Interpreting Neural Networks through the Polytope Lens

Sid Black∗ Lee Sharkey∗ Leo Grinsztajn Eric Winsor Dan Braun
Jacob Merizian Kip Parker Carlos Ramón Guevara Beren Millidge
Gabriel Alfour Connor Leahy

Abstract

Mechanistic interpretability aims to explain what a neural network has learned at a nuts-and-bolts level. What are the fundamental primitives of neural network representations? What basic objects should we use to describe the operation of neural networks mechanistically? Previous mechanistic descriptions have used individual neurons or their linear combinations to understand the representations a network has learned. But there are clues that neurons and their linear combinations are not the correct fundamental units of description—directions cannot describe how neural networks use nonlinearities to structure their representations. Moreover, many instances of individual neurons and their combinations are polysemantic (i.e. they have multiple unrelated meanings). Polysemanticity makes interpreting the network in terms of neurons or directions challenging since we can no longer assign a specific feature to a neural unit. In order to find a basic unit of description that doesn’t suffer from these problems, we zoom in beyond just directions to study the way that piecewise linear activation functions (such as ReLU) partition the activation space into numerous discrete polytopes. We call this perspective the ‘polytope lens’. Although this view introduces new challenges, we think they are surmountable and that more careful consideration of the impact of nonlinearities is necessary in order to build better high-level abstractions for a mechanistic understanding of neural networks. The polytope lens makes concrete predictions about the behavior of neural networks, which we evaluate through experiments on both convolutional image classifiers and language models. Specifically, we show that polytopes can be used to identify monosemantic regions of activation space (while directions are not in general monosemantic) and that the density of polytope boundaries reflect semantic boundaries. We also outline a vision for what mechanistic interpretability might look like through the polytope lens.

∗equal contribution

Preprint. Under review.
1 Introduction

How should we carve a neural network at the joints? Traditionally, mechanistic descriptions of neural circuits have been posed in terms of neurons, or linear combinations of neurons also known as ‘directions’. Describing networks in terms of these neurons and directions has let us understand a surprising amount about what they’ve learned (Cammarata et al., 2020a). But these descriptions often possess undesirable properties - such as polysemanticity and inability to account for nonlinearity - which suggest to us that they don’t always carve a network at its joints.

If not neurons or directions, then what should be the fundamental unit of a mechanistic description of what a neural network has learned? Ideally, we would want a description in terms of some object that throws away unnecessary details about the internal structure of a neural network while simultaneously retaining what’s important. In other words, we’d like a less ‘leaky’ abstraction for describing a neural network’s mechanisms.

We propose that a particular kind of mathematical object – a ‘polytope’ – might serve us well in mechanistic descriptions of neural networks with piecewise-linear activations.¹ We

¹And, with some relaxations to ‘soft’ polytopes, the polytope lens might also let us mechanistically describe neural networks with activations such as GELU and Swish. Some prior work exists that extends the polytope lens to such activations (Balestriero and Baraniuk, 2018b). See the Appendix C for further discussion.
believe they might let us build less leaky abstractions than individual neurons and directions alone, while still permitting mechanistic understandings of neural networks of comparable length and complexity.

To help explain how the polytope lens could underlie mechanistic descriptions of neural networks, we first look at the problems that arise when using individual neurons (both biological and artificial) and then when using directions as the basic units of description and suggest how this perspective offers a potential solution.

1.1 Are individual neurons the fundamental unit of neural networks?

Studying the function of single neurons has a long history. The dominant view in neuroscience for approximately one hundred years was the ‘neuron doctrine’ (Yuste, 2015). The neuron doctrine contended that the way to understand neural networks is to understand the responses of individual neurons and their role in larger neural circuits. This led to significant successes in the study of biological neural circuits, most famously in the visual system. Early and important discoveries within this paradigm included cells in the frog retina that detect small patches of motion (fly detectors) (Lettvin et al., 1959); cells in the visual cortex with small receptive fields that detect edges (Hubel and Wiesel, 1962), cells in the higher visual system that detect objects as complex as faces (Sergent et al., 1992), and many even highly abstract multimodal concepts appear to be represented in single neurons (Quian Quiroga et al., 2005, 2009).

