CS 489: Introduction to Natural Language Processing

Lecture 13: Mechanistic Interpretability

Instructor: Freda Shi

November 3, 2026



UNIVERSITY OF WATERLOO

Outline

1. What computation produces a prediction?
   Circuits and induction heads
2. Which internal states affect that prediction?
   Activation patching
3. Which features inside those states matter?
   Superposition and sparse autoencoders

From Accessible Information to Computation

Lecture 12: A probe reads information from frozen hidden states.

A successful readout does not identify the computation producing the LM's prediction.

From Accessible Information to Computation

Lecture 12: A probe reads information from frozen hidden states.

A successful readout does not identify the computation producing the LM's prediction.

Mechanistic interpretability aims to explain behavior through the computations that produce it.

Start with a temporary codebook: propose a lookup computation, then test its predictions with interventions.

In-Context Learning: A Temporary Codebook

Toy task: Complete the last row using this prompt's animal codes.

Prompt 1

cat	7
dog	2
bird	9
cat	___

In-Context Learning: A Temporary Codebook

Toy task: Complete the last row using this prompt's animal codes.

Prompt 1

cat	7
dog	2
bird	9
cat	7

Prompt 2

cat	4
dog	2
bird	9
cat	___

In-Context Learning: A Temporary Codebook

Toy task: Complete the last row using this prompt's animal codes.

Prompt 1

cat	7
dog	2
bird	9
cat	7

Prompt 2

cat	4
dog	2
bird	9
cat	4

Same LM, separate prompts: changing the codebook changes the answer.
In-context learning: Use the current prompt's examples, with no weight update.

Induction: Copy the Code After a Match

Simplify Prompt 1: treat each animal name and code as one token.

cat 7 dog 2 bird 9 cat $\longrightarrow$ 7

Induction: Copy the Code After a Match

Simplify Prompt 1: treat each animal name and code as one token.

$$\texttt{cat}\ \texttt{7}\quad \texttt{dog}\ \texttt{2}\quad \texttt{bird}\ \texttt{9}\quad \texttt{cat} \longrightarrow \texttt{7}$$

$$\underbrace{[A][B]}_{\text{earlier example}} \quad \ldots \quad \underbrace{[A]}_{\text{query}} \longrightarrow [B]$$

Here $A = \texttt{cat}$ is the lookup token and $B = \texttt{7}$ is its code.

Induction: Copy the Code After a Match

Simplify Prompt 1: treat each animal name and code as one token.

`cat 7  dog 2  bird 9  cat` $\longrightarrow$ `7`

$$\underbrace{[A][B]}_{\text{earlier example}} \quad \ldots \quad \underbrace{[A]}_{\text{query}} \quad \longrightarrow \quad [B]$$

Here $A =$ `cat` is the lookup token and $B =$ `7` is its code.

Induction: Match an earlier A and copy the following B.
We will trace a **circuit** that implements this lookup pattern.

What Is a Circuit?

How could the earlier *A* and *B* lead to a prediction of *B* at the current position?

What Is a Circuit?

How could the earlier *A* and *B* lead to a prediction of *B* at the current position?

A **circuit** is a set of components and connections proposed to implement a particular behavior.

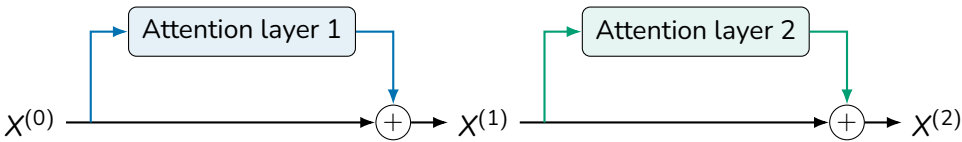
A circuit explanation should specify:

1. What information each component reads and writes.
2. How those writes influence later components and the output.
3. Which interventions support the proposed computation.

Elhage et al. (2021), *A Mathematical Framework for Transformer Circuits*.

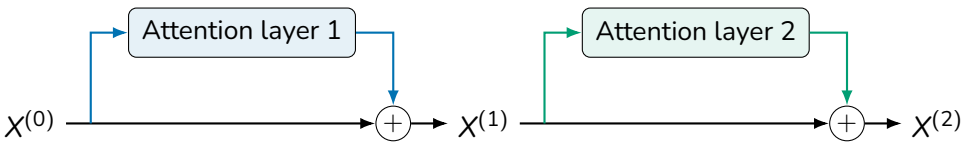
How Can One Head Pass Information to Another?

