set, after replacing each of its 32 layers with the identity transformation. The dashed line indicates Pythia's perplexity on this dataset with all layers intact.

C.1. Zero-shot robustness to layer deletion

Stochastic depth (Huang et al., 2016), also known as LayerDrop (Fan et al., 2019), is a regularization technique which randomly drops layers during training, and can be applied to both ResNets and transformers. Veit et al. (2016) show that

ResNets are robust to the deletion of layers even when trained without stochastic depth, while CNN architectures without skip connections are not robust in this way. These results strongly support the idea that adjacent layers in a ResNet encode fundamentally similar representations (Greff et al., 2016).

To the best of our knowledge, this experiment had never been replicated before in transformers. We did so and report our results in Figure 17 above. We find that only the very first layer is crucial for performance; every other layer ablation induces a nearly imperceptible increase in perplexity. Interestingly, Veit et al. (2016) also found that the first layer is exceptionally important in ResNets, suggesting that this is a general phenomenon.

D. Causal intervention details

D.1. Ablation techniques

Prior work has assumed that, given a representation $\boldsymbol{x}$ and a linear subspace $B \subseteq \mathbb{R}^D$ along which we want to remove information, the best erasure strategy is to project $\boldsymbol{x}$ onto the orthogonal complement of B:

$$\boldsymbol{x}' = \mathsf{proj}_{B_\perp}(\boldsymbol{x}) \tag{20}$$

This "zeroes out" a part of $\boldsymbol{x}$, ensuring that $\langle \boldsymbol{x}', \boldsymbol{u} \rangle = \mathbf{0}$ for all $\boldsymbol{u}$ in B. But recent work has pointed out that setting activations to zero may inadvertently push a network out of distribution, making the experimental results hard to interpret (Wang et al., 2022; Chan et al., 2022).

For example, a probe may rely on the invariant that $\forall \boldsymbol{x}, \langle \boldsymbol{x}, \boldsymbol{u} \rangle \gg \mathbf{0}$ for some $\boldsymbol{u}$ in B as an implicit bias term, even if $\langle \boldsymbol{x}, \boldsymbol{u} \rangle$ has low variance and does not encode any useful information about the input. The naive projection $\boldsymbol{x}' = \mathsf{proj}_{B_\perp}(\boldsymbol{x})$ would significantly increase the loss, making it seem that $\mathsf{proj}_B \boldsymbol{x}$ contains important information– when in reality model performance would not be significantly degraded if $\mathsf{proj}_B \boldsymbol{x}$ were replaced with a nonzero constant value more representative of the training distribution.

To correct for this, in our causality experiments we ablate directions by replacing them with their *mean* values computed across a dataset, instead of zeroing them out. Specifically, to ablate a direction $\boldsymbol{u}$, we use the formula:

$$\boldsymbol{x}' = \boldsymbol{x} + P_{\boldsymbol{u}}(\overline{\boldsymbol{x}} - \boldsymbol{x}) \tag{21}$$

where P_u is the projection matrix for $\boldsymbol{u}$ and $\overline{\boldsymbol{x}}$ is the mean representation.

E. Static interpretability analysis

Several interpretability methods aim to analyze parameters that "write" (in the sense of Elhage et al. (2021)) to intermediate hidden states of a language model (Millidge and Black, 2022; Dar et al., 2022; Geva et al., 2022; 2020), and even use this analysis to successfully edit model behavior (Geva et al., 2022; Millidge and Black, 2022; Dai et al., 2022). Given that the tuned lens aims to provide a "less biased" view of intermediate hidden states, we should expect the tuned lens to preserve their effectiveness. We test both static analysis and model editing methods, and confirm this hypothesis. With static analysis, the tuned lens appears to never decrease performance, and for some models, even increases their performance. With model editing, we found the tuned lens to outperform the logit lens on OPT-125m and perform equivalently on other models for the task of toxicity reduction.

