Faculty of Mathematics and Information Sciences

WARSAW UNIVERSITY OF TECHNOLOGY

Introduction to Sparse Autoencoders

Scaling and evaluating sparse autoencoders (2024)

Overview

- Problem of Superposition in Neural Networks

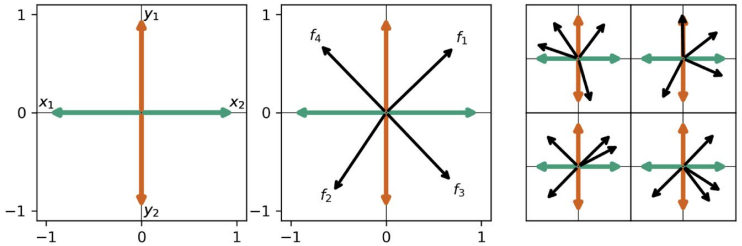
- What are Sparse Autoencoders?
- How to evaluate Sparse Autoencoders

- Problems with Sparse Autoencoders
- Scaling and evaluating sparse autoencoders
- Fun Part!!

- Future Problems with Sparse Autoencoders

Why neural network are not monosemantic (superposition)?

superposition - “a hypothesized phenomenon where a neural network represents more independent "features" of the data than it has neurons by assigning each feature its own linear combination of neurons.”



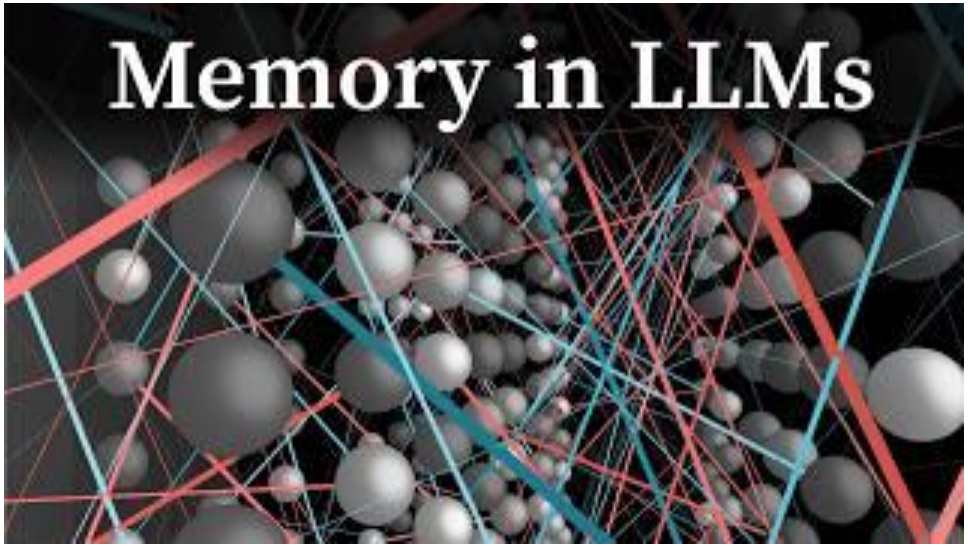
Linear algebraic structure of word senses, with applications to polysemy
S. Arora, Y. Li, Y. Liang, T. Ma, A. Risteski.
Transactions of the Association for Computational Linguistics, Vol 6, pp. 483--495. MIT Press. 2018.

Decoding The Thought Vector [link]
G. Goh. 2016.

Zoom In: An Introduction to Circuits
C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, S. Carter.
Distill. 2020.
DOI: 10.23915/distill.00024.001

Toy Models of Superposition
N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec, Z. Hatfield-Dodds, R. Lasenby, D. Drain, C. Chen, R. Grosse, S. McCandlish, J. Kaplan, D. Amodei, M. Wattenberg, C. Olah.
Transformer Circuits Thread. 2022.

Why neural network are not monosemantic (superposition)?



Do every neuron is polisemantic?

Language models can explain neurons in language models

AUTHORS

Steven Bills*, Nick Cammarata*, Dan Mossing*, Henk Tillman*, Leo Gao*, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu*, William Saunders*

* Core Research Contributor; Author contributions statement below. Correspondence to interpretability@openai.com.

AFFILIATION

OpenAI

PUBLISHED

May 9, 2023

Overall, our subjective sense was that neurons for more capable models tended to be more interesting, although we spent the majority of our efforts looking at GPT-2 XL neurons rather than more modern models.

out of a total of 307,200 neurons, 5,203 (1.7%) have top-and-random scores above 0.7 (explaining roughly half the variance), using our default methodology. With random-only scoring, this drops to 732 neurons (0.2%). Only 189 neurons (0.06%) have top-and-random scores above 0.9, and 86 (0.03%) have random-only scores above 0.9.

