Introducing Mechanistic Interpretability:

Demistify black boxes with **Circuit Analaysis**[1] & **Monosemanticity**[2]

J. Setpal

February {1, 8}, 2024

MACHINE LEARNING @ PURDUE

[1] https://transformer-circuits.pub/2021/framework/

[2] https://transformer-circuits.pub/2023/monosemantic-features/

Outline

1. Background & Intuition
2. Transformer Circuit Analysis
3. Towards Monosemanticity

Outline

What is Interpretability?

What is Interpretability?

Interpretability within Machine Learning is the **degree** to which we can understand the **cause** of a decision, and use it to consistently <u>predict the model's prediction</u>.

What is Interpretability?

Interpretability within Machine Learning is the **degree** to which we can understand the **cause** of a decision, and use it to consistently <u>predict the model's prediction</u>.

This is easy for shallow learning.

What is Interpretability?

Interpretability within Machine Learning is the **degree** to which we can understand the **cause** of a decision, and use it to consistently predict the model's prediction.

This is easy for shallow learning. For deep learning however, it is a **lot harder**.

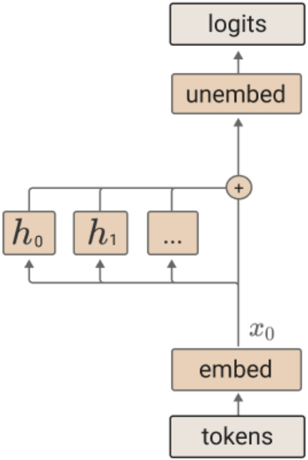
What is Interpretability?

Interpretability within Machine Learning is the **degree** to which we can understand the **cause** of a decision, and use it to consistently predict the model's prediction.

This is easy for shallow learning. For deep learning however, it is a **lot harder**.

Today, we will interpret deep neural networks (transformer).

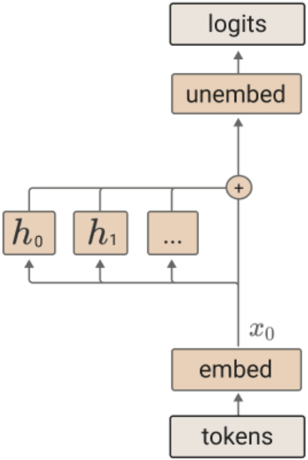
What will we Achieve Today?



Specifically, we'll analyze the 1-layer attention model.

For mathematical simplicity, this model ignores biases, layer-norm and dense layers.

What will we Achieve Today?

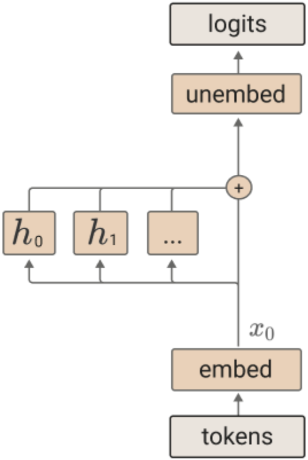


Specifically, we'll analyze the 1-layer attention model.

For mathematical simplicity, this model ignores biases, layer-norm and dense layers.

Why is this useful?

What will we Achieve Today?



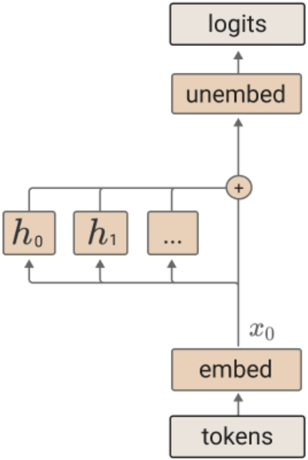
Specifically, we'll analyze the 1-layer attention model.

For mathematical simplicity, this model ignores biases, layer-norm and dense layers.

Why is this useful?
If we are able to *completely understand* a toy model, we can:

- understand why attention works.

What will we Achieve Today?



Specifically, we'll analyze the 1-layer attention model.

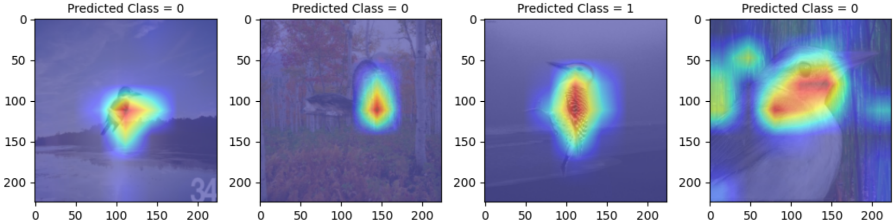
For mathematical simplicity, this model ignores biases, layer-norm and dense layers.

Why is this useful?
If we are able to *completely understand* a toy model, we can:

- understand why attention works.
- observe recurring patterns in complex models.

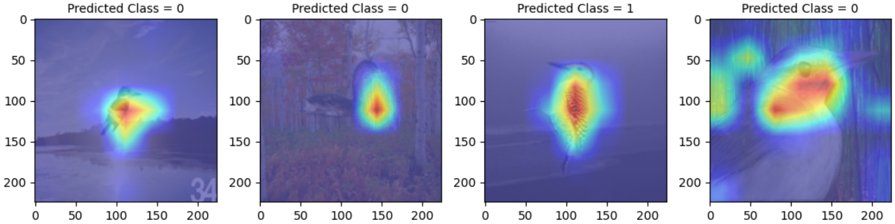
What is *Mechanistic* Interpretability?