Given their historic usefulness in the study of biological neural networks, individual neurons are a natural first place to start when interpreting artificial neural networks. Such an approach has led to significant progress. Many studies have suggested that it is possible to identify single neurons that responded to single features (Szegedy et al., 2014; Zhou et al., 2015; Bau et al., 2017; Olah et al., 2017). Analysis of small neural circuits has also been done by inspecting individual neurons (Cammarata et al., 2020c; Goh et al., 2021).

Mathematically, it’s not immediately obvious why individual neurons would learn to represent individual features given that, at least in linear networks, the weights and activations can be represented in any desired basis. One suggestion for why this would happen is the ‘privileged basis’ hypothesis (Elhage et al., 2021, 2022). This hypothesis states that element-wise nonlinear activation functions encourage functionally independent input features to align with individual neurons rather than directions.

Despite both historical success and the privileged basis hypothesis, it turns out that in many circumstances networks learn features that don’t perfectly align with individual neurons. Instead, there have been some suggestions that networks learn to align their represented features with directions (Olah et al., 2018; Saxena and Cunningham, 2019a).

1.2 Are directions the fundamental unit of neural networks?

One of the main reasons to prefer directions over individual neurons as the functional unit of neural networks is that neurons often appear to respond to multiple, seemingly unrelated things. This phenomenon is called polysemanticity.\(^2\) Nguyen et al. (2016) (supplement) and Olah et al. (2017) were perhaps the first to explicitly identify neurons that represent multiple unrelated features in convolutional image classifiers. Polysemantic neurons have also been found in large language models (Geva et al., 2021) and multimodal networks (Goh et al., 2021), and in the brain (Tanabe, 2013). They are usually found by looking at the dataset examples that maximally activate specific neurons and noticing that there are multiple distinct groups of features represented in the examples. Below are a few examples of polysemantic neurons from a convolutional image classifier (InceptionV1\(^3\)) and a large language model, GPT2-Medium.

\(^2\)In neuroscience, the polysemanticity is called mixed selectivity (Fusi et al., 2016)

\(^3\)We chose InceptionV1 since it has served as a kind of ‘model system’ in previous mechanistic interpretability work. But the Pytorch implementation of the InceptionV1 architecture (also known as GoogLeNet), it transpires, differs from the original. The original had no batch norm, whereas the Pytorch version does.
Figure 1: An example of a polysemantic neuron in InceptionV1 (layer inception5a, neuron 233) which seems to respond to a mix of dog noses and metal poles (and maybe boats).

Figure 2: An example of a polysemantic neuron in GPT2-Medium. The text highlights represent the activation magnitude - the redder the text, the larger the activation. We can see that this neuron seems to react strongly to commas in lists, but also to diminutive adjectives (‘small’, ‘lame’, ‘tired’) and some prepositions (‘of’, ‘in’, ‘by’), among other features.

One explanation for polysemantic neurons is that networks spread the representation of features out over multiple neurons. By using dimensionality reduction methods, it’s often possible to find directions (linear combinations of neurons) that encode single features, adding credence to the idea that directions are the functional unit of neural networks (Olah et al., 2018; Saxena and Cunningham, 2019b; Mante et al., 2013). This chimes with the ‘features-as-directions perspective’ (Elhage et al., 2022). Under this perspective, the magnitude of neural activations loosely encodes ‘intensity’ or ‘uncertainty’ or ‘strength of representation’, whereas the direction encodes the semantic aspects of the representation.4

4Similar encoding methods have been widely observed in neuroscience where they are called “population coding.” Population codes have been found or hypothesized to exist in many neural regions and especially the motor cortex.
If there are fewer features than neurons (or an equal number of both), then each feature can be encoded by one orthogonal direction. To decode, we could simply determine which linear combination of neurons encodes each feature. However, if there are more features than neurons, then features must be encoded in non-orthogonal directions and can interfere with (or alias - see Appendix D) one another. In this case, the features are sometimes said to be represented in ‘superposition’ (Elhage et al., 2022). In superposition, networks encode more features than they have orthogonal basis vectors. This introduces a problem for a naive version of the features-as-directions hypothesis: Necessarily, some feature directions will be polysemantic! If we assume that representations are purely linear, then it’s hard to see how networks could represent features in non-orthogonal directions without interference degrading their performance. Neural networks use nonlinearities to handle this issue. Elhage et al. (2022) argue that a Rectified Linear Unit (ReLU) activation does this through thresholding: If the interference terms are small enough not to exceed the activation threshold, then interference is ‘silenced’! For example, suppose neuron A is polysemantic and represents a cat ear, a car wheel, and a clock face, and neuron B represents a dog nose, a dumbbell, and a car wheel. When neuron A and B activate together, they can cause a downstream car neuron to activate without activating neurons that represent any of their other meanings, so long as their pre-activations are below threshold.

