# The Missing Curve Detectors of InceptionV1: Applying Sparse Autoencoders to InceptionV1 Early Vision

**Liv Gorton** [1]

## Abstract

Recent work on sparse autoencoders (SAEs) has shown promise in extracting interpretable features from neural networks and addressing challenges with polysemantic neurons caused by superposition. In this paper, we apply SAEs to the early vision layers of InceptionV1, a well-studied convolutional neural network, with a focus on curve detectors. Our results demonstrate that SAEs can uncover new interpretable features not apparent from examining individual neurons, including additional curve detectors that fill in previous gaps. We also find that SAEs can decompose some polysemantic neurons into more monosemantic constituent features. These findings suggest SAEs are a valuable tool for understanding InceptionV1, and convolutional neural networks more generally.

## 1. Introduction

The original project of mechanistic interpretability was a thread of work (Cammarata et al., 2020a) trying to reverse engineer InceptionV1 (Szegedy et al., 2014). However, this work had a major limitation that was recognised at the time: polysemantic neurons which respond to unrelated stimuli. It was hypothesised that this was due to *superposition*, where combinations of neurons are used to represent features (Arora et al., 2018; Elhage et al., 2022). This was a significant barrier to the original Circuits project, since uninterpretable polysemantic neurons were a roadblock for neuron-based circuit analysis.

Since then, significant progress has been made on addressing superposition. In particular, it has been found that applying a variant of dictionary learning (Olshausen & Field, 1997; Elad, 2010) called a sparse autoencoder (SAE) can extract interpretable features from language models (Cunningham et al., 2023; Yun et al., 2021; Bricken et al., 2023).

[1]Cellware Labs, San Francisco, CA, United States of America. Correspondence to: Liv Gorton <liv@livgorton.com>.

*Copyright 2024 by the author(s).*

| Layer | $\lambda$ | Expansion Factor |
|---|---|---|
| conv2d0 | $1e-7$ | $2\times$ |
| conv2d1 | $5e-6$ | $4\times$ |
| conv2d2 | $5e-6$ | $4\times$ |
| mixed3a | $3.5e-6$ | $8\times$ |
| mixed3b | $3.5e-6$ | $8\times$ |

*Table 1.* Default sparse autoencoder hyperparameters for each layer.

This creates an opportunity for us to return to the original project of understanding InceptionV1 with new tools. In this paper, we apply sparse autoencoders to InceptionV1 early vision (Olah et al., 2020) and especially curve detectors (Cammarata et al., 2020b; 2021). We find that at least some previously uninterpretable neurons contribute to representing interpretable features. We also find "missing features" which are invisible when examining InceptionV1 in terms of neurons but are revealed by features. This includes a number of additional curve detector features which were missing from the well-studied neuron family.

## 2. Methods

### 2.1. Sparse Autoencoders

Following previous work, we decompose any activation vector $x$ into $x \simeq b + \sum_i f_i(x) d_i$ where $f_i(x)$ is the feature activation and $d_i$ is the feature direction. Throughout the paper, we'll denote these features as `<layer>/f/<i>` (and neurons as `<layer>/n/<i>`).

We achieve this decomposition using SAEs trained on image activations sampled from InceptionV1 over its training set, ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Addison Howard, 2018). Recall that, as a conv net, InceptionV1 has a grid of activation vectors corresponding to image positions. For each of the images in ILSVRC, we sample activations from 10 such positions. We trained for approximately 500 epochs on a dataset of shuffled activations (until a total of 5 billion activations were trained on).

