THE SHAPE OF DEPTH: CONSISTENT DECODER GEOMETRY IN SPARSE AUTOENCODERS ACROSS LLM LAYERS

Anonymous authors
Paper under double-blind review

ABSTRACT

To disentangle the superposed semantic units inside large language models (LLMs), sparse autoencoders (SAEs) have become a go-to choice. Yet, they remain layer-wise snapshots that are blind to how representations evolve throughout the full forward pass to the next token. In this work, we interrogate the geometrical structure of SAE decoders across layers of GPT-2 and find a kernel of information that undergoes a consistent rotation from layer to layer – perhaps a form of LLM working memory. We thus show that the top activation directions of different layers exhibit a clear rotational trend in cosine similarity, and canonical correlation analysis of decoder structure further reveals that residual activation dimensions map into a persistent structure. To show practical benefit, we construct a plane spanned by the leading activation directions of two layers via Gram–Schmidt orthogonalization and derive a rotation operator on this plane. We control this rotation to applied activation-concept spaces during cross-layer transfer learning, which improves SAE training, and yields fewer dead SAE features by treating rotation angle as a tunable hyperparameter. These insights suggest that transformations of a geometric object persistent across layers can be monitored and leveraged, such as to enhance SAE training and performance.

1 INTRODUCTION

Polysemanticity of neurons occurs when they activate for different reasons (Bricken et al., 2023). However, despite the representational efficiency of polysemanticity, understanding how large language models (LLMs) process information becomes more challenging. This has led to the idea of modeling superposition, where sparsely active semantic features live in and get revealed in a higher dimensional space (Elhage et al., 2022). To disentangle the activations of these networks, sparse autoencoders (SAEs) have been used for their high-dimensional, sparse, overcomplete basis of features, which are more easily interpretable and monosemantic (Bricken et al., 2023; Huben et al., 2024; Templeton et al., 2024). However, SAEs have failed to isolate robust circuits throughout models, and do not give a full picture of how information trickles through layers.

Today's SAE approaches are just a layer-wise snapshot as they do not consider information carried in all layers at once. This lack of purview has lead to research into other methods such as transcoders or cross layer transcoders, which work in a similar fashion, but connect across layers (Ameisen et al., 2025; Dunefsky et al., 2024). However, we look further for connections between SAEs-detected geometries across layers to better understand language transformations at the systems level, while still being minimally invasive. We start from the voiced, but rarely tested assumption that each transformer block only adds or slightly edits the representational content in the residual stream, and leaves much of the information carried in it untouched. In fact, this key property of LLM architectures has not been fully quantified in general or exploited in LLM explainable AI (XAI) research in particular.

LLMs are known to refine outcome predictions across layers (Kongmanee, 2025). Thus it is useful to test if there are consistent changes between them, so researchers can better understand and attempt to steer token vector behaviour towards more desirable outputs. In addition, some GPT-type models, like GPT-2, have the unembedding matrix that is the transpose of the embedding matrix (Radford

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

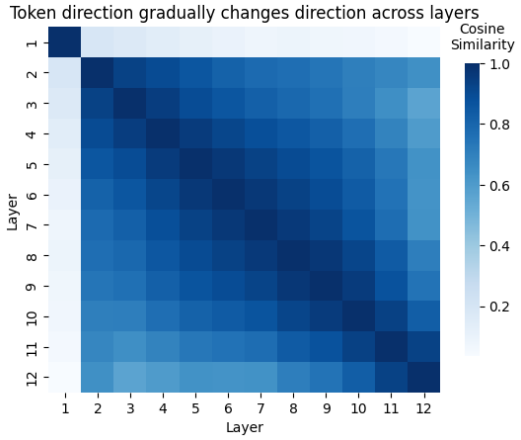


Figure 1: **LLMs continously refine output predictions across layers.** We present empirical evidence that suggests alignment of dominant activation directions across the residual stream activations of each LLM layer. Cosine similarity of 2nd token vector of a sample text is taken across layers. The same token vector is used for each layer, with the cosine similarity taken with it at each layer. The token vector is taken from the residual stream before attention at each layer, thus the first layer does not have token context taken into account, which may cause it to stand out from the rest.