Most of interpretability seeks to extract representations from weights:



What is *Mechanistic* Interpretability?

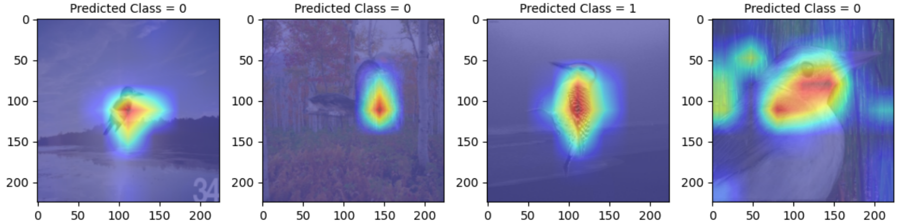
Most of interpretability seeks to extract representations from weights:



Mechanistic Interpretability is a subset of interpretability, that places a focus on **reverse engineering neural networks**.

What is *Mechanistic* Interpretability?

Most of interpretability seeks to extract representations from weights:

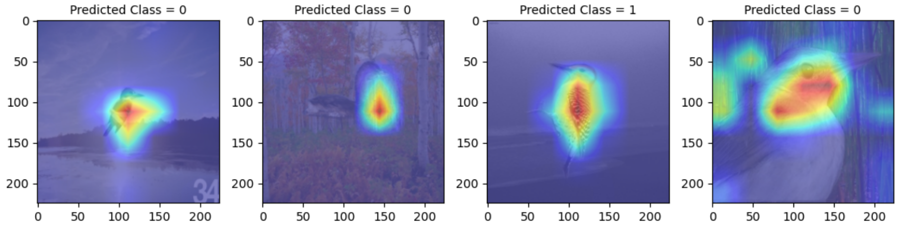


Mechanistic Interpretability is a subset of interpretability, that places a focus on **reverse engineering neural networks**.

It seeks to understand functions that *individual neurons* play in the inference of a neural network.

What is *Mechanistic* Interpretability?

Most of interpretability seeks to extract representations from weights:



Mechanistic Interpretability is a subset of interpretability, that places a focus on **reverse engineering neural networks**.

It seeks to understand functions that *individual neurons* play in the inference of a neural network.

This can subsequently be used to offer high-level explanations for decisions, as well as guarantees during inference.

Outline

Self-Attention Synopsis

n-gram models used the following incorrect assumption:

$$p(x_t | \{x_i\}_{i=1}^{t-1}; \theta) \not\approx p(x_t | x_{t-1}; \theta) \tag{1}$$

Self-Attention Synopsis

n-gram models used the following incorrect assumption:

$$p(x_t | \{x_i\}_{i=1}^{t-1}; \theta) \not\approx p(x_t | x_{t-1}; \theta) \tag{1}$$

Why $\not\approx$?

Self-Attention Synopsis

n-gram models used the following <u>incorrect</u> assumption:

$$p(x_t|\{x_i\}_{i=1}^{t-1};\theta) \not\approx p(x_t|x_{t-1};\theta) \tag{1}$$

Why $\not\approx$? It's because context is important!

Self-Attention Synopsis

n-gram models used the following incorrect assumption:

$$p(x_t | \{x_i\}_{i=1}^{t-1}; \theta) \not\approx p(x_t | x_{t-1}; \theta) \tag{1}$$

Why $\not\approx$? It's because context is important!

But, so is *efficiency*. Self-Attention solves this by effectively creating a **trainable database**.

Self-Attention Synopsis

n-gram models used the following incorrect assumption:

$$p(x_t|\{x_i\}_{i=1}^{t-1};\theta) \not\approx p(x_t|x_{t-1};\theta) \tag{1}$$

Why $\not\approx$? It's because context is important!

But, so is *efficiency*. Self-Attention solves this by effectively creating a **trainable database**.

We query it to subset the important tokens. For $\{x_i\}_{i=1}^{t}$,

$$\alpha_i = \sigma_{softmax}\left(\frac{q_i k_i^T}{\sqrt{d_k}}\right) \tag{2}$$

$$\tag{3}$$

Where q_i, k_i, v_i are each independent parameter matrices.

Self-Attention Synopsis

n-gram models used the following incorrect assumption:

$$p(x_t | \{x_i\}_{i=1}^{t-1}; \theta) \not\approx p(x_t | x_{t-1}; \theta) \tag{1}$$

Why $\not\approx$? It's because context is important!

But, so is *efficiency*. Self-Attention solves this by effectively creating a **trainable database**.

We query it to subset the important tokens. For $\{x_i\}_{i=1}^{t}$,

$$\alpha_i = \sigma_{softmax}\left(\frac{q_i k_i^T}{\sqrt{d_k}}\right) \tag{2}$$

$$h(x) = \sum_{i=1}^{t} \alpha_i v_i \tag{3}$$

Where q_i, k_i, v_i are each independent parameter matrices.