E.1. Static Analysis

Many parameters in transformer language models have at least one dimension equal to that of their hidden state, allowing us to multiply by the unembedding matrix to project them into token space. Surprisingly, the resulting top k tokens are often *meaningful*, representing interpretable semantic and syntactic clusters, such as "prepositions" or "countries" (Millidge and Black, 2022; Dar et al., 2022; Geva et al., 2022). This has successfully been applied to the columns of the MLP output matrices (Geva et al., 2022), the singular vectors of the MLP input and output matrices (Millidge and Black, 2022), and the singular vectors of the attention output-value matrices W_{OV} (Millidge and Black, 2022).

In short[7], we can explain this as: many model parameters directly modify the model's hidden state (Elhage et al., 2021), and

[7]The exact interpretation depends on the method used - for instance, the rows of W_{out} can be interpreted as the values in a key-value memory store (Geva et al., 2020).

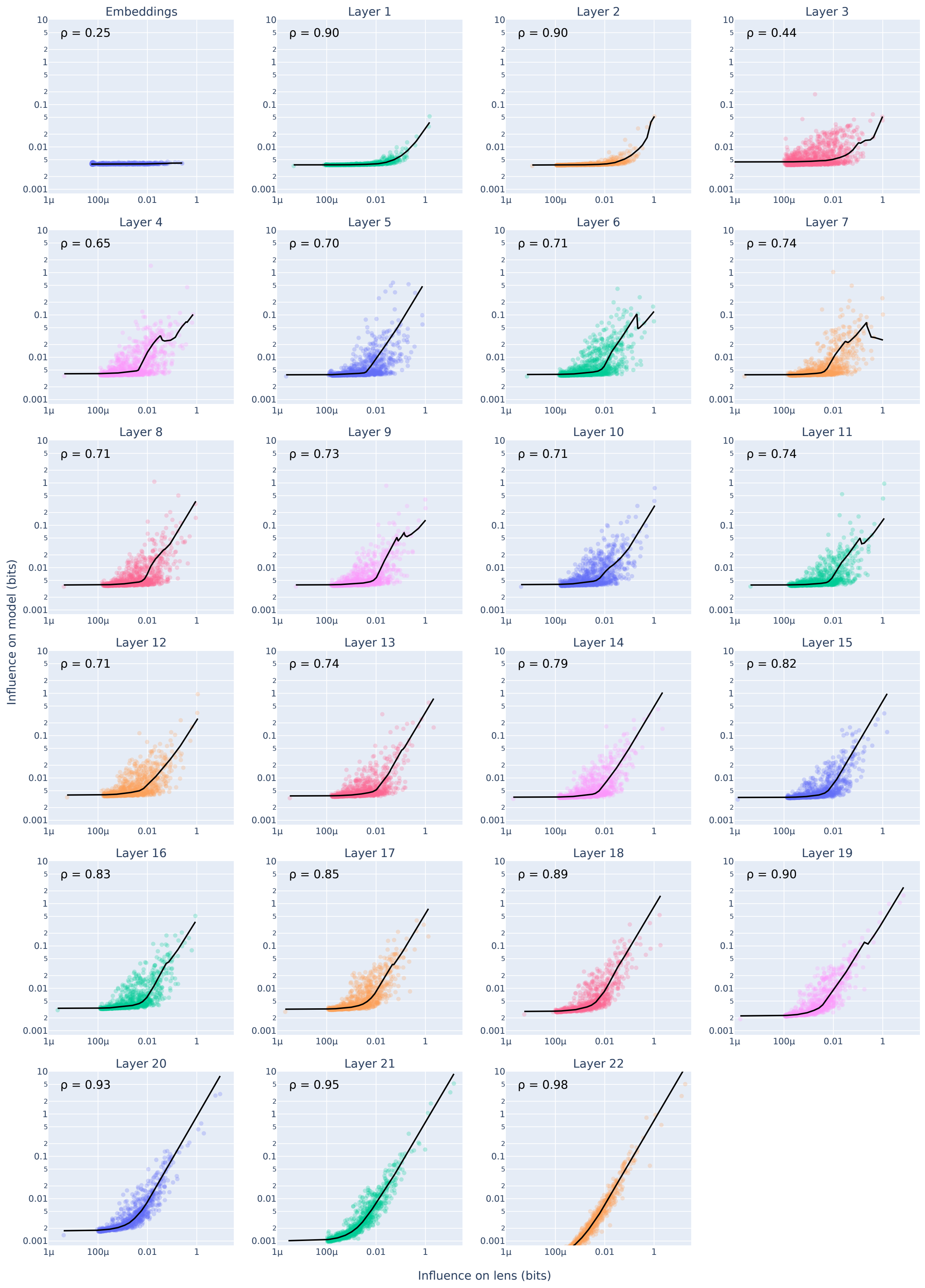

Figure 18. Causal fidelity in Pythia 410M across layers. In the top left corner of each plot is the Spearman rank correlation between the causal influence of each feature on the tuned lens and its influence on the model.

	OV SVD (L)	OV SVD (R)	QK SVD (L)	QK SVD (R)	W_{in} SVD	W_{in} columns	W_{out} SVD	W_{out} rows
Logit lens (real)	0.0745	0.1164	0.0745	0.1164	0.0745	0.0864	0.0728	0.0796
Logit lens (random)	0.0698	0.0695	0.0697	0.0688	0.0689	0.0689	0.0691	0.0688
Tuned lens (real)	0.1240	0.1667	0.1240	0.1667	0.1193	0.1564	0.1196	0.1630
Tuned lens (random)	0.1164	0.1157	0.1177	0.1165	0.1163	0.1262	0.1163	0.1532

Table 3. The mean interpretability scores (as measured in Appendix E.3) for Pythia 125M, with several different interpretability techniques (Millidge and Black, 2022; Geva et al., 2020), comparing both the tuned lens and logit lens to randomly generated matrices. Where applicable, the notation (L) and (R) indicates that the results are for the left and right singular vectors, respectively.

when viewed from the right angle, the modifications they make are often interpretable. These "interpretable vectors" occur significantly more often than would be expected by random chance (Geva et al., 2022), and can be used to edit the model's behaviour, like reducing probability of specific tokens (Millidge and Black, 2022), decreasing toxicity (Geva et al., 2022), or editing factual knowledge (Dai et al., 2022).

