Circuit Tracing: Revealing Computational Graphs in Language Models

Ameisen et al. (Anthropic)

Presented by Conor Miller-Lynch

Agenda

- Problem Statement
- Prior Work
- Approach
- Results
- Limitations
- Conclusion

Problem Statement

- We want to explain LMs’ behavior using computational graphs
- The nodes and edges should be meaningful to humans
- Construct graphs for one prompt at a time

Prior Work

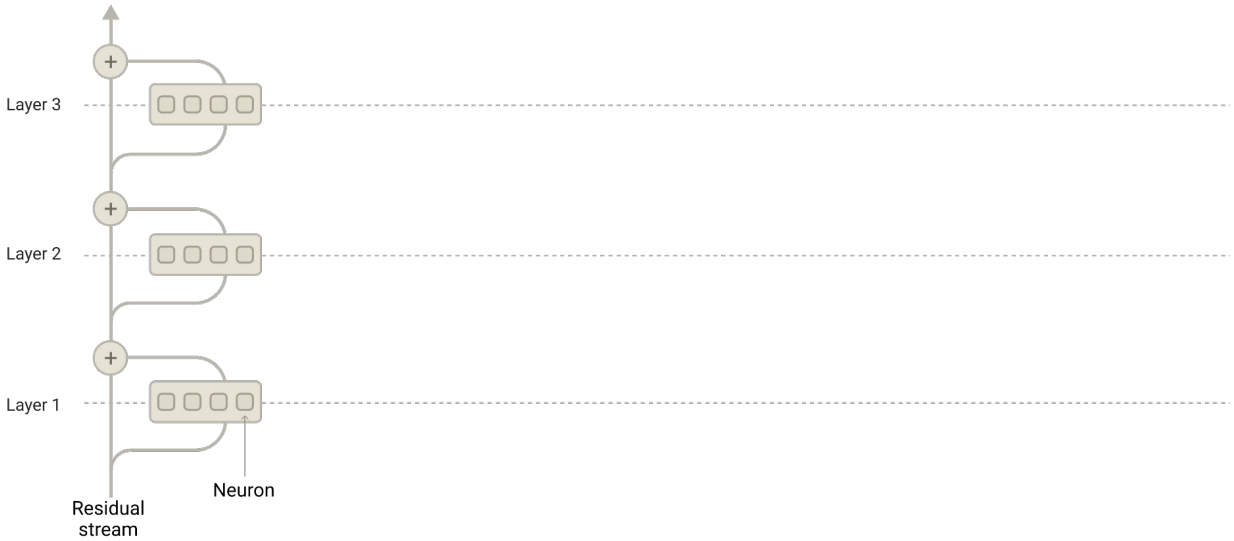
- Cammarata et al. [1] constructed circuits out of **raw neurons**, but neurons are often polysemantic (have multiple unrelated roles)
- Several other works have used sparse coding approaches:
  - **Sparse autoencoders (SAEs)** [2], which attempt to reconstruct the output of MLPs
  - **Transcoders** [3], which attempt to emulate MLPs
  - **Cross-Layer Transcoders** [4], transcoders that output to multiple layers

[1] Cammarata, et al., “Thread: Circuits”, Distill, 2020.
[2] Bricken, et al., “Towards Monosemanticity: Decomposing Language Models With Dictionary Learning”, Transformer Circuits Thread, 2023.
[3] Dunefsky, et al., “Transcoders find interpretable LLM feature circuits”, NeurIPS, 2025.
[4] CLT: Lindsey, et al., "Sparse Crosscoders for Cross-Layer Features and Model Diffing", Transformer Circuits Thread, 2024.

Prior Work

- This paper is the first full paper to use cross-layer transcoders (CLTs)
- These model the computation performed by an LM’s MLPs
- Their cross-layer nature results in simpler computational paths

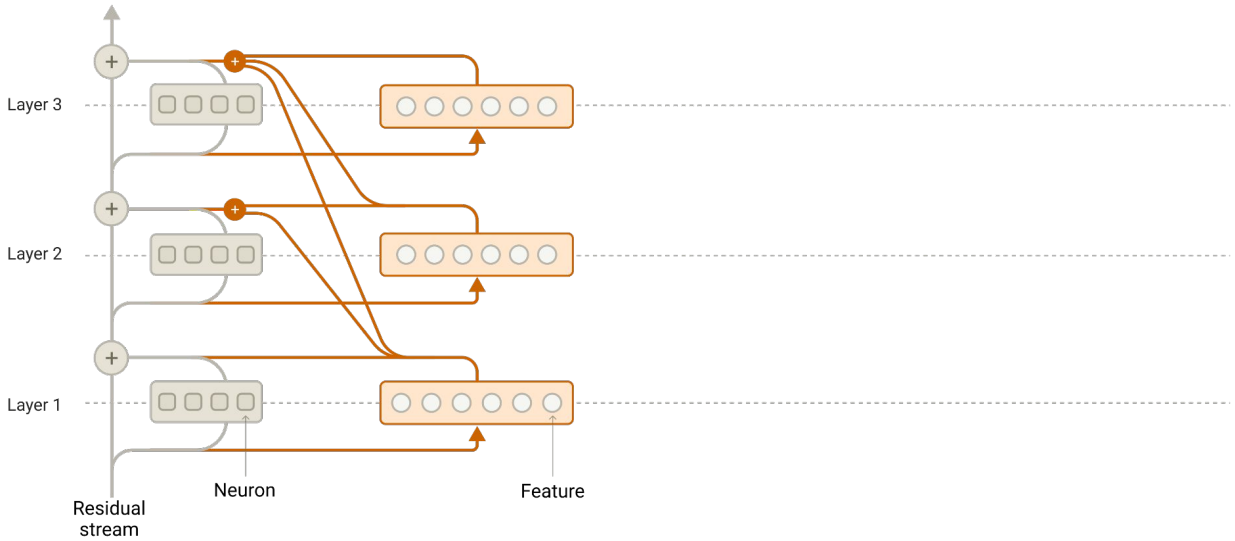
Approach: Overview



- Begin with an existing LM

Figure: Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.

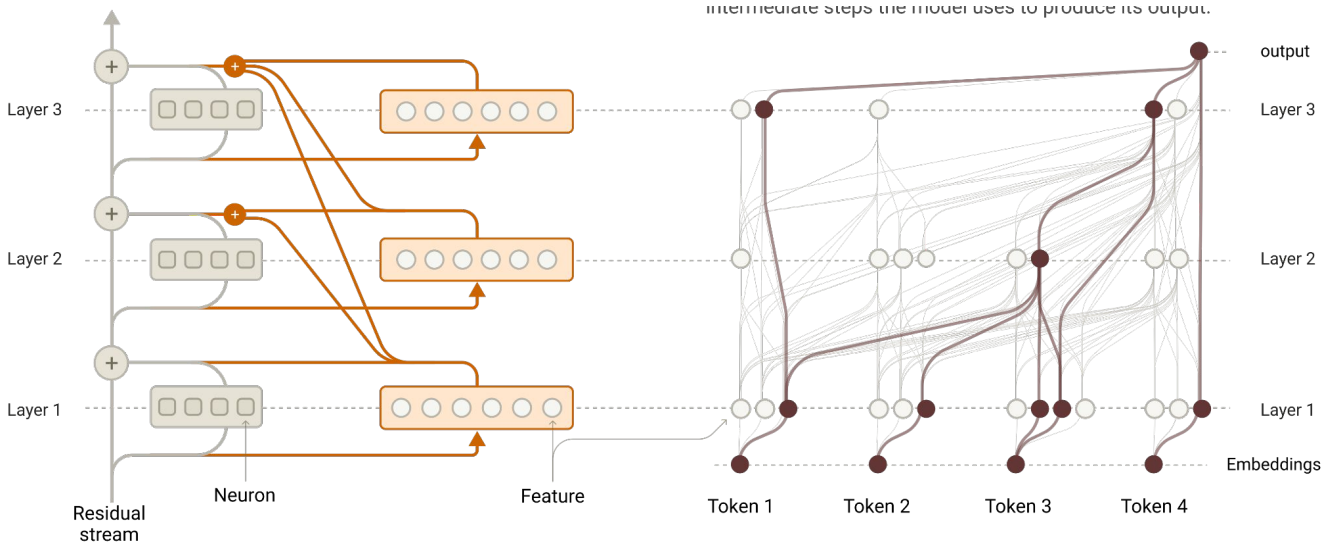
Approach: Overview



- Begin with an existing LM
- Train a *replacement model*

Figure: Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.