CLIP-DISSECT: AUTOMATIC DESCRIPTION OF NEURON REPRESENTATIONS IN DEEP VISION NETWORKS

Tuomas Oikarinen
UCSD CSE
toikarinen@ucsd.edu

Tsui-Wei Weng
UCSD HDSI
lweng@ucsd.edu

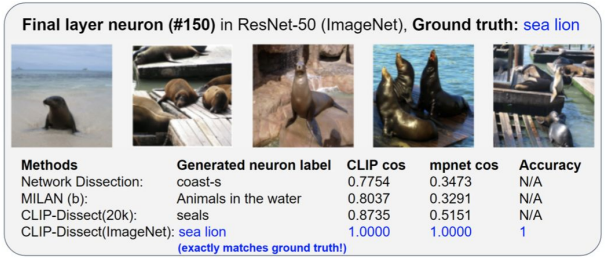


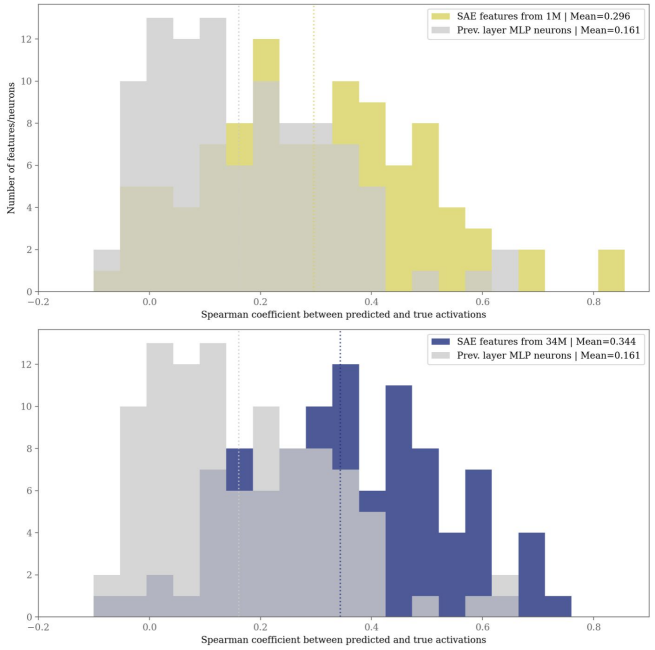
Figure 3: Example of a final layer neuron: we compare the descriptions generated by different methods and our metrics. Accuracy only evaluated for CLIP-Dissect with ImageNet labels as concept set since it is the only method where exact correct answer is a possible choice and therefore accuracy makes sense.

Table 2: Performance when labeling final layer neurons of a ResNet18 trained on Places365. Accuracy measured on 267/365 neurons whose label is a directly included in Broden labels.

Method	D_{probe}	Concept set $\mathcal{S}$	gt label annotation	Top1 Acc	CLIP cos	mpnet cos
Net-Dissect (baseline)	Broden	Broden	Yes	43.82%	0.8887	0.6697
CLIP-Dissect (ours)	Broden	Broden	No	**58.05%**	**0.9106**	**0.7024**

Do every neuron is polisemantic? Are they?

Unfortunately, the most natural computational unit of the neural network – the neuron itself – turns out not to be a natural unit for human understanding. This is because many neurons are *polysemantic*: they respond to mixtures of seemingly unrelated inputs. In the vision model *Inception v1*, a single neuron responds to faces of cats and fronts of cars [1]. In a small language model we discuss in this paper, a single neuron responds to a mixture of academic citations, English dialogue, HTTP requests, and Korean text. Polysemanticity makes it difficult to reason about the behavior of the network in terms of the activity of individual neurons.



We additionally evaluated the specificity of random neurons and SAE features using the automated specificity rubric above. We find that the activations of a random selection of SAE features are significantly more specific than those of the neurons in the previous layer.

What are Sparse Autoencoders?

- The idea was explored long ago
- Inspired by the sparse coding hypothesis in neuroscience.
- Dictionary learning
- Mechanistic interpretability

Sparse dictionary learning (also known as **sparse coding** or **SDL**) is a representation learning method which aims to find a sparse representation of the input data in the form of a linear combination of basic elements as well as those basic elements themselves.

[PDF] **Sparse autoencoder**
A Ng - CS294A Lecture notes, 2011 - graphics.stanford.edu
… Then, we show how this is used to construct an **autoencoder**, which is an unsupervised learning algorithm. Finally, we build on this to derive a **sparse autoencoder**. Because these notes …
☆ Save 99 Cite Cited by 1934 Related articles All 7 versions »

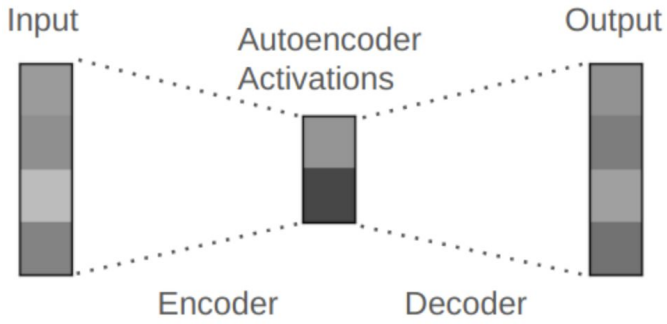
Sparse coding [edit]

The sparse code is when each item is encoded by the strong activation of a relatively small set of neurons. For each item to be encoded, this is a different subset of all available neurons. In contrast to sensor-sparse coding, sensor-dense coding implies that all information from possible sensor locations is known.

$$J_{\text{sparse}}(W,b) = J(W,b) + \beta \sum_{j=1}^{s_2} \text{KL}(\rho||\hat{\rho}_j),$$

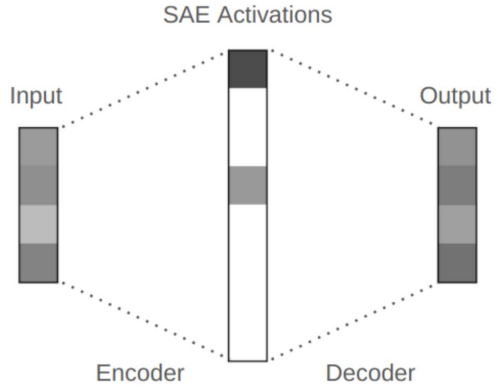
What are Sparse Autoencoders?

Typical Autoencoder



- Dense Bottleneck
- l << n

Sparse Autoencoder



- Sparse Bottleneck
- l >> n

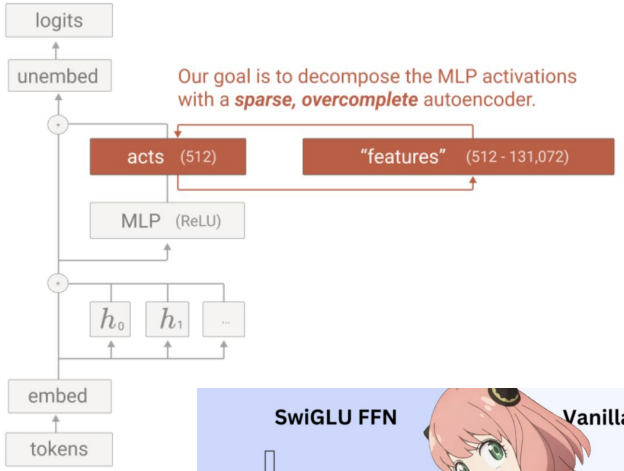
What do Sparse Autoencoders have to do with interpretability?