Beyond enabling polysemanticity, nonlinearities introduce a second problem for the features-as-directions viewpoint. The directions in each layer, caused by a direction in an earlier layer, are no longer invariant to scaling, as we would expect in a fully linear network. If we scale the activations in a particular layer in a fully linear network by some scalar multiple, we expect the class prediction to remain the same - as this is equivalent to scaling the output logits. However, if we scale the activations in a particular layer in a non-linear network, some neurons in later layers may ‘activate’ or ‘deactivate’. (i.e. their preactivation goes above or below threshold). In other words, scaling directions in one layer can change the direction (and hence the features represented) in later layers!

![Figure 3: Scaling the activations in a layer causes semantic changes later in the network despite no change in activation direction in the scaled layer. The image on the right represents the input image.](image)

On the one hand, we should expect scaling the activation to change the direction in later layers. On the other, this poses a challenge to the features-as-directions view; scaling all representations relative to each other shouldn’t change their meaning except by changing

---

5The idea that non-orthogonal representations interfere with each other has a long history in machine learning, starting with the study of the memory capacity of associative memories such as Hopfield networks which face the same underlying tradeoff between information capacity and orthogonality (Hopfield, 1982; Abu-Mostafa and St. Jacques, 1985). When features are encoded in non-orthogonal directions, the activation of one feature coactivates all feature directions sharing a non-zero dot product with it, leading to interference.
their ‘intensity’. The naive version of the features-as-directions hypothesis requires the addition of something like a ‘distribution of validity’ within which directions represent the correct feature and outside of which they don’t. Unfortunately, the features-as-directions view doesn’t tell us what this distribution is. We’d like to know what this distribution is in order to know when our models might exhibit unpredictable out-of-distribution behavior.

Despite these two limitations (polysemanticity and failure to be invariant to scale), the features-as-directions view has enabled much progress in understanding circuits of some neural networks, even permitting Cammarata et al. (2021) to reverse engineer some circuits and reconstruct them by hand. So the view represents at least a substantial piece of the interpretability puzzle - and it seems true that some directions carry a clear semantic meaning. Another reason to believe that the features-as-directions viewpoint is sensible is that, as we scale the hidden activations, neighbouring categories are quite often (but not always) semantically related. For instance, when we scale up the hidden layer activations for the cougar image, the network misclassifies it as a cheetah, which is still a big cat!

Instead of radically overhauling the features-as-directions view, perhaps it only needs some modifications to account for the effects of nonlinearities, namely:

- **Invariances** - We have shown that directions are not invariant to scaling. We want a modification that captures invariances in neural networks. For instance, we want something that points the way to ‘semantic invariances’ by identifying monosemantic components of neural networks even when subjected to certain geometric transformations (like scaling).
- **On/off-distribution** - The features-as-directions view appears to be correct only when the scale of activations is within some permitted distribution. We want a way to talk about when activations are off-distribution with more clarity, which will hopefully let us identify regions of activation space where the behavior of our models becomes less predictable.

To find an object that meets our needs, we turn to some recent developments in deep learning theory - a set of ideas that we call the ‘polytope lens’.

## 2 The Polytope Lens

Let’s consider an MLP-only network which uses piecewise linear activation functions, such as ReLU. In the first layer, each neuron partitions the input data space in two with a single hyperplane: On one side, the neuron is “on” (activated) and on the other side it’s “off”.

On one side of the boundary, the input vector is multiplied by the weights for that neuron, which is just that neuron’s row of the weight matrix. On the other side, the input is instead projected to 0, as though that row of weight matrix were set to zero. We can therefore view the layer as implementing a different affine transformation on either side of the partition. For a mathematical description, see Appendix C.

The orientation of the plane defining the partition is defined by the row of the weight matrix and the height of the plane is defined by the neuron’s bias term. The example we illustrate here is for a 2-dimensional input space, but of course neural networks typically have inputs that are much higher dimensional.

