A Practical Guide to Mechanistic Interpretability:

Demistifying black boxes with **Sparse AutoEncoders**[1][2][3]

J. Setpal

February 13, 2025

1 https://transformer-circuits.pub/2023/monosemantic-features/

2 https://arxiv.org/abs/2404.16014

3 https://www.arena.education/

Outline

Outline

1. Background & Intuition
2. Sparse AutoEncoders
3. Applications & Practical Detail

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 predict the model's prediction.

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.

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?

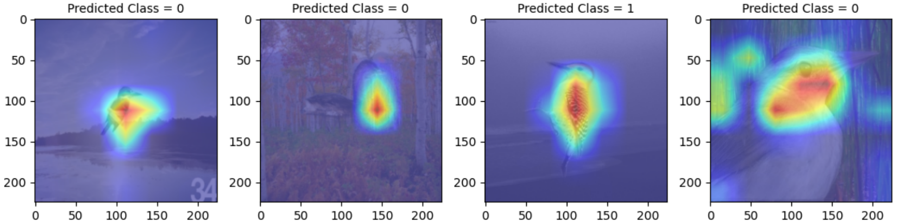
Interpretability within Machine Learning is the **degree** to which we can understand the **cause** of a decision, and use it to consistently predict the model's prediction.

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

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

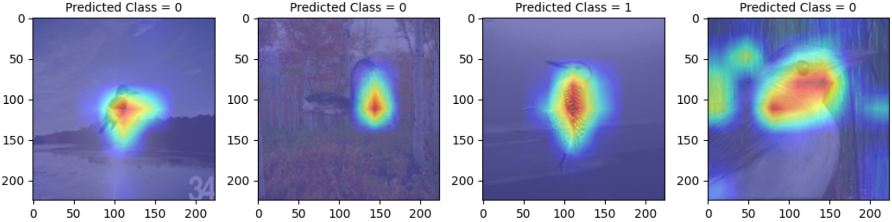
What is *Mechanistic* Interpretability?

Most of interpretability seeks to extract representations from weights:



What is *Mechanistic* Interpretability?

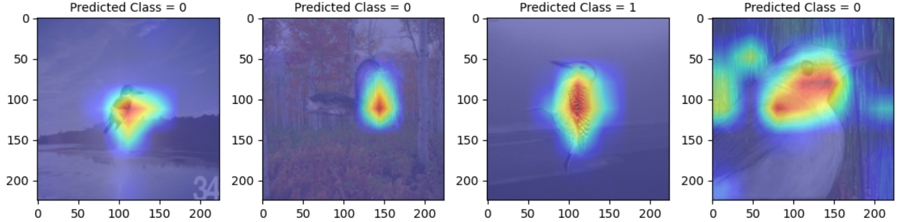
Most of interpretability seeks to extract representations from weights:



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:

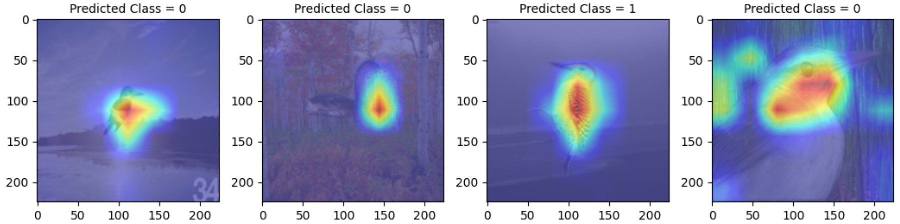


Mechanistic Interpretability is a subset of interpretability, that places a focus on **reverse engineering neural networks**.

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.