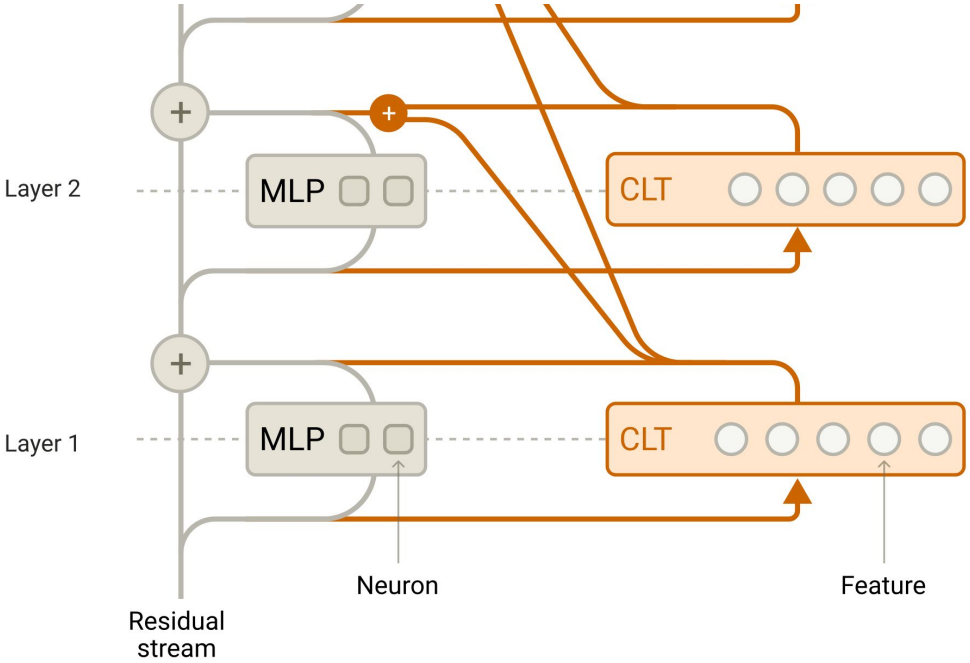
Approach: Overview



- Begin with an existing LM
- Train a *replacement model*
- Construct an *attribution graph*

Figure: Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.

Approach: Local Replacement Model



- Train a cross-layer transcoder (CLT) to replace the MLPs in an LM
- Each layer has:
  - An encoder
  - A non-linearity (features)
  - A decoder for each subsequent layer

Figure: Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.
CLT: Lindsey, et al., "Sparse Crosscoders for Cross-Layer Features and Model Diffing", Transformer Circuits Thread, 2024.

Approach: Local Replacement Model

- “Local”: for a specific prompt
- Fix attention weights and LayerNorm denominators
- This makes connections between features linear

Approach: Local Replacement Model

$$L_{\text{MSE}} = \sum_{\ell=1}^{L} \|\hat{\mathbf{y}}^{\ell} - \mathbf{y}^{\ell}\|^2$$ **Reconstruction Error Loss**

$\hat{\mathbf{y}}^{\ell}$ MLP output reconstruction

$\mathbf{y}^{\ell}$ Original MLP output

ℓ Layer index

Approach: Local Replacement Model

$$L_{\text{sparsity}} = \lambda \sum_{\ell=1}^{L} \sum_{i=1}^{N} \tanh(c \cdot \|\mathbf{W}_{\mathbf{dec,i}}^{\boldsymbol{\ell}}\| \cdot a_i^{\ell})$$ **Sparsity penalty**

c Hyperparameter

$\mathbf{W}_{\mathbf{dec,i}}^{\boldsymbol{\ell}}$ Decoder weights

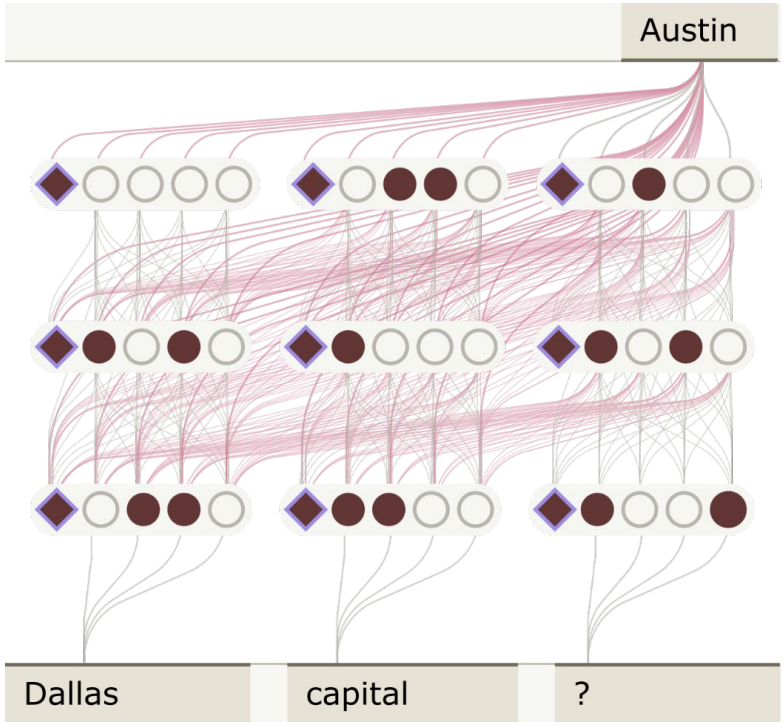
a_i^{ℓ} Feature activation

i Feature index

ℓ Layer index

λ Hyperparameter

Approach: Local Replacement Model



Reconstruction Error

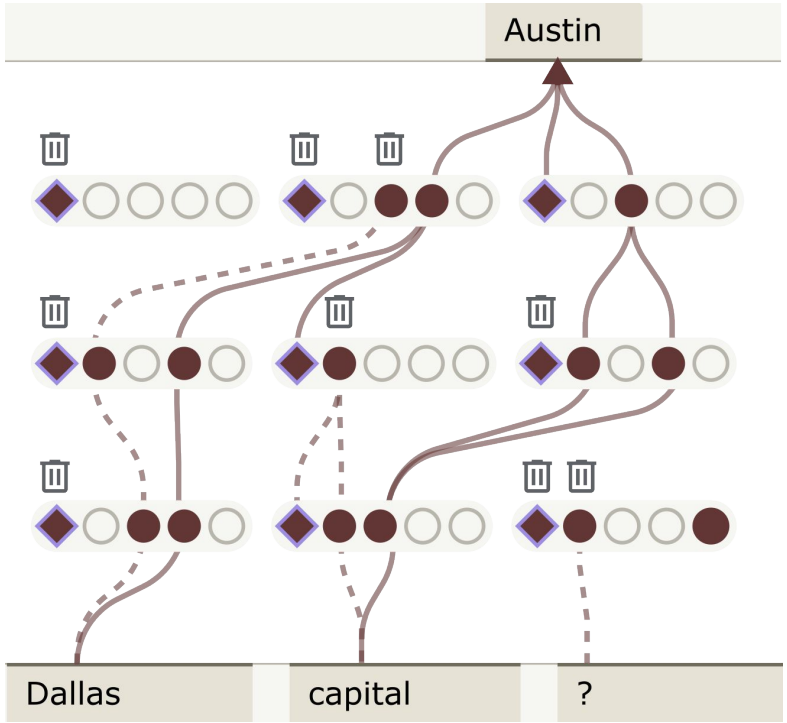
Error nodes represent the difference between the original MLP output and the replacement model's reconstruction

Attention-mediated weights

Attention patterns are frozen to their value in the original model, allowing us to define weights between features in different token positions

Figure: Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.