Unless otherwise stated, the hyperparameters used for each

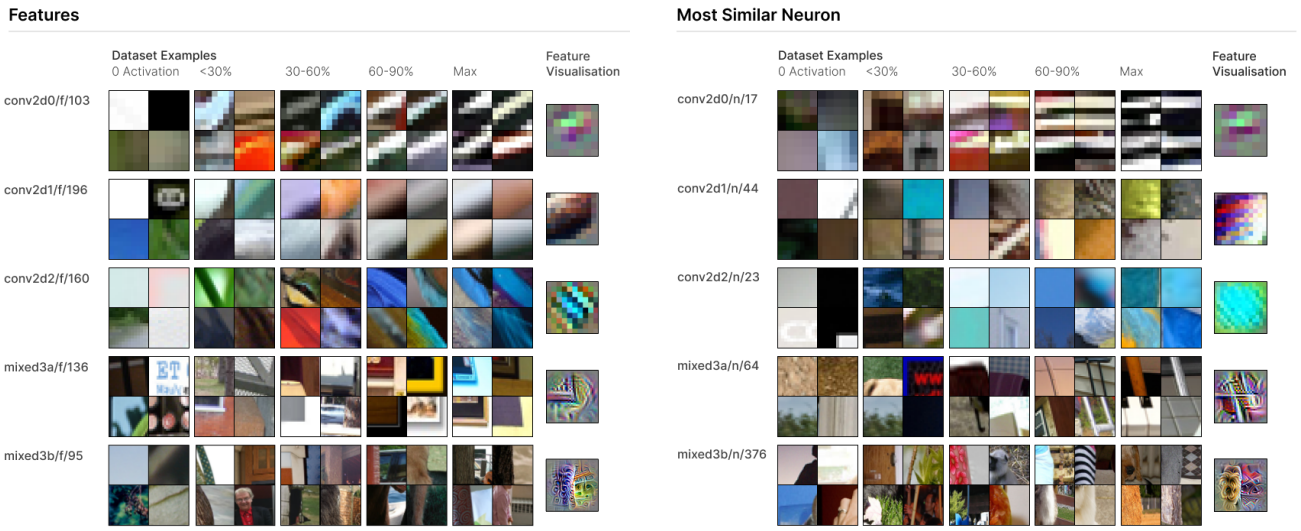


*Figure 1.* This figure presents examples of one interpretable feature learned by the SAE at each layer. For each feature, we present dataset examples at varying activation levels and a feature visualisation. We also show the neuron most similar to this feature (i.e. the neuron with the largest weight to it), and corresponding dataset examples and feature visualisations.

SAE can be found in Table 1. Our specific SAE setup followed some modifications recommended by Conerly et al. (2024). In particular, we didn't constrain the decoder norm and instead scaled feature activations by it.[1] We also made two additional modifications to work around compute constraints, described in the following subsections.

#### 2.1.1. Over Sampling Large Activations

In InceptionV1, the majority of image positions produce small activations (for example, on backgrounds). To avoid spending lots of compute modelling these small activations, we oversampled large activations by sampling activations from positions in the image proportional to their activation magnitude.

#### 2.1.2. Branch Specific SAEs

InceptionV1 divides these layers into branches with different convolution sizes (Szegedy et al., 2014). For layers `mixed3a` and `mixed3b`, we perform dictionary learning only on the `3x3` and `5x5` convolutional branches respectively. This seems relatively principled as with such limited space, features are likely isolated to a branch that has the most advantageous filter size.

### 2.2. Analysis Methods

In analysing features, we primarily rely on *dataset examples* and *feature visualisation* (Erhan et al., 2009; Olah et al., 2017). Dataset examples show how a neuron or feature behaves on distribution, while feature visualisation helps isolate what causes a feature to activate. This is supplemented by more specific methods like synthetic dataset examples from Cammarata et al. (2020b) for curve detectors.

#### 2.2.1. Dataset Examples

We collect dataset examples where the feature activates over the ILSVRC dataset (Addison Howard, 2018), since that is what InceptionV1 was trained on. In using dataset examples, it's important to not just look at top examples to avoid interpretability illusions (Bolukbasi et al., 2021). Instead, we collect dataset examples randomly sampled to be within ten activation intervals, evenly spaced between 0 and the feature's maximum observed activation.

#### 2.2.2. Feature Visualisation

Feature visualisation is performed following Olah et al. (2017). We use the Lucent library (Swee Kiat, 2021), with some small modifications to their port of InceptionV1 to ensure compatibility with the original model.[2]

[1] This seemed to help avoid dead neurons.