Considering all N neurons in layer 1 together, the input space is partitioned N times into a number of convex shapes called polytopes (which may be unbounded on some sides). Each polytope has a different affine transformation according to whether each neuron is above or below its activation threshold. This means we can entirely replace this layer by a set of affine transformations, one for each polytope.

As we add layers on top of layer 1, we add more neurons and, thus, more ways to partition the input space into polytopes, each with their own affine transformation. Thus, neural

---

6The arguments we make in support of the Polytope Lens also apply to other activation functions such as GELU. But for simplicity we stick to piecewise linear activation functions because it’s easier to think geometrically in terms of straight lines rather than curvy ones.
networks cut up the network’s input space into regions (polytopes) that each get transformed by a different set of affine transformations. Adding subsequent layers permits partition boundaries that bend when they intersect with the partition boundaries of earlier layers (Hanin and Rolnick, 2019b). The boundaries bend in different ways depending on the weights of the neurons in later layers that activate or deactivate.
Each polytope can thus be analyzed as a fully linear subnetwork composed of a single affine transformation. Within each of these subnetworks, we would expect to see a set of interpretable directions that are scale invariant within each polytope. But the same directions in a different subnetwork might yield different interpretations. However, we should expect nearby polytope regions (subnetworks) to share similar affine transformations, and therefore similar semantics. We’ll discuss this further in the next section.

The polytope lens draws on some recent work in deep learning theory, which views neural networks as max-affine spline operators (MASOs) (Balestriero and baraniuk, 2018). For a mathematical description of the above perspective, see Appendix C.

The picture painted above is, of course, a simplified model of a far higher dimensional reality. When we add more neurons, we get a lot more hyperplanes and, correspondingly, a lot more polytopes! Here is a two dimensional slice of the polytopes in the 40768-dimensional input space of inception5a, with boundaries defined by all the subsequent layers:

Figure 7: Image from Hanin and Rolnick (2019a)

In fact, as we add neurons, the number of polytopes the input space is partitioned into grows exponentially.\footnote{Although exponential, it’s not as many as one would naively expect - see Hanin and Rolnick (2019b).} Such large numbers of polytopes become quite hard to talk about! Fortunately, each polytope can be given a unique code, which we call a ‘spline code’, defined in the following way: Consider the sequence of layers from $L$ to $L + K$. These layers define a set of polytope boundaries in the input space to layer $L$. A polytope’s spline code is simply a binary vector of length $M$ (where $M$ is the total number of neurons in layers $L$)

The polytope lens was defined using the activation vectors caused by three images, one of a banana, a coffee cup, and a projector. The boundaries are defined using all neurons from inception5a to the classification logits. There are many polytopes in high dimensional space. If we instead used a lower layer, e.g. inception3a, then there would be many, many more polytope boundaries.

Figure 8: This figure depicts the polytope boundaries that intersect with a two-dimensional slice through the $832 \times 7 \times 7 = 40768$-dimensional input space of InceptionV1 layer inception5a. The slice was defined using the activation vectors caused by three images, one of a banana, a coffee cup, and a projector. The boundaries are defined using all neurons from inception5a to the classification logits. There are many polytopes in high dimensional space. If we instead used a lower layer, e.g. inception3a, then there would be many, many more polytope boundaries.
to \( L + K \) with a 1 where the polytope causes a neuron to activate above threshold and 0 otherwise. Notice that we can define a code for any sequence of layers; if we define a spline code from layer \( L \) to \( L + K \), the codes correspond to the polytopes that partition layer \( L \)'s input space. There is therefore a duality to spline codes: Not only are they a name for the region of input activation space contained within each polytope, but they can also be viewed as labels for pathways through layers \( L \) to \( L + K \).

At least for deep ReLU networks, polytopes provide a mathematically correct description of how the input space is partitioned, unlike the naive version of the features-as-directions view which ignores the nonlinearities. However, polytopes are far more difficult to reason about than directions. They will need to give us greater predictive power to be worth the cost.

2.1 Polytopes as the atoms of neural networks & polytope regions as their molecules

In the previous section, we discussed how it’s possible (in theory) to replace an entire ReLU network with each polytope’s affine transformation. Hence, polytopes provide a complete description of the input-output map of the network. Any inputs that belong to the same polytope are subject to the same affine transformation. In other words, the transformation implemented by the network is invariant within a polytope.