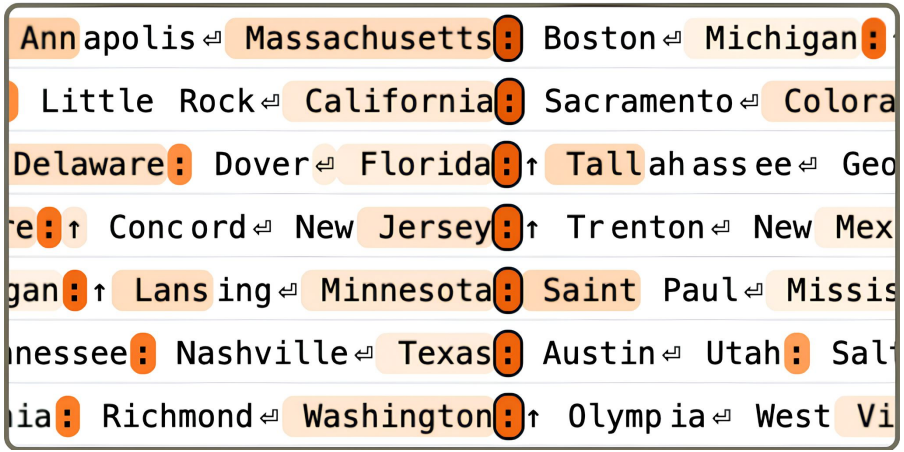
Approach: Attribution Graph



- Eliminate weights that don’t affect the output
- This produces an attribution graph
- We still need to label the nodes

Figure: Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.

Approach: Attribution Graph



- Features are labeled manually using visualizations on dataset examples
- This feature is active right before saying a state capital

Figure: Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.

Results: `Fact: Michael Jordan plays the sport of`

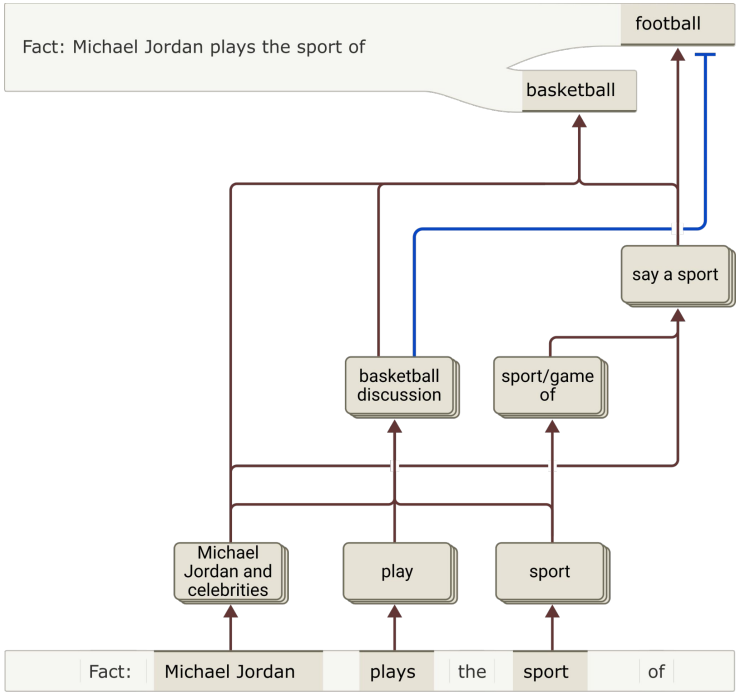


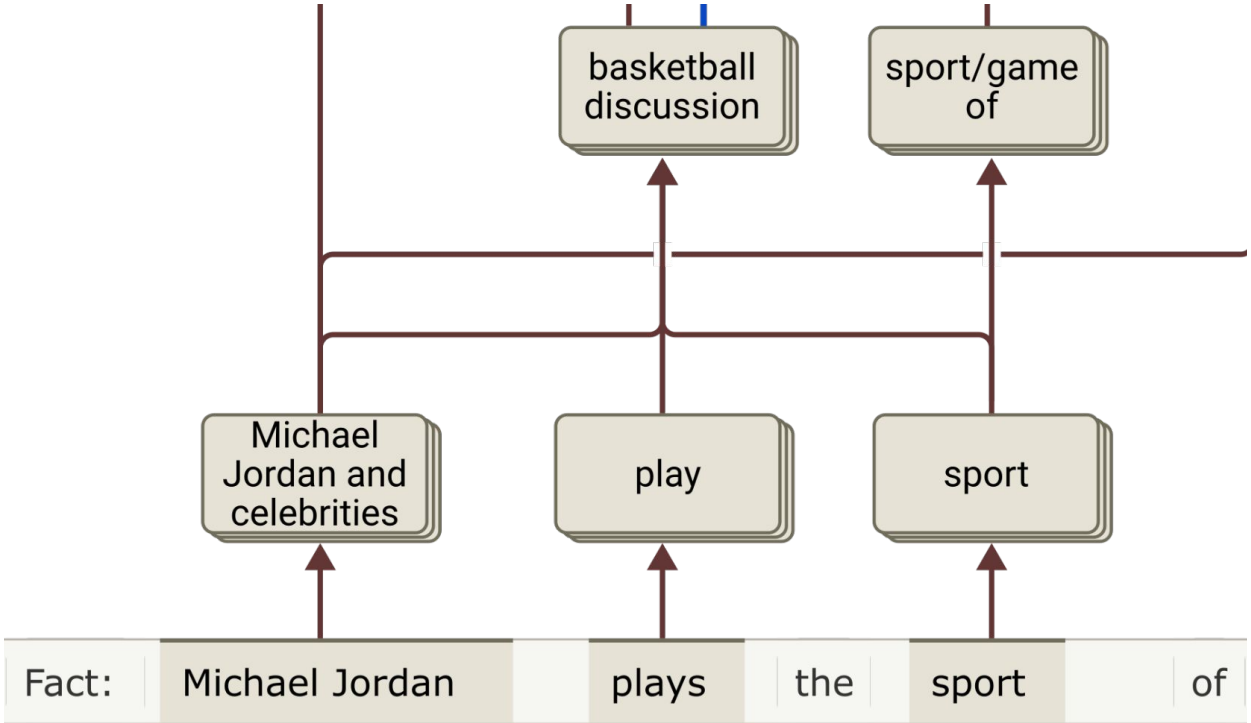
Figure 13: A simplified diagram of the attribution graph for 18L recalling a simple fact.

Let's look at the simplified attribution graph for this prompt:

`Fact: Michael Jordan plays the sport of`

Figure: Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.

