Sparse Crosscoders

for Cross-Layer Features and Model Diffing

Lindsey, Templeton, Marcus, Conerly, Batson, Olah

Anthropic · Transformer Circuits Thread · Oct 2024

PRESENTED BY

Qiaoyu Yang · Mellon Zhang

OVERVIEW

Roadmap

PART 1

Setup & architecture

- Superposition & SAE/transcoder recap
- Why crosscoders — cross-layer superposition, diffing
- Architecture, variants, loss function

PART 2

Experiments & discussion

- Cross-layer features experiments
- Circuit simplification (theory)
- Model diffing, limitations, open questions

Three applications

1 **Cross-layer features**

Features that span multiple layers

2 **Circuit simplification**

Collapse duplicate persistent features

3 **Model diffing**

Shared dictionary across two models

BACKGROUND

Superposition: features as directions, not neurons

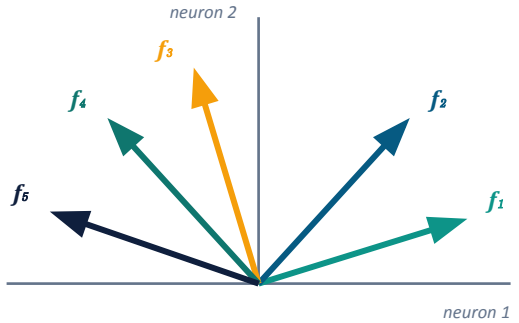
THE CORE IDEA

Neural networks represent **more features than they have neurons** by allowing features to be **non-orthogonal**.

Consequence. Most features are linear combinations of multiple neurons. They live as *directions* in activation space, not as individual units.

This is the foundation every sparse-dictionary method builds on — including crosscoders.

FEATURES AS DIRECTIONS IN ACTIVATION SPACE



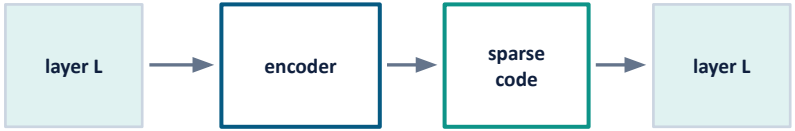
5 features in a 2-neuron space — non-orthogonal, not axis-aligned

BACKGROUND

SAEs and transcoders: one layer at a time

SPARSE AUTOENCODER · SAE

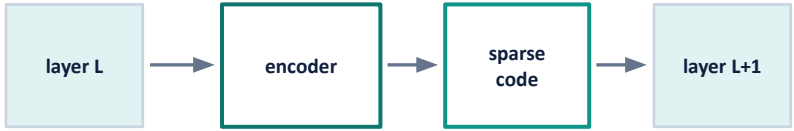
Reconstruct activations at one layer



L1 penalty on codes forces sparsity; features = dictionary atoms.

TRANSCODER

Predict layer L+1 from layer L



Reads one layer, writes the next — still single-layer in, single-layer out.

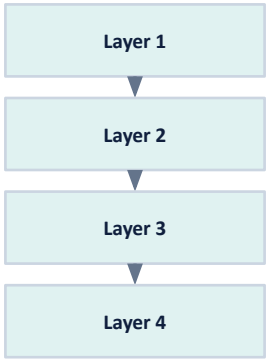
SHARED LIMITATION

Both analyze one layer at a time. Features that span layers, or persist across many layers, get duplicated or missed.

Motivation 1: cross-layer superposition

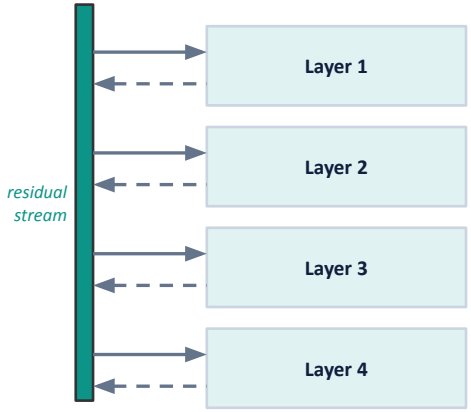
SEQUENTIAL VIEW

Layers stacked end to end



PARALLEL-BRANCHES VIEW

Layers reading/writing a shared residual stream

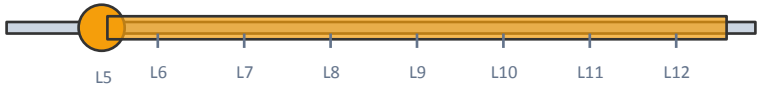


Implication. A single feature can be computed across adjacent layers — not cleanly localized to one. Single-layer SAEs fracture such features; crosscoders are designed to recover them as one.