et al., 2018). This also motivated us to look for consistent rotations occurring between layer representations, since the token vectors initially point to current positions in the sequences, but at the output the tokens point to next positions. Indeed, we see a consistent change (and thus rotation) in cosine similarity for a token across layers that we found to be typical across processed tokens (Figure 1). We ran principal component analysis (PCA) of the SAE decoders of different layers of the LLM – each layer decoder mapping internal concept space to observable residual stream activation space. We examined the absolute value of cosine similarities between the directions of the top 5 PC directions of layers with a middle layer, and tested for a pattern in the changes. We saw there is a rotation that occurs between layers that showed a clear trend. We then applied projection of PC1s onto a plane made of two of the PC1s, and found that rotating the decoder on this plane could improve SAE training results. Our aim was to test for the existence of such a coherent across-layer signal. Practical advantages in model training and model performances also became apparent, as we harness our across-layer rotation operator as a new hyper-parameter.

2 Related Work

Balcells et al. (2024) compare SAE feature activations across layers and show that some features may be consistent across layers, while others may disappear, be new, or cluster. Also, some features may function as AND/OR gates of preceding features. Thus, while their research provides evidence of significant inconsistency of features, some consistency is observable, which may be translated across SAEs across layers. They look at feature labels to look for patterns, while we look for general geometric relations between layer activations.

Lan et al. (2025) show that there is a significant amount of feature universality between SAEs, through representational similarity measures. They use Singular Value Canonical Correlation Analysis (SVCCA) to measure feature space alignment between different SAE instances. However, they find that the matching is not exact, as some feature subspaces differ. This method also only covers pairs of layers, and thus is not a holistic approach. They use SVCCA since it is rotationally invariant, which gives evidence that this is an aspect that could be studied in and of itself.

Li et al. (2025) break down SAE geometry into three scales: the ”atom” crystal structure scale, the ”brain” modular structure scale, and the ”galaxy” point cloud structure scale. At the atom scale, they find traits such as shapes similar to parallelograms that connect phenomena such as capitals to their countries. At the brain scale, they find traits such as clustering based on cosine similarity between SAE decoder vectors, and then compare this to co-occurring features. An example is that features

that are mostly activated on math and code tend to agglomerate in spatial clusters. At the galaxy scale, they make obserations such as the point cloud being anisotropic. This can be seen from a 3D view of the top three PCA components, which show a concentration of variance. While they look at traits per layer and compare, we place special emphasis on how information flows across processing layers.

Ghilardi et al. (2024) study transfer learning of SAEs, where weights from one SAE are used as initialization weights for SAEs of adjacent layers, going both forwards and backwards. These investigators find that training SAEs with such transfer learning is generally more efficient (e.g. faster convergence in less gradient descent steps) than random weight initializations. Thus, they may give evidence for consistent alignment of dominant activation directions between layers. We apply transfer learning across layers, using rotation of concept-activation mapping functions based on more than just one adjacent layer, for improvements.

Paulo & Belrose (2026) show that different random seeds return different SAE features . These authors point out encoder rows and corresponding decoder columns of different SAEs may be permuted. Hence, the features must be matched when comparing SAE instances of different parameter initialization seeds. They find that around 30% of features may be shared across seeds, although this depends on LLM and SAE size. This shows that SAEs may not find universal sets of features. These investigators base their results on the cosine similarity of encoder and decoder directions. However, cosine similarity does not necessarily indicate semantic equivalence, and their results may actually be affected by changes in directions of related semantic features. We thus look at the rotation of the activation-concept structure, because rotation may give evidence for semantic feature consistency that may be missed otherwise.

Sun et al. (2025) study the geometry of dense SAE latent representaion directions, which these authors consider as those that activate on over 10% of input tokens . Dense latents had generally been considered non-intepretable. However, they report that some do exhibit discernable features, and some have high cosine similarity with alignment of dominant activation directions. We further investigate traits of PC1's, but in relations and usage across layers.

