# Residual Stream Analysis with Multi-Layer SAEs

Tim Lawson Lucy Farnik Conor Houghton Laurence Aitchison

University of Bristol

tim.lawson@bristol.ac.uk

## Motivation

Sparse autoencoders (SAEs) learn interpretable directions or **latents** in the representation spaces of transformer language models.

But we want to understand and control model behaviors, which span **multiple layers**. There are two options to link latents across layers:

- Match latents from SAEs trained at different layers, like Balcells et al. (2024), Balagansky et al. (2024), and Paulo et al. (2024)
- Learn latents that represent the same concept at multiple layers, like Yun et al. (2023) and Ghilardi et al. (2024)

## Multi-Layer SAEs

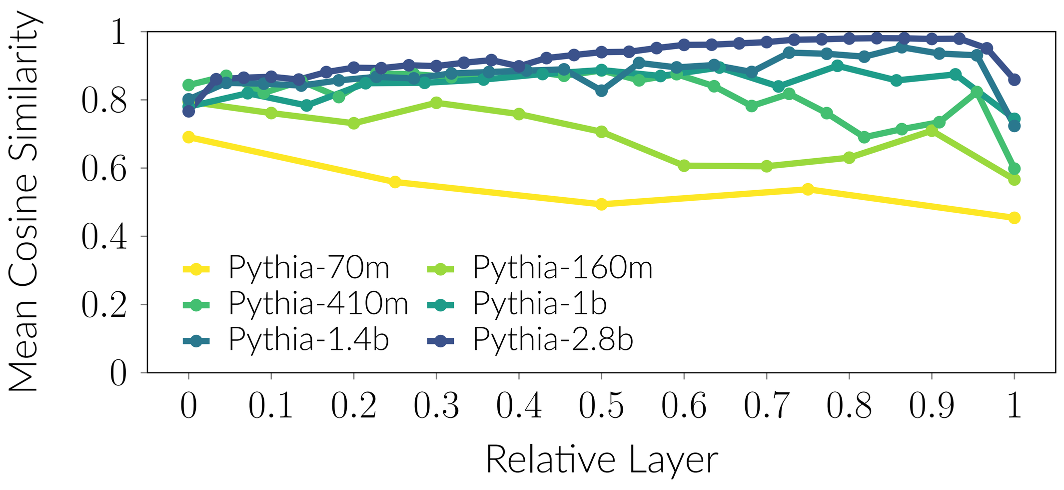
We train a single SAE on the residual stream activation vectors from **every** layer of a transformer.

## How similar are transformer layers?

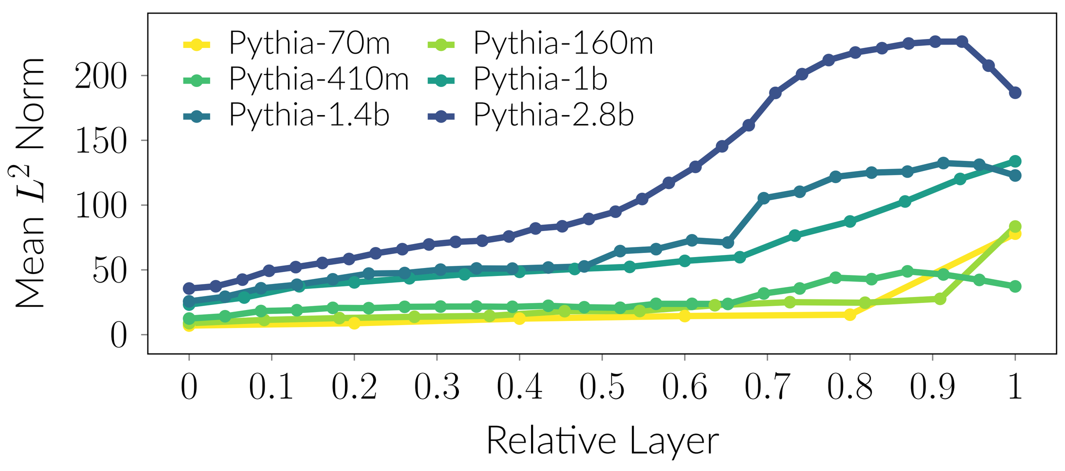
We expect the vector spaces at different layers to be similar:

- Intuitively, due to residual connections (e.g., Elhage et al. 2021)
- Empirically, from path patching (e.g., Goldowsky-Dill et al. 2023)

The **larger** the model, the **more similar** the residual stream activation vectors at adjacent layers (cf. Lad et al. 2024):




But the **magnitude** ($L^2$ norm) of residual stream activation vectors **increases with depth** (Heimersheim and Turner 2023):



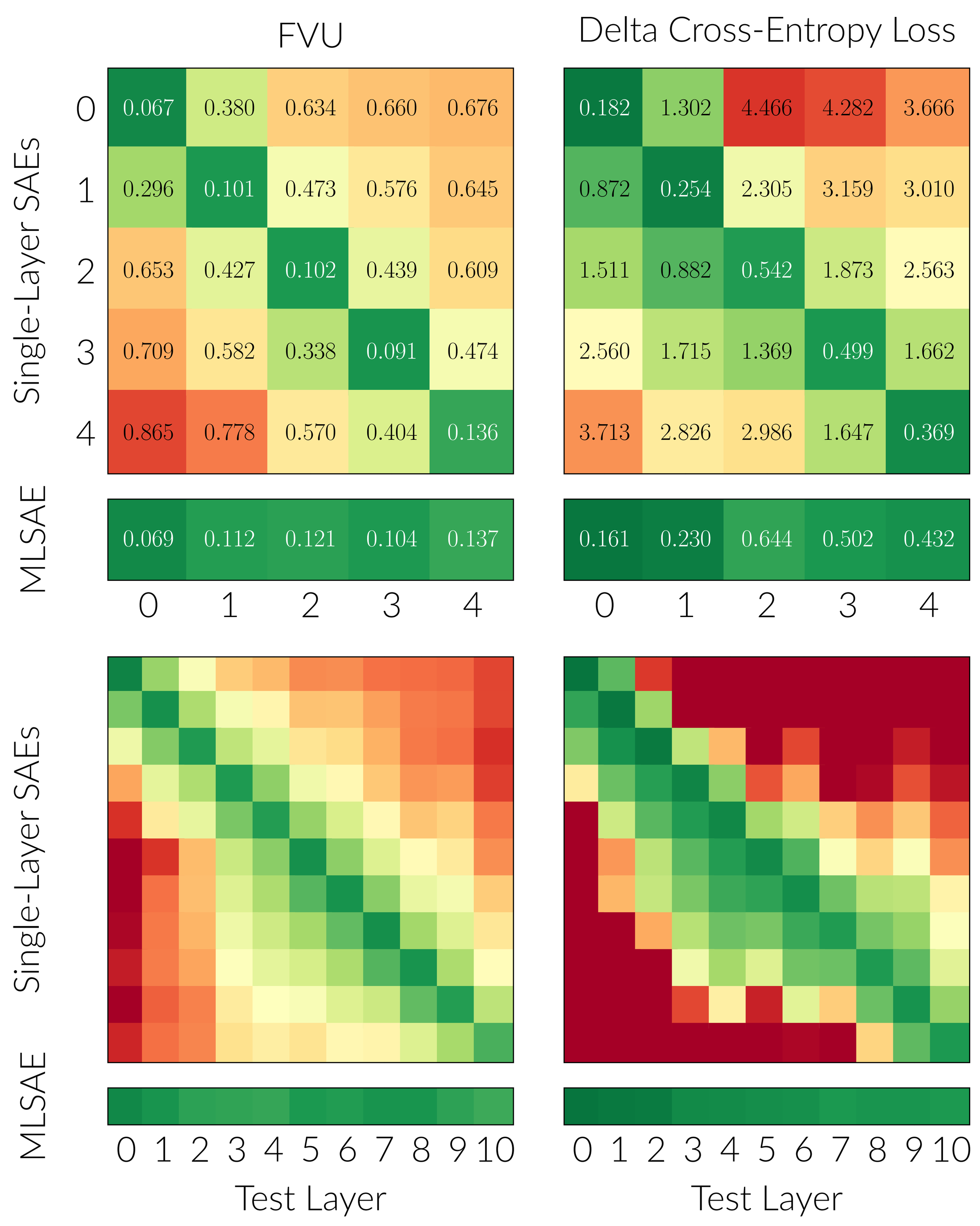

## How faithful are multi-layer SAEs?

We use the **fraction of variance unexplained** (FVU) reconstruction error, and compute the **delta cross-entropy loss** when activation vectors are replaced by their reconstruction (Gao et al. 2024).