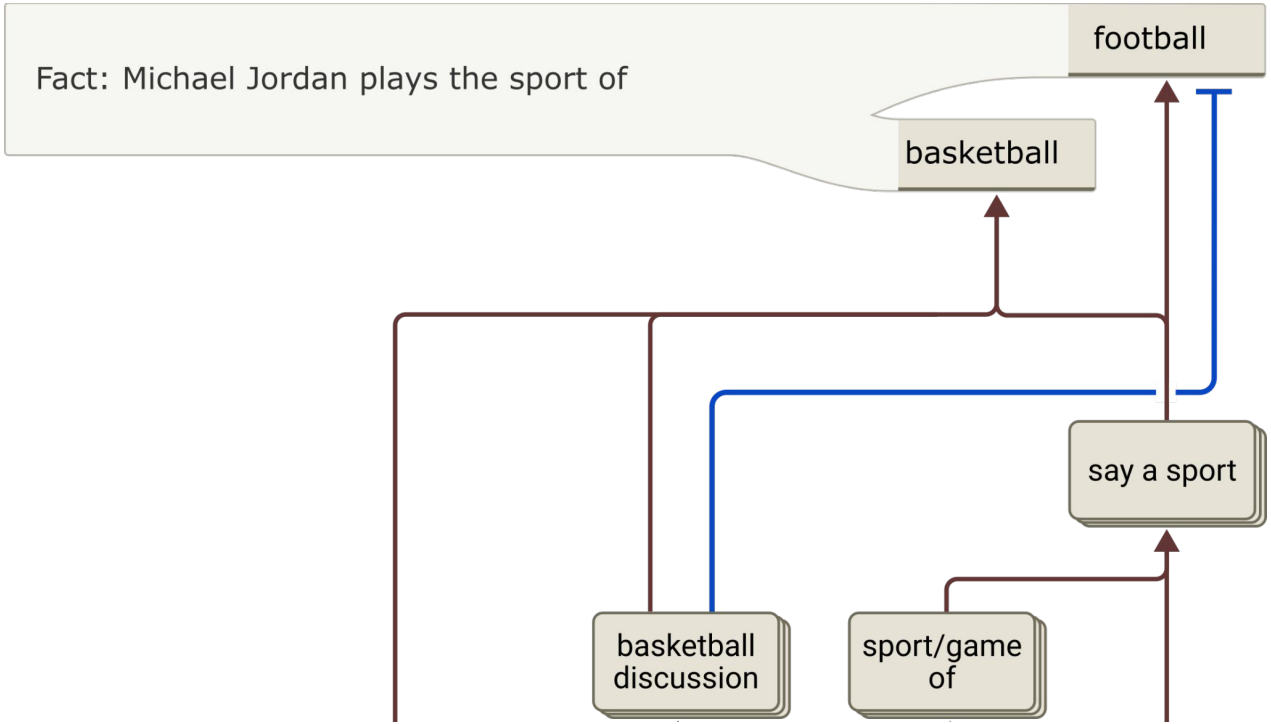
Results: `Fact: Michael Jordan plays the sport of`



- Some early features just activate on specific tokens
- Later features represent more abstract concepts

Figure: Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.

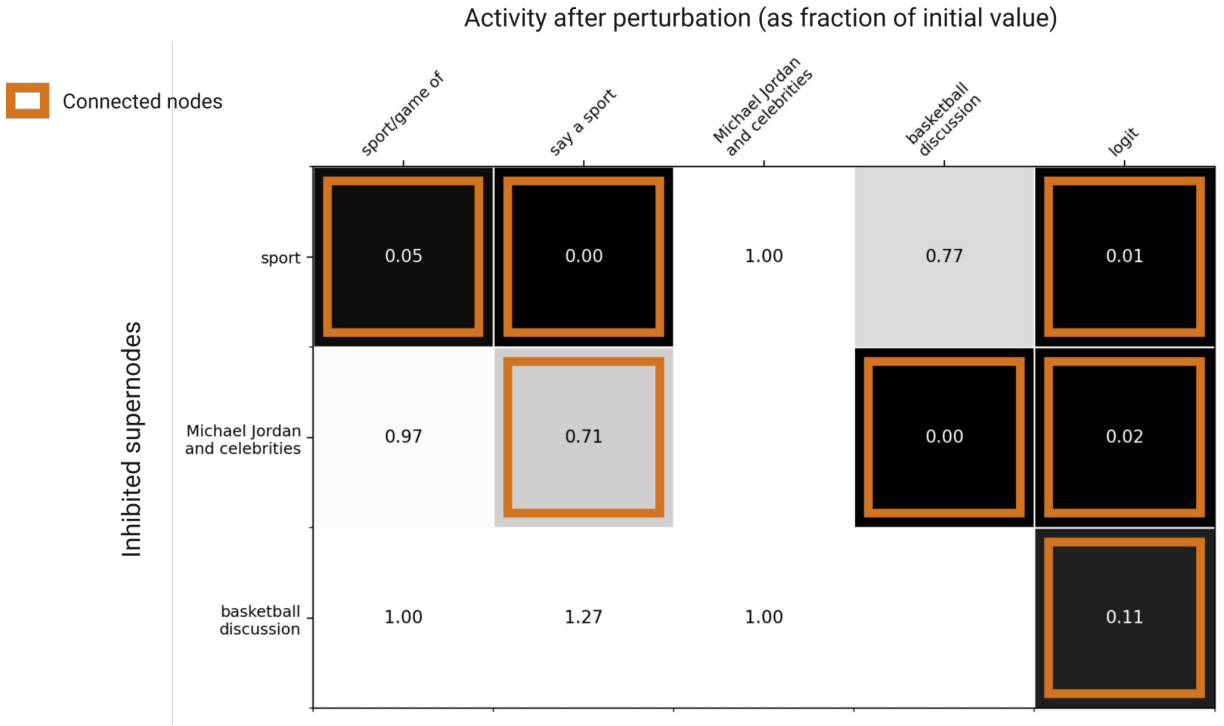
Results: `Fact: Michael Jordan plays the sport of`



- Features interact to produce the final answer
- Here, `basketball discussion` inhibits `football`

Figure: Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.

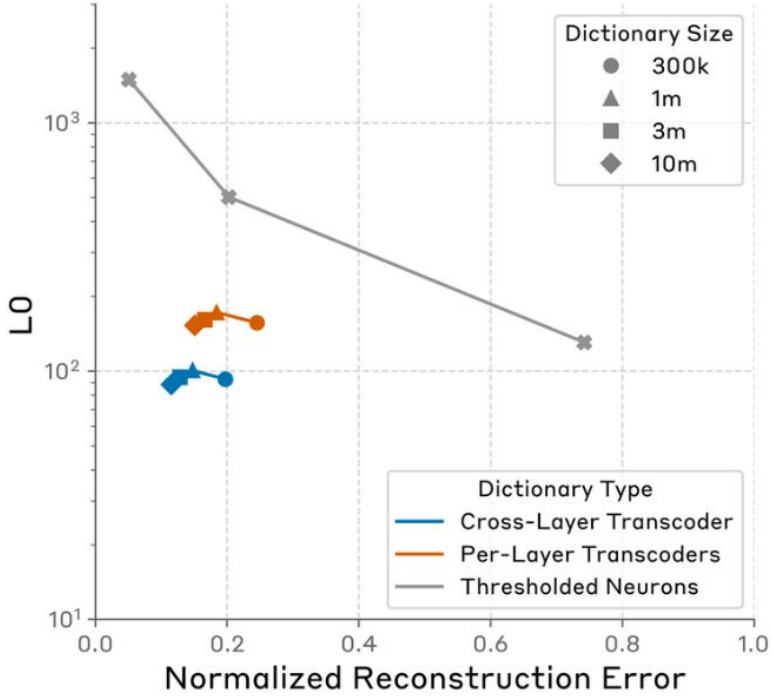
Results: `Fact: Michael Jordan plays the sport of`



- Inhibiting features changes activations and the output
- This validates the features

Figure: Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.