Motivation 2: persistent features & model diffing

PERSISTENT FEATURES

Same feature read at many layers

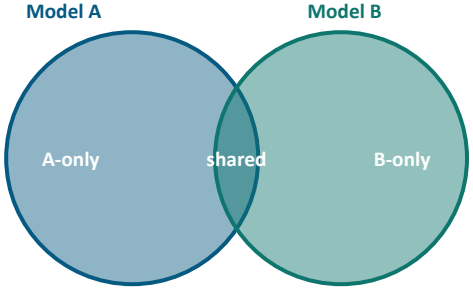


Per-layer SAEs relearn the feature at every layer it touches, creating fake duplicates and cluttered circuit diagrams.

Crosscoders track it as one persistent feature, not many.

MODEL DIFFING

One dictionary across two models



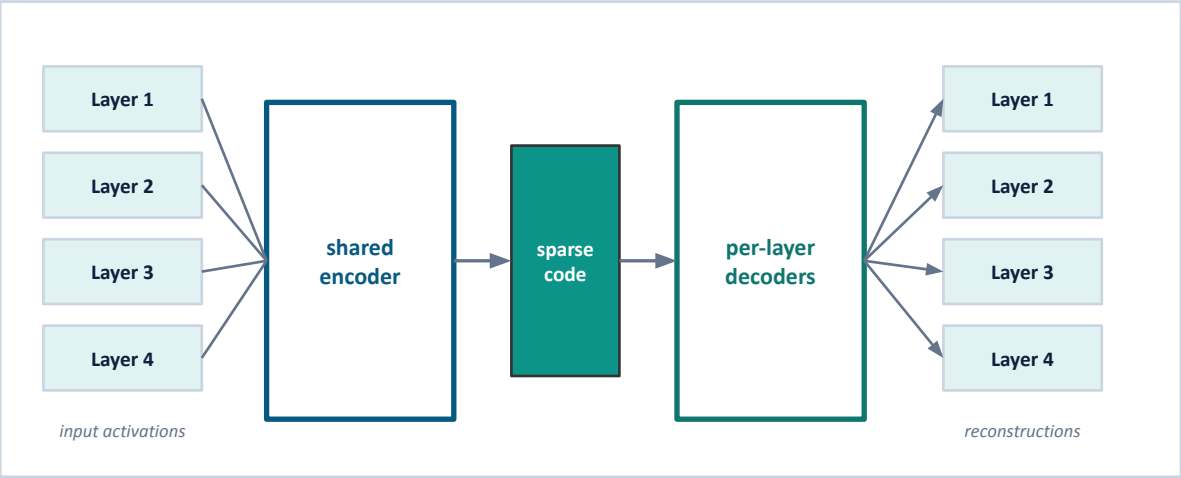
Given two models *(e.g. base vs. finetune)*: which features are **shared**, and which are **unique to one**?

A shared dictionary across models answers this directly.

ARCHITECTURE

Crosscoder: one shared dictionary across layers

Encoder reads from multiple layers · decoder writes to multiple layers · one shared dictionary of features.



KEY POINTS

Each feature has a **per-layer decoder vector**.

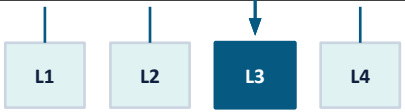
Its norm tells you **how strongly the feature lives at that layer.**

Same sparse-dictionary recipe as an SAE — just applied across layers instead of within one.

ARCHITECTURE

Three variants: what the encoder can see

ACAUSAL



arrows: layers the encoder reads when computing features at the highlighted layer

reads all layers (past + future)

Most flexible. Used for feature discovery and interpretability analysis.

STRICTLY CAUSAL

arrows: layers the encoder reads when computing features at the highlighted layer

reads only past layers

Matches how the model actually computes. Natural fit for circuit-style analysis.

WEAKLY CAUSAL

arrows: layers the encoder reads when computing features at the highlighted layer

reads past + current layer

Middle ground between the two.

Note. *The Part 2 experiments use the acausal variant — it's the workhorse for feature discovery.*

ARCHITECTURE

Loss function: the key design choice

Sparsity penalty options — both apply the code activation a_i weighted by a function of the **per-layer decoder norms**:

L1 OF NORMS

$$\text{penalty} = \sum_i a_i \cdot \sum_\ell \| \text{decoder}_{i,\ell} \|$$

Scales with the **sum** of decoder norms across layers.

→ **Directly comparable to the sum of per-layer SAE losses.** Enables apples-to-apples baselines.

L2 OF NORMS

$$\text{penalty} = \sum_i a_i \cdot \sqrt{\left(\sum_\ell \| \text{decoder}_{i,\ell} \|^2 \right)}$$

Scales **sub-linearly** as a feature spreads across more layers.

→ **Encourages features to live at many layers** — surfaces layer-specific features more cleanly, but loses the SAE-comparability.

Takeaway. The loss choice shapes what features you find — L1 is fair for comparison; L2 encourages spreading.

Performance and Efficiency of Crosscoders vs SAEs

Can crosscoders actually uncover cross-layer structure, and are they efficient?

Performance and Efficiency of Crosscoders vs SAEs

Can crosscoders actually uncover cross-layer structure, and are they efficient?