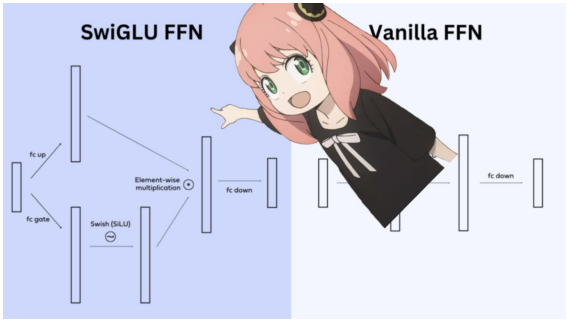
Sparse dictionary learning (also known as **sparse coding** or **SDL**) is a representation learning method which aims to find a sparse representation of the input data in the form of a linear combination of basic elements as well as those basic elements themselves.

SAE decompose the activations of a model into more interpretable pieces.

Can be applied to **any** activation representation

Residual streams are believed to add new information to the representation. Residual stream dimension 128 and internal MLP dimension 512.





As a preprocessing step we apply a scalar normalization to the model activations so their average squared L2 norm is the residual stream dimension, D. We denote the normalized activations as $\mathbf{x} \in \mathbb{R}^D$, and attempt to decompose this vector using F features as follows:

$$\hat{\mathbf{x}} = \mathbf{b}^{dec} + \sum_{i=1}^{F} f_i(\mathbf{x}) \mathbf{W}^{dec}_{\cdot,i}$$

where $W^{dec} \in \mathbb{R}^{D \times F}$ are the learned SAE decoder weights, $\mathbf{b}^{dec} \in \mathbb{R}^D$ are learned biases, and f_i denotes the activity of feature *i*. Feature activations are given by the output of the encoder:

$$f_i(x) = \mathrm{ReLU}\left(\mathbf{W}^{enc}_{i,\cdot} \cdot \mathbf{x} + b^{enc}_i\right)$$

where $W^{enc} \in \mathbb{R}^{F \times D}$ are the learned SAE encoder weights, and $\mathbf{b}^{enc} \in \mathbb{R}^F$ are learned biases.

The loss function $\mathcal{L}$ is the combination of an L2 penalty on the reconstruction loss and an L1 penalty on feature activations.

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}}\left[\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \lambda \sum_i f_i(\mathbf{x}) \cdot \|\mathbf{W}^{dec}_{\cdot,i}\|_2\right]$$

Including the factor of $\|\mathbf{W}^{dec}_{\cdot,i}\|_2$ in the L1 penalty term allows us to interpret the unit-normalized decoder vectors $\frac{\mathbf{W}^{dec}_{\cdot,i}}{\|\mathbf{W}^{dec}_{\cdot,i}\|_2}$ as "feature vectors" or "feature directions," and the product $f_i(\mathbf{x}) \cdot \|\mathbf{W}^{dec}_{\cdot,i}\|_2$ as the feature activations[2] . Henceforth we will use "feature activation" to refer to this quantity.

Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet

AUTHORS

Adly Templeton*, Tom Conerly*, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, Alex Tamkin, Esin Durmus, Tristan Hume, Francesco Mosconi, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, Tom Henighan

AFFILIATIONS

Anthropic

PUBLISHED

May 21, 2024

* Core Contributor; Correspondence to henighan@anthropic.com; Author contributions statement below.

As a preprocessing step we apply a scalar normalization to the model activations so their average squared L2 norm is the residual stream dimension, D. We denote the normalized activations as $\mathbf{x} \in \mathbb{R}^D$, and attempt to decompose this vector using F features as follows:

$$\hat{\mathbf{x}} = \mathbf{b}^{dec} + \sum_{i=1}^{F} f_i(\mathbf{x}) \mathbf{W}^{dec}_{\cdot,i}$$

where $W^{dec} \in \mathbb{R}^{D \times F}$ are the learned SAE decoder weights, $\mathbf{b}^{dec} \in \mathbb{R}^D$ are learned biases, and f_i denotes the activity of feature *i*. Feature activations are given by the output of the encoder:

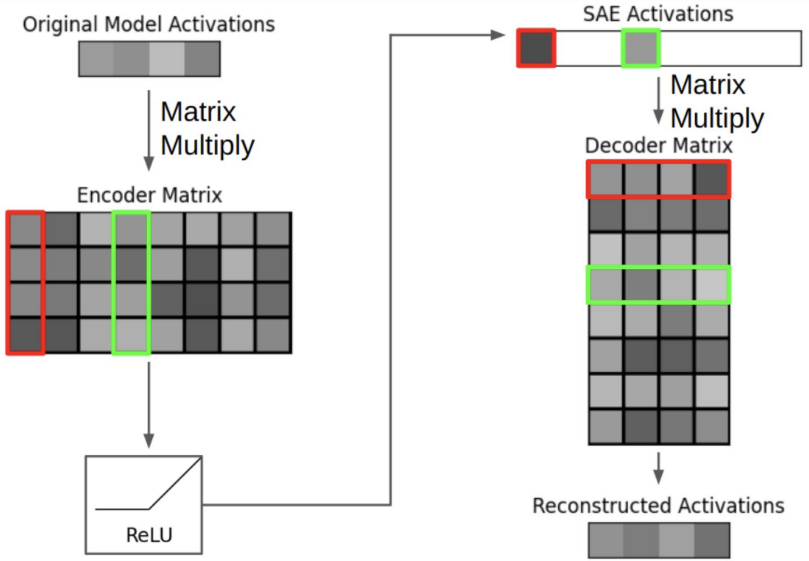
$$f_i(x) = \text{ReLU}\left(\mathbf{W}^{enc}_{i,\cdot} \cdot \mathbf{x} + b^{enc}_i\right)$$

where $W^{enc} \in \mathbb{R}^{F \times D}$ are the learned SAE encoder weights, and $\mathbf{b}^{enc} \in \mathbb{R}^F$ are learned biases.