At each position t, the **residual stream** is a vector $\mathbf{x}_t^{(\ell)} \in \mathbb{R}^d$.
Write $X^{(\ell)}$ for the collection of these vectors.



How Can One Head Pass Information to Another?

At each position t, the **residual stream** is a vector $\mathbf{x}_t^{(\ell)} \in \mathbb{R}^d$.
Write $X^{(\ell)}$ for the collection of these vectors.



Attention reads states across positions and adds an output at each position.
An earlier head's write can become part of a later head's key or value.

Match A, Retrieve the Following B

$$\underbrace{[A]_{s-1}[B]_s}_{\text{earlier in context}} \quad \dots \; [A]_t \quad \longrightarrow \quad [B]$$

Matching the current A to the earlier A reads the value at position $s - 1$.

Match A, Retrieve the Following B

$$\underbrace{[A]_{s-1}[B]_{s}}_{\text{earlier in context}} \quad \dots \; [A]_{t} \quad \longrightarrow \quad [B]$$

Matching the current A to the earlier A reads the value at position $s - 1$.

The token we want is at position s, where B occurs.

Match A, Retrieve the Following B

$$\underbrace{[A]_{s-1}[B]_s}_{\text{earlier in context}} \quad \dots \; [A]_t \quad \longrightarrow \quad [B]$$

Matching the current A to the earlier A reads the value at position $s-1$.

The token we want is at position s, where B occurs.

Plan: Make B's position carry information about the preceding A.

- Form the key from A to locate the right position.
- Form the value from B to retrieve the continuation.

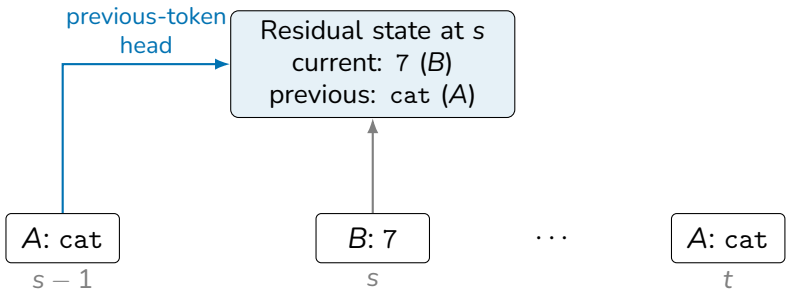
Step 1: Write A at B's Position

A **previous-token head** attends to the immediately preceding position.



Step 1: Write A at B's Position

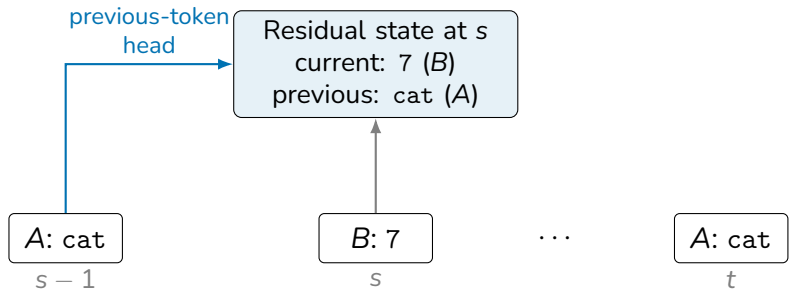
A **previous-token head** attends to the immediately preceding position.



At position s, the head reads position $s-1$ and writes information about A.
The residual state at s carries information about both A and B.

Step 1: Write A at B's Position

A **previous-token head** attends to the immediately preceding position.



At position s, the head reads position $s-1$ and writes information about A. The residual state at s carries information about both A and B.

A later head can form its key from A and value from B.

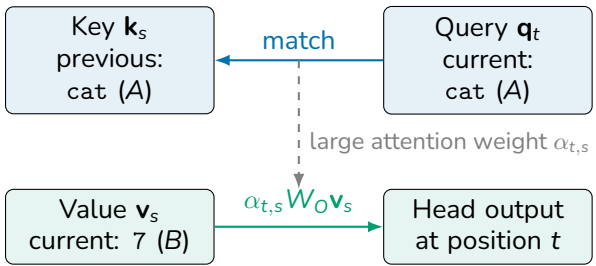
Step 2: Match the Key, Copy the Value

At the query `cat` (*t*), the induction head reads the earlier code `7` (*s*).