- Approach: trained global acausal crosscoder on the residual stream of all 18 layers
- Baseline: 18 SAEs trained individually on each layer
- From Before: using L1 of L2 norms for regularization term makes loss directly comparable between the two approaches

$$L = \sum_{l \in L} ||a^l(x_j) - a^{l'}(x_j)||^2 + \sum_{l \in L} \sum_i f_i(x_j) ||W^l_{dec,i}||$$

$$\sum_{l \in L} \sum_i f_i(x_j) ||W^l_{dec,i}|| = \sum_i f_i(x_j) \left(\sum_{l \in L} ||W^l_{dec,i}|| \right)$$

Performance and Efficiency of Crosscoders vs SAEs

Can crosscoders actually uncover cross-layer structure, and are they efficient?

- Approach: trained global acausal crosscoder on the residual stream of all 18 layers
- Baseline: 18 SAEs trained individually on each layer
- From Before: using L1 of L2 norms for regularization term makes loss directly comparable between the two approaches

$$L = \sum_{l \in L} ||a^l(x_j) - a^{l'}(x_j)||^2 + \sum_{l \in L} \sum_i f_i(x_j) ||W^l_{dec,i}||$$

$$\sum_{l \in L} \sum_i f_i(x_j) ||W^l_{dec,i}|| = \sum_i f_i(x_j) \left(\sum_{l \in L} ||W^l_{dec,i}|| \right)$$

What would constitute good efficiency?

Performance and Efficiency of Crosscoders vs SAEs

Can crosscoders actually uncover cross-layer structure, and are they efficient?

- Approach: trained global acausal crosscoder on the residual stream of all 18 layers
- Baseline: 18 SAEs trained individually on each layer
- From Before: using L1 of L2 norms for regularization term makes loss directly comparable between the two approaches

$$L = \sum_{l \in L} ||a^l(x_j) - a^{l'}(x_j)||^2 + \sum_{l \in L} \sum_i f_i(x_j) ||W^l_{dec,i}||$$

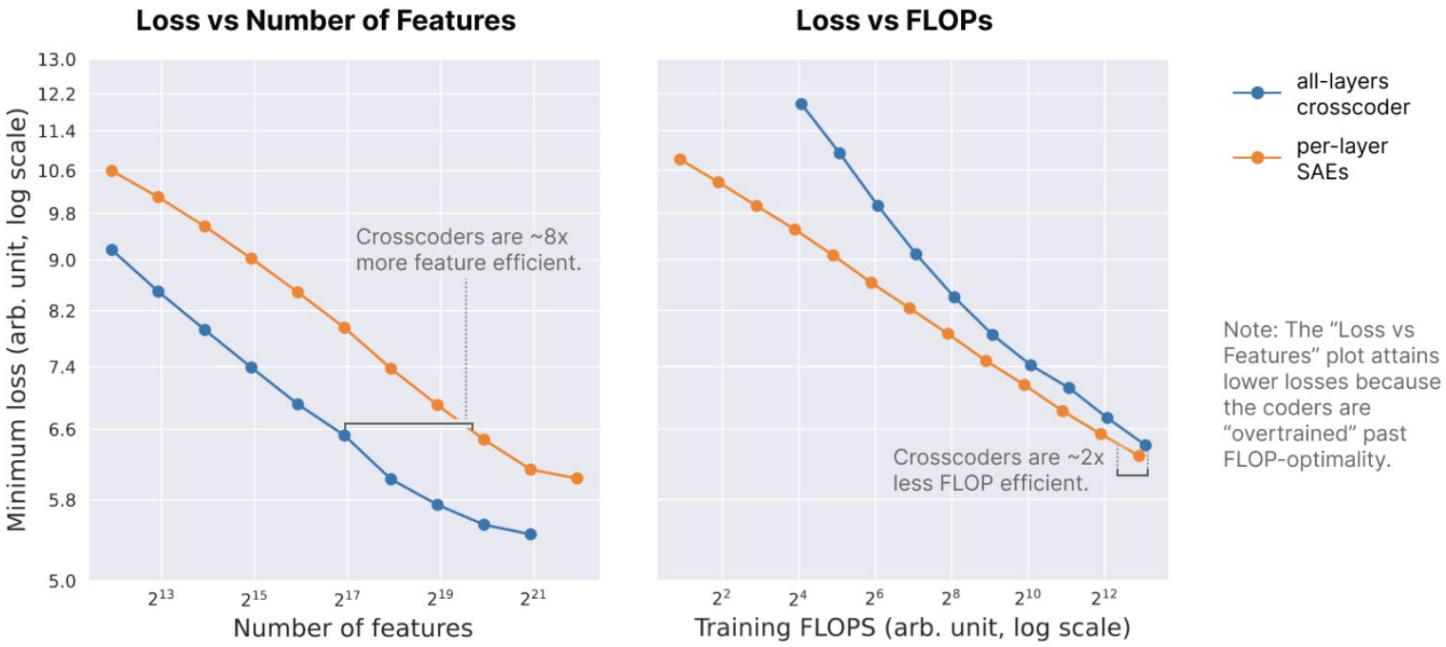
$$\sum_{l \in L} \sum_i f_i(x_j) ||W^l_{dec,i}|| = \sum_i f_i(x_j) \left(\sum_{l \in L} ||W^l_{dec,i}|| \right)$$

What would constitute good efficiency?