The loss function $\mathcal{L}$ is the combination of an L2 penalty on the reconstruction loss and an L1 penalty on feature activations.

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}}\left[\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \lambda \sum_i f_i(\mathbf{x}) \cdot \|\mathbf{W}^{dec}_{\cdot,i}\|_2\right]$$

Including the factor of $\|\mathbf{W}^{dec}_{\cdot,i}\|_2$ in the L1 penalty term allows us to interpret the unit-normalized decoder vectors $\frac{\mathbf{W}^{dec}_{\cdot,i}}{\|\mathbf{W}^{dec}_{\cdot,i}\|_2}$ as "feature vectors" or "feature directions," and the product $f_i(\mathbf{x}) \cdot \|\mathbf{W}^{dec}_{\cdot,i}\|_2$ as the feature activations[2] . Henceforth we will use "feature activation" to refer to this quantity.



As a preprocessing step we apply a scalar normalization to the model activations so their average squared L2 norm is the residual stream dimension, D. We denote the normalized activations as $\mathbf{x} \in \mathbb{R}^D$, and attempt to decompose this vector using F features as follows:

$$\hat{\mathbf{x}} = \mathbf{b}^{dec} + \sum_{i=1}^{F} f_i(\mathbf{x}) \mathbf{W}^{dec}_{\cdot,i}$$

where $W^{dec} \in \mathbb{R}^{D \times F}$ are the learned SAE decoder weights, $\mathbf{b}^{dec} \in \mathbb{R}^D$ are learned biases, and f_i denotes the activity of feature *i*. Feature activations are given by the output of the encoder:

$$f_i(x) = \text{ReLU}\left(\mathbf{W}^{enc}_{i,\cdot} \cdot \mathbf{x} + b^{enc}_i\right)$$

where $W^{enc} \in \mathbb{R}^{F \times D}$ are the learned SAE encoder weights, and $\mathbf{b}^{enc} \in \mathbb{R}^F$ are learned biases.

The loss function $\mathcal{L}$ is the combination of an L2 penalty on the reconstruction loss and an L1 penalty on feature activations.

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}}\left[\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \lambda \sum_i f_i(\mathbf{x}) \cdot \|\mathbf{W}^{dec}_{\cdot,i}\|_2\right]$$

Including the factor of $\|\mathbf{W}^{dec}_{\cdot,i}\|_2$ in the L1 penalty term allows us to interpret the unit-normalized decoder vectors $\frac{\mathbf{W}^{dec}_{\cdot,i}}{\|\mathbf{W}^{dec}_{\cdot,i}\|_2}$ as "feature vectors" or "feature directions," and the product $f_i(\mathbf{x}) \cdot \|\mathbf{W}^{dec}_{\cdot,i}\|_2$ as the feature activations[2] . Henceforth we will use "feature activation" to refer to this quantity.

Newer Version!

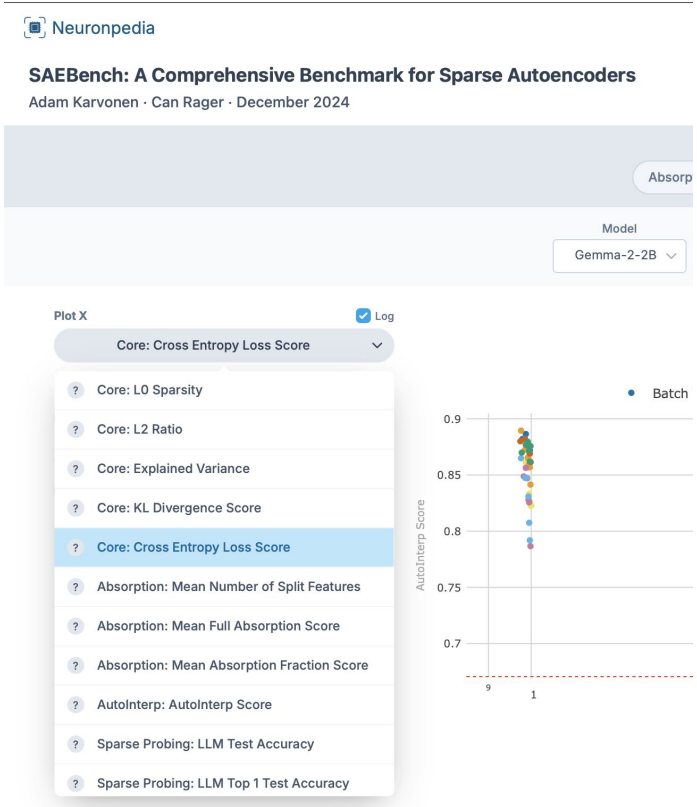
Let n be the input and output dimension and m be the autoencoder hidden layer dimension. Let s be the size of the dataset. Given encoder weights $W_e \in \mathbb{R}^{m \times n}$, decoder weights $W_d \in \mathbb{R}^{n \times m}$, and biases $\mathbf{b}_e \in \mathbb{R}^m$, $\mathbf{b}_d \in \mathbb{R}^n$, the operations and loss function over a dataset $X \in \mathbb{R}^{s,n}$ are:

$$\begin{aligned}
\mathbf{f}(x) &= \text{ReLU}(W_e\mathbf{x} + \mathbf{b}_e) \\
\hat{\mathbf{x}} &= W_d\mathbf{f}(x) + \mathbf{b}_d \\
\mathcal{L} &= \frac{1}{|X|}\sum_{\mathbf{x} \in X} ||\mathbf{x} - \hat{\mathbf{x}}||_2^2 + \lambda \sum_i |\mathbf{f}_i(x)|||W_{d,i}||_2
\end{aligned}$$

Note that the columns of W_d have an unconstrained L2 norm (in Towards Monosemanticity they were constrained to norm one) and the sparsity penalty (second term) has been changed to include the L2 norm of the columns of W_d. We believe this was the most important change we made from Towards Monosemanticity.