3 METHODS

Early research training SAEs on LLM internal activation read-outs tended to implement L1 penalty terms into their loss function to ensure sparsity (Huben et al., 2024). More recently, other methods such as *TopK* sparse encoders have gained popularity and simplify usage, keeping the top-k activating features, which avoids reducing the magnitude of all latent activations with the L1 penalty (Gao et al., 2025). However, we used L1 sparsity, as it serves as a basic SAE example. Encoder activations $z \in \mathbb{R}^n$ with n SAE internal concept dimensions can be found with $z = \mathrm{ReLU}(W_{enc}(x - b_{pre}) + b_{enc})$, where residual stream activation $x \in \mathbb{R}^d$, with model size d, serves as the input for the SAE model (n is commonly chosen to be a multiple of d), $W_{enc} \in \mathbb{R}^{n \times d}$ is the encoder matrix, and $b_{pre} \in \mathbb{R}^d$ and $b_{enc} \in \mathbb{R}^n$ are bias terms. The decoder then estimates a reconstruction of the original inputs (i.e. residual stream activation vector for a token at hand) with $\hat{x} = W_{dec}z + b_{pre}$, where $W_{dec} \in \mathbb{R}^{d \times n}$ is the decoder matrix. The sparsity is created with the training loss $\mathcal{L} = ||x - \hat{x}||_2^2 + \lambda||z||_1$, where λ is a hyperparameter that controls the strength of imposed parsimony (Gao et al., 2025). The encoder rows can be seen to activate the decoder column directions in an overcomplete basis. The linear combination of sparse feature vectors in the decoder reconstructs the input.

To systematically examine SAE decoder rotation between layers, to search for consistent movement of information, we took the top 5 principal components PC $\in \mathbb{R}^d$ of the transpose of the decoder weights W_{dec}^T (thus PCA with SAE dimensions as n rows and LLM dimensions d as columns) in each of 12 total layers. Checking the cosine similarity of PCs between layers gave evidence of consistent rotations. However, in high dimensions, rotations are not unique. Yet, rotations on a plane are unique and allow for control of information. To check for consistent rotations on a plane, we made a plane spanned by two PCs, from different layers. We used the top PC, PC1, from each layer. To create a plane S to check for consistent unique rotations of PC1s, we used the Gram–Schmidt process with PC1s from two layers (Cheney & Kincaid, 2009).

We can consider a PC1 from one layer as v_1, and a PC1 from another layer as v_2. We can then subtract the projection of v_2 on v_1 and normalize to obtain an orthonormal basis for S, e_1 and e_2, as,

$$u_1 = v_1, \quad e_1 = \frac{u_1}{||u_1||} \tag{1}$$

$$u_2 = v_2 - \text{proj}_{u_1}(v_2) = v_2 - \frac{< v_2, u_1 >}{< u_1, u_1 >} u_1, \quad e_2 = \frac{u_2}{||u_2||} \tag{2}$$

We then obtain the projections of a PC1 onto S as,

$$\text{proj}_{e_1}(\text{PC1}) =< \text{PC1}, e_1 > e_1, \quad \text{proj}_{e_2}(\text{PC1}) =< \text{PC1}, e_2 > e_2 \tag{3}$$

This could be used to compare cosine similarities and angles between PCs on planes spanned by two PCs, to check for consistency of unique rotations. The PC1 could then be broken up into its parallel and orthogonal components with S, as,

$$\text{PC1} = \text{PC1}_{\parallel S} + \text{PC1}_{\perp S} = \text{PC1}_{\parallel e_1} + \text{PC1}_{\parallel e_2} + \text{PC1}_{\perp S} \tag{4}$$

The PC1 projection on S is then,

$$\text{proj}_S(\text{PC1}) =< \text{PC1}, e_1 > e_1 + < \text{PC1}, e_2 > e_2 = e_1 e_1^{\text{T}} \text{PC1} + e_2 e_2^{\text{T}} \text{PC1} \tag{5}$$

To rotate on the plane S, we could obtain the rotated parallel components $\text{PC1}'_{\parallel \text{S}}$ with the common rotation matrix in 2D,

$$\begin{bmatrix} < \text{PC1}', e_1 > \\ < \text{PC1}', e_2 > \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} < \text{PC1}, e_1 > \\ < \text{PC1}, e_2 > \end{bmatrix} \tag{6}$$