[2] It was important to us to ensure our model was identical to the one studied by the original Circuits thread. In cross-validating the torch version with the original TensorFlow one, we found that some small differences were introduced to the local response normalisation layer when Lucent ported the model to PyTorch. These significantly modify model behaviour. A PR for these fixes can be found at: https://github.com/greentfrapp/lucent/pull/51

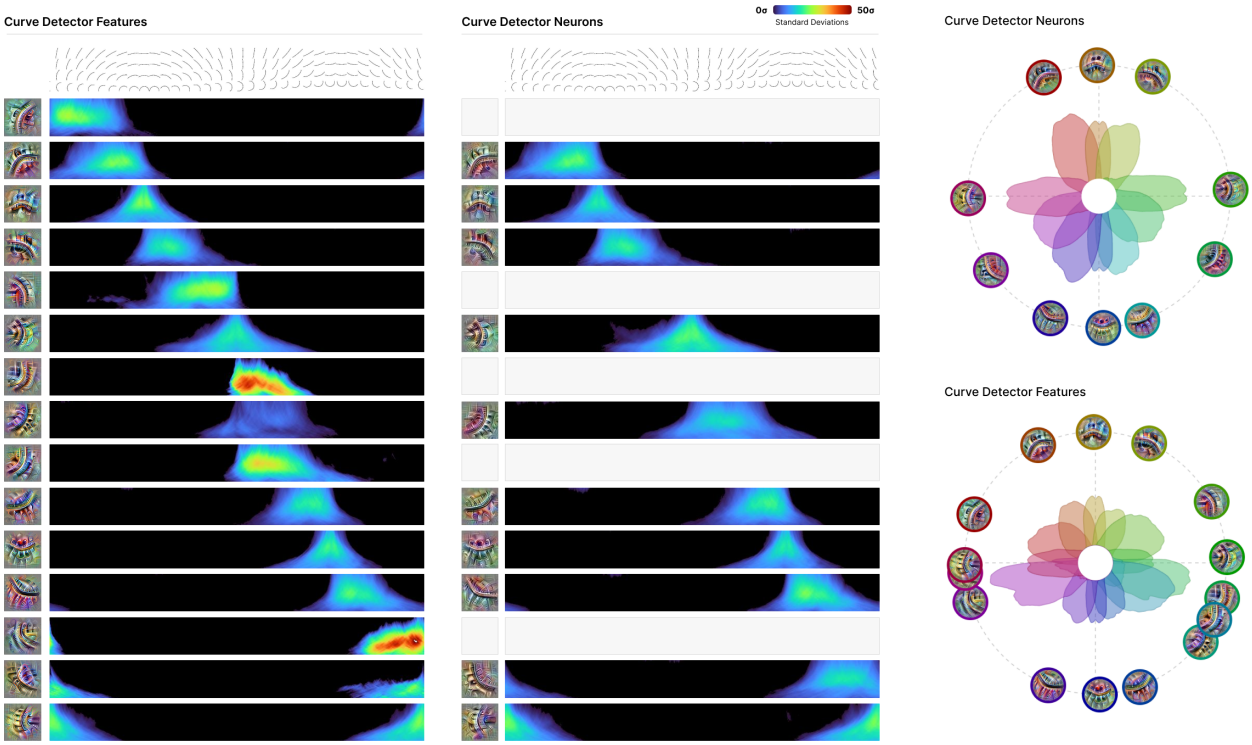


*Figure 2.* **Left & Middle:** Synthetic data plots, showing how curve detector features (left) and neurons (middle) respond to synthetic curve stimuli, as in Cammarata et al. (2020b). Activations are denominated in standard deviations. A subset of the new curve detectors are shown, with gaps between the neurons representing a previously missing curve detector. **Right:** The same data is shown on radial tuning curve plots, again following Cammarata et al. (2020b). The curve radius at a given orientation denotes activation, again measured in standard deviations of activation.

## 3. Results

This paper is a very early report on applying SAEs to InceptionV1. We present a variety of *examples* of SAEs producing more interpretable features in various ways. Our sense is that these results are representative of a more general trend, but we don't aim to defend this in a systematic way at this stage.

Concretely, we claim the following:

1. There exist relatively interpretable SAE features which were not visible in terms of neurons.
2. There are new curve detector features which fill missing "gaps" among curve detectors found by Cammarata et al. (2020b).
3. Some polysemantic neurons can be seen to decompose into more monosemantic features.