Step 2: Match the Key, Copy the Value

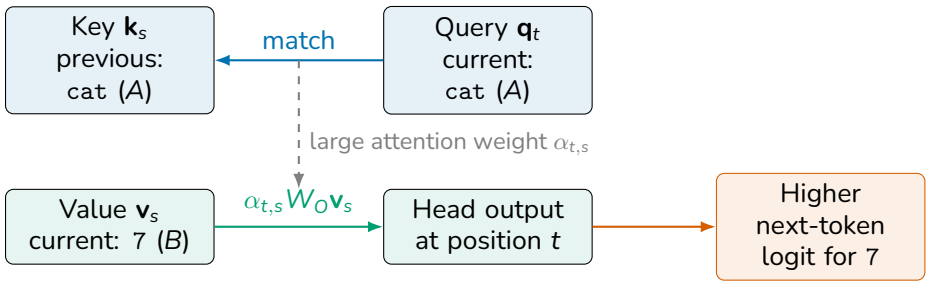
At the query `cat` (t), the induction head reads the earlier code `7` (s).



The key matches A, but the value carries B.
The output is written at t, which predicts token $t + 1$.

Step 2: Match the Key, Copy the Value

At the query `cat` (*t*), the induction head reads the earlier code `7` (*s*).



The key matches *A*, but the value carries *B*.
The output is written at *t*, which predicts token $t + 1$.

In this circuit, the value–output (OV) mapping writes a direction that promotes *B* as the next token.

Predictions of the Proposed Circuit

If the prompt assigns a different code to `cat`, what should the circuit copy?

Context	Predicted continuation
`cat 7 ... cat`	
`cat 4 ... cat`	

Predictions of the Proposed Circuit

If the prompt assigns a different code to `cat`, what should the circuit copy?

Context	Predicted continuation
`cat 7 ... cat`	`7`
`cat 4 ... cat`	`4`

Test many temporary codebooks to check whether the model follows the prompt's assignments.

Predictions of the Proposed Circuit

If the prompt assigns a different code to `cat`, what should the circuit copy?

Context	Predicted continuation
`cat 7 ... cat`	`7`
`cat 4 ... cat`	`4`

Test many temporary codebooks to check whether the model follows the prompt's assignments.

Ablation: Remove or replace a proposed head's output, then recompute downstream states.

What Is the Evidence for Induction?

Does this copying mechanism contribute to prediction beyond one repeated pair?

Olsson et al. (2022) compare early and late next-token loss:

$$G = L_{\text{early}} - L_{\text{late}}, \qquad L = \text{average next-token loss.}$$

Larger G means better prediction on later tokens relative to early tokens.

What Is the Evidence for Induction?

Does this copying mechanism contribute to prediction beyond one repeated pair?

Olsson et al. (2022) compare early and late next-token loss:

$$G = L_{\text{early}} - L_{\text{late}}, \qquad L = \text{average next-token loss.}$$

Larger G means better prediction on later tokens relative to early tokens.

- **Training dynamics:** Induction heads emerge around a sharp increase in this improvement in the studied models.
- **Interventions:** Ablating induction heads reduces this improvement in small attention-only models.

What Is the Evidence for Induction?

Does this copying mechanism contribute to prediction beyond one repeated pair?

Olsson et al. (2022) compare early and late next-token loss:

$$G = L_{\text{early}} - L_{\text{late}}, \qquad L = \text{average next-token loss.}$$

Larger G means better prediction on later tokens relative to early tokens.

- **Training dynamics:** Induction heads emerge around a sharp increase in this improvement in the studied models.
- **Interventions:** Ablating induction heads reduces this improvement in small attention-only models.

Ablation supports a causal contribution in these models.
Caveat: Induction is not a complete account of in-context learning.

Activation Patching

Replace an activation in a **target run** with the corresponding activation from a **source run**, then continue the computation.

Run	Prompt
Source	`cat 7  dog 2  bird 9  cat`
Target	`cat 4  dog 2  bird 9  cat`

As before, each animal name and code is one token.

Activation Patching

Replace an activation in a **target run** with the corresponding activation from a **source run**, then continue the computation.

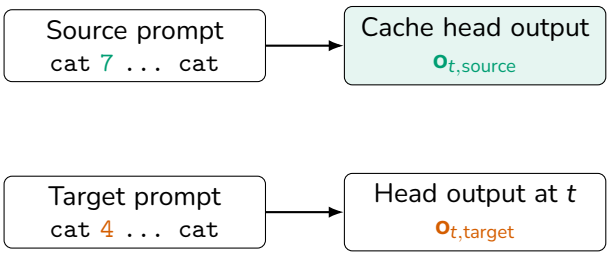
Run	Prompt
Source	`cat 7 dog 2 bird 9 cat`
Target	`cat 4 dog 2 bird 9 cat`

As before, each animal name and code is one token.