To obtain the full rotated vector, we could solve for the orthogonal component in equation equation 4, noting that, since the orthogonal components to the plane are not rotated, $\text{PC1}_{\perp S} = \text{PC1'}_{\perp S}$. We then combine parallel and orthogonal components to the plane and get,

$$\begin{aligned} \text{PC1}' =& < \text{PC1}, e_1 > \cos(\theta) e_1 - < \text{PC1}, e_2 > \sin(\theta) e_1 + < \text{PC1}, e_1 > \sin(\theta) e_2 \\ &+ < \text{PC1}, e_2 > \cos(\theta) e_2 + \text{PC1} - < \text{PC1}, e_1 > e_1 - < \text{PC1}, e_2 > e_2 \end{aligned} \tag{7}$$

This then gives a rotation matrix $\text{R}_S \in \mathbb{R}^{d \times d}$ that could be applied to any vector with dimensions d of the LLM activation space, e.g. the columns of W_{dec},

$$\text{R}_S = \text{I} + (\cos(\theta) - 1)(e_1 e_1^{\text{T}} + e_2 e_2^{\text{T}}) + \sin(\theta)(e_2 e_1^{\text{T}} - e_1 e_2^{\text{T}}) \tag{8}$$

which could then provide a handle as a new experimental factor or introduce added flexibility in the form of a new hyperparameter.

4 Experimental details

To explore the alignability of dominant activation directions across LLM layers, we look at cosine similarity between PC's. SAE features are considered PCA observations in this scenario, with LLM dimensions as PCA features. Visible trends in dominant PC directions across layer activations are taken as evidence for consistent rotations. For transfer learning, while we do consider reconstruction error rates of the SAEs, we do not report them since we do not find significant differences. We look at dead features and L0 sparsity to determine differences between runs. Dead features count the number of features that have not activated over a dead feature window, 1k training steps (batches optimized), before the current step. L0 sparsity is an average number of feature activations per token in the current step.

For a text dataset, for training SAEs, we use the Open Web Text dataset, which has over 8 million rows of text with a minimum of 128 tokens each (Gokaslan et al., 2019). This dataset is used since it was used to train pretrained SAEs on the same LLM (GPT-2-small) (Bloom et al., 2024). It was also used to train GPT-2 (Gokaslan et al., 2019). A sample from the dataset with truncated text is, "text": "The opinions expressed by columnists are their own and do not represent the views of..." There is no train/test split, evaluation metrics are computed on the same data as used for training.

To maximize the comparability of our research, we provide an implementation of out SAE training based on SAELens (Bloom et al., 2024). The configurations were mostly taken from tutorials or pretrained SAEs. The same configuration was used for each layer, aside from random,

transfer learning, or rotated transfer learning modifications. For all SAE training configurations, we use total training steps=30k, batch size=4096, and thus total training tokens=122,880,000. For Model + Training Distribution in LanguageModelSAERunnerConfig (all config values in this paragraph are included in this runner config), for rotated decoder layer transfer learning, we used model name=gpt2-small, hook name=blocks.3.hook_resid_pre (the residual stream at layer input), dataset path=Skylion007/openwebtext, is dataset tokenized=False, and streaming=True. For SAE parameters, sae=StandardTrainingSAEConfig, we used d in=768, d sae=32 * 768 (the max number of SAE features), apply b dec to input=True, normalize activations='none', reshape activations='none', and l1 coefficient=1. Training parameters were lr=4e-4, prepend bos=True, model from pretrained kwargs center writing weights = True, adam beta1=0.9, adam beta2=0.999, lr scheduler name="constant", lr warm up steps=0, lr decay steps=total training steps // 5 , and train batch size tokens=batch size. Activation store parameters were context size=128, n batches in buffer=64, training tokens=total training tokens, store batch size prompts=16. Resampling protocols were feature sampling window=1000, dead feature window=1000, dead feature threshold=1e-4. The LoggingConfig was, log to wandb=True, wandb log frequency=30, eval every n wandb logs=20. The wandb project and run name depended on the run. Other config settings were device=cuda, seed=42, checkpoints=0, checkpoint path="checkpoints", and dtype="float32".