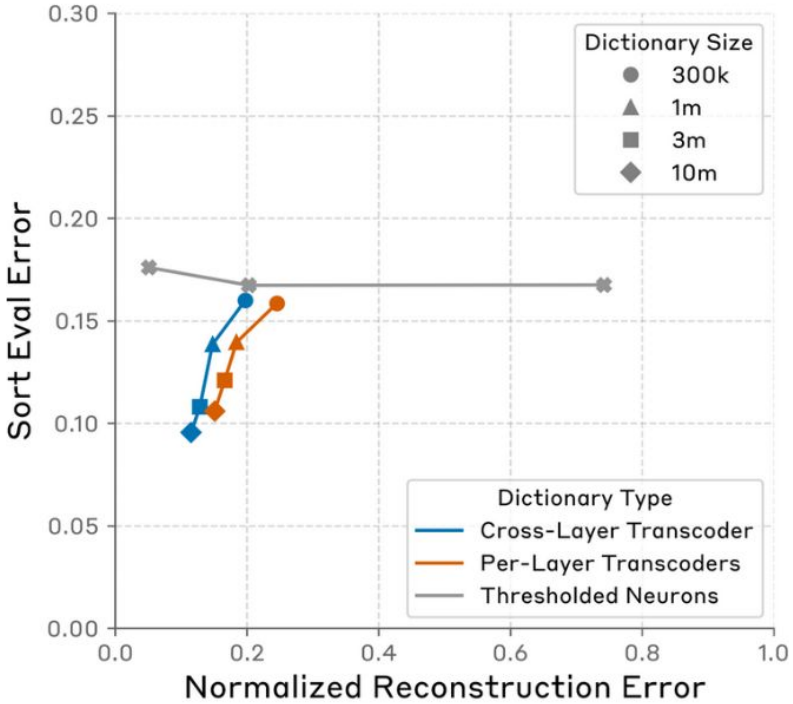
Results: Quantitative



L0: The average number of active features

Figure: Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.

Results: Quantitative

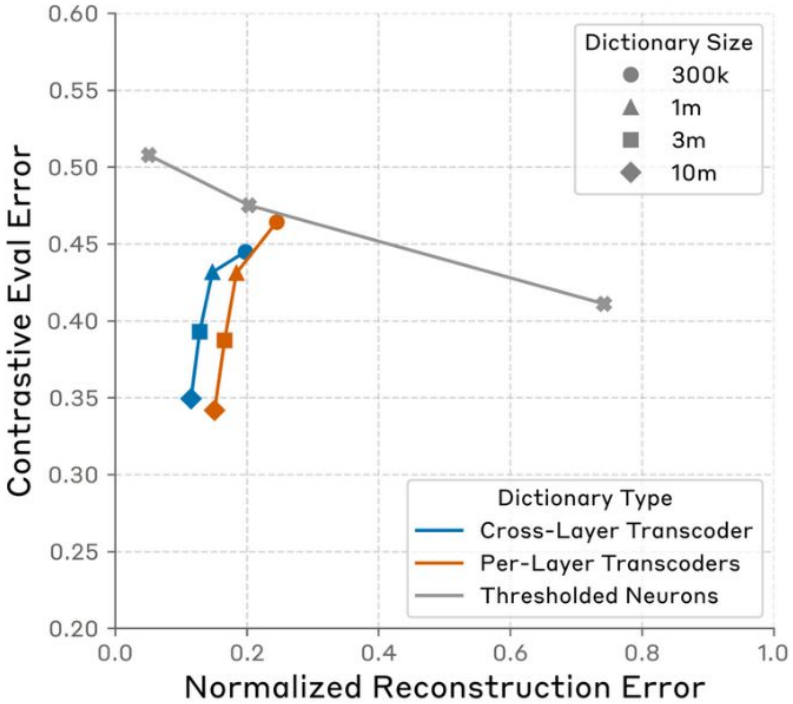


Sort Eval:

1. Randomly sample two features
2. Find the dataset examples that most strongly activate each feature
3. Provide these examples and feature activation information to Claude
4. For a new example, ask Claude which feature it activates

Figure: Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.

Results: Quantitative



Contrastive Eval:

1. Generate a pair of similar prompts with one key difference
2. Identify features that only activate on one of them
3. For each feature, provide the prompts and feature visualizations to Claude
4. Ask Claude to guess which prompt causes the feature to activate

Figure: Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.

Limitations

- Only focuses on explaining OV circuits
- Primarily focuses on local replacement models
- Parts of the approach are highly qualitative

Conclusion

- Cross-layer transcoders learn interpretable features
- Models can use a variety of heuristics to arrive at an answer
- There is still much work to be done

References

Ameisen, et al., "Circuit Tracing: Revealing Computational Graphs in Language Models", Transformer Circuits, 2025.

Cammarata, et al., “Thread: Circuits”, Distill, 2020.

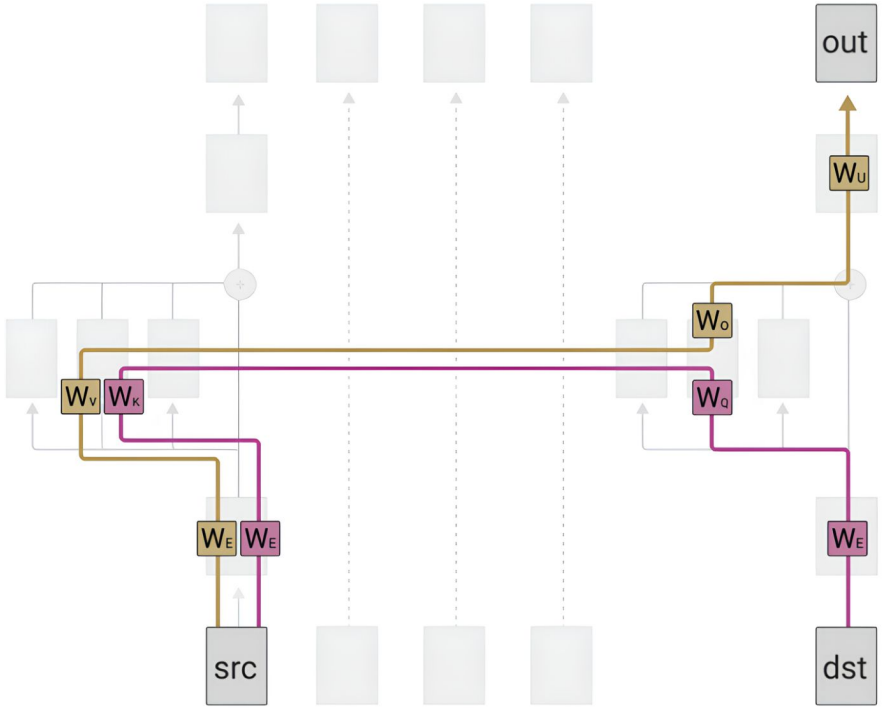
Bricken, et al., “Towards Monosemanticity: Decomposing Language Models With Dictionary Learning”, Transformer Circuits Thread, 2023.

Dunefsky, et al., “Transcoders find interpretable LLM feature circuits”, NeurIPS, 2025.

Lindsey, et al., "Sparse Crosscoders for Cross-Layer Features and Model Diffing", Transformer Circuits Thread, 2024.

Elhage, et al., "A Mathematical Framework for Transformer Circuits", Transformer Circuits Thread, 2021.

Background: QK Circuits and OV Circuits



The **OV ("output-value") circuit** determines how attending to a given token affects the logits.