| Model | Mean FVU | Mean Delta CE Loss |
|---|---|---|
| Pythia-70m | 0.097 | 0.565 |
| Pythia-160m | 0.106 | 0.432 |
| Pythia-410m | 0.081 | 0.414 |
| Pythia-1b | 0.095 | 0.404 |

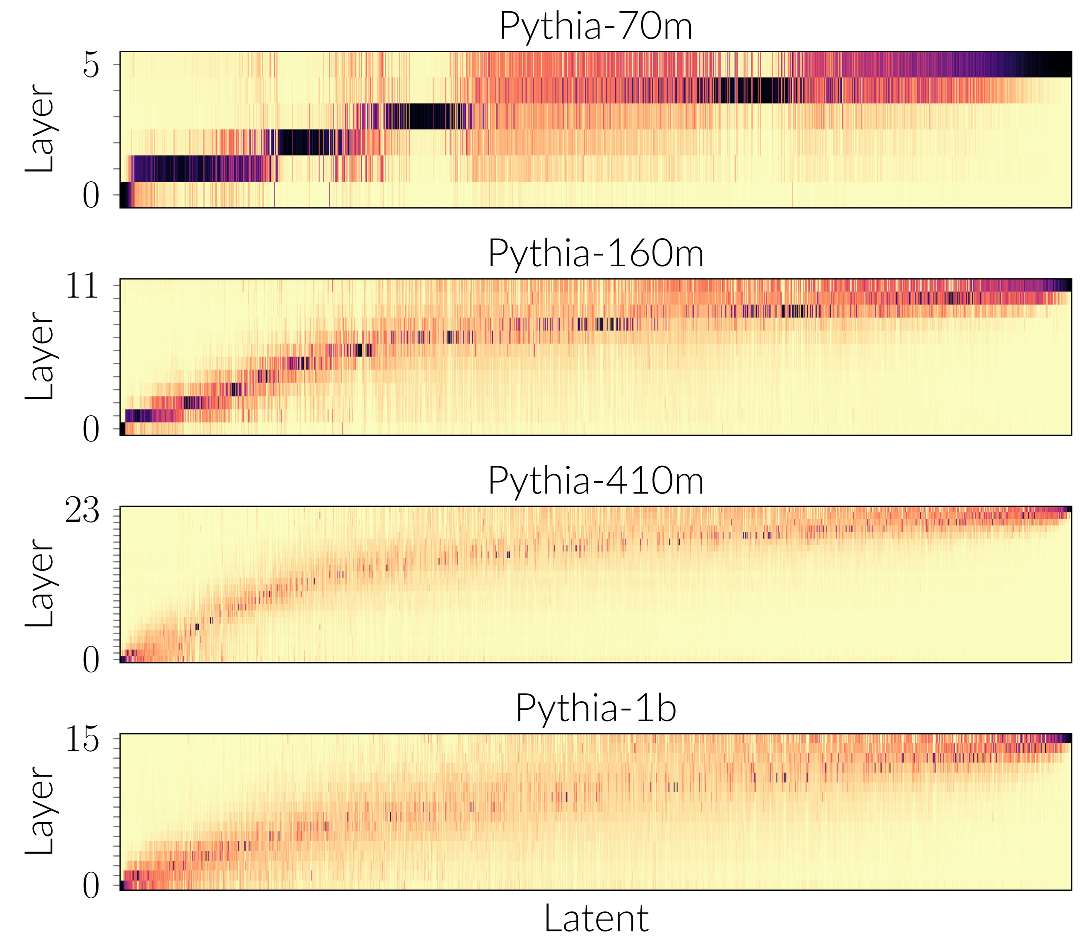
With our setup on Pythia-70m and 160m, MLSAEs perform:

- Similarly to single-layer SAEs on their own layer (diagonal)
- Better than single-layer SAEs on the other layers (off-diagonal)