Standard scientific python packages used include google-genai (Google, 2026), Pytorch (Ansel et al., 2024), Numpy (Harris et al., 2020), Pandas (The pandas development team, 2024), Seaborn (Waskom, 2021), and Sci-kit Learn (Pedregosa et al., 2011). SAE relevant packages include SAELens (Bloom et al., 2024), TransformerLens (Nanda & Bloom, 2022), Wandb (Biewald, 2020), and Hugging Face Datasets (Lhoest et al., 2021).

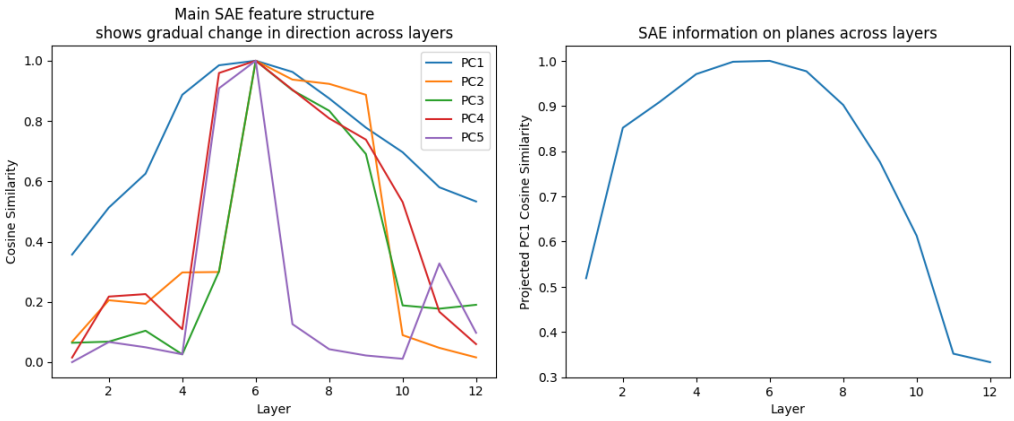
5 RESULTS

Rotation of SAE decoder weight principal directions of representation across layers show consistency of information flow. The absolute values of cosine similarities between the top 5 decoder PCs of layer 6 and the top 5 decoder PCs of each of the other layers shows evidence of consistent rotations (Figure 2a). This is especially the case for PC1. Higher number PCs may also show some evidence of consistent rotations. However, the pattern is less clear with PC5. We also found in the explained variance ratios that for the PC1s across layers, they all fall in the range of $\sim 2 - 3\%$ explained variance, which may imply there is somewhat of a main structure shared across layers.

However, in high dimensions, rotations can occur along many axes. To be able to utilize the rotations in a unique manner, they would need to occur along a plane. The projection of the PC1s on a plane made from PC1s of layers 1 and 3 and the cosine similarities with PC1 of layer 6 projected on that plane show a rotation pattern (Figure 2b). Though the rotation pattern is not the same, and the angles between the PCs not projected on the plane may be different, a clear pattern of consistent rotation does occur, somewhat similar to the non-plane-projected case. The rotation on the plane may only account for a portion of the angle difference between non-projected PC1s.

We also computed the top 38 PC scores $\in \mathbb{R}^n$ of the decoder (d layer activation rows and n SAE feature columns) for each layer and ran canonical correlation analysis (CCA) between these aggregated matrices. The value of 38 PC scores was chosen since it gives about a $20\times$ row to column ratio for the CCA. We then found high correlation between the top covariates between each layer, showing consistency of overall residual dimension structure in the SAEs between layers (Figure 3). This aligns with the residual stream adding values at each layer, thereby maintaining some consistency. The CCA does not provide direct information on SAE features, but it does provide justification of using the d dimensions of the SAEs as feature columns for the PCs checked for rotation, since the dimensions are used in a consistent manner across layers.