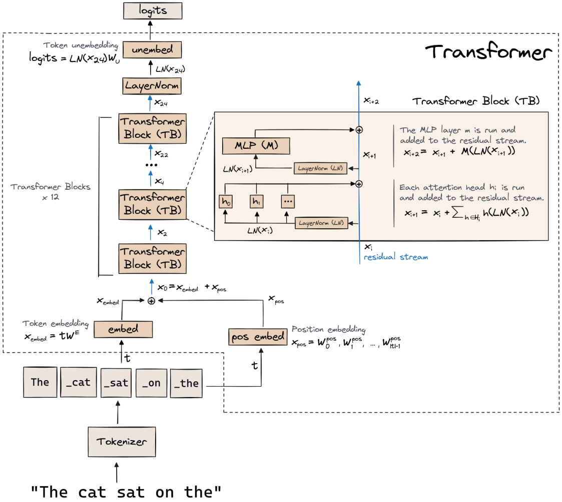
This can subsequently be used to offer high-level explanations for decisions, as well as guarantees during inference.

Outline

Transformers Mini-Review

Crucial Aside: Treat residual connections as "memory"; all other layers "read from", "process", and "write-to" memory!

Crucial Aside: Treat residual connections as "memory"; all other layers "read from", "process", and "write-to" memory!



Problem Setup

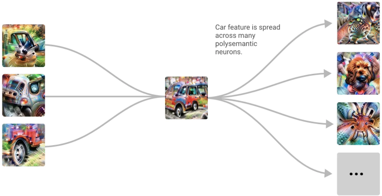
Q: Now, given the framework we just discussed, what stops from directly analyzing MLP activations?

Problem Setup

Q: Now, given the framework we just discussed, what stops from directly analyzing MLP activations?
A: Enter **polysemanticity** & **superposition**.

Problem Setup

Q: Now, given the framework we just discussed, what stops from directly analyzing MLP activations?
A: Enter **polysemanticity** & **superposition**.



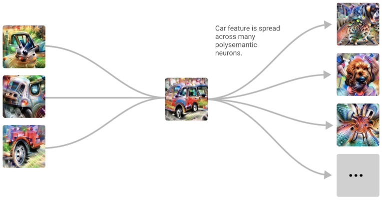
When we perform an indvidual analysis of neurons, we observe it fires for unrelated concepts.

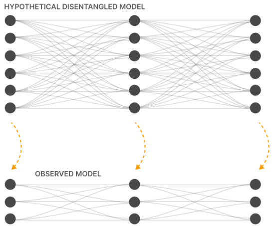
This is **polysemanticity**.

Problem Setup

Q: Now, given the framework we just discussed, what stops from directly analyzing MLP activations?

A: Enter **polysemanticity** & **superposition**.





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

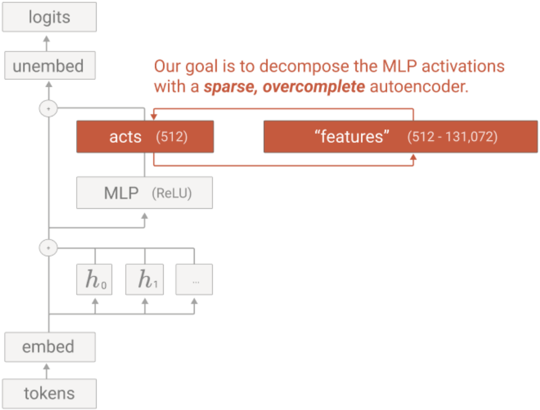
This is **polysemanticity**.

We observe learning compresses larger models to smaller footprints using denser parameters.

This is **superposition**.

Analytical Setup

We will explore the following setup:



Training Setup

	Transformer	Sparse Autoencoder
Layers	1 Attention Block 1 MLP Block	1 ReLU 1 Linear
MLP Size	512	$512 \times f \in \{1, \ldots, 256\}$[4]
Dataset	The Pile (100B tokens)	Activations (8B samples)
Loss	Autoregressive Log-Likelihood	$L2$ Reconstruction $L1$ on hidden-layer activation

[4] $f = 8$ for our analysis

Training Setup

	Transformer	Sparse Autoencoder
Layers	1 Attention Block 1 MLP Block	1 ReLU 1 Linear
MLP Size	512	$512 \times f \in \{1, \ldots, 256\}$[4]
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**.