Reframing using Tensorization (1/3)

We can represent attention using **tensor products**:

$$h(x) = (I \otimes W_O) \cdot (A \otimes I) \cdot (I \otimes W_V) \cdot x \tag{4}$$

$$\tag{5}$$

Reframing using Tensorization (1/3)

We can represent attention using **tensor products**:

$$h(x) = (I \otimes W_O) \cdot (A \otimes I) \cdot (I \otimes W_V) \cdot x \tag{4}$$

$$= (A \otimes W_O W_V) \cdot x \tag{5}$$

Reframing using Tensorization (1/3)

We can represent attention using **tensor products**:

$$h(x) = (I \otimes W_O) \cdot (A \otimes I) \cdot (I \otimes W_V) \cdot x \tag{4}$$

$$= (A \otimes W_O W_V) \cdot x \tag{5}$$

The *disjointed* nature of A, $W_O W_V$ tells us a lot!

Reframing using Tensorization (1/3)

We can represent attention using **tensor products**:

$$h(x) = (I \otimes W_O) \cdot (A \otimes I) \cdot (I \otimes W_V) \cdot x \tag{4}$$

$$= (A \otimes W_O W_V) \cdot x \tag{5}$$

The *disjointed* nature of A, $W_O W_V$ tells us a lot!

a. A and $W_O W_V$ are <u>fundamentally independent entities</u>.

Reframing using Tensorization (1/3)

We can represent attention using **tensor products**:

$$h(x) = (I \otimes W_O) \cdot (A \otimes I) \cdot (I \otimes W_V) \cdot x \tag{4}$$
$$= (A \otimes W_O W_V) \cdot x \tag{5}$$

The *disjointed* nature of A, $W_O W_V$ tells us a lot!

a. A and $W_O W_V$ are fundamentally independent entities.
b. A describes which token information moves through, $W_O W_V$ describes which residual subspace to read from and write to.

Reframing using Tensorization (1/3)

We can represent attention using **tensor products**:

$$h(x) = (I \otimes W_O) \cdot (A \otimes I) \cdot (I \otimes W_V) \cdot x \quad (4)$$

$$= (A \otimes W_O W_V) \cdot x \quad (5)$$

The *disjointed* nature of A, $W_O W_V$ tells us a lot!

a. A and $W_O W_V$ are fundamentally independent entities.

b. A describes which token information moves through, $W_O W_V$ describes which residual subspace to read from and write to.

$$MHA(x_0) = x_0 + \sum_{h \in H} (A^h \otimes W_O^h W_V^h) \cdot x_0 \quad (6)$$

Reframing using Tensorization (1/3)

We can represent attention using **tensor products**:

$$h(x) = (I \otimes W_O) \cdot (A \otimes I) \cdot (I \otimes W_V) \cdot x \tag{4}$$
$$= (A \otimes W_O W_V) \cdot x \tag{5}$$

The *disjointed* nature of A, $W_O W_V$ tells us a lot!

a. A and $W_O W_V$ are fundamentally independent entities.
b. A describes which token information moves through, $W_O W_V$ describes which residual subspace to read from and write to.

$$MHA(x_0) = x_0 + \sum_{h \in H} (A^h \otimes W_O^h W_V^h) \cdot x_0 \tag{6}$$

Our final transformer has the following equation:

$$T(t_0) = (I \otimes W_U) \cdot MHA((I \otimes W_E) \cdot t_0) \tag{7}$$

Reframing using Tensorization (1/3)

We can represent attention using **tensor products**:

$$h(x) = (I \otimes W_O) \cdot (A \otimes I) \cdot (I \otimes W_V) \cdot x \tag{4}$$
$$= (A \otimes W_O W_V) \cdot x \tag{5}$$

The *disjointed* nature of A, $W_O W_V$ tells us a lot!

a. A and $W_O W_V$ are fundamentally independent entities.
b. A describes which token information moves through, $W_O W_V$ describes which residual subspace to read from and write to.

$$MHA(x_0) = x_0 + \sum_{h \in H} (A^h \otimes W_O^h W_V^h) \cdot x_0 \tag{6}$$

Our final transformer has the following equation:

$$T(t_0) = (I \otimes W_U) \cdot MHA((I \otimes W_E) \cdot t_0) \tag{7}$$

Why is this important?

Reframing using Tensorization (2/3)

We begin by simplifying to just T:

$$T = (I \otimes W_U) \cdot MHA(I \otimes W_E) \tag{8}$$

$$\tag{9}$$

$$\tag{10}$$

Reframing using Tensorization (2/3)

We begin by simplifying to just T:

$$T = (I \otimes W_U) \cdot MHA(I \otimes W_E) \tag{8}$$

$$= (I \otimes W_U) \cdot (I \otimes W_E + \sum_{h \in H} (A^h \otimes W_O^h W_V^h) \cdot I \otimes W_E) \tag{9}$$

$$\tag{10}$$

Reframing using Tensorization (2/3)

We begin by simplifying to just T:

$$T = (I \otimes W_U) \cdot MHA(I \otimes W_E) \tag{8}$$

$$= (I \otimes W_U) \cdot (I \otimes W_E + \sum_{h \in H} (A^h \otimes W_O^h W_V^h) \cdot I \otimes W_E) \tag{9}$$

$$T = W_U W_E + \sum_{h \in H} (A^h \otimes W_U W_O^h W_V^h W_E) \tag{10}$$

Reframing using Tensorization (2/3)

We begin by simplifying to just T:

$$T = (I \otimes W_U) \cdot MHA(I \otimes W_E) \tag{8}$$

$$= (I \otimes W_U) \cdot (I \otimes W_E + \sum_{h \in H} (A^h \otimes W_O^h W_V^h) \cdot I \otimes W_E) \tag{9}$$

$$T = W_U W_E + \sum_{h \in H} (A^h \otimes W_U W_O^h W_V^h W_E) \tag{10}$$

Here's the breakdown:

a. $W_U W_E$ approximate bigram statistics.