Each of the following subsections will support one of these claims.

### 3.1. SAEs Learn New, Interpretable Features

Figure 1 presents five SAE features, one from each "early vision" layer we applied SAEs to. (These are the same five layers studied by Olah et al. (2020).) In each case, the feature is quite interpretable across the activation spectrum (see dataset examples in Figure 1). The neurons most involved in representing the feature are, to varying extents, vaguely related or polysemantic neurons, which are difficult to interpret.

For example, `conv2d1/f/196` appears to be a relatively monosemantic brightness gradient detector. However, the most involved neuron `conv2d1/n/44` was unable to be categorised by Olah et al. (2020) and appears quite polysemantic. (In fact, our SAE decomposes `conv2d1/n/44` into a variety of features responding to brightness gradients of various orientations, colour contrast, and complex Gabor filters.) On the other hand, there are features which have more monosemantic corresponding neurons, but are still more precise and monosemantic. For example, `mixed3b/f/95` is a relatively generic boundary detector receiving its largest weight from `mixed3b/n/376`, which

responds to boundaries with a preference for fur on one side. The SAE appears to decompose `mixed3b/n/376` into a generic boundary detector and a fur-specific boundary detector in the same orientation.

### 3.2. SAEs Discover Additional Curve Detectors

Curve detectors are likely the best characterised neurons in InceptionV1, studied in detail by Cammarata et al. (2020b; 2021). They were also found to be concentrated within the `5x5` branch of `mixed3b` by Voss et al. (2021). This existing literature makes them a natural target for understanding the behaviour of SAEs.

In Figure 2, we use the "synthetic data" approach of Cammarata et al. (2020b) to systematically show how our curve detector features respond to curve stimuli of a range of orientations and radii. We find curve detector features which closely match the curve detector neurons known to exist, and others which fill in "gaps" where there wasn't a curve detector for a particular orientation. Following existing work, we report the stimuli activations in terms of standard deviations to make it clear how unusual the kind of intense reaction they produce is. To see the way the new curve detector features "fill in" gaps, we also create "radial tuning curve" plots from Cammarata et al. (2020b), seen in Figure 2. The features fill in several gaps where there is no neuron responding to curves in a given orientation.

### 3.3. SAEs Split Polysemantic Neurons

We've already mentioned several examples of polysemantic neurons being split into more monosemantic features. In this section, we'll further support this pattern with a particularly striking example. Olah et al. (2020) reported the existence of "double curve" detector neurons, which respond to curves in two very different orientations. It's very natural to suspect these are an example of superposition.

As shown in Figure 3, the double curve detector `mixed3b/n/359` primarily decomposes into three features: two monosemantic curve detectors and a double curve detector. We suspected the additional double curve detector might be a failure of the SAE. To study this, we trained two additional SAEs with higher L1 coefficients. As L1 the coefficient increases, the maximum activation of the analogous double curve features falls, while the left and right curve features maximum activations correspondingly increases (see bottom of Figure 3).

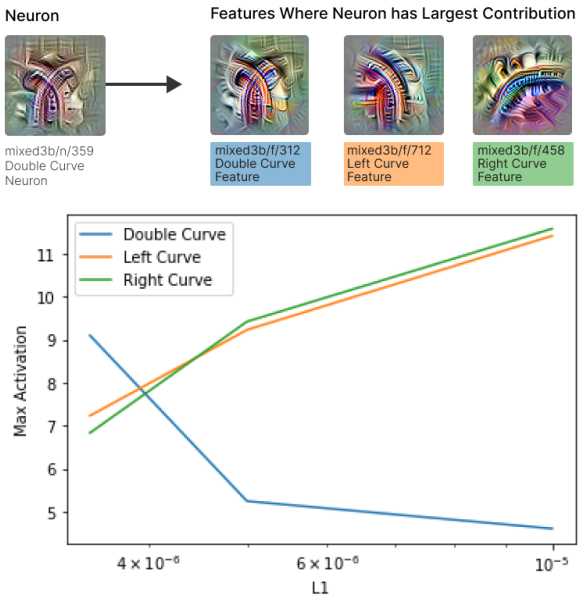


*Figure 3.* **Top:** The three SAE features most strongly weighted to `mixed3b/n/359`, previously identified by Olah et al. (2020) as a double curve detector that was likely polysemantic. **Bottom:** Max activations of analogous features across SAEs with different L1 coefficients. As L1 increases, the double curve feature becomes smaller, while the left and right curves correspondingly grow.