[4] $f = 8$ for our analysis

Training Setup

	Transformer	Sparse Autoencoder
Layers	1 Attention Block 1 MLP Block	1 ReLU 1 Linear
MLP Size	512	$512 \times f \in \{1, \ldots, 256\}$[4]
Dataset	The Pile (100B tokens)	Activations (8B samples)
Loss	Autoregressive Log-Likelihood	$L2$ Reconstruction $L1$ on hidden-layer activation

Objective: *polysemantic activations* $\overset{Tr}{\rightarrow}$ **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.

[4] $f = 8$ 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:

$$||X - DR||_F^2 \approx 0 \tag{1}$$

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:

$$||X - DR||_F^2 \approx 0 \tag{1}$$

We can motivate our objective transformation by linear factorization:

$$x^j \approx b_D + \sum_i f_i(x^j) d_i \tag{2}$$

$$f_i = \sigma_{ReLU}(W_E(x - b_D) + b_E) \tag{3}$$

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:

$$||X - DR||_F^2 \approx 0 \tag{1}$$

We can motivate our objective transformation by linear factorization:

$$x^j \approx b_D + \sum_i f_i(x^j) d_i \tag{2}$$

$$f_i = \sigma_{ReLU}(W_E(x - b_D) + b_E) \tag{3}$$

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:

$$||X - DR||_F^2 \approx 0 \tag{1}$$

We can motivate our objective transformation by linear factorization:

$$x^j \approx b_D + \sum_i f_i(x^j) d_i \tag{2}$$

$$f_i = \sigma_{ReLU}(W_E(x - b_D) + b_E) \tag{3}$$

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:

$$||X - DR||_F^2 \approx 0 \tag{1}$$

We can motivate our objective transformation by linear factorization:

$$x^j \approx b_D + \sum_i f_i(x^j) d_i \tag{2}$$

$$f_i = \sigma_{ReLU}(W_E(x - b_D) + b_E) \tag{3}$$

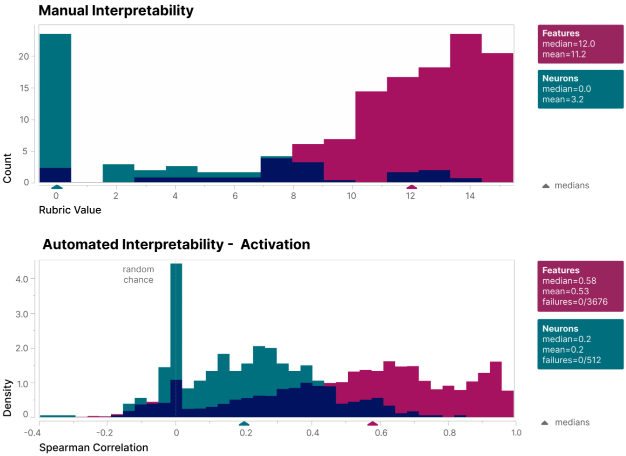
where d_i is the 'feature direction' represented as columns of the W_D.

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!

Analyzing Arabic Features

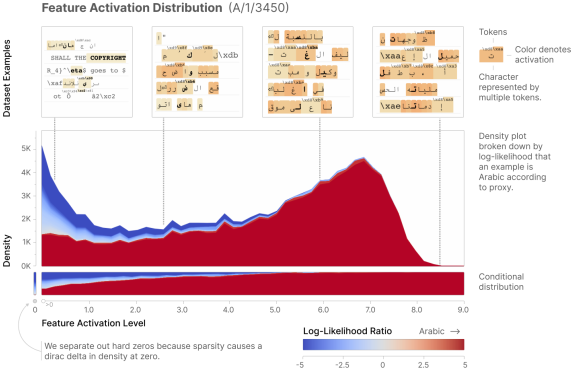
Let's analyze feature **A/1/3450**, that fires on Arabic Script.