How to evaluate Sparse Autoencoders

- mean L0-norm of feature activations
- Mean Squared Error
- fraction of variance unexplained (FVU)
- delta LM loss - cross-entropy loss experienced by the model when activation is forwarded through SAE into the LM’s forward pass
- delta LM loss measured by KL divergence
- Sparse Probing
- AutoInterp Score



Problems with Sparse Autoencoders

- How to interpret SAE features
- Shrinkage Problem
- Scaling of SAE
- Feature Splitting
- Composition of canonical units

How to interpret SAE features

Just use LLM!

For each feature draw:

- 5 highest activating examples
- 20 random, non-activating examples

The model based on the k highest and n random samples suggest the description

Autointerpretable score is calculated by giving example and description and asking model to predict the magnitude of the feature.

Finally the correlation is calculated between ground truth and LLM prediction

Problems with Sparse Autoencoders

- ~~How to interpret SAE features~~
- Shrinkage Problem
- Scaling of SAE
- Feature Splitting
- Composition of canonical units

Shrinkage Problem

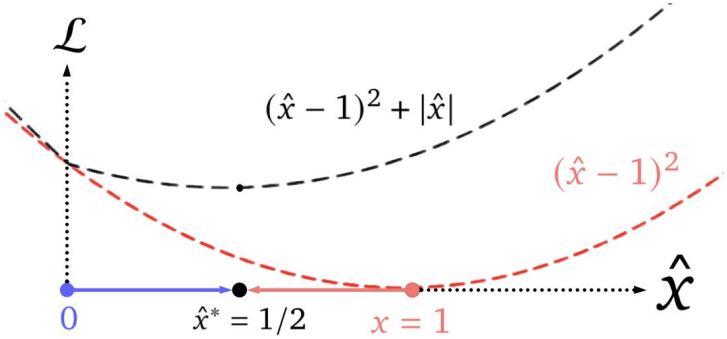
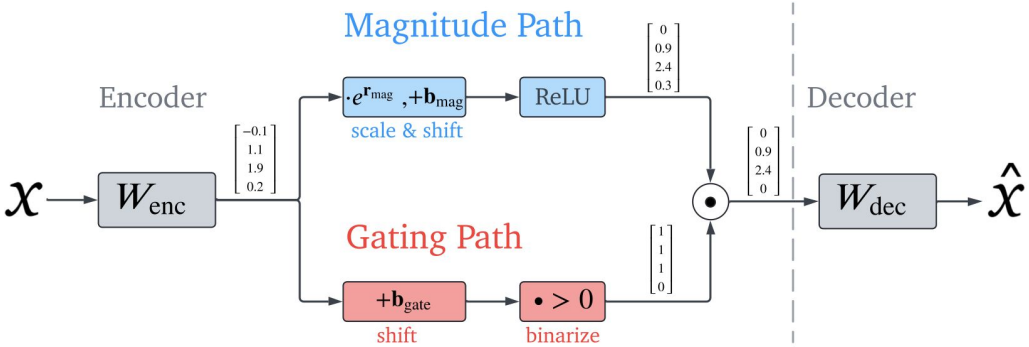
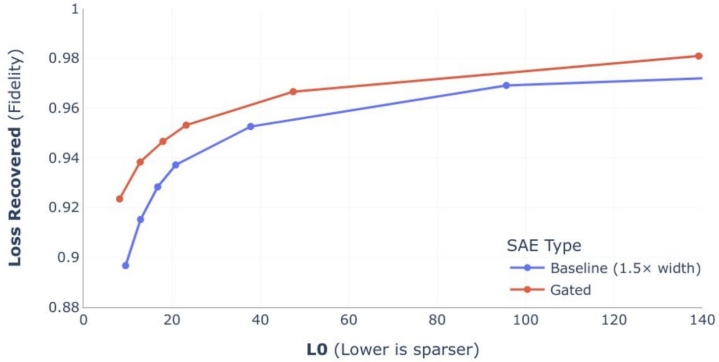


Figure 2 | The L1 penalty in sparse autoencoder causes *shrinkage* – reconstructions are biased towards smaller norms, even when perfect reconstruction is possible.
E.g. a single-feature SAE (with L1 coefficient $\lambda = 1$) reconstructs 1/2 rather than 1 when minimizing Equation (4).





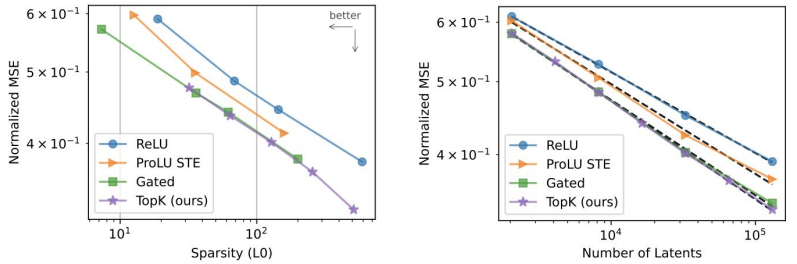
Problems with Sparse Autoencoders

- ~~How to interpret SAE features~~
- ~~Shrinkage Problem~~
- Scaling of SAE (dead latents)
- Feature Splitting
- Composition of canonical units

Scaling of SAE (dead neurons)

Anthropic - “dead” features as those which were not active over a sample of 10^7 tokens. The proportion of dead features was roughly 2% for the 1M SAE, 35% for the 4M SAE, and 65% for the 34M SAE

ReLU SAE is not so good at higher sparsity and bigger number of lattents



(a) At a fixed number of latents ($n = 32768$), TopK has a better reconstruction-sparsity trade off than ReLU and ProLU, and is comparable to Gated.

(b) At a fixed sparsity level ($L_0 = 128$), scaling laws are steeper for TopK than ReLU.[6]

Figure 2: Comparison between TopK and other activation functions.