## 4. Conclusion

Recent progress on SAEs appears to open up exciting new mechanistic interpretability directions in the context of language models, such as circuit analysis based on features (Marks et al., 2024). Our results suggest that this promise isn't limited to language models. At the very least, SAEs seem promising for understanding InceptionV1.

## Acknowledgements

I would like to thank my co-founder, Shae McLaughlin, for her ongoing support. I am also grateful to Chris Olah for insightful discussions and feedback that helped shape this work.

## Impact Statement

This paper presents work whose goal is to advance the field of AI safety. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

## References

Addison Howard, Eunbyung Park, W. K. Imagenet object localization challenge, 2018. URL https://kaggle.com/competitions/imagenet-object-localization-challenge.

Arora, S., Li, Y., Liang, Y., Ma, T., and Risteski, A. Linear

algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495, 2018.

Bolukbasi, T., Pearce, A., Yuan, A., Coenen, A., Reif, E., Viégas, F., and Wattenberg, M. An interpretability illusion for bert. *arXiv preprint arXiv:2104.07143*, 2021.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Hatfield-Dodds, Z., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Cammarata, N., Carter, S., Goh, G., Olah, C., Petrov, M., Schubert, L., Voss, C., Egan, B., and Lim, S. K. Thread: Circuits. *Distill*, 2020a. doi: 10.23915/distill.00024. https://distill.pub/2020/circuits.

Cammarata, N., Goh, G., Carter, S., Schubert, L., Petrov, M., and Olah, C. Curve detectors. *Distill*, 2020b. doi: 10.23915/distill.00024.003. https://distill.pub/2020/circuits/curve-detectors.

Cammarata, N., Goh, G., Carter, S., Voss, C., Schubert, L., and Olah, C. Curve circuits. *Distill*, 2021. doi: 10.23915/distill.00024.006. https://distill.pub/2020/circuits/curve-circuits.

Conerly, T., Templeton, A., Bricken, T., Marcus, J., and Henighan, T. Update on how we train saes, 2024.

Cunningham, H., Ewart, A., Riggs, L., Huben, R., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models, 2023.

Elad, M. *Sparse and redundant representations: from theory to applications in signal and image processing*, volume 2. Springer, 2010.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Erhan, D., Bengio, Y., Courville, A., and Vincent, P. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

Marks, S., Rager, C., Michaud, E. J., Belinkov, Y., Bau, D., and Mueller, A. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*, 2024.

Olah, C., Mordvintsev, A., and Schubert, L. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. An overview of early vision in inceptionv1. *Distill*, 2020. doi: 10.23915/distill.00024.002. https://distill.pub/2020/circuits/early-vision.

Olshausen, B. A. and Field, D. J. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

Swee Kiat, L. Lucent, 2021. URL https://github.com/greentfrapp/lucent.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL http://arxiv.org/abs/1409.4842.

Voss, C., Goh, G., Cammarata, N., Petrov, M., Schubert, L., and Olah, C. Branch specialization. *Distill*, 2021. doi: 10.23915/distill.00024.008. https://distill.pub/2020/circuits/branch-specialization.

Yun, Z., Chen, Y., Olshausen, B. A., and LeCun, Y. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors. *arXiv preprint arXiv:2103.15949*, 2021.

## A. Branch Specialisation of Learned Features

After the first three convolution layers, InceptionV1 has a branched structure (Szegedy et al., 2014). Prior work has found that the early branches often specialise; for example, `mixed3b` seems specialize on curve and boundary detectors (Voss et al., 2021).