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

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

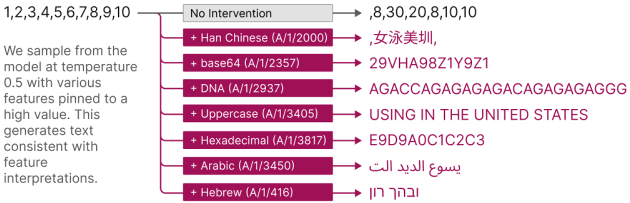
$$LL(t) = \log\left(P(t|\text{Arabic})/P(t)\right) \quad (4)$$

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



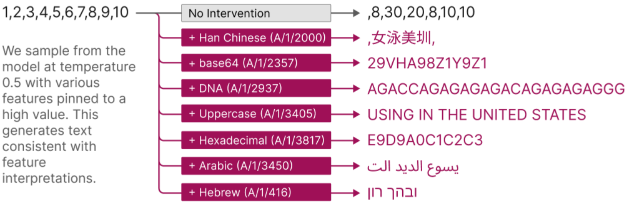
Pinned Feature Sampling

They can be used to steer generation.



Pinned Feature Sampling

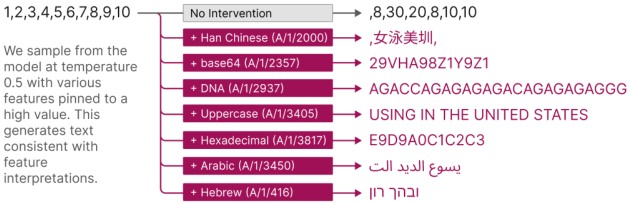
They can be used to steer generation.



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**.

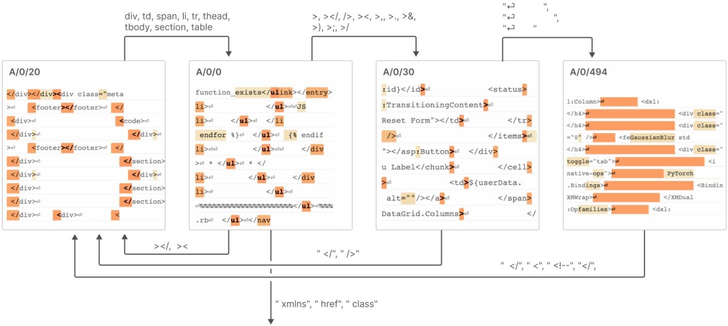
Unlike circuits, these work by daisy chaining features that increase the probability of another feature firing in a loop-like fashion.

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:



Modern (Gated) SAEs (1/2)

Quick review of the structure of the original SAE:

$$f(x) := \sigma_{\mathrm{ReLU}}(W_E(x - b_D) + b_E) \tag{5}$$

$$\hat{x}(f(x)) := W_D f(x) + b_D \tag{6}$$

$$\min_{W_E, W_D, b_D, b_e} \mathcal{L}(x) = \min_{W_E, W_D, b_D, b_e} \underbrace{\|x - \hat{x}(f(x))\|_2^2}_{\text{reconstruction error}} + \underbrace{\lambda \|f(x)\|_1}_{\text{sparsity penalty}} \tag{7}$$

Modern (Gated) SAEs (1/2)

Quick review of the structure of the original SAE:

$$f(x) := \sigma_{\text{ReLU}}(W_E(x - b_D) + b_E) \tag{5}$$

$$\hat{x}(f(x)) := W_D f(x) + b_D \tag{6}$$

$$\min_{W_E, W_D, b_D, b_e} \mathcal{L}(x) = \min_{W_E, W_D, b_D, b_e} \underbrace{\|x - \hat{x}(f(x))\|_2^2}_{\text{reconstruction error}} + \underbrace{\lambda \|f(x)\|_1}_{\text{sparsity penalty}} \tag{7}$$

We evaluate the SAE by how much loss increases when **activations are substituted with the reconstructions** during forward pass.