We then found that the rotation angle can be treated as a hyperparameter that may improve training results during transfer learning, where SAE weights from one layer are used in the next layer. As seen in Figure 4b, a decoder rotated -164 degrees on the plane shows fewer dead features than random weights by the end of training. Also, transfer learning, in general, shows an improved (lower/sparser) L0 (number of features that activate) over random weights (Figure 4a). Thus Figure 4b shows how rotations affect dead feature counts, with a trend that follows rotation direction. Reconstruction loss was similar across all training runs (SAE model estimations based on the subject



(a) (b)

Figure 2: **Major activation-concept directions capture flow of representation transformations from layer to layer.** (a) Absolute values of cosine similarities of the top 5 decoder weight PCs of layer 6 and the top 5 decoder PCs of all other layers. We estimated the major sources of variation in the activation-concept space by feeding PCA with the SAE decoder matrix: SAE features vectors as rows (n) and dimensions of those feature vectors as columns (d), of a specific GPT-2 layer. (b) Absolute values of cosine similarities with PC1 of layer 6 with PC1s projected on a plane that was constructed with PC1s of layers 1 and 3.

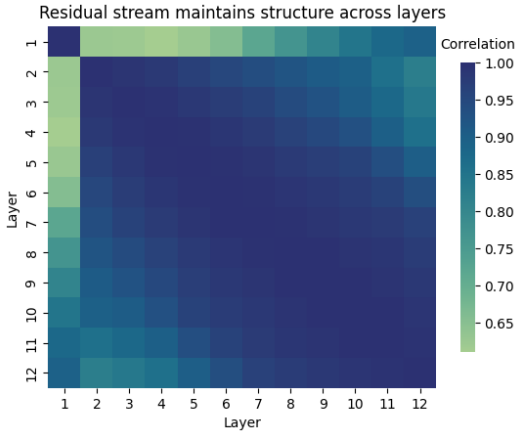
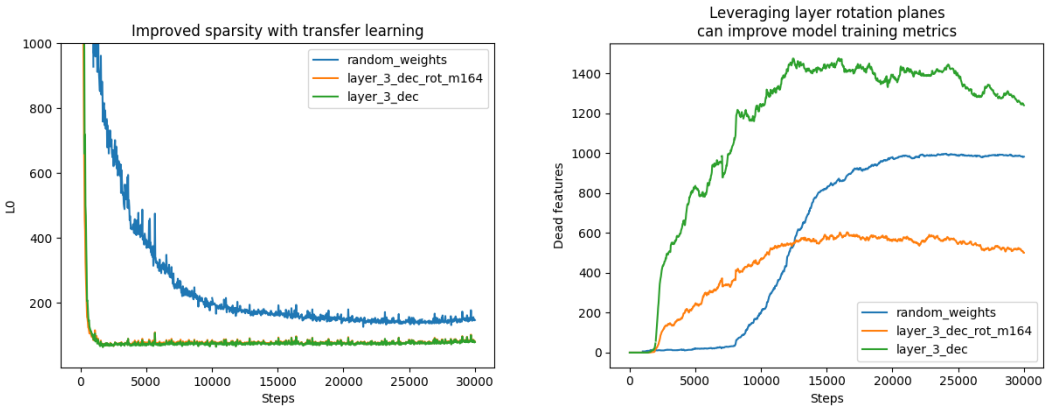


Figure 3: **LLM layer activations can be expressed in a joint structure.** In addition to dominant activation-concept directions, we also examined the alignability of a wider spectrum of layer activation subspaces. CCA was used to quantify this layer-layer pairwise correspondance by searching through projection vectors for each side (each cube indicates one such layer-layer relationship). Shown is correlation strength of top CCA covariates computed from top 38 PCA component scores of the decoder weight matrices that correspond to two different GPT-2 layers.



(a) (b)

Figure 4: **Transfer learning improves model training and performance by exploiting layer-to-layer rotations.** Transfer learning takes decoder weights from one layer to use in another. (a) L0 sparsity for random, transfer learning, and rotated transfer learning weights. A lower L0 means more sparsity, which can enable improved interpretability. Green and orange lines overlap, both showing improved sparsity. (b) Dead features for random, transfer learning, and rotated transfer learning weights on projection plane of PC1s of layers 1 and 3. Weights from layer 3 were transferred for use as initial weights for the SAE of the subsequent layer 4 (layer_3). Decoder weights and biases, the encoder bias, and the encoder weights as the transpose of the decoder, were transferred (dec). The decoder is also rotated on the plane of PC1s of layers 1 and 3, with the encoder weights set to the transpose of that (dec_rot). The negative (m) of the angle rotated on the plane in degrees (164).