In the main portion of this paper, we trained sparse autoencoders on these branches as a principled way to save compute, with the intuition that the most computationally efficient way to represent a given feature is to utilise the most advantageous (i.e. the smallest) convolution size that can capture the feature. If this were true, the combination of neurons that represent a given feature would exist on the same branch.

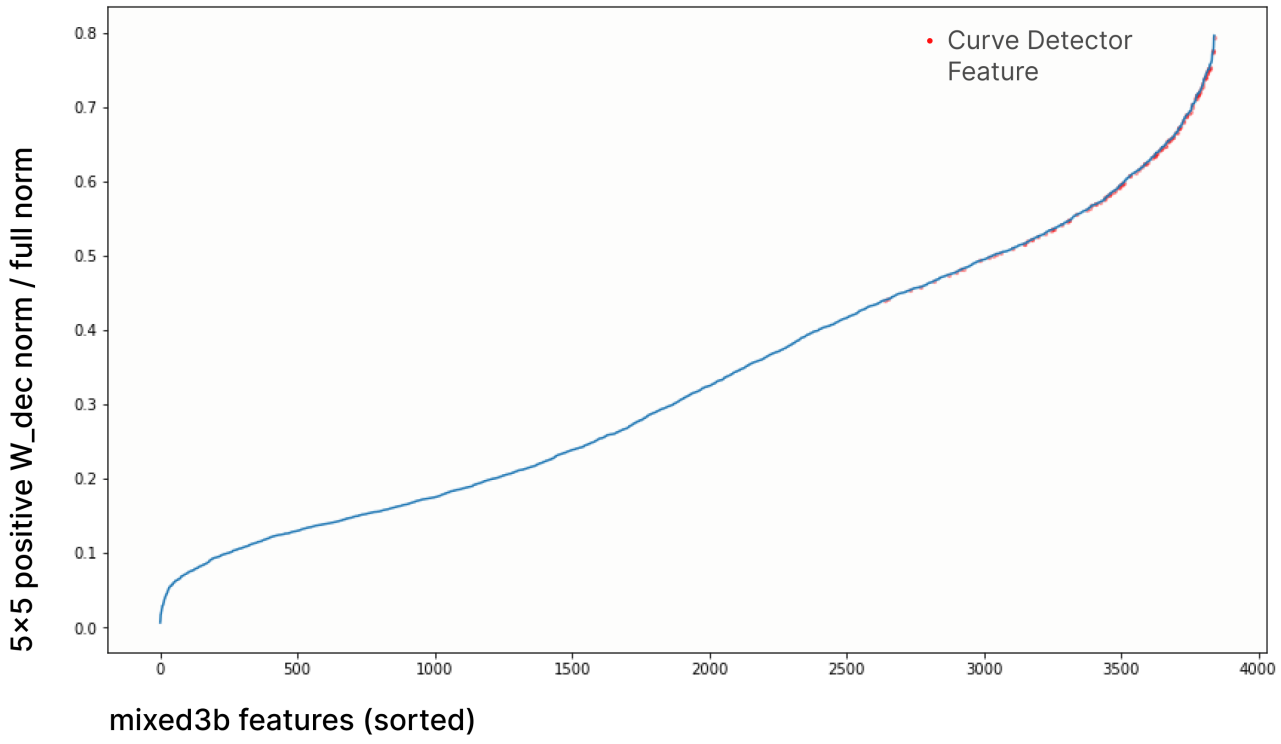


*Figure 4.* The extent to which each learned feature of a sparse autoencoder trained across all branches of `mixed3b`, is represented by neurons on the $5 \times 5$ branch. Negative weights were excluded when computing the norm. Curve detectors are amongst the features with the greatest branch specialisation.

We now train sparse autoencoders on the entire `mixed3b` layer (ie. all branches jointly). We find that features are not uniformly distributed across branches. Some features are highly branch specific (as much as 80% localized to `5x5`) and others are highly concentrated on other branches (nearly 0% localized to `5x5`). On the other hand, while it's not clear where the boundaries for "concentrated on a branch" should be (especially since `5x5` is smaller than other branches), but it seems likely that many features should be understood as spread across branches.

Interestingly, the features concentrated on `5x5` disproportionately tend to be curve detectors, consistent with Voss et al. (2021).