Modern (Gated) SAEs (1/2)

Quick review of the structure of the original SAE:

$$f(x) := \sigma_{\text{ReLU}}(W_E(x - b_D) + b_E) \tag{5}$$

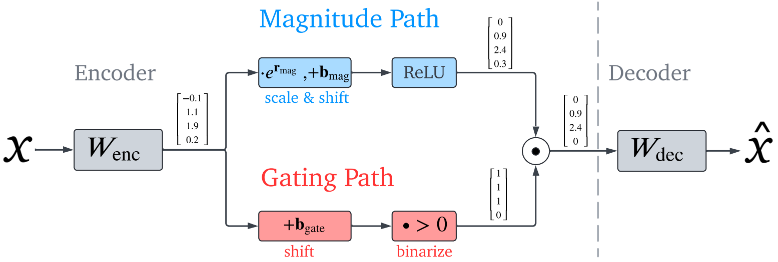
$$\hat{x}(f(x)) := W_D f(x) + b_D \tag{6}$$

$$\min_{W_E, W_D, b_D, b_e} \mathcal{L}(x) = \min_{W_E, W_D, b_D, b_e} \underbrace{\|x - \hat{x}(f(x))\|_2^2}_{\text{reconstruction error}} + \underbrace{\lambda \|f(x)\|_1}_{\text{sparsity penalty}} \tag{7}$$

We evaluate the SAE by how much loss increases when **activations are substituted with the reconstructions** during forward pass.

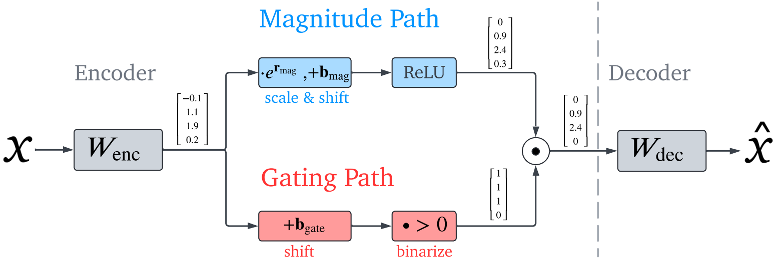
Observation: $\|\cdot\|_1$ motivates *shrinkage* – minimizing sparsity is "easier" than reconstructing sparse features, and motivates under-activation of reconstructed features.

Idea: Let's disentangle feature importance with feature existance:



Modern (Gated) SAEs (2/2)

Idea: Let's disentangle feature importance with feature existance:

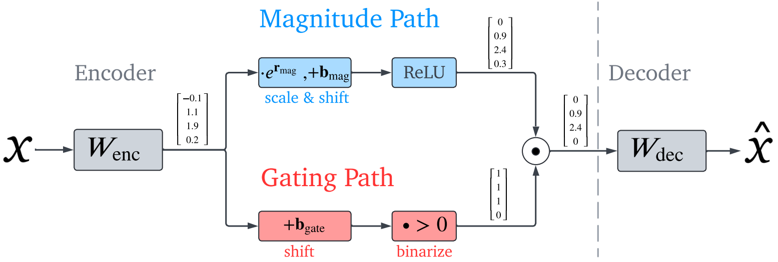


For this, the authors also define the following loss function:

$$\mathcal{L}(x) := \|x - \hat{x}(f(x))\|_2^2 + \underbrace{\lambda \|\sigma_{\text{ReLU}}(f_g(x))\|_1}_{f_g := \text{ pre-activation}} + \|x - \hat{x}(\sigma_{\text{ReLU}}(f_g(x)))\|_2^2$$

Modern (Gated) SAEs (2/2)

Idea: Let's disentangle feature importance with feature existance:



For this, the authors also define the following loss function:

$$\mathcal{L}(x) := \|x - \hat{x}(f(x))\|_2^2 + \underbrace{\lambda\|\sigma_{\text{ReLU}}(f_g(x))\|_1}_{f_g := \text{ pre-activation}} + \|x - \hat{x}(\sigma_{\text{ReLU}}(f_g(x)))\|_2^2$$

Finally, they also use weight-tying to reduce parameter explosion.

Outline

Training SAEs

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

Dashboard Interpretation

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

Feature Steering with SAEs

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

Thank you!

Have an awesome rest of your day!

Slides: `https://jinen.setpal.net/slides/sae.pdf`