Reframing using Tensorization (2/3)

We begin by simplifying to just T:

$$T = (I \otimes W_U) \cdot MHA(I \otimes W_E) \tag{8}$$

$$= (I \otimes W_U) \cdot (I \otimes W_E + \sum_{h \in H} (A^h \otimes W_O^h W_V^h) \cdot I \otimes W_E) \tag{9}$$

$$T = W_U W_E + \sum_{h \in H} (A^h \otimes W_U W_O^h W_V^h W_E) \tag{10}$$

Here's the breakdown:

a. $W_U W_E$ approximate bigram statistics.

b. A^h dictates where the attention heads attend.

Reframing using Tensorization (2/3)

We begin by simplifying to just T:

$$T = (I \otimes W_U) \cdot MHA(I \otimes W_E) \tag{8}$$

$$= (I \otimes W_U) \cdot (I \otimes W_E + \sum_{h \in H} (A^h \otimes W_O^h W_V^h) \cdot I \otimes W_E) \tag{9}$$

$$T = W_U W_E + \sum_{h \in H} (A^h \otimes W_U W_O^h W_V^h W_E) \tag{10}$$

Here's the breakdown:

a. $W_U W_E$ approximate bigram statistics.

b. A^h dictates where the attention heads attend.

c. $W_U W_O^h W_V^h W_E$ describes the **behavior of logits if we attend to a given token**.

Reframing using Tensorization (2/3)

We begin by simplifying to just T:

$$T = (I \otimes W_U) \cdot MHA(I \otimes W_E) \tag{8}$$

$$= (I \otimes W_U) \cdot (I \otimes W_E + \sum_{h \in H} (A^h \otimes W_O^h W_V^h) \cdot I \otimes W_E) \tag{9}$$

$$T = W_U W_E + \sum_{h \in H} (A^h \otimes W_U W_O^h W_V^h W_E) \tag{10}$$

Here's the breakdown:

a. $W_U W_E$ approximate bigram statistics.

b. A^h dictates where the attention heads attend.

c. $W_U W_O^h W_V^h W_E$ describes the **behavior of logits if we attend to a given token**.

Observation: The equation is linear, <u>if we fix attention patterns</u>.

Reframing using Tensorization (3/3)

Finally, let's also unpack attention in **tensor-product form**.

Reframing using Tensorization (3/3)

Finally, let's also unpack attention in **tensor-product form**.

First, we can display key-value matrix operations:

$$q_i = (I \otimes W_Q W_E) \cdot t_0 \tag{11}$$

$$k_i = (I \otimes W_K W_E) \cdot t_0 \tag{12}$$

Reframing using Tensorization (3/3)

Finally, let's also unpack attention in **tensor-product form**.

First, we can display key-value matrix operations:

$$q_i = (I \otimes W_Q W_E) \cdot t_0 \tag{11}$$

$$k_i = (I \otimes W_K W_E) \cdot t_0 \tag{12}$$

And then apply them to unnormalized[3] attention:

$$A = \sigma_{softmax}\left([q_i k_j^T]_{i,j}\right) \tag{13}$$

$$= \sigma_{softmax}\left(t_0^T \cdot (I \otimes W_E^T W_Q^T) \cdot (I \otimes W_K W_E) \cdot t_0\right) \tag{14}$$

$$= \sigma_{softmax}\left(t_0^T \cdot W_E^T W_Q^T W_K W_E \cdot t_0\right) \tag{15}$$

[3] to ease computation.

Unravelling QK, OV Circuits (1/3)

Here's the two tensor equations combined:

$$T = W_U W_E + \sum_{h \in H} (A^h \otimes W_U W_O^h W_V^h W_E) \tag{10}$$

$$A = \sigma_{softmax} \left(t_0^T \cdot W_E^T W_Q^T W_K W_E \cdot t_0 \right) \tag{15}$$

Unravelling QK, OV Circuits (1/3)

Here's the two tensor equations combined:

$$T = W_U W_E + \sum_{h \in H} (A^h \otimes W_U W_O^h W_V^h W_E) \tag{10}$$

$$A = \sigma_{softmax}\left(t_0^T \cdot W_E^T W_Q^T W_K W_E \cdot t_0\right) \tag{15}$$

Q: Is there anything interesting about these two? (similarities, differences)

Unravelling QK, OV Circuits (1/3)

Here's the two tensor equations combined:

$$T = W_U W_E + \sum_{h \in H} (A^h \otimes W_U W_O^h W_V^h W_E) \tag{10}$$

$$A = \sigma_{softmax} \left(t_0^T \cdot W_E^T W_Q^T W_K W_E \cdot t_0 \right) \tag{15}$$

Q: Is there anything interesting about these two? (similarities, differences)

Here's my observations:

a. It's a much *simpler* recomposition of feedforward inference.

Unravelling QK, OV Circuits (1/3)

Here's the two tensor equations combined:

$$T = W_U W_E + \sum_{h \in H} (A^h \otimes W_U W_O^h W_V^h W_E) \tag{10}$$

$$A = \sigma_{softmax}\left(t_0^T \cdot W_E^T W_Q^T W_K W_E \cdot t_0\right) \tag{15}$$

Q: Is there anything interesting about these two? (similarities, differences)

Here's my observations:

a. It's a much *simpler* recomposition of feedforward inference.
b. A is the *only* non-linear operation.