What about other layers? In a preliminary SAE experiment on `mixed5b`, we find that a small fraction of features are very concentrated on the `5x5` branch, while most are more distributed. `mixed4e` preliminarily looks more similar to `mixed3b`. The nature of those branch-localized features is left to future work. Perhaps there is previously unknown branch specialization, which was previously obscured by superposition.

Figure 5 shows how specialised each feature is to the $5 \times 5$ branch of `mixed3b`. No features are represented entirely on the

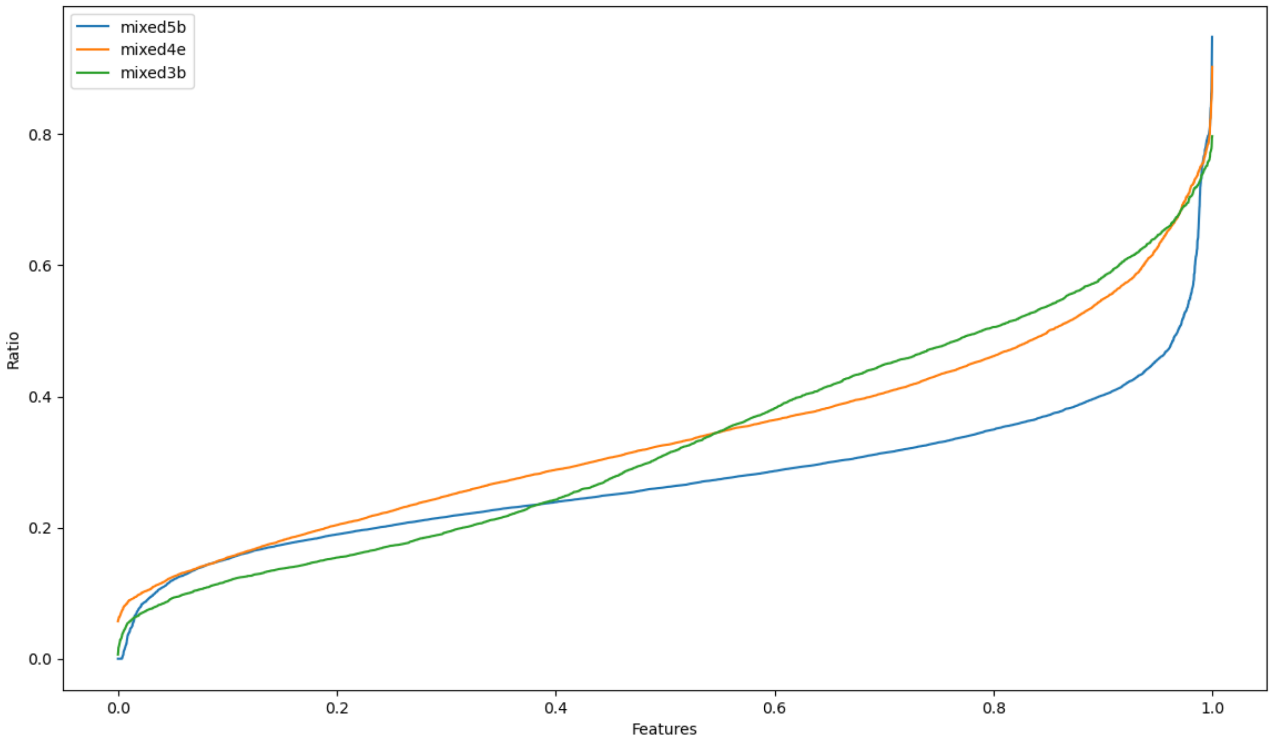


*Figure 5.* The extent to which each learned feature of sparse autoencoders trained across all branches of `mixed3b`, `mixed4e`, and `mixed5b` are represented by neurons on the $5 \times 5$ branch. Negative weights were excluded when computing the norm.

$5 \times 5$ branch, with the most specialised feature having approximately $80\%$ of its positive weighted neurons on it.

It is worth emphasising that this doesn't change the correctness the features identified in this paper but instead means that branch-specific SAEs will have features still hidden by cross-branch superposition.