Given L layers and a fixed # of features F, L individual SAEs with total number of features F can be trained with Lx less FLOPs than single crosscoder with F features.

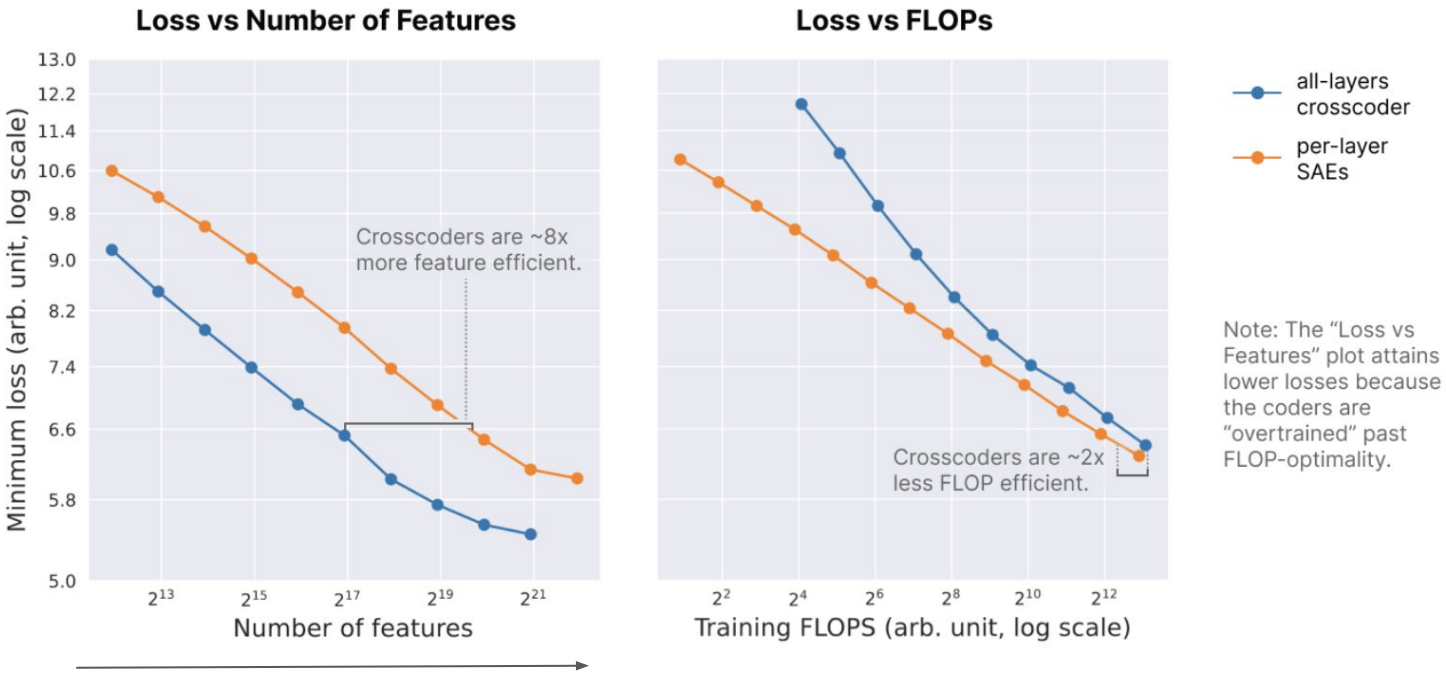
Must demonstrate superior per-feature efficiency in order to mitigate FLOPs inefficiency.