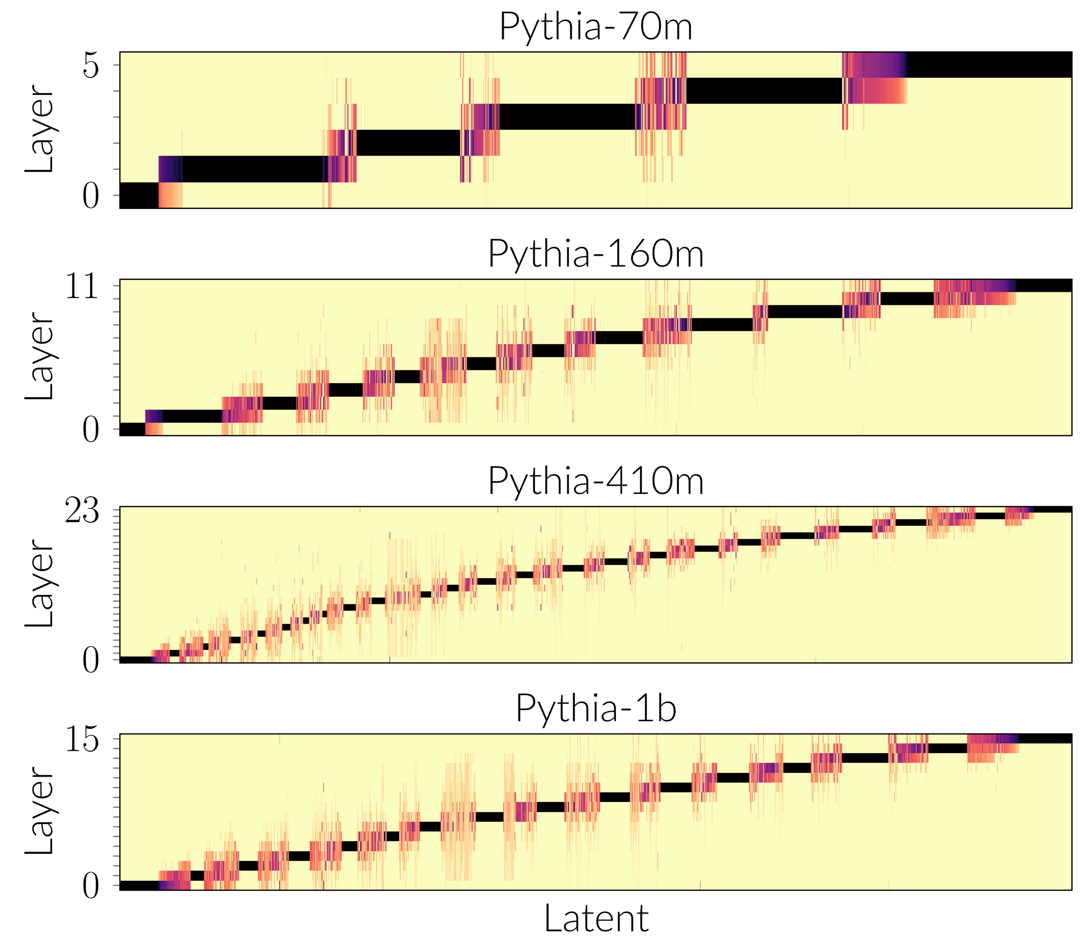



## Are latents shared between layers?

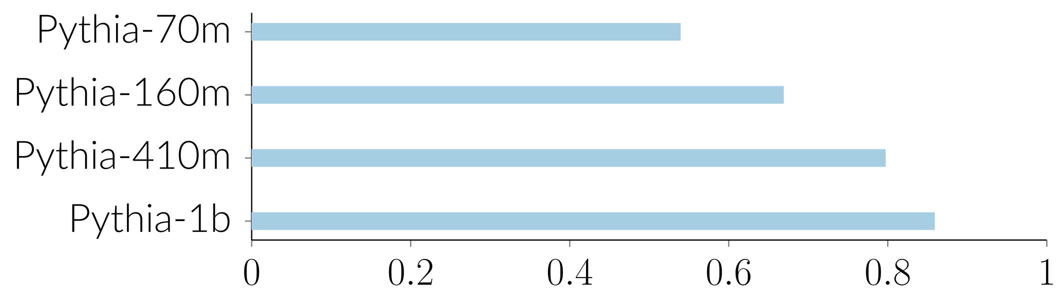
**Over 10 million tokens**, we find most latents are activated by inputs from multiple transformer layers:



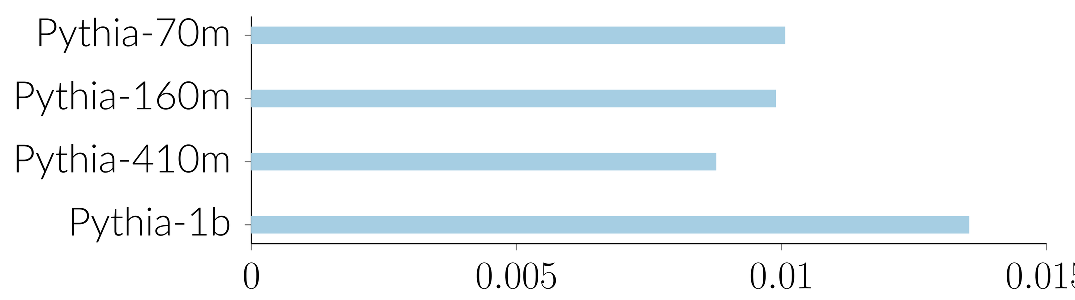

**Given an example prompt**, we find more latents are activated by inputs from a single transformer layer:




We can see the difference by the distribution of latent activations over layers and the **variance of the layer index** (see right):




Variance for one latent, aggregating over tokens, as a proportion of the total variance over all latents




Variance for one token and latent as a proportion of the total variance for that latent

The variance of the layer index is more than an order of magnitude **smaller** for a single token than when aggregating over tokens.

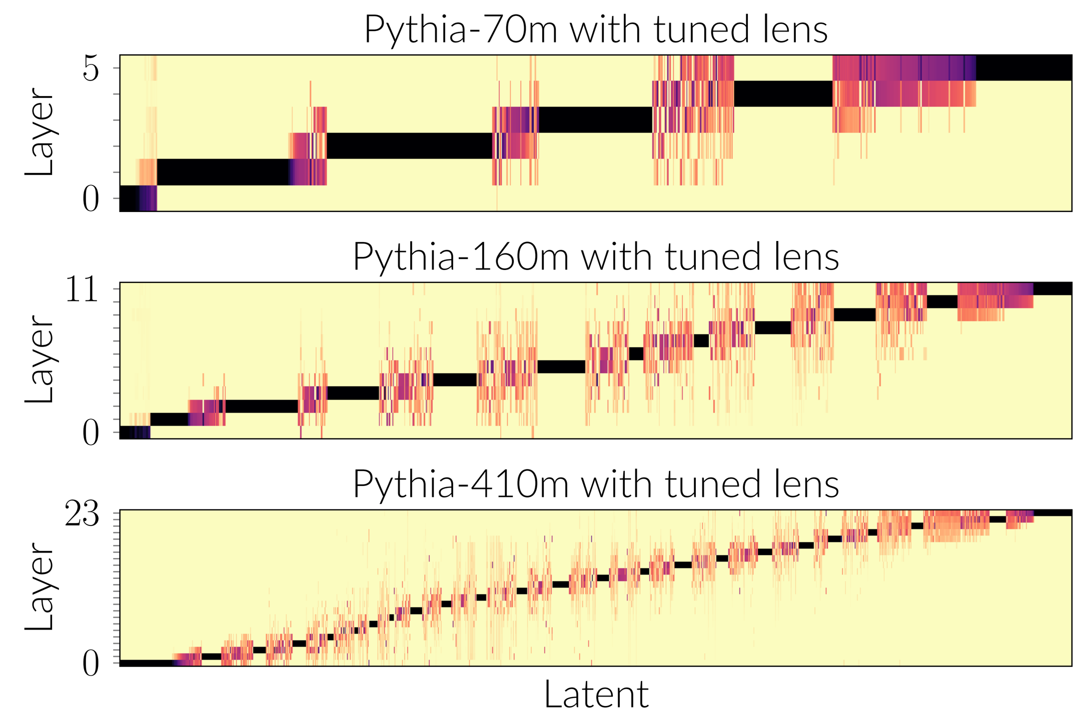
## Can we reduce representation drift?

The 'logit lens' method decodes hidden states into token predictions (nostalgebraist 2020), but assumes no representation drift.

The **tuned lens** method transforms the activations at each layer into a more similar basis to the output layer (Belrose et al. 2023).

We applied these transformations to the input activations before passing them to multi-layer SAEs to reduce representation drift.

Given an example prompt, this **slightly increased** the proportion of latents activated by inputs from multiple transformer layers:




## Implications

A single, multi-layer SAE trained on the residual stream activations from every layer **performs well** compared to single-layer SAEs.