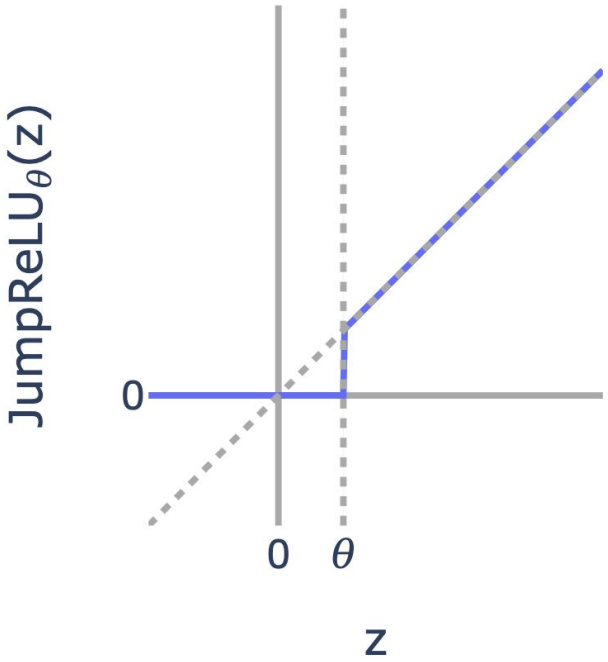
$$W_U W_O W_V W_E$$

The **QK ("query-key") circuit** controls which tokens the head prefers to attend to.

$$W_E^T W_Q^T W_K W_E$$

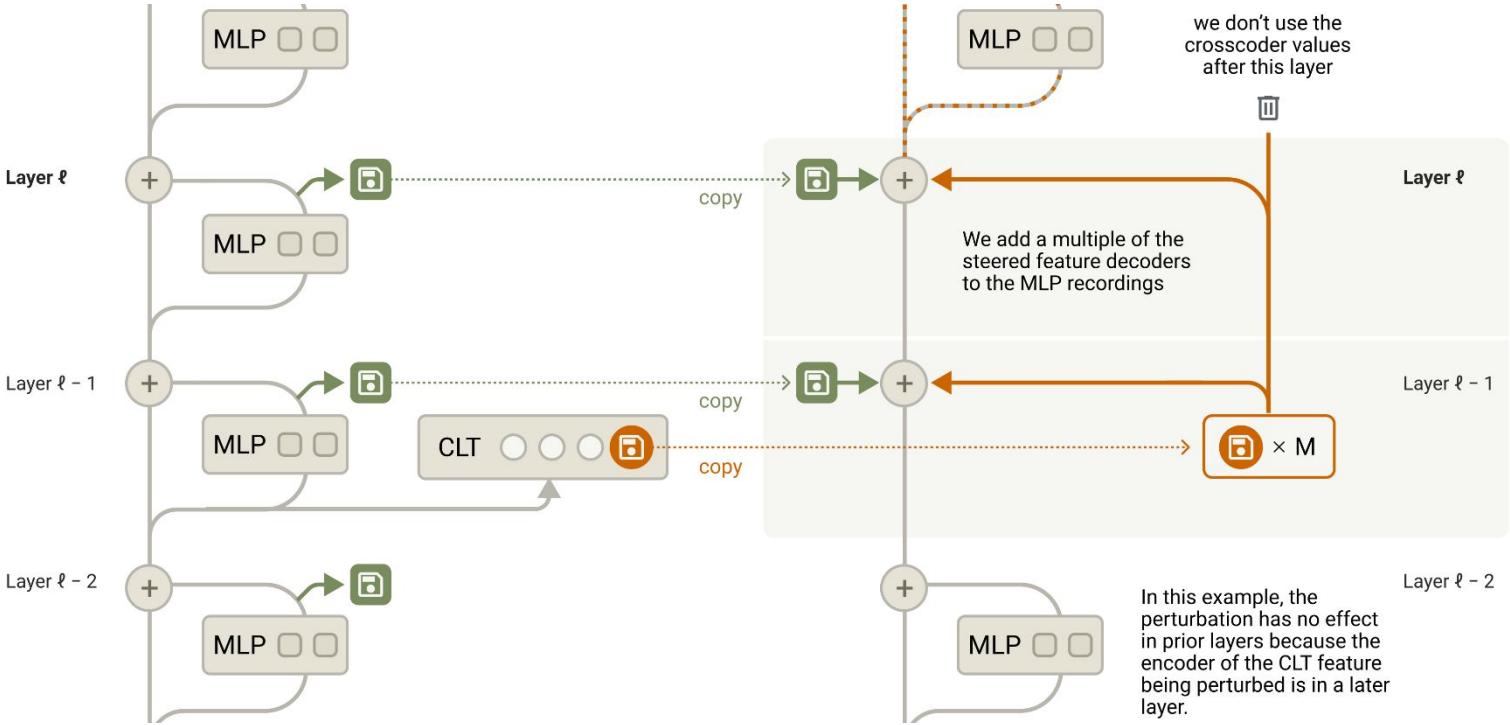
Elhage, et al., "A Mathematical Framework for Transformer Circuits", Transformer Circuits Thread, 2021.

Approach: Local Replacement Model



- CLTs use the JumpReLU activation function
- This helps enable sparsity

Approach: Validation



Approach: Global Weights

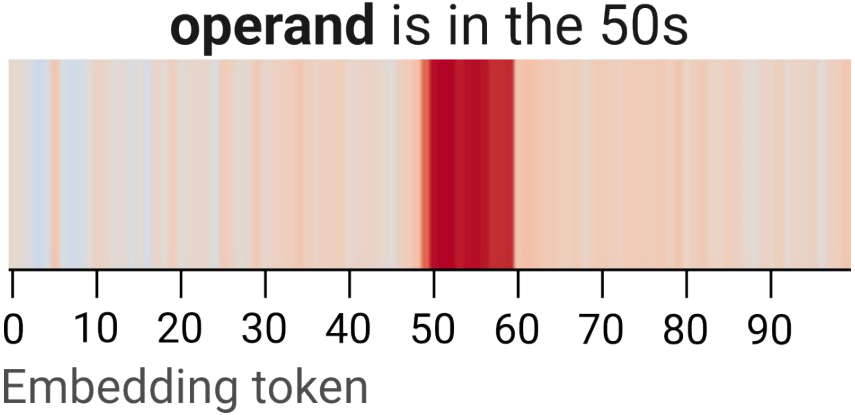
$$V_{st} = \left\langle \sum_{\ell \in L_{st}} W_{\text{dec}}^{s,\ell}, W_{\text{enc}}^{t} \right\rangle$$

$$V_{ij}^{\text{TWERA}} = \frac{\mathbb{E}\left[a_j a_i\right]}{\mathbb{E}\left[a_j\right]} V_{ij}$$

Results: `calc: 36+59=`

Embedding Weight Plots

The direct effect of embeddings for tokens 0–99 on the feature encoder. Visualized for features active on **a** or **b**.

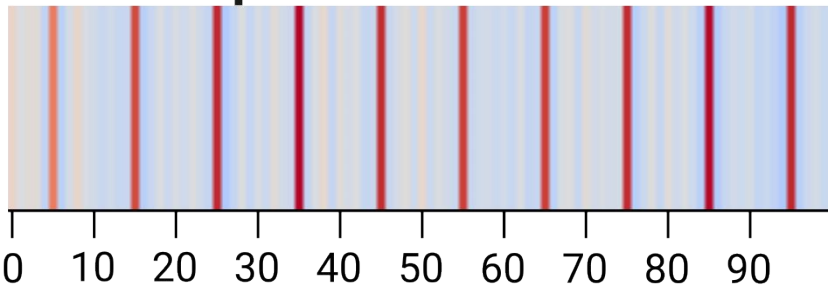


Results: `calc: 36+59=`

Output Weight Plots

The direct effect of the sum of a feature's decoders on the outputs for tokens 0–99. Visualized for features active on =.