Unravelling QK, OV Circuits (1/3)

Here's the two tensor equations combined:

$$T = W_U W_E + \sum_{h \in H} (A^h \otimes W_U W_O^h W_V^h W_E) \tag{10}$$

$$A = \sigma_{softmax} \left(t_0^T \cdot W_E^T W_Q^T W_K W_E \cdot t_0 \right) \tag{15}$$

Q: Is there anything interesting about these two? (similarities, differences)

Here's my observations:

a. It's a much *simpler* recomposition of feedforward inference.
b. A is the *only* non-linear operation.
c. A **learns independently** from the rest of the tensor equation.

Unravelling QK, OV Circuits (1/3)

Here's the two tensor equations combined:

$$T = W_U W_E + \sum_{h \in H} (A^h \otimes W_U W_O^h W_V^h W_E) \tag{10}$$

$$A = \sigma_{softmax} \left(t_0^T \cdot W_E^T W_Q^T W_K W_E \cdot t_0 \right) \tag{15}$$

Q: Is there anything interesting about these two? (similarities, differences)

Here's my observations:

a. It's a much *simpler* recomposition of feedforward inference.
b. A is the *only* non-linear operation.
c. A **learns independently** from the rest of the tensor equation.

However, we're still missing one.

Unravelling QK, OV Circuits (2/3)

Importantly, both equations have $(|voc|, |voc|)$ size matrices:

$$T = W_U W_E + \sum_{h \in H} (A^h \otimes W_U W_O^h W_V^h W_E) \tag{10}$$

$$A = \sigma_{softmax} \left(t_0^T \cdot W_E^T W_Q^T W_K W_E \cdot t_0 \right) \tag{15}$$

These chained tensor operations are our **circuits**, and lie at the heart of the transformer architecture.

Unravelling QK, OV Circuits (2/3)

Importantly, both equations have $(|voc|, |voc|)$ size matrices:

$$T = W_U W_E + \sum_{h \in H} (A^h \otimes W_U W_O^h W_V^h W_E) \tag{10}$$

$$A = \sigma_{softmax}\left(t_0^T \cdot W_E^T W_Q^T W_K W_E \cdot t_0\right) \tag{15}$$

These chained tensor operations are our **circuits**, and lie at the heart of the transformer architecture.

a. The **Output-Value(OV) Circuit** $W_U W_O^h W_V^h W_E$: determines how attending to a token affects logits.

Unravelling QK, OV Circuits (2/3)

Importantly, both equations have $(|voc|, |voc|)$ size matrices:

$$T = W_U W_E + \sum_{h \in H} (A^h \otimes W_U W_O^h W_V^h W_E) \tag{10}$$

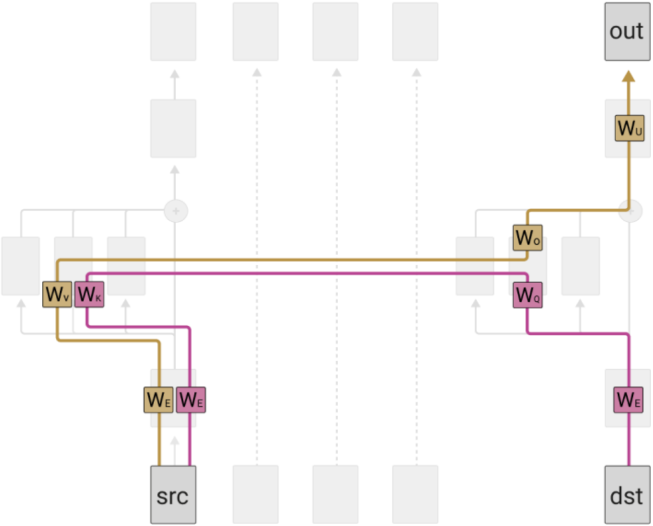
$$A = \sigma_{softmax}\left(t_0^T \cdot W_E^T W_Q^T W_K W_E \cdot t_0\right) \tag{15}$$

These chained tensor operations are our **circuits**, and lie at the heart of the transformer architecture.

a. The **Output-Value(OV) Circuit** $W_U W_O^h W_V^h W_E$: determines how attending to a token affects logits.

b. The **Query-Key(QK) Circuit** $W_E^T W_Q^T W_K W_E$: determines which tokens to attend to.

Unravelling QK, OV Circuits (3/3)



Interpretation as Skip-Trigrams

We can think through inference procedure with *single* source token.[4]

[4] for simplicity.

Interpretation as Skip-Trigrams

We can think through inference procedure with *single* source token.[4]

From there, we look at the largest QK and OV entries.