Patch site: The induction head's output at the final `cat` (position t).
Both runs use the same model, with unchanged weights.

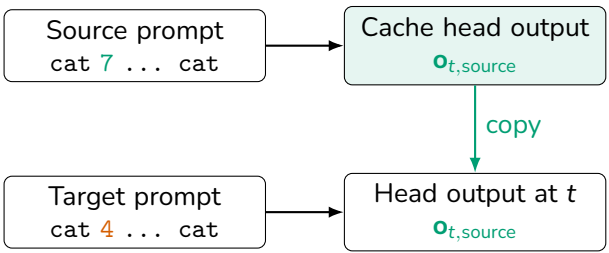
Patch the Head Output, Then Continue

Let $\mathbf{o}_t$ be the head's output after W_O, before residual addition.



Patch the Head Output, Then Continue

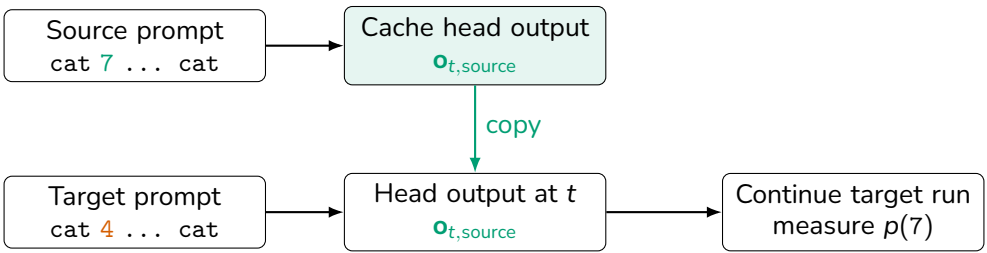
Let $\mathbf{o}_t$ be the head's output after W_O, before residual addition.



Copy between the same head and token position in the two runs.

Patch the Head Output, Then Continue

Let $\mathbf{o}_t$ be the head's output after W_O, before residual addition.



Copy between the same head and token position in the two runs.

Keep all other activations at this stage from the target run.
Recompute the residual addition and the remaining network.

Measure the Change in the Target Prediction

Both runs below use the target prompt: `cat 4 dog 2 bird 9 cat`.

Run	**Head output at** t	**Probability of** 7
Target (no patch)	$\mathbf{o}_{t,\text{target}}$	$p_{\text{target}}(7)$

Measure the Change in the Target Prediction

Both runs below use the target prompt: `cat 4 dog 2 bird 9 cat`.

Run	**Head output at t**	**Probability of 7**
Target (no patch)	$\mathbf{o}_{t,\text{target}}$	$p_{\text{target}}(7)$
Target (patched)	$\mathbf{o}_{t,\text{source}}$	$p_{\text{patched}}(7)$

Circuit prediction: The patch should increase $p(7)$.

Measure the Change in the Target Prediction

Both runs below use the target prompt: `cat 4 dog 2 bird 9 cat`.

Run	Head output at t	Probability of 7
Target (no patch)	$\mathbf{o}_{t,\text{target}}$	$p_{\text{target}}(7)$
Target (patched)	$\mathbf{o}_{t,\text{source}}$	$p_{\text{patched}}(7)$

Circuit prediction: The patch should increase $p(7)$.

An increase supports the proposed code-copying role of this head.

The copied vector can contain several features.

From Neuron Coordinates to Feature Directions

A **polysemantic neuron** responds in several distinct kinds of contexts.
Individual coordinates can mix information about different features.

From Neuron Coordinates to Feature Directions

A **polysemantic neuron** responds in several distinct kinds of contexts.
Individual coordinates can mix information about different features.

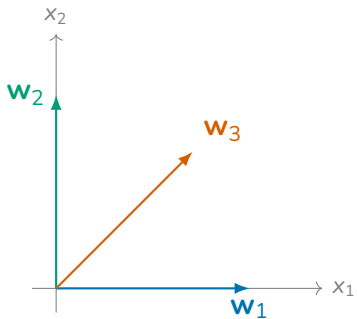
Superposition hypothesis: Represent more feature directions than the activation space has dimensions.

$$\mathbf{x} \approx \sum_{i=1}^{M} f_i \mathbf{w}_i, \qquad \mathbf{w}_i \in \mathbb{R}^d, \quad M > d.$$

Here f_i is a feature's activity and $\mathbf{w}_i$ is its direction.