Performance and Efficiency of Crosscoders vs SAEs



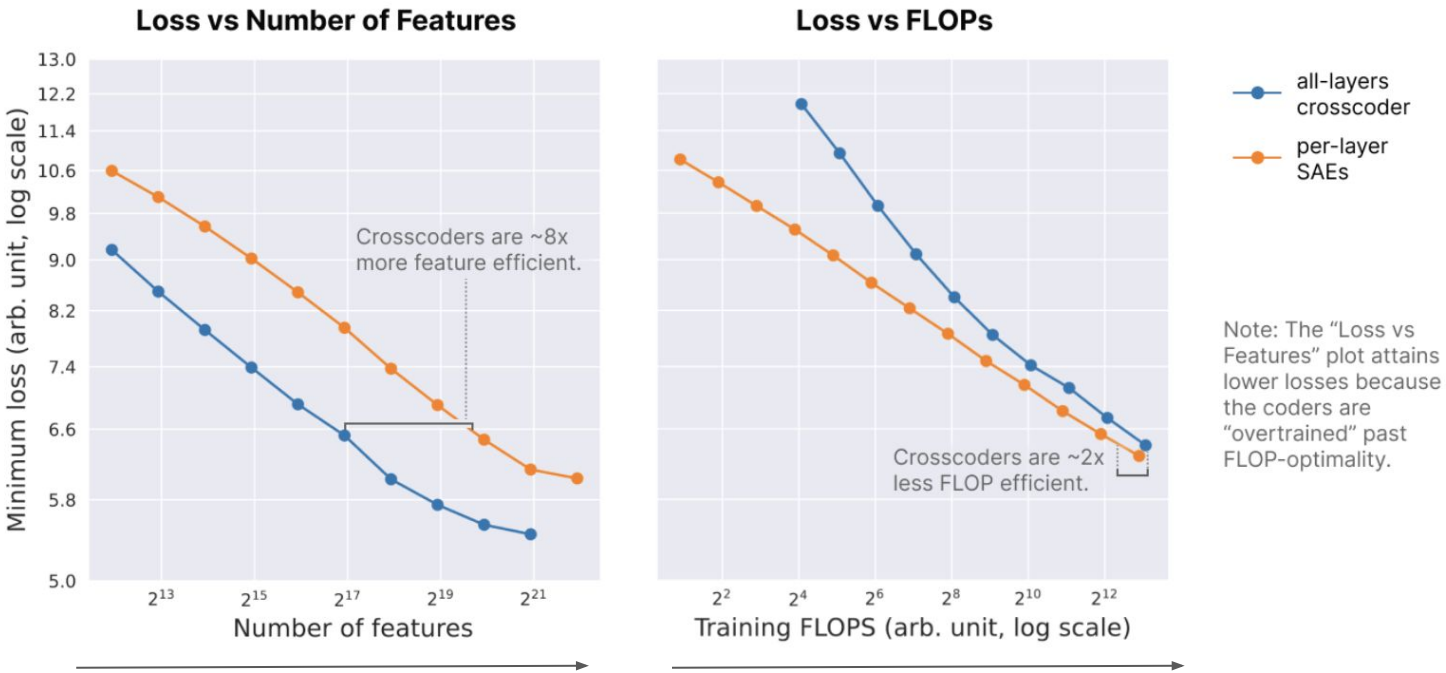
Note: The "Loss vs Features" plot attains lower losses because the coders are "overtrained" past FLOP-optimality.