Although these interpretability methods differ in precise details, we can give a generic model for the interpretation of a model parameter W using the unembedding matrix:[8]

$$T(W) = \text{top-}k(f_i(W)W_U)$$

where f_i is some function from our parameter to a vector with the same dimensions as the hidden state, and W_U is the unembedding matrix. In words: we extract a hidden state vector from our parameter W according to some procedure, project into token space, and take the top k matching tokens. The resulting $T(W)$ will be a list of tokens of size k, which functions as a human interpretable view of the vector $f_i(W)$, by giving the k tokens most associated with that vector. As an example, the model parameter W could be the MLP output matrix W_{out} for a particular layer, and f_i the function selecting the ith column of the matrix.

With the tuned lens, we modify the above to become:

$$T(W) = \text{top-}k(L_\ell(f_i(W))W_U)$$

where L_ℓ is the tuned lens for layer number ℓ, projecting from the hidden state at layer ℓ to the final hidden state.

To test this hypothesis, we developed a novel automated metric for evaluating the performance of these parameter interpretability methods, based on the pretrained BPEmb encoder (Heinzerling and Strube, 2018), enabling much faster and more objective experimentation than possible with human evaluation. We describe this method in Appendix E.3.

The results for Pythia 125M can be seen in Table 3. The parameters under both the tuned and logit lens consistently appear more interpretable than random. And the tuned lens appears to show benefit: the difference between random/real average scores is consistently higher with the tuned lens than the logit lens. However, preliminary experiments found much less improvement with larger models, where both the tuned and logit lens appeared to perform poorly.

E.2. Interpretability score distributions

The distribution of interpretability scores for a single parameter (the OV circuit) can be seen in Figure 19. (Interpretability distributions of other parameters appear to share these features as well.) The plot displays a major complication for static analysis methods: most singular vectors are not more interpretable than random, except for a minority lying in a long right tail. It also shows the necessity of comparison to randomly generated matrices: naively comparing the interpretability scores for the tuned lens against those for the baseline would imply a far greater improvement than actual, because the interpretability scores under the tuned lens are higher even for randomly generated matrices.