- Feature directions need not align with neuron coordinates.
- Sparse activity limits interference: only a few features are active at once.

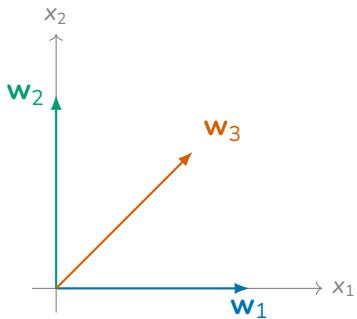
Three Feature Directions in Two Dimensions



Toy dictionary, with unit-length columns:

$$\mathbf{w}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{w}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{w}_3 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Three Feature Directions in Two Dimensions

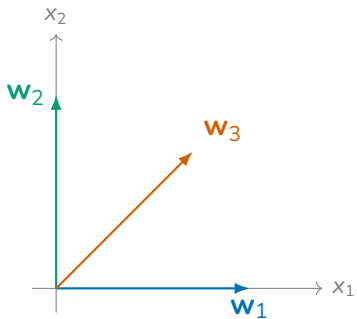


Toy dictionary, with unit-length columns:

$$\mathbf{w}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{w}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{w}_3 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Compare $\mathbf{x} = \mathbf{w}_1$ with $\mathbf{x} = \sqrt{2}\,\mathbf{w}_3$.
Both have $x_1 = 1$, but point in different directions.

Three Feature Directions in Two Dimensions



Toy dictionary, with unit-length columns:

$$\mathbf{w}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{w}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{w}_3 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Compare $\mathbf{x} = \mathbf{w}_1$ with $\mathbf{x} = \sqrt{2}\,\mathbf{w}_3$.
Both have $x_1 = 1$, but point in different directions.

Under a **one-active-feature** assumption, the direction identifies which feature is active.

In a real LM, the dictionary is unknown.
How can we learn directions from many activation vectors?

Sparse Autoencoders: Learn a Dictionary

Collect activation vectors $\mathbf{x} \in \mathbb{R}^d$ at one site in a frozen LM.
Learn a dictionary and sparse codes from these vectors, without human feature labels.



Sparse Autoencoders: Learn a Dictionary

Collect activation vectors $\mathbf{x} \in \mathbb{R}^d$ at one site in a frozen LM.
Learn a dictionary and sparse codes from these vectors, without human feature labels.



$$\mathbf{f} = \mathrm{ReLU}(W_{\mathrm{enc}}(\mathbf{x} - \mathbf{b}_{\mathrm{dec}}) + \mathbf{b}_{\mathrm{enc}}),$$

$$\hat{\mathbf{x}} = W_{\mathrm{dec}}\mathbf{f} + \mathbf{b}_{\mathrm{dec}} = \sum_{i=1}^{M} f_i \mathbf{w}_i + \mathbf{b}_{\mathrm{dec}}.$$

The decoder columns $\mathbf{w}_i$ are candidate feature directions.
Usually $M > d$, but few entries of $\mathbf{f}$ should be nonzero for each input.

Reconstruct Well, Use Few Features

A basic L_1-regularized SAE minimizes

$$\mathbb{E}_{\mathbf{x}}\left[\underbrace{\|\mathbf{x}-\hat{\mathbf{x}}\|_2^2}_{\text{reconstruction error}} + \lambda \underbrace{\|\mathbf{f}\|_1}_{\text{feature activity}}\right], \qquad \lambda > 0.$$

- Reconstruction rewards retaining information from the LM's activations.
- The L_1 penalty encourages sparse feature activity.

Reconstruct Well, Use Few Features

A basic L_1-regularized SAE minimizes

$$\mathbb{E}_{\mathbf{x}}\left[\underbrace{\|\mathbf{x}-\hat{\mathbf{x}}\|_2^2}_{\text{reconstruction error}} + \lambda \underbrace{\|\mathbf{f}\|_1}_{\text{feature activity}}\right], \qquad \lambda > 0.$$

- Reconstruction rewards retaining information from the LM's activations.
- The L_1 penalty encourages sparse feature activity.