Some examples of large entries QK/OV circuit

Source Token	Destination Token	Out Token	Example Skip Tri-grams
“ perfect”	“ are”, “ looks”, “ is”, “ provides”	“ perfect”, “ super”, “ absolute”, “ pure”	“ perfect... are perfect”, “ perfect... looks super”
“ large”	“ contains”, “ using”, “ specify”, “ contain”	“ large”, “ small”, “ very”, “ huge”	“ large... using large”, “ large... contains small”
“ two”	“ One”, “\n ”, “ has”, “\r\n ”, “One”	“ two”, “ three”, “ four”, “ five”, “ one”	“ two... One two”, “ two... has three”
“lambda”	“ $\\”, “}{\\”, “+\\”, “(\\”, “ ${\\”	“lambda”, “sorted”, “ lambda”, “operator”	“lambda... $\\lambda”, “lambda... +\\lambda”
“nbsp”	“&”, “ \"&”, “}&”, “>&”, “=&”	“nbsp”, “01”, “gt”, “00012”, “nbs”, “quot”	“nbsp... ”, “nbsp... > ”
“Great”	“The”, “ The”, “ the”, “ contains”, “ /”	“ Great”, “ great”, “ poor”, “ Every”	“Great... The Great”, “Great... the great”

[4] for simplicity.

Eigenvalue Analysis

Most of the prominent behaviours include copying. We can identify this using **eigenvalue analysis**.

Eigenvalue Analysis

Most of the prominent behaviours include copying. We can identify this using **eigenvalue analysis**. Recall from the definition of eigenvectors,

$$Wv = \lambda v; \lambda \in \mathbb{C} \tag{16}$$

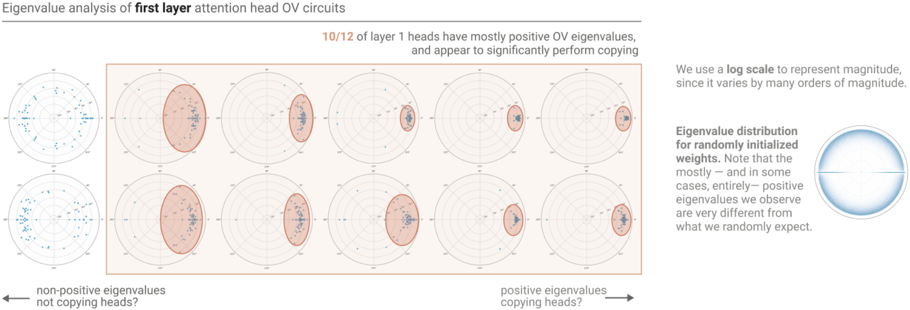
This is useful when we map a vector space upon itself.

Eigenvalue Analysis

Most of the prominent behaviours include copying. We can identify this using **eigenvalue analysis**. Recall from the definition of eigenvectors,

$$Wv = \lambda v; \lambda \in \mathbb{C} \tag{16}$$

This is useful when we map a vector space upon itself.



Importantly, note that positive eigenvalues mean they are copying 'on average', and are not definitive.

Outline

Problem Setup

Q: Is anyone familiar with the the curse of dimensionality?

Problem Setup

Q: Is anyone familiar with the the curse of dimensionality?
A: For NNs, basically latent space $\propto |\text{layers}|^c$.

This makes them tough to analyze at scale.

Problem Setup

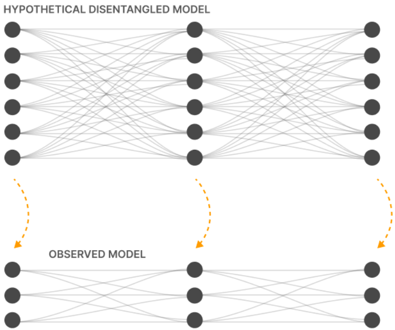
Q: Is anyone familiar with the the curse of dimensionality?
A: For NNs, basically latent space $\propto |\text{layers}|^c$.

This makes them tough to analyze at scale. In addition, models are *incredibly efficient* at information compression.

Problem Setup

Q: Is anyone familiar with the the curse of dimensionality?
A: For NNs, basically latent space $\propto |\text{layers}|^c$.

This makes them tough to analyze at scale. In addition, models are *incredibly efficient* at information compression.

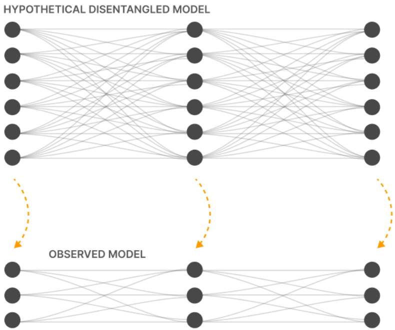


This is **superposition**.

Problem Setup

Q: Is anyone familiar with the the curse of dimensionality?
A: For NNs, basically latent space $\propto |\text{layers}|^c$.

This makes them tough to analyze at scale. In addition, models are *incredibly efficient* at information compression.



This is **superposition**.

When we perform an indvidual analysis of neurons, it fires for unrelated concepts.

This is **polysemanticity**.

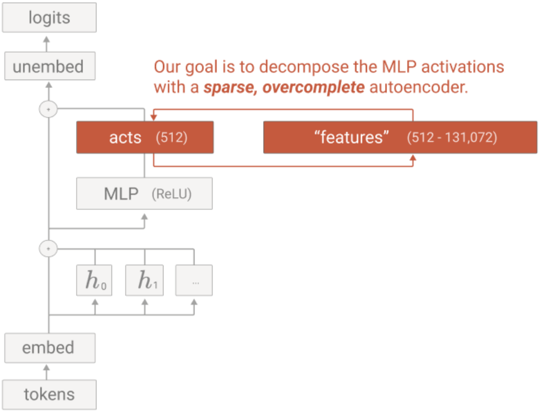
Updated Architecture

Previously, we used an **attention-only** model, since the MLP was too hard to analyze mathematically.