SCALING AND EVALUATING SPARSE AUTOENCODERS

Leo Gao* **Tom Dupré la Tour**† **Henk Tillman**† **Gabriel Goh** **Rajan Troll**
Alec Radford **Ilya Sutskever** **Jan Leike** **Jeffrey Wu**†
OpenAI
lg@openai.com

ABSTRACT

Sparse autoencoders provide a promising unsupervised approach for extracting interpretable features from a language model by reconstructing activations from a sparse bottleneck layer. Since language models learn many concepts, autoencoders need to be very large to recover all relevant features. However, studying the properties of autoencoder scaling is difficult due to the need to balance reconstruction and sparsity objectives and the presence of dead latents. We propose using k-sparse autoencoders (Makhzani & Frey, 2013) to directly control sparsity, simplifying tuning and improving the reconstruction-sparsity frontier. Additionally, we find modifications that result in few dead latents, even at the largest scales we tried. Using these techniques, we find clean scaling laws with respect to autoencoder size and sparsity. We also introduce several new metrics for evaluating feature quality based on the recovery of hypothesized features, the explainability of activation patterns, and the sparsity of downstream effects. These metrics all generally improve with autoencoder size. To demonstrate the scalability of our approach, we train a 16 million latent autoencoder on GPT-4 activations for 40 billion tokens. We release code and autoencoders for open-source models, as well as a visualizer.

TopK Sparse Autoencoders

A.2 Auxiliary loss

We define an auxiliary loss (AuxK) similar to "ghost grads" [Jermyn and Templeton, 2024] that models the reconstruction error using the top-k_{aux} dead latents (typically $k_{\text{aux}} = 512$). Latents are flagged as dead during training if they have not activated for some predetermined number of tokens (typically 10 million). Then, given the reconstruction error of the main model $e = x - \hat{x}$, we define the auxiliary loss $\mathcal{L}_{\text{aux}} = ||e - \hat{e}||_2^2$, where $\hat{e} = W_{\text{dec}} z$ is the reconstruction using the top-k_{aux} dead latents. The full loss is then defined as $\mathcal{L} + \alpha\mathcal{L}_{\text{aux}}$, where α is a small coefficient (typically $1/32$).

We use a k-sparse autoencoder [Makhzani and Frey, 2013], which directly controls the number of active latents by using an activation function (TopK) that only keeps the k largest latents, zeroing the rest. The encoder is thus defined as:

$$z = \text{TopK}(W_{\text{enc}}(x - b_{\text{pre}})) \tag{2}$$

and the decoder is unchanged. The training loss is simply $\mathcal{L} = ||x - \hat{x}||_2^2$.

Using k-sparse autoencoders has a number of benefits:

- It removes the need for the L1 penalty (shrinkage problem).
- It enables setting the L0 directly, as opposed to tuning an L1 coefficient λ
- It empirically outperforms baseline ReLU autoencoders on the sparsity-reconstruction frontier (Figure 2a), and this gap increases with scale (Figure 2b).
- It increases monosemanticity of random activating examples by effectively clamping small activations to zero.

A lot of SCALING!!!!

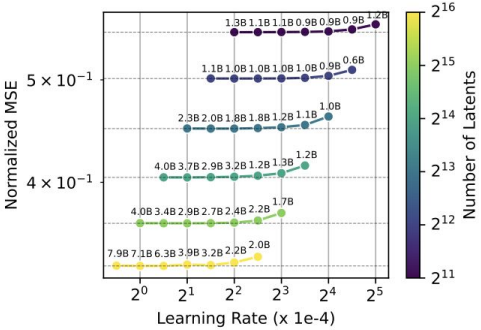


Figure 3: Varying the learning rate jointly with the number of latents. Number of tokens to convergence shown above each point.

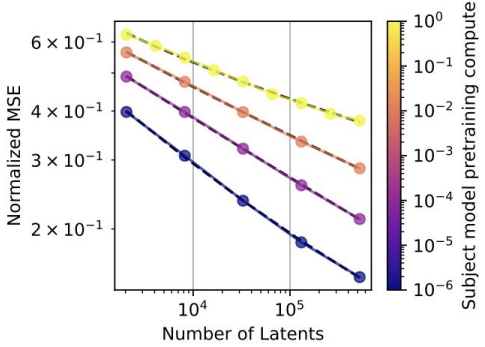
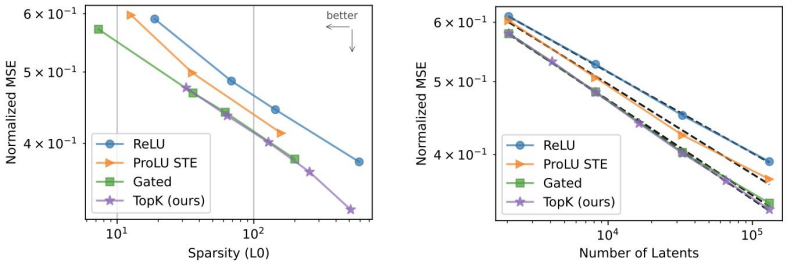


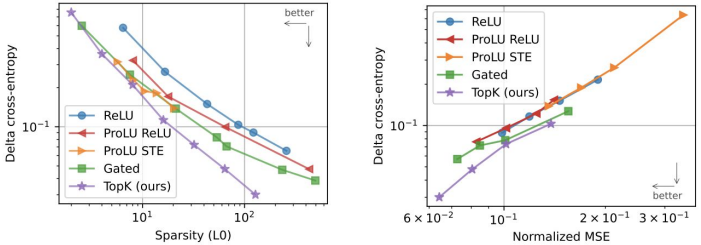
Figure 4: Larger subject models in the GPT-4 family require more latents to get to the same MSE ($k = 32$).



(a) At a fixed number of latents ($n = 32768$), TopK has a better reconstruction-sparsity trade off than ReLU and ProLU, and is comparable to Gated.

(b) At a fixed sparsity level ($L_0 = 128$), scaling laws are steeper for TopK than ReLU.[6]

Figure 2: Comparison between TopK and other activation functions.