But relatively **few latents** are activated by inputs from **multiple** transformer layers at a given token position.

**Representation drift** is a significant obstacle:

- Information from earlier layers may be obscured by increasingly large activation vectors
- Methods like the logit lens and direct logit attribution may underestimate representation drift

We use TopK SAEs (Gao et al. 2024), but our approach can be combined with **any SAE architecture** and objective.

## Links

- Contact: `tim.lawson@bristol.ac.uk`
- Paper: `https://arxiv.org/abs/2409.04185`
- Repository: `https://github.com/tim-lawson/mlsae`
- Models: `https://huggingface.co/tim-lawson`
- Metrics: `https://wandb.ai/timlawson-/mlsae`

(a) Paper (b) Repository

## The small print

### Methods

- We trained multi-layer SAEs on transformers from the Pythia suite, but we are working on GPT-2, Llama-3.2, and Gemma 2
- We take the residual stream activations after each block, exclude the input embeddings, and skip the final layer norm
- We use a $k$-sparse autoencoder, a.k.a. a TopK SAE, but we are working on other SAE architectures and objectives
- Our default hyperparameters are an expansion factor of $R = 64$ and sparsity $k = 32$, but we explore others in the appendix
- We use tuned lenses trained by FAR.AI

### Latent distributions over layers

- The observed distribution of latent activations over layers is the sum for inputs from each layer, normalized by the sum for all layers: $P(L = \ell \mid T = t, J = j) = h_j(\mathbf{x}_{t,\ell}) / \sum_{\ell'} h_j(\mathbf{x}_{t,\ell'})$
- We sort latent indices in the heatmaps in ascending order of the expected value of the layer index $\mathbb{E}[L \mid J = j]$
- The variance for one latent aggregating over tokens, as a proportion of the total variance over all latents, is $\frac{\mathbb{E}[\mathrm{Var}(L|J)]}{\mathrm{Var}(L)}$
- The variance for one token and latent as a proportion of the total variance for that latent is $\frac{\mathbb{E}[\mathrm{Var}(L|J,T)]}{\mathbb{E}[\mathrm{Var}(L|J)]}$

**Acknowledgements** Tim Lawson and Lucy Farnik are funded by the UKRI Centre for Doctoral Training in Interactive Artificial Intelligence.

## References

Balagansky, N. et al. (Oct. 2024). *Mechanistic Permutability: Match Features Across Layers*. DOI: `10.48550/arXiv.2410.07656`.

Balcells, D. et al. (Nov. 2024). *Evolution of SAE Features Across Layers in LLMs*. DOI: `10.48550/arXiv.2410.08869`.

Belrose, N. et al. (Nov. 2023). *Eliciting Latent Predictions from Transformers with the Tuned Lens*. DOI: `10.48550/arXiv.2303.08112`.

Elhage, N. et al. (2021). *A Mathematical Framework for Transformer Circuits*. URL: `https://transformer-circuits.pub/2021/framework/index.html`.

Gao, L. et al. (June 2024). *Scaling and evaluating sparse autoencoders*. DOI: `10.48550/arXiv.2406.04093`.

Ghilardi, D. et al. (Oct. 2024). *Efficient Training of Sparse Autoencoders for Large Language Models via Layer Groups*. DOI: `10.48550/arXiv.2410.21508`.

Goldowsky-Dill, N. et al. (May 2023). *Localizing Model Behavior with Path Patching*. DOI: `10.48550/arXiv.2304.05969`.

Heimersheim, S. and A. Turner (May 2023). *Residual stream norms grow exponentially over the forward pass*. URL: `https://www.alignmentforum.org/posts/8mizBCm3dyc432nK8/residual-stream-norms-grow-exponentially-over-the-forward`.

Lad, V. et al. (June 2024). *The Remarkable Robustness of LLMs: Stages of Inference?* DOI: `10.48550/arXiv.2406.19384`.

nostalgebraist (Aug. 2020). *Interpreting GPT: the logit lens*. URL: `https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens`.

Paulo, G. et al. (Oct. 2024). *Automatically Interpreting Millions of Features in Large Language Models*. DOI: `10.48550/arXiv.2410.13928`.

Yun, Z. et al. (Apr. 2023). *Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors*. DOI: `10.48550/arXiv.2103.15949`.