E.3. Automated interpretability analysis method

While humans can easily tell if a list of words represents some coherent category, this can be subjective and time-consuming to evaluate manually. Some previous work has attempted to automate this process, using GPT-3 in Millidge and Black

[8]In most transformer language models, the function here should technically be $T(W) = \text{top-}k(\text{softmax}(\text{LN}(f_i(W)W_U)))$, including both softmax and LayerNorm. However, the softmax is not necessary because it preserves rank order, and the LayerNorm can be omitted for similar reasons (and assuming that either $f_i(W)$ is zero-mean or that W_U has been left-centered).

[8]Random shuffling applied to each matrix (head-wise for attention matrices), to approximate the element-wise marginal distribution.

[8]Similar to above footnote, random shuffling applied to each W_{OV} for each head.

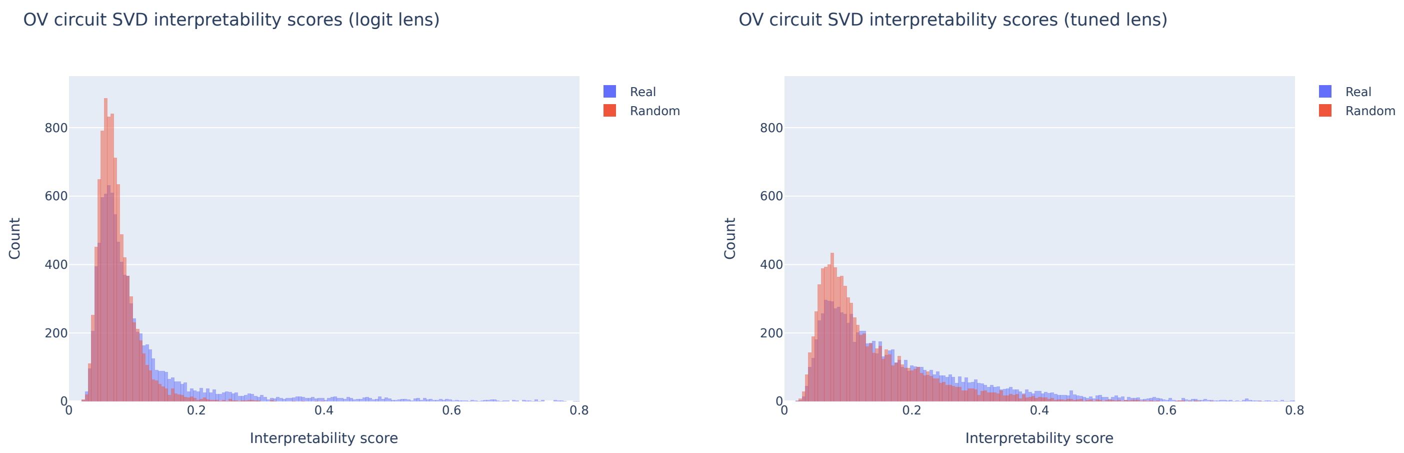

Figure 19. The interpretability scores for the right singular vectors of the OV matrices W_{OV} (following Millidge and Black (2022)) in Pythia 125M, compared with a randomly generated matrix, for both the logit lens (left), and the tuned lens (right).

(2022) or the Perspective API in Geva et al. (2022), but these can still be too slow for quick experimentation.

We created our own method, motivated by the observation that much of the human interpretable structure of these tokens appears to be related to their mutual similarity. Then, the mutual similarity of words can easily be measured by their cosine similarity under a word embedding (Sitikhu et al., 2019). In particular, we use the pretrained BPEmb encoder (Heinzerling and Strube, 2018) [9]. We encode each token, measure the cosine similarity pairwise between all tokens, and take the average value:

$$I(T(W)) = \frac{\sum_{i,j}^{k,k} E(T_i(W)) \cdot E(T_j(W))}{k^2}$$

where the subscript on $T_i(W)$ denotes the i-th token of $T(W)$, and E denotes the normalized BPEmb encoding function. This creates an interpretability metric I that measures something like the “monosemanticity" of a set of tokens.