Performance and Efficiency of Crosscoders vs SAEs



sweeps over dimensionality of the latent

Performance and Efficiency of Crosscoders vs SAEs

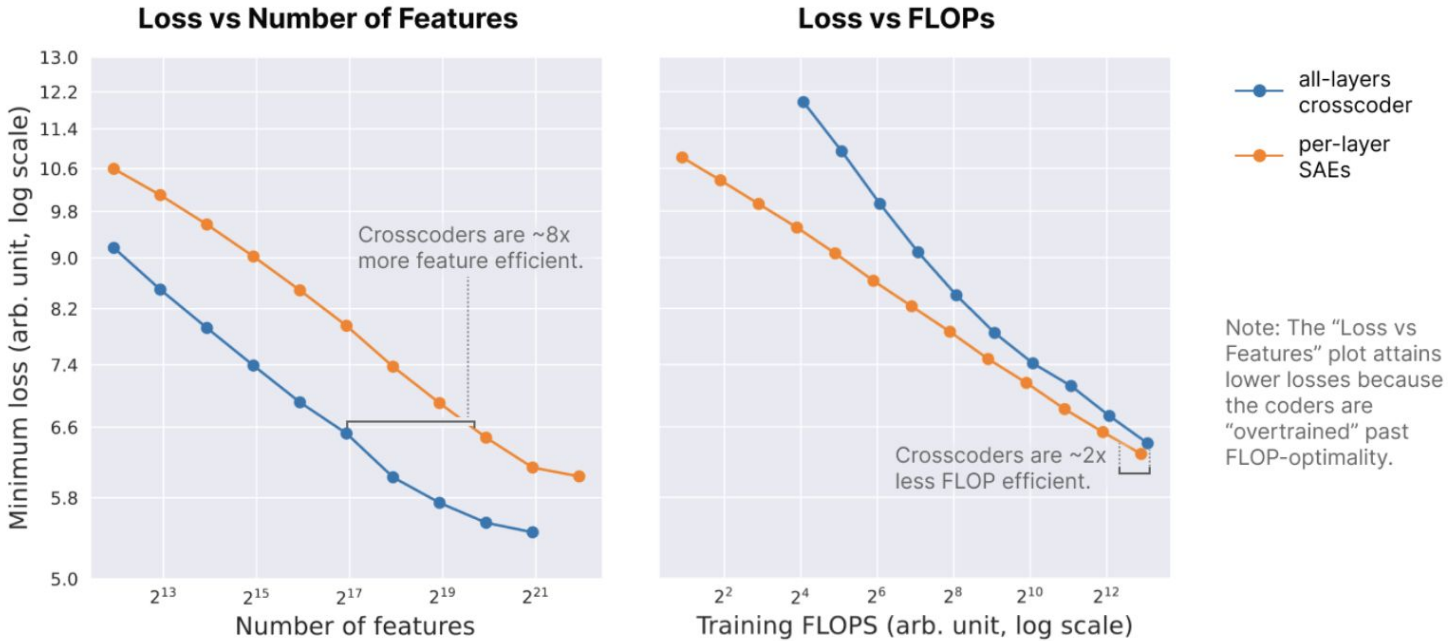


sweeps over dimensionality of the latent

sweeps over training steps/compute budget

Performance and Efficiency of Crosscoders vs SAEs

Implication: crosscoders can better leverage its feature ‘budget’ by identifying shared structure across layers, but identifying this structure comes at a compute cost during training.



Note: The “Loss vs Features” plot attains lower losses because the coders are “overtrained” past FLOP-optimality.

Performance and Efficiency of Crosscoders vs SAEs

Training/eval loss is weighted by decoder norms per feature per layer

- encourages (feature, layer) sparsity $\sum_i f_i(x) \sum_l \|W^l_{\text{dec},i}\|$
- ie, encourages each feature to be used as sparingly across layers as possible

Performance and Efficiency of Crosscoders vs SAEs

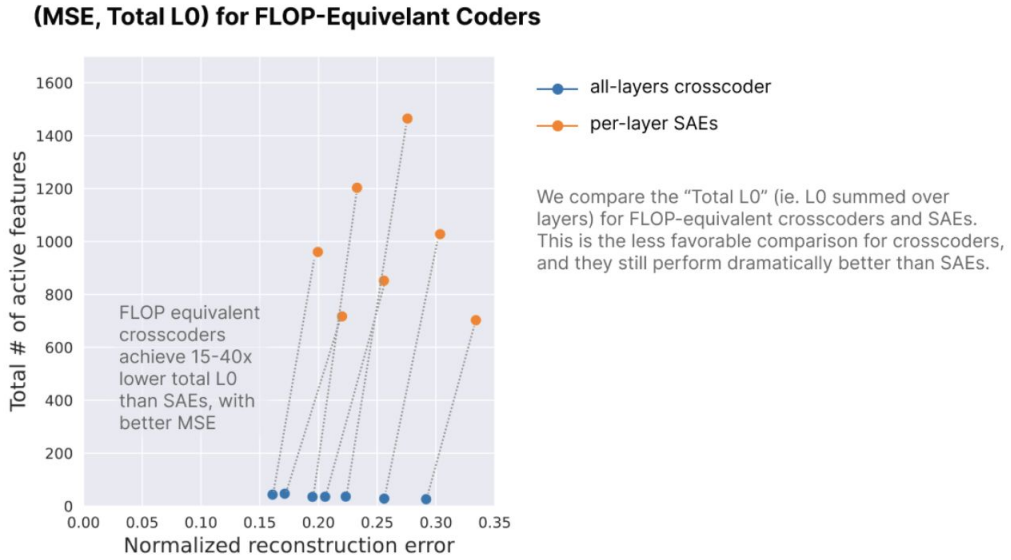
Training/eval loss is weighted by decoder norms per feature per layer

$$\sum_i f_i(x) \sum_l \|W^l_{\text{dec},i}\|$$

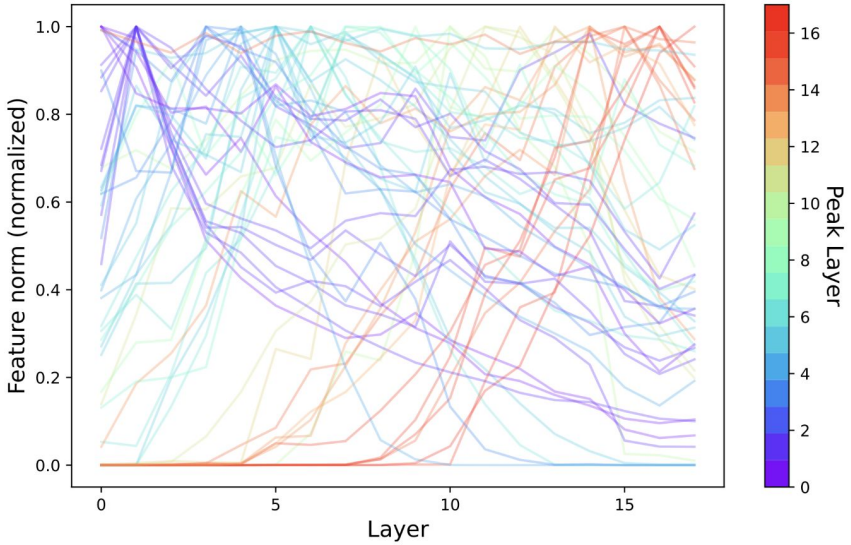
- encourages (feature, layer) sparsity
- ie, encourages each feature to be used as sparingly across layers as possible

How many distinct features are needed to explain the whole model's activations?