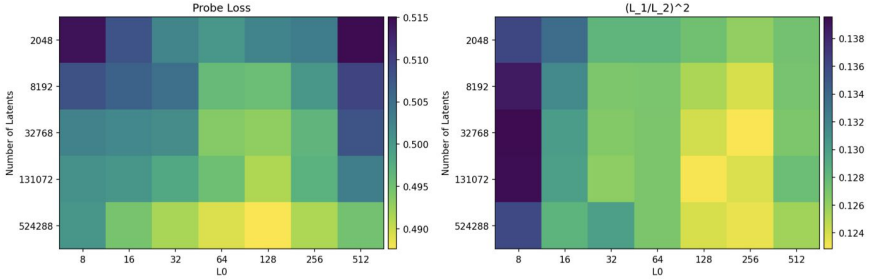
(a) For a fixed number of latents ($n = 2^{17} = 131072$), the downstream-loss/sparsity trade-off is better for TopK autoencoders than for other activation functions.

(b) For a fixed sparsity level ($L_0 = 128$), a given MSE level leads to a lower downstream-loss for TopK autoencoders than for other activation functions.

Figure 5: Comparison between TopK and other activation functions on downstream loss. Comparisons done for GPT-2 small, see Figure 13 for GPT-4.

If we can choose the sparsity, how sparse SAE should be?

Promisingly, models trained with larger k have latents with sparser effects. However, the trend reverses at k=512, indicating that as k approaches d=768, the autoencoder learns latents with less interpretable effects.

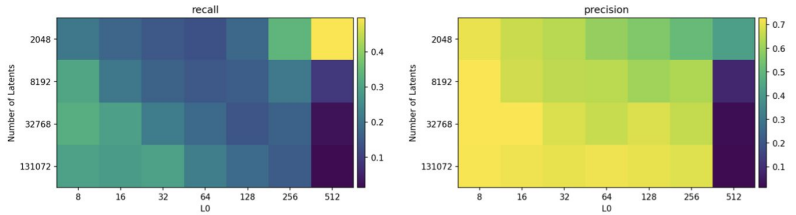


(a) Probe loss (b) Logit diff sparsity

Figure 6: The probe loss and logit diff metrics as a function of number of total latents n and active latents k, for GPT-2 small autoencoders. More total latents (higher n) generally improves all metrics (yellow = better). Both metrics are worse at $L_0 = 512$, a regime in which solutions are dense (see subsection E.5).

In general, autoencoders with more total latents and fewer active latents are easiest to model with N2G (autointerp). Wider and sparser SAE code more canonical features.

Figure 24: TopK beats ReLU on N2G F1 score. Its N2G explanations have noticeably higher recall, but worse precision. (higher is better)



(a) Recall of N2G explanations $P(\text{n2g} > 0 | \text{act} > 0)$

(b) Precision of N2G explanations $P(\text{act} > 0 | \text{n2g} > 0)$

Figure 25: Neuron2graph precision and recall. The average autoencoder latent is generally easier to explain as k decreases and n increases. However, $n = 2048, k = 512$ latents are easy to explain since many latents activate extremely densely (see Section E.5).

Progressive recovery and Multi-TopK

Training with TopK only gives a progressive code *up to the value of k used during training*.

Multi-TopK **improves** generalization to larger k

Limitations: TopK forces every token to use exactly k latents, which is likely suboptimal.

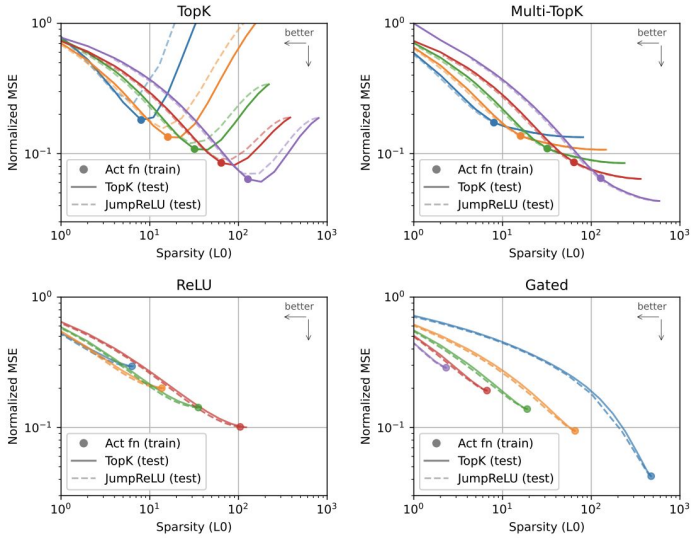


Figure 10: Sparsity levels can be changed at test time by replacing the activation function with either TopK(k) or JumpReLU(θ), for a given value k or θ. TopK tends to overfit to the value of k used during training, but using Multi-TopK improves generalization to larger k.

Fun Part!!

- Paper released on ArXiv: 6 Jun 2024
- Submitted to ICLR 2025; 6 months after
- Got Oral
- No Progressive recovery, Multi-TopK and Fixed sparsity versus fixed threshold
- One 3 (reject), One 8, Two 10

Paper Thoughts

Public Comment by Neel Nanda 15 Nov 2024 at 00:48 (modified: 27 Nov 2024 at 11:52) Everyone Revisions

Comment:

Commenting as an unrelated mechanistic interpretability researcher, I consider this to be **one of the most important mechanistic interpretability papers of the past year, and worthy of being highlighted at the conference**. Sparse autoencoders seemed a highly promising technique, but had so far mostly been demonstrated on small-ish models, with limited proxy metrics, and had several known issues like shrinkage. This work:

- Efficiently scaled SAEs to GPT-4 - an extremely impressive engineering feat, and the furthest they have yet been scaled, along with contributing to our knowledge for how to efficiently train large SAEs with things like scaling laws. A key risk in mech interp is that the techniques used only work for small models, this work showed that was false for SAEs.
- Showed that the TopK architecture was an improvement, both in terms of performance and in terms of resolving issues like shrinkage, along with advantages like being able to set the L0. This is a practically useful finding for all SAE work, and some later papers/open source SAE releases have used this architecture or built on it. Though topK SAEs were a known technique in other contexts, demonstrating they were valuable for interpretability was a real contribution.
- Explored creative ways of measuring SAE performance - the standard methods that we eg used in Rajamanoharan et al (interpretability via max activating dataset examples, and sparsity-reconstruction trade-offs) are useful, but only just proxies. The methods used in this paper are also just proxies, of course, but I appreciate that a broader range was explored. In particular, sparse probing is a very reasonable approach that I hadn't seen used before with SAEs - make a dataset for a concept we expect to be learned, and see if it's in there

(Disclosure: I was not at all involved in this paper, but I do know the authors and likely have some positive bias due to that. No one asked me to write this)

(Note also: I don't feel confident in the norms here, so feel free to ignore this comment if these kinds of thoughts are not welcome! I don't see other researchers doing this, but I figure that as ICLR has made a deliberate choice to allow public comments during the rebuttal process, they likely want this kind of public feedback from uninvolved researchers, so long as it's constructive)

Add: Public Comment

Public Comment by Johan Edstedt

Public Comment by Johan Edstedt 20 Nov 2024 at 09:44 Everyone

Comment:

This basically breakes double blind. Maybe not a suitable comment to make.