Updated Architecture

Previously, we used an **attention-only** model, since the MLP was too hard to analyze mathematically.

Let's instead analyze the following architecture *empirically*:



Training Setup

	Transformer	**Sparse Autoencoder**
Layers	1 Attention Block 1 MLP Block	1 ReLU 1 Linear
MLP Size	512	$512 \times f \in \{1, \ldots, 256\}$[5]
Dataset	The Pile (100B tokens)	Activations (8B samples)
Loss	Autoregressive Log-Likelihood	$L2$ Reconstruction $L1$ on hidden-layer activation

[5] $f = 8$ for our analysis

Training Setup

	Transformer	**Sparse Autoencoder**
Layers	1 Attention Block 1 MLP Block	1 ReLU 1 Linear
MLP Size	512	$512 \times f \in \{1, \ldots, 256\}$[5]
Dataset	The Pile (100B tokens)	Activations (8B samples)
Loss	Autoregressive Log-Likelihood	$L2$ Reconstruction $L1$ on hidden-layer activation

Objective: *polysemantic activations* $\xrightarrow{Tr}$ **monosemantic features**.

[5] $f = 8$ for our analysis

Training Setup

	Transformer	Sparse Autoencoder
Layers	1 Attention Block 1 MLP Block	1 ReLU 1 Linear
MLP Size	512	$512 \times f \in \{1, \ldots, 256\}$[5]
Dataset	The Pile (100B tokens)	Activations (8B samples)
Loss	Autoregressive Log-Likelihood	$L2$ Reconstruction $L1$ on hidden-layer activation

Objective: *polysemantic activations* $\xrightarrow{Tr}$ **monosemantic features**.

The sparse, overcomplete autoencoder is trained against this objective.

1. **Sparse** because we constrain activations (L1 penalty).
2. **Overcomplete** because the hidden layer exceeds the input dimension.

[5] $f = 8$ for our analysis

Sparse Dictionary Learning

Given $X := \{x^j\}_{j=1}^K; x_i \in \mathbb{R}^d$, we wish to find $D \in \mathbb{R}^{d \times n}, R \in \mathbb{R}^n$ s.t:

$$||X - DR||_F^2 \approx 0 \tag{17}$$

Sparse Dictionary Learning

Given $X := \{x^j\}_{j=1}^{K}; x_i \in \mathbb{R}^d$, we wish to find $D \in \mathbb{R}^{d \times n}, R \in \mathbb{R}^n$ s.t:

$$||X - DR||_F^2 \approx 0 \tag{17}$$

We can motivate our objective transformation by linear factorization:

$$x^j \approx b + \sum_i f_i(x^j) d_i \tag{18}$$

$$f_i = \sigma_{ReLU}(W_E(x - b_D) + b_E) \tag{19}$$

where d_i is the 'feature direction' represented as columns of the W_D.

Sparse Dictionary Learning

Given $X := \{x^j\}_{j=1}^K; x_i \in \mathbb{R}^d$, we wish to find $D \in \mathbb{R}^{d \times n}, R \in \mathbb{R}^n$ s.t:

$$||X - DR||_F^2 \approx 0 \tag{17}$$

We can motivate our objective transformation by linear factorization:

$$x^j \approx b + \sum_i f_i(x^j) d_i \tag{18}$$

$$f_i = \sigma_{ReLU}(W_E(x - b_D) + b_E) \tag{19}$$

where d_i is the 'feature direction' represented as columns of the W_D.

Some interesting implementation notes:

a. Training data $\propto n$(interpretable features).

Sparse Dictionary Learning

Given $X := \{x^j\}_{j=1}^{K}; x_i \in \mathbb{R}^d$, we wish to find $D \in \mathbb{R}^{d \times n}, R \in \mathbb{R}^n$ s.t:

$$||X - DR||_F^2 \approx 0 \tag{17}$$

We can motivate our objective transformation by linear factorization:

$$x^j \approx b + \sum_i f_i(x^j) d_i \tag{18}$$

$$f_i = \sigma_{ReLU}(W_E(x - b_D) + b_E) \tag{19}$$

where d_i is the 'feature direction' represented as columns of the W_D.

Some interesting implementation notes:

a. Training data $\propto n$(interpretable features).
b. Tying b_D before the encoder and after the decoder improves performance.

Sparse Dictionary Learning

Given $X := \{x^j\}_{j=1}^{K}; x_i \in \mathbb{R}^d$, we wish to find $D \in \mathbb{R}^{d \times n}, R \in \mathbb{R}^n$ s.t:

$$||X - DR||_F^2 \approx 0 \tag{17}$$

We can motivate our objective transformation by linear factorization:

$$x^j \approx b + \sum_i f_i(x^j) d_i \tag{18}$$

$$f_i = \sigma_{ReLU}(W_E(x - b_D) + b_E) \tag{19}$$

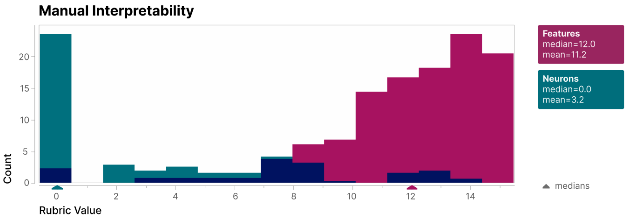
where d_i is the 'feature direction' represented as columns of the W_D.