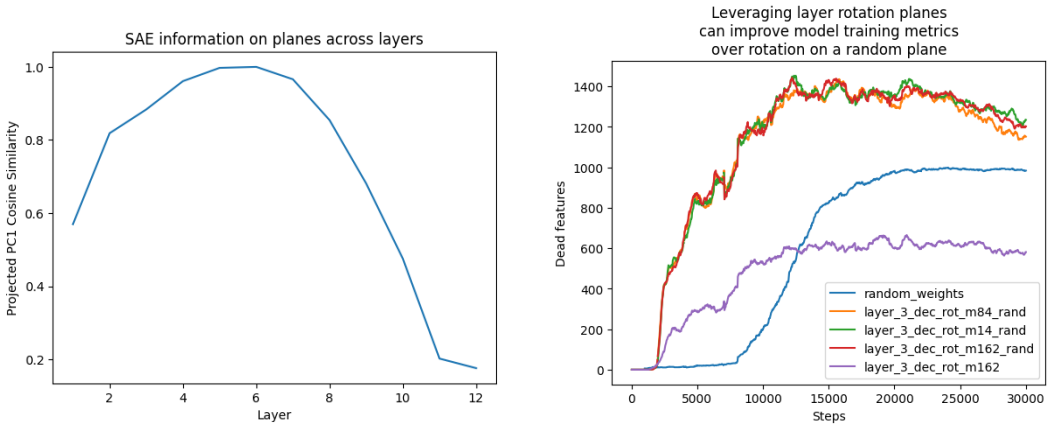
LLM), and thus is not shown. This demonstrated that rotation on a plane of PCs can readily serve as a tunable hyperparameter useful for transfer learning across layers.

Dead features may not be a significant issue with models such as TopK SAEs that use auxiliary loss (AuxK) (Gao et al., 2025). However, the creation of features from these dead latents is not done under ”normal” sparsity constraints, and directionality may be affected. Thus we may want to keep open the possibility of other methods to combat dead features that may be more interpretable, if they use the same sparsity constraint as other features for creation, such as with rotated transfer learning, as we do here. To provide evidence for the practical usefulness of the method, we check for rotation on a similar plane and a random plane. A plane from PC1s from layers 1 and 2 shows a similar rotation pattern as that of layers 1 and 3 (Figure 5a). We then also construct a random plane from random vector directions of the same number of dimensions. We find that rotation on the plane of PC1s of layers 1 and 2 show a similar decrease in dead features for a similar rotation, $-162°$, along with rotations on the random plane not showing much difference from transfer learning without rotation (Figure 5b). Sparsity differences between transferred and random weights were similar to the previous case. This shows that there is useful information on the planes of PC1s used, which is shown by the varying number of dead features found. However, the insight of information flow along planes may be more useful in itself for prospective uses.

6 Discussion

We have opened a new window onto how we can understand the evolution of the geometry of LLM layer activations, and how notions of across-layer coherence can yield a new hyperparameter with great practical benefits. An SAE decoder here serves as a mapping function from concept space to LLM internal activations (residual stream) of a particular layer. We have demonstrated that there is high level correspondences in the concept-LLM mapping function across layers – potentially a new primitive pointing to a form of working memory inside LLM representations.

This is a platform on which across-layer large scale decoding and circuit discovery may be possible. As an action-able step in this direction, we here offer a format and a tool to measure the similarity in how SAE concept projections pass on internal processing content from one layer to the next



(a) (b)

Figure 5: **Multiple planes hold information more useful than a random plane projection of layer pairs.** (a) Absolute values of cosine similarities with PC1 of layer 6 with PC1s projected on a plane made with PC1s of layers 1 and 2. This plot can be compared to rotation on PC1s of layers 1 and 3 (Figure 2b) – a similar rotation pattern is visible. Thus we find that rotation on the plane of PC1s of layers 1 and 2 show major SAE directions (b) Dead features for random, randomly rotated transfer learning, and rotated transfer learning weights on projection plane of PC1s of layers 1 and 2. More dead features early during training process is prefereable, since, in sparse autoencoder training, having more features die early can be seen as "good". It means the sparsity pressure is quickly pruning redundant or unproductive directions while there is still ample training time left to detect and resample those dead features into useful ones.

layer. Rotation of the dominant activation directions occurs in a somewhat consistent manner across layers. This enables the rotation of the decoder on a plane (which may have information trends to inspect itself) spanned by PC1s of different layers that can improve training results, i.e. fewer dead features - an important desiderata when attempting to make the nature of LLM activations amenable to human understanding.