Add: Public Comment

????

Public Comment by Naomi Saphra 20 Nov 2024 at 18:40 Everyone

Comment:

I like the paper, but direct appeals to the referees from high-profile allies cannot be part of the review process. If public discussion opens the ACs to Oscar-style award campaigns, then ICLR shouldn't allow comments at all until after decisions are finalized.

Add: Public Comment

Replying to Official Comment by Authors

Official Comment by Reviewer CUAP

Official Comment by Reviewer CUAP 28 Nov 2024 at 13:12 Everyone

Comment:

I address below the response in order:

1. I believe this is such a crucial component that it needs to be in the paper. Otherwise the paper is a very good proof of concept but it is not mature enough yet for a publication. The proposed TopK is already seen as ineffective evident by this recent paper (https://openreview.net/pdf?id=d4dpOCqybL). Thanks to the public contributor for drawing my attention to their work.

Replying to Official Comment by Reviewer CUAP

Official Comment by Reviewer c1TT

Official Comment by Reviewer c1TT 28 Nov 2024 at 14:32 Everyone

Comment:

Dear fellow reviewer.

On your point 1. That work is contemporaneous under ICLR's guidelines, and build on top of the work we are reviewing here. A further instance of work in this line is JumpReLU SAEs, as they take the work further by learning k.

Since this paper came out sooner than JumpReLU and batch top-k, and both works were inspired by it, we should consider wether we want the impact this paper had after publication to be a positive factor or a negative factor in our evaluation.

Sincerely, your fellow reviewer.

Add: **Public Comment**

Replying to Official Comment by Reviewer c1TT

Official Comment by Reviewer CUAP

Official Comment by Reviewer CUAP 28 Nov 2024 at 15:23 Everyone

Comment:

The other work was referenced as an additional data point that the topK approach is very strict and the variable k sparsity is a potential solution.

I do not see how this paper came out sooner than batch top-k although batch top-k is already published while we are still reviewing this paper here? Considering this is a double-blind review process, I trust that this paper and any others addressing similar topics were developed independently by separate teams, without any intent to influence my impartial evaluation of this work. If these teams happen to be reviewing each other's papers, I trust that they will adhere to the highest standards of fairness and objectivity in their evaluations.

Add: **Public Comment**

Timeline

[Submitted on 6 Jun 2024]

Scaling and evaluating sparse autoencoders

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, Jeffrey Wu

BatchTopK Sparse Autoencoders

Bart Bussmann, Patrick Leask, Neel Nanda

Published: 10 Oct 2024, Last Modified: 09 Nov 2024 SciForDL Poster Everyone Revisions BibTeX CC BY 4.0

[Submitted on 9 Dec 2024]

BatchTopK Sparse Autoencoders

Bart Bussmann, Patrick Leask, Neel Nanda

[3] Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders, 2024.

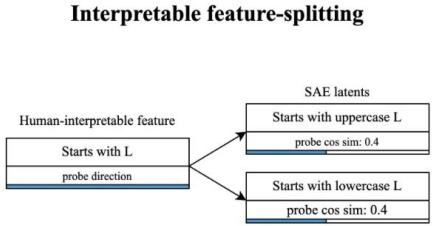
Scaling and evaluating sparse autoencoders

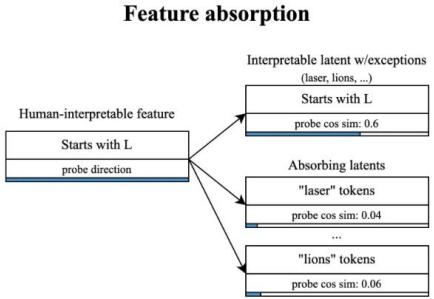
Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, Jeffrey Wu

Published: 22 Jan 2025, Last Modified: 27 Feb 2025 ICLR 2025 Oral Everyone Revisions BibTeX CC BY 4.0

Problems with Sparse Autoencoders

- ~~How to interpret SAE features~~
- ~~Shrinkage Problem~~
- ~~Scaling of SAE (dead latents)~~
- Feature Splitting
- Composition of canonical units





Problems with Sparse Autoencoders

- ~~How to interpret SAE features~~
- ~~Shrinkage Problem~~
- ~~Scaling of SAE (dead latents)~~
- ~~Feature Splitting~~
- Composition of canonical units

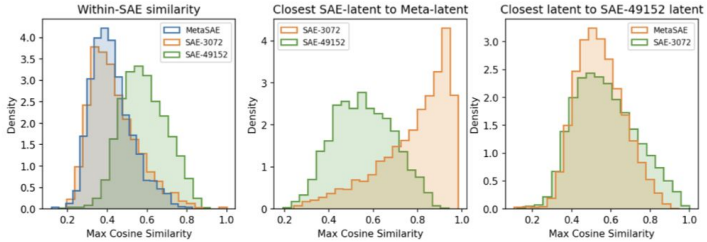


Figure 6: Cosine similarity between SAE latents and meta-SAE latents. Note the high maximum cosine similarity between latents from a meta-SAE with 2304 latents, and a standard SAE with 3072 latents.

Faculty of Mathematics and Information Sciences

WARSAW UNIVERSITY OF TECHNOLOGY

Introduction to Sparse Autoencoders

Scaling and evaluating sparse autoencoders (2024)