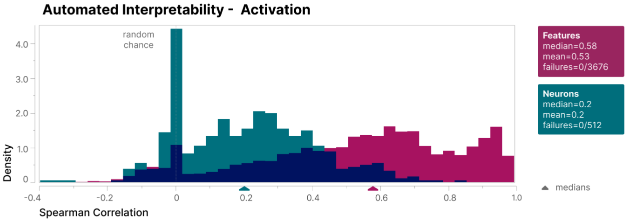
Some interesting implementation notes:

a. Training data $\propto n$(interpretable features).
b. Tying b_D before the encoder and after the decoder improves performance.
c. Dead neurons are periodically *resampled* to improve feature representations.

Evaluating Interpretability

Reliable evaluations on interpretability were scored based on a rubric:





Features were found to be interpretable when score > 8.

Analyzing Arabic Features

Let's analyze feature **A/1/3450**, that fires on Arabic Script.

Analyzing Arabic Features

Let's analyze feature **A/1/3450**, that fires on Arabic Script.

This is effectively *invisible* when viewed through the polysemantic model!

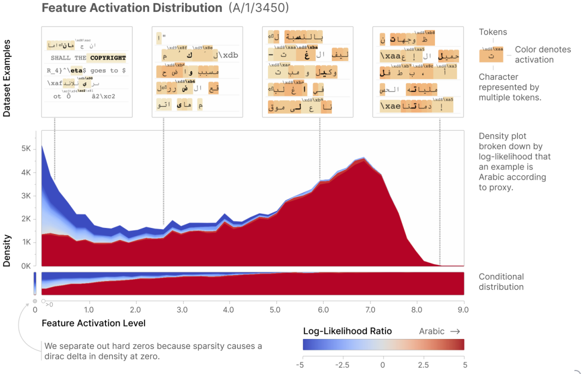
Let's analyze feature **A/1/3450**, that fires on Arabic Script.

This is effectively *invisible* when viewed through the polysemantic model!

We can evaluate each token using the log-likelihood ratio:

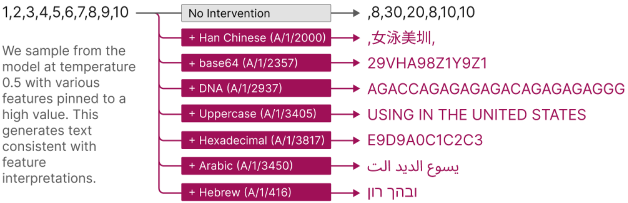
$$LL(t) = \log\left(P(t|\text{Arabic})/P(t)\right) \tag{20}$$

Despite representing 0.13% of training data, arabic script makes up **81% of active tokens**:



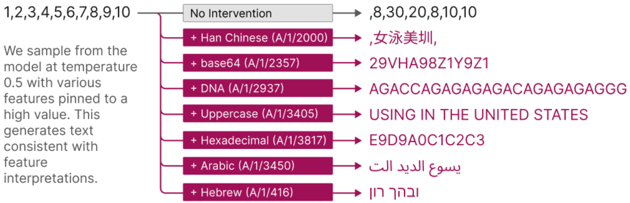
Pinned Feature Sampling

They can be used to steer generation.



Pinned Feature Sampling

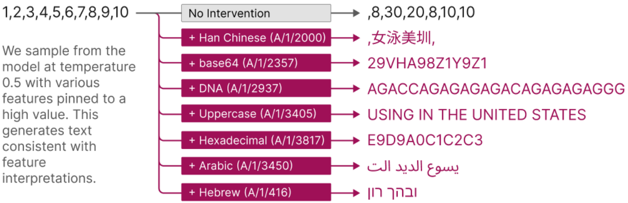
They can be used to steer generation.



Approach: Set high values of features demonstrating desired behaviors, and then sample from the model.

Pinned Feature Sampling

They can be used to steer generation.



Approach: Set high values of features demonstrating desired behaviors, and then sample from the model.

We observe that interpreted features are actively used by the model.

Finite State Automaton

A unique feature of features is their role as **finite state automaton**.

Finite State Automaton

A unique feature of features is their role as **finite state automaton**.

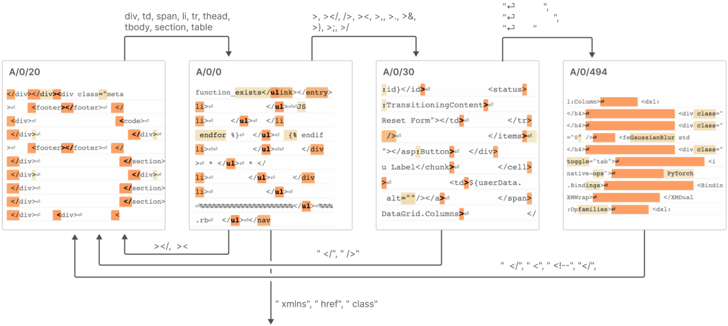
Unlike circuits, these work by daisy chaining features that increase the probability of another feature firing in a loop-like fashion.

Finite State Automaton

A unique feature of features is their role as **finite state automaton**.

Unlike circuits, these work by daisy chaining features that increase the probability of another feature firing in a loop-like fashion.

These present partial explanations of **memorizations** within transformers:



Reimplementation

If you can view this screen, I am making a mistake.

Thank you!

Have an awesome rest of your day!

Slides: `https://cs.purdue.edu/homes/jsetpal/slides/mechinterp.pdf`