Constrain decoder columns, for example $\|\mathbf{w}_i\|_2 = 1$.
Otherwise, scaling $\mathbf{w}_i$ up and f_i down preserves the reconstruction while reducing the sparsity penalty.

Reconstruct Well, Use Few Features

A basic L_1-regularized SAE minimizes

$$\mathbb{E}_{\mathbf{x}}\left[\underbrace{\|\mathbf{x}-\hat{\mathbf{x}}\|_2^2}_{\text{reconstruction error}} + \lambda \underbrace{\|\mathbf{f}\|_1}_{\text{feature activity}}\right], \qquad \lambda > 0.$$

- Reconstruction rewards retaining information from the LM's activations.
- The L_1 penalty encourages sparse feature activity.

Constrain decoder columns, for example $\|\mathbf{w}_i\|_2 = 1$.
Otherwise, scaling $\mathbf{w}_i$ up and f_i down preserves the reconstruction while reducing the sparsity penalty.

A larger λ puts more pressure on sparsity, potentially increasing reconstruction error. This objective does not explicitly reward human-readable meanings.

What the Sparsity Penalty Prefers

Fix our toy dictionary with zero decoder bias. For $\mathbf{x} = (1, 1)^\top$, compare two exact reconstructions using these directions:

	Two active features	**One active feature**
$\mathbf{f}$	$(1, 1, 0)^\top$	$(0, 0, \sqrt{2})^\top$
$W_{\text{dec}}\mathbf{f}$	$\mathbf{w}_1 + \mathbf{w}_2$	$\sqrt{2}\,\mathbf{w}_3$
Reconstruction error	0	0

What the Sparsity Penalty Prefers

Fix our toy dictionary with zero decoder bias. For $\mathbf{x} = (1, 1)^\top$, compare two exact reconstructions using these directions:

	Two active features	**One active feature**
$\mathbf{f}$	$(1, 1, 0)^\top$	$(0, 0, \sqrt{2})^\top$
$W_{\text{dec}}\mathbf{f}$	$\mathbf{w}_1 + \mathbf{w}_2$	$\sqrt{2}\,\mathbf{w}_3$
Reconstruction error	0	0
$\|\mathbf{f}\|_1$	2	$\sqrt{2} \approx 1.41$

The one-feature code has lower L_1 cost among these exact reconstructions.

What the Sparsity Penalty Prefers

Fix our toy dictionary with zero decoder bias. For $\mathbf{x} = (1, 1)^\top$, compare two exact reconstructions using these directions:

	Two active features	**One active feature**
$\mathbf{f}$	$(1, 1, 0)^\top$	$(0, 0, \sqrt{2})^\top$
$W_{\text{dec}}\mathbf{f}$	$\mathbf{w}_1 + \mathbf{w}_2$	$\sqrt{2}\,\mathbf{w}_3$
Reconstruction error	0	0
$\|\mathbf{f}\|_1$	2	$\sqrt{2} \approx 1.41$

The one-feature code has lower L_1 cost among these exact reconstructions.

If features 1 and 2 generated this vector, the preferred code is wrong. Sparsity alone does not guarantee recovery of the original features.

Interpreting and Testing SAE Features

Bricken et al. (2023) report interpretable features in a one-layer Transformer, including features associated with DNA sequences and Base64 text.

For a candidate feature:

1. Inspect high-activation examples and propose an interpretation.
2. Check that interpretation on new examples and near misses.
3. Change its contribution to the LM's activation and measure prediction effects.

From Behavior to a Tested Mechanism

1. **Define the behavior.**
   Retrieve the code 7 from `cat 7 ... cat`.

2. **Propose a computation.**
   Previous-token write $\rightarrow$ key match $\rightarrow$ value copying.

3. **Choose variables to intervene on.**
   Head outputs, residual states, or candidate SAE feature directions.

4. **Test the predicted changes.**
   Change inputs or internal states, then recompute the output.

From Behavior to a Tested Mechanism

1. **Define the behavior.**
   Retrieve the code 7 from `cat 7 ... cat`.
2. **Propose a computation.**
   Previous-token write $\rightarrow$ key match $\rightarrow$ value copying.
3. **Choose variables to intervene on.**
   Head outputs, residual states, or candidate SAE feature directions.
4. **Test the predicted changes.**
   Change inputs or internal states, then recompute the output.

Compare outcomes with the circuit's predictions across inputs and interventions.

Next Lecture

Evaluation and Benchmarking

How do we measure model quality and make reliable comparisons?

Metrics, benchmark reliability, and evaluation of generated text.