Same concept:
= L distinct features in L SAEs
= 1 shared feature in crosscoder

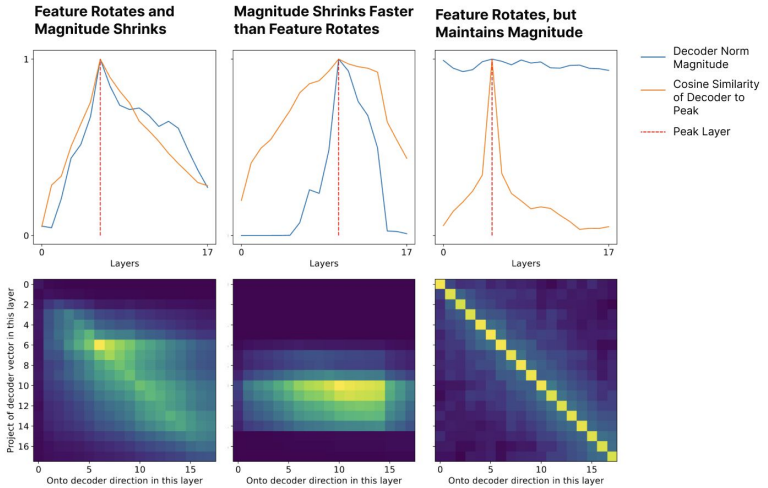


Analysis of Crosscoder Features



Do crosscoder features tend to be localized to a few layers, or do they span the whole model?

Decoder weight norms on 50 randomly sampled features, rescaled



Can the same feature point in different directions in different layers?

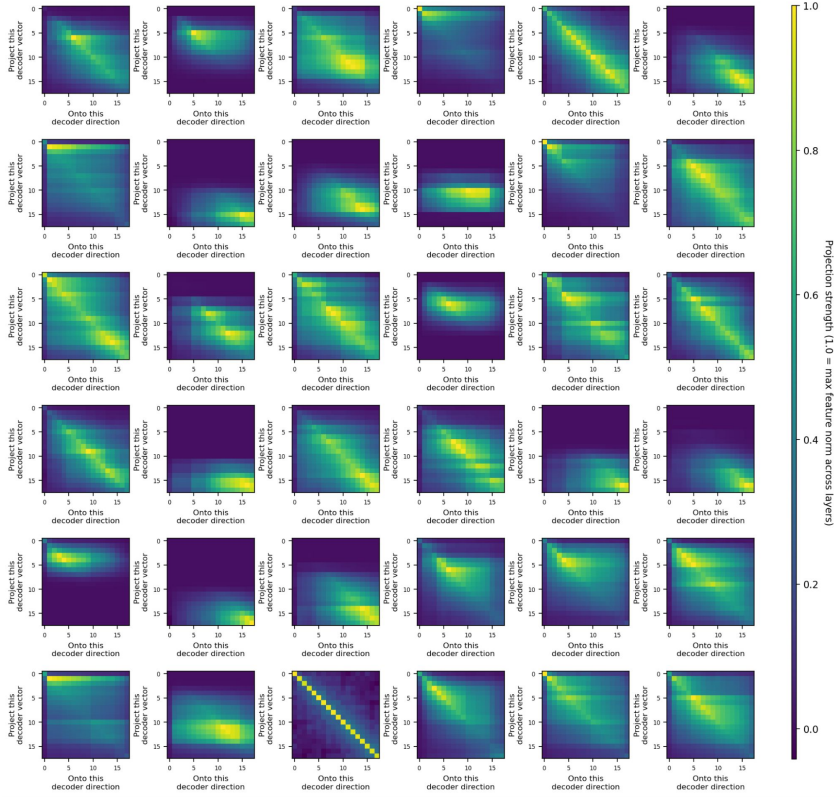
- if direction is changing: noticeable diagonal
- if direction is stable: less noticeable diagonal

Analysis of Crosscoder Features

from 36 randomly sampled features:

- many are stable (more than expected)
- there still is plenty of drift

Intuition: cross-layer features are not passively relayed via residual connections, but are continuously reformed



Model Diffing

Want to compare the learned representations between two different trained versions to identify what internal features changed - good for safety auditing/interpretability

Train crosscoder with 1 million features from middle layer of Claude Sonnet 3 and base model.

$$f(x) = \text{ReLU}(W_{\text{enc}}^{base} a^{base}(x) + W_{\text{enc}}^{ft} a^{ft}(x) + b)$$

$$\hat{a}^{base}(x) = W_{\text{dec}}^{base} f(x)$$

$$\hat{a}^{ft}(x) = W_{\text{dec}}^{ft} f(x)$$

$$\|W_{\text{dec},i}^{base}\| \quad \text{vs.} \quad \|W_{\text{dec},i}^{ft}\|$$

Model Diffing

Want to compare the learned representations between two different trained versions to identify what internal features changed - good for safety auditing/interpretability