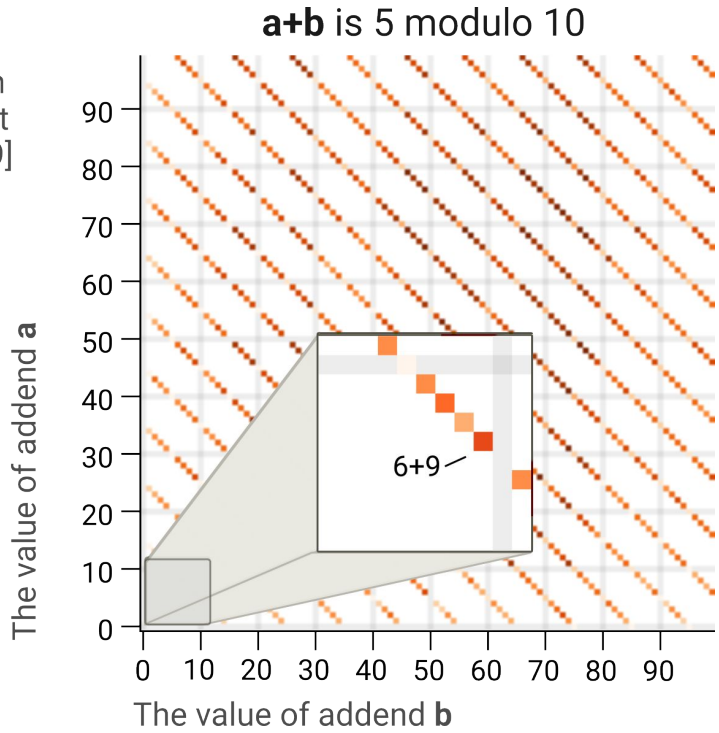
output is 5 modulo 10



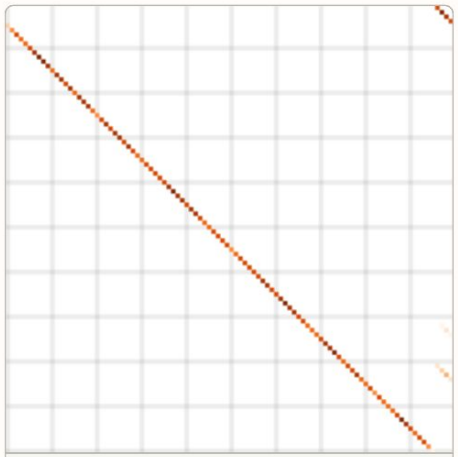
Results: `calc: 36+59=`

Operand Plots

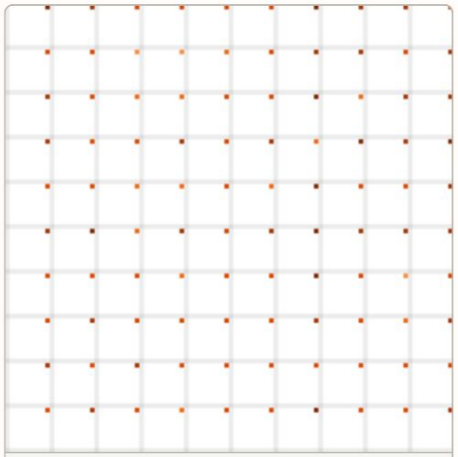
The activity of a feature on the “=” token of the prompt “calc: a+b=”, for a,b ∈ [0,99]



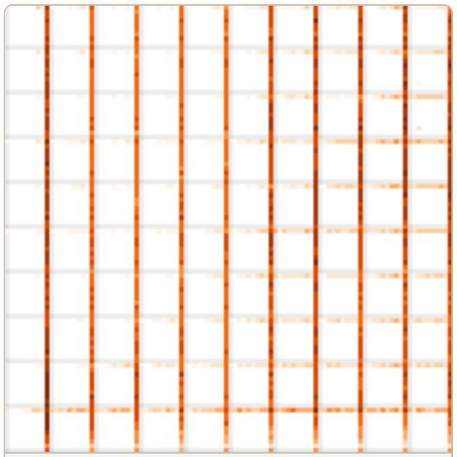
Results: `calc: 36+59=`



sum = _95



_9 + _9



add 9

Results: `calc: 36+59=`

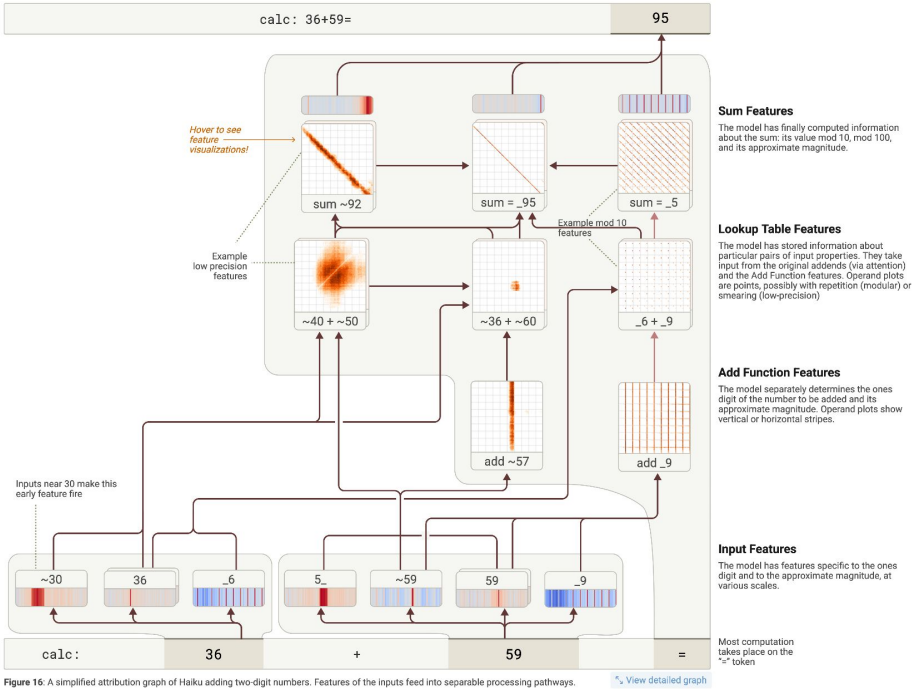


Figure 16: A simplified attribution graph of Haiku adding two-digit numbers. Features of the inputs feed into separable processing pathways.