AI USE STATEMENT

In this work, we used generative AI tools for known coding methods searches, known math relation searches, aid with searches for related work, checking text for errors, and use as a thesaurus. We have not used generative AI tools for helping develop conceptual frameworks, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, or interpret results, and the rest of the required disclosure tasks are not applicable to this work. No novel coding or math claims are made, the core contribution being the overall rotation process which we devised ourselves. We have reviewed all AI-assisted work. We reviewed papers ourselves, checked all code and math, and text was written by ourselves at the level of complete sentences or longer. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

A notebook is provided that includes the plots and can reproduce the plots with some modifications for angles used. A Wandb account is required to run fully.

REFERENCES

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan,

Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/methods.html.

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024. doi: 10.1145/3620665.3640366. URL https://docs.pytorch.org/assets/pytorch2-2.pdf.

Daniel Balcells, Benjamin Lerner, Michael Oesterle, Ediz Ucar, and Stefan Heimersheim. Evolution of sae features across layers in llms, 2024. URL https://arxiv.org/abs/2410.08869.

Lukas Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Joseph Bloom, Curt Tigges, Anthony Duong, and David Chanin. Saelens. https://github.com/decoderesearch/SAELens, 2024.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Elliott W. Cheney and David Kincaid. *Linear algebra: theory and applications*. Jones and Bartlett Publishers, Sudbury, Mass, 1. ed edition, 2009. ISBN 978-0-7637-5020-6. URL https://books.google.ca/books?id=Gg3Uj1GkHK8C&pg=PA544.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable LLM feature circuits. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=J6zHcScAo0.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/toy_model/index.html.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tcsZt9ZNKD.

Davide Ghilardi, Federico Belotti, Marco Molinari, and Jaehyuk Lim. Accelerating sparse autoencoder training via layer-wise transfer learning in large language models. In *The 7th BlackboxNLP Workshop*, 2024. URL https://openreview.net/forum?id=GI5j6OMTju.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Google. Google gen ai python sdk. https://github.com/googleapis/python-genai, 2026. Accessed: [Insert Date Here].

Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020. doi: 10.1038/s41586-020-2649-2.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.

Jaturong Kongmanee. Unraveling token prediction refinement and identifying essential layers in language models, 2025. URL https://arxiv.org/abs/2501.15054.

Michael Lan, Philip Torr, Austin Meek, Ashkan Khakzar, David Krueger, and Fazl Barez. Quantifying feature space universality across large language models via sparse autoencoders, 2025. URL https://arxiv.org/abs/2410.06981.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 175–184, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.emnlp-demo.21.

Yuxiao Li, Eric J. Michaud, David D. Baek, Joshua Engels, Xiaoqing Sun, and Max Tegmark. The Geometry of Concepts: Sparse Autoencoder Feature Structure. *Entropy*, 27(4):344, March 2025. ISSN 1099-4300. doi: 10.3390/e27040344. URL https://www.mdpi.com/1099-4300/27/4/344.

Neel Nanda and Joseph Bloom. Transformerlens. https://github.com/TransformerLensOrg/TransformerLens, 2022.

Gonçalo Paulo and Nora Belrose. Sparse autoencoders trained on the same data learn different features. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=EjInprGpk9.

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.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018. URL https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Xiaoqing Sun, Alessandro Stolfo, Joshua Engels, Ben Peng Wu, Senthooran Rajamanoharan, Mrinmaya Sachan, and Max Tegmark. Dense SAE latents are features, not bugs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=p8lKcNkJRi.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan.

Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

The pandas development team. pandas-dev/pandas: Pandas, 2024. URL https://github.com/pandas-dev/pandas.

Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6 (60):3021, 2021. doi: 10.21105/joss.03021. URL https://doi.org/10.21105/joss.03021.