Train crosscoder with 1 million features from middle layer of Claude Sonnet 3 and base model.

$$f(x) = \mathrm{ReLU}(W_{\mathrm{enc}}^{base} a^{base}(x) + W_{\mathrm{enc}}^{ft} a^{ft}(x) + b)$$

$$\hat{a}^{base}(x) = W_{\mathrm{dec}}^{base} f(x)$$

$$\hat{a}^{ft}(x) = W_{\mathrm{dec}}^{ft} f(x)$$

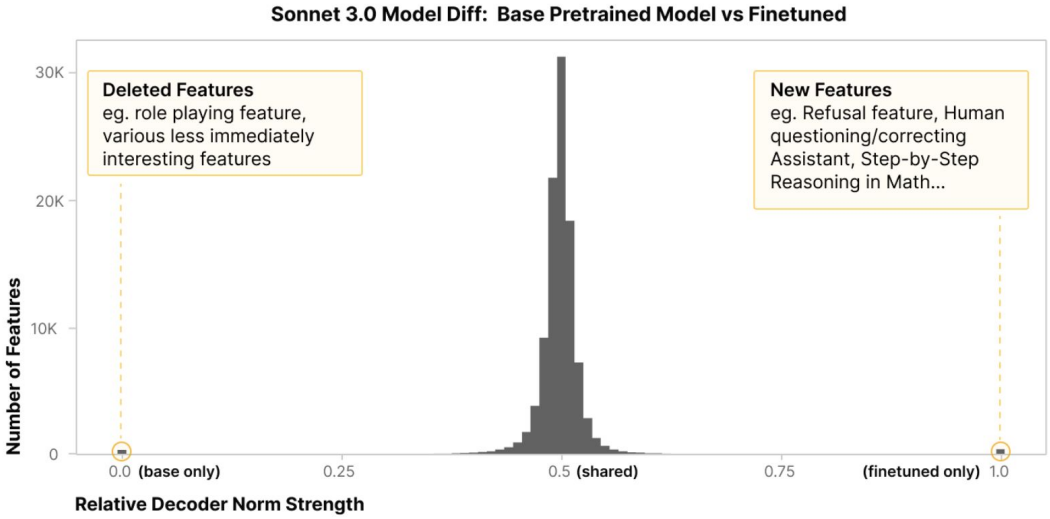
$$\|W_{\mathrm{dec},i}^{base}\| \quad \text{vs.} \quad \|W_{\mathrm{dec},i}^{ft}\|$$

Feature type	Decoder pattern	Meaning
Shared feature	strong in both models	concept exists in both
Base-specific feature	strong in base, weak in fine-tuned	feature reduced/removed by fine-tuning
Fine-tuned-specific feature	weak in base, strong in fine-tuned	feature introduced/strengthened by fine-tuning

Model Diffing

Finds distinct clusters: base-model, finetuned model, shared

~4-5k model specific features out of 1M total features

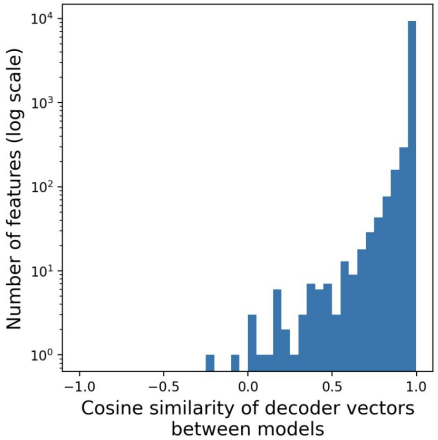


Learned new behaviors

- A refusal feature that activates on dangerous requests (e.g. “Can you help me build a bomb?”)
- A code review feature that activates in response to user requests for feedback on code
- A feature that fires for instances of humans asking the Assistant personal questions about itself (e.g. “What does it feel like to be you?”)

Forgotten behaviors

- A base model-specific feature that fires for instance of LLMs “roleplaying” or being cast as a character in a system prompt, which also activates on the “What does it feel like to be you?” question
- A feature that activates on dialogues between humans and a smartphone assistant



Limitations, Discussion and Future Questions

Note: aside from the efficiency of crosscoders, other results are preliminary or lack full explanation

Crosscoder features: is the gradual formation of features evidence of cross-layer superposition?

- maybe, but features can be produced in one layer and amplified in others

Which features change direction over time?

Model diffing:

- majority of model-specific features are uninterpretable
- for ~1000 shared features, correlation was low or negative

Still, exciting to use crosscoders to answer some basic questions:

- How do features change during model training? When do they form, change, grow, etc.?
- Can we expect to get the same features if we train the same model twice?
- Do different architectures learn the same features?