Results: `calc: 36+59=`

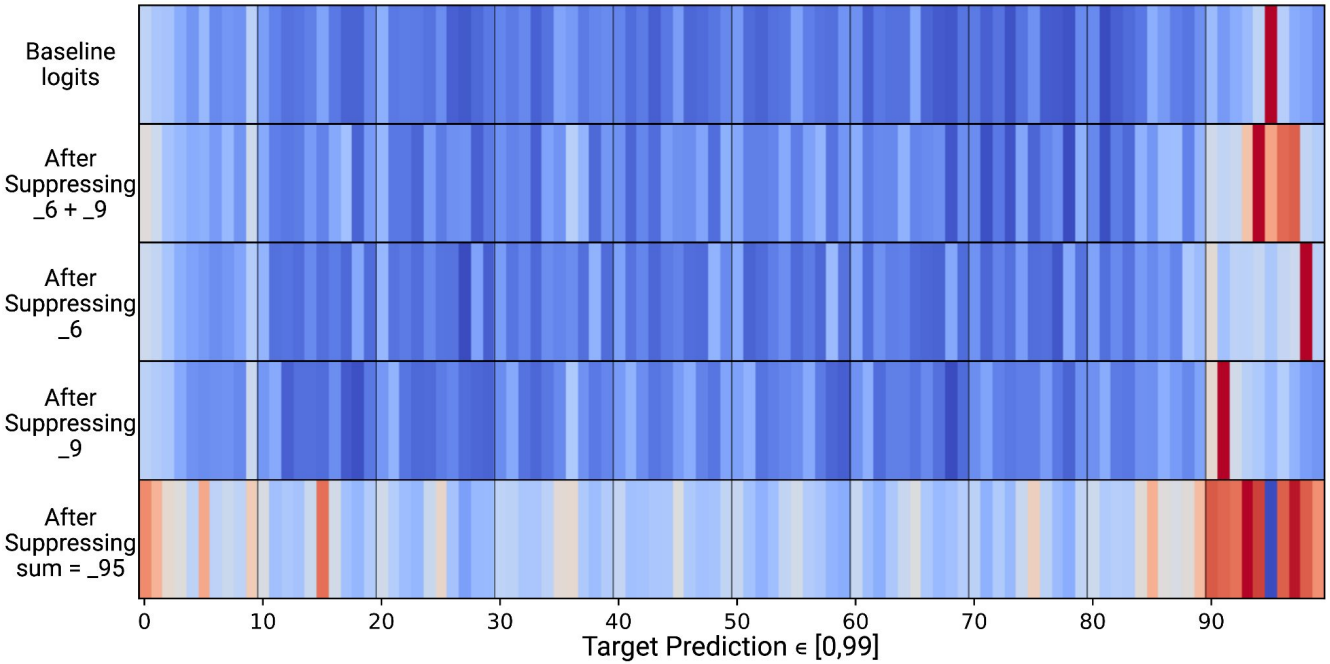


Figure 19: Target prediction logits for different interventions on “calc: 36+59=”.

Results: Quantitative

Path Length:

- How long are the paths in the computation graph?
- Shorter paths can be more interpretable

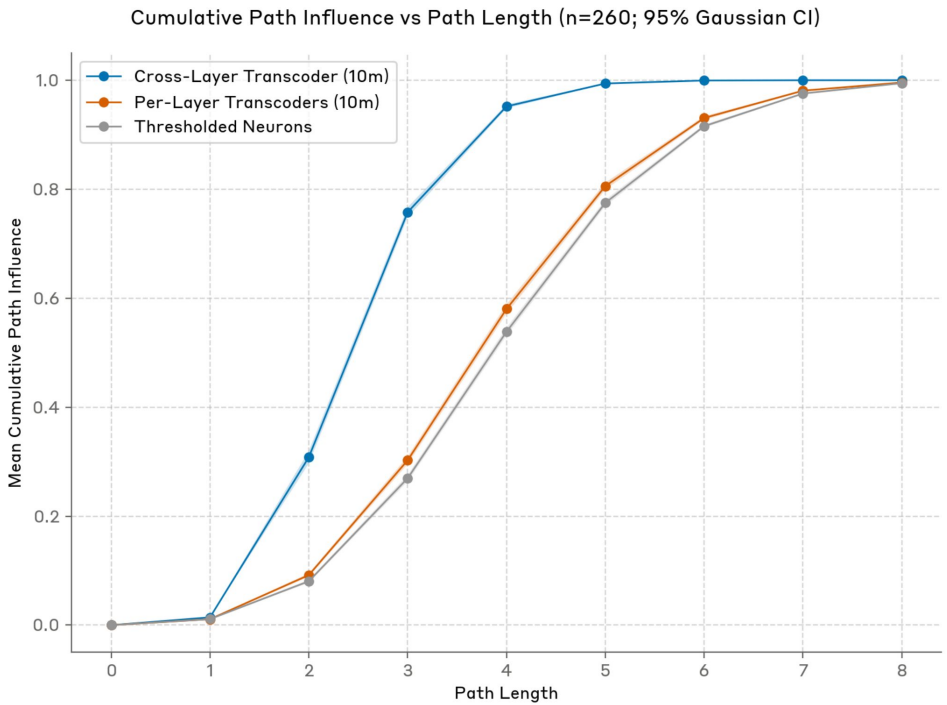
$$B_\ell = \sum_{i=0}^{\ell} A^i$$

(Compute sum of strengths of all paths between connected nodes)

$$P_\ell = \sum_e B_{t,e}^\ell$$

(Average across embedding nodes)

Results: Quantitative



Full Author List

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, Joshua Batson

Motivation

- Language models are powerful, but difficult to interpret
- Neurons cannot be

Motivation

- Language models are powerful, but difficult to interpret
- This limits their trustworthiness and makes it hard to systematically prevent unwanted behavior
- This in turn limits their applications

Background: QK Circuits and OV Circuits

QK Circuit: Determined by **Query** and **Key** matrices

OV Circuit: Determined by **Output** and **Value** matrices

Elhage, et al., "A Mathematical Framework for Transformer Circuits", Transformer Circuits Thread, 2021.

Background: QK Circuits and OV Circuits

“The **QK circuit** determines which "source" token the present "destination" token attends back to and copies information from”

“[T]he **OV circuit** describes what the resulting effect on the "out" predictions for the next token is”

Elhage, et al., "A Mathematical Framework for Transformer Circuits", Transformer Circuits Thread, 2021.

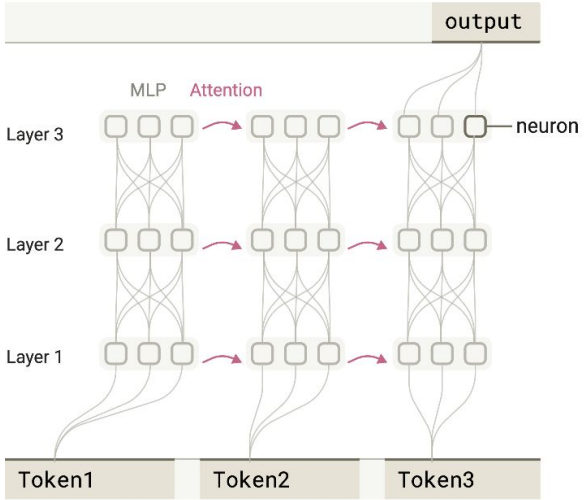
Approach: Overview

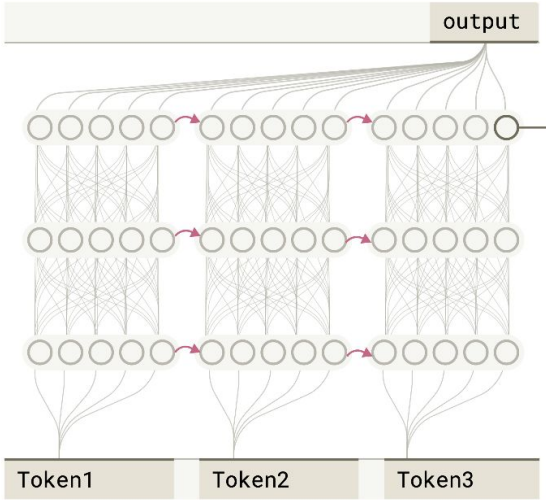
Original Transformer Model

The underlying model that we study is a transformer-based large language model.

Replacement Model

We replace the neurons of the original model with *features*. There are typically more features than neurons. Features are sparsely active and often represent interpretable concepts.





Feature

To understand what a feature represents, we use a *feature visualization*, which shows dataset examples for which the feature is most strongly active. In this example, the feature fires strongly when the model is about to say a state capital.

Figure 2: The replacement model is obtained by replacing the original model's neurons with the cross-layer transcoder's sparsely-active features.

Approach: Perturbation

Approach: Global Weights

Discussion: Paper

- Concepts are introduced in a logical order
- Interactive examples help communicate concepts
- The authors are transparent about the weaknesses of their approach