Because BPEmb is a pretrained model, unlike most CBOW or skip-gram models, there is no ambiguity in the training data used to produce word similarities and results are easy to replicate. We find the interpretability scores given by the model to correspond with human judgment in most cases: it rarely identifies human-uninterpretable lists of tokens as interpretable, although it occasionally struggles to recognize some cases of interpretable structure, like syntactically related tokens, or cases where a list as a whole is more interpretable than a pairwise measure can capture.

E.4. Improved Model Editing

Geva et al. (2022) edit GPT-2-medium to be less toxic by changing the coefficients which feed into the columns of the MLP output matrix, under the key-value framework developed in Geva et al. (2020). They define both the value vector v_i^ℓ (i.e. the columns of the MLP's W_{out}) and a coefficient m_i^ℓ (i.e. the output of W_{in} before the activation function) for each column i and each layer ℓ. This is used to reduce the toxicity of GPT-2-medium by editing the coefficients of specific value vectors. They find:

$$T(W_{\text{out}^l}) = \text{top-30}(f_i(W_{\text{out}}^\ell)W_U)$$

where:

$$f_i(W_{\text{out}}^\ell) = v_i^\ell$$

They then concatenate these top-30 tokens for every layer and column, sending it to Perspective API, a Google API service which can return a toxicity score. They then sampled from the vectors that scored < 0.1 toxicity (where a score > 0.5 is classified as toxic), set each value vector's respective coefficient to 3 (because 3 was a higher than average activation), and ran the newly edited model through a subset of REALTOXICPROMPTS (Gehman et al., 2020) to measure the change in toxicity compared to the original model.

[9]We also tried the fastText encoder (Bojanowski et al., 2016), but found BPEmb to give results better matching human judgment.

	Toxicity	Severe Toxicity	Sexually explicit	Threat	Profanity	Identity attack	Perplexity
Original/ Unedited	0.50	0.088	0.159	0.058	0.40	0.043	**27.66**
Logit Lens Top-5	0.47	0.077	0.148	0.057	0.38	0.043	27.68
Logit Lens Top-10	0.45	0.073	0.144	0.056	0.36	0.040	27.70
Logit Lens Top-20	0.43	0.072	0.141	0.057	0.33	0.041	27.73
Tuned Lens Top-5	0.45	0.079	0.143	0.056	0.36	0.041	**27.66**
Tuned Lens Top-10	0.42	0.063	0.143	0.057	0.33	**0.034**	27.67
Tuned Lens Top-20	**0.39**	**0.061**	**0.138**	**0.052**	**0.31**	0.035	27.73

Table 4. Percentage of prompt generations labeled as toxic for OPT-125m. Best results in **bold**. Across all respective settings, the tuned lens outperformed the logit lens with no significant increases in perplexity. Although the value vectors were only graded & ranked by toxicity, there were still decreases in other metrics (severe toxicity, sexually explicit, etc) that were not directly selected against. However, these results did not generalize to other models tested.

Changing the coefficients of value vectors can generate degenerate text such as always outputting " the" which is scored as non-toxic. To measure this effect, they calculate the perplexity of the edited models, where a higher perplexity may imply these degenerate solutions.

We perform a similar experiment, but found the vast majority of value vectors that scored < 0.1 toxicity to be non-toxic as opposed to anti-toxic (e.g. semantic vectors like " the", " and", etc are scored as non-toxic). Alternatively, we set the k most toxic value vector's coefficient to 0, as well as projecting with both the logit and tuned lens on OPT-125m, as seen in Table 4.

We additionally tested pythia-125m, pythia-350m, gpt-2-medium, and gpt-neo-125m; however, there were no significant decreases in toxicity (i.e. $> 2\%$) compared to the original, unedited models.

E.5. Implementation

The initial implementation of many of our static analysis experiments was helped by the use of the `transformer_lens` library (Nanda, 2022).