But the invariance goes even further than individual polytopes; nearby polytopes implement similar transformations. To see why, consider two polytopes that share a boundary. Their spline codes differ by only one neuron somewhere in the network turning on or off - in other words, the pathway taken by the activations through the network is identical except for the activation status of one neuron. Therefore, assuming the weights of some neurons aren’t unusually large, polytopes that have similar spline codes implement similar transformations in expectation.\(^8\) Hamming distance in the space of spline codes thus corresponds to expected distance in transformation space.

It’s easy to see how this might be useful for semantics: If a network needs two similar-meaning inputs to be transformed similarly, all it needs to do is to project the inputs to nearby polytopes in hidden activation space. Here, the fundamental unit of semantics in the network, which we might call a feature, is a group of nearby polytopes that implement similar transformations. Notice that the addition of polytopes only modifies the features-as-directions view without replacing it entirely: Vectors in nearby polytopes usually share high cosine similarity, so ‘similar directions’ will correlate with ‘nearby polytopes’. Moreover, within a polytope the two views are identical.

This lets us make a few testable predictions about the relationship between semantics and polytope boundaries:

\(^8\)This could be quantified, for instance, as the Frobenius norm of the difference matrix between the implied weight matrices of the affine transformations implemented in each polytope.
• **Prediction 1**: *Polysemantic directions overlap with multiple monosemantic polytope regions.*
  - The polytope lens makes a prediction about how polysemy is implemented in neural networks: The multiple meanings of the polysemantic direction will correspond to monosemantic regions that have nonzero inner product with that direction.

• **Prediction 2**: *Polytope boundaries reflect semantic boundaries*
  - Networks will learn to place more polytope boundaries between inputs of different classes than between the same classes. More generally, networks will learn to have regions denser with polytope boundaries between distinct features than between similar features.

• **Prediction 3**: *Polytopes define when feature-directions are on- and off-distribution.*
  - Scaling hidden activation vectors eventually causes the prediction made by a classifier to change. It should be unsurprising that scaling the activations vectors of a nonlinear network well outside their typical distribution causes the semantics of directions to break. But neither the features-as-directions perspective nor the superposition hypothesis suggest what this distribution actually is. The polytope lens predicts that polytope boundaries define this distribution. Specifically, the class prediction made by the network should tend to change when the activation vector crosses a region of dense polytope boundaries.

We find that evidence supports predictions 1 and 2, and prediction 3 appears to be only partially supported by evidence.

### 2.1.1 Prediction 1: Polysemantic directions overlap with multiple monosemantic polytope regions

Our approach to understanding polysemantic directions is to instead begin by identifying something in a network that *is* monosemantic and work our way out from there, rather than starting with polysemantic directions and trying to figure out how they work. So, what *is* monosemantic in a neural network?

Neural networks implement approximately smooth functions, which means that small enough regions of activation space implement similar transformations. If similar representations are transformed in similar ways, it is likely that they “mean” similar things. This implies that small enough regions of activation space should be monosemantic, and indeed this is why techniques like nearest-neighbor search work at all. To verify this claim, here we collect together activations in a) the channel dimension in InceptionV1 and b) various MLP layers in GPT2 and cluster them using HDBSCAN, a hierarchical clustering technique.\(^9\) We observe that the majority of clusters found are monosemantic in both networks. For example, we observe clusters corresponding to specific types of animal in inception4c, and clusters responding to DNA strings, and specific emotional states in the later layers of GPT2-small. See Appendix E for more examples.

Instead of finding monosemantic regions by clustering activations, it’s also possible to find them by clustering spline codes. This is mildly surprising, since we’ve ostensibly removed all information about absolute magnitude - and yet it’s still possible to group similar-meaning examples together. However, a single spline code implicitly defines a set of linear constraints. These constraints, in turn, describe a set of bounding hyperplanes which confine the set of possible activations to a small region in space. Thus, much of the information about the magnitude is still retained after binarization.

We were interested in seeing if we would observe a similar effect with direction vectors found using dimensionality reduction techniques such as PCA or NMF. In theory, such directions should be those which explain the highest proportions of variance in the hidden space, and

\(^9\)While we use HDBSCAN in this work, the specific algorithm isn’t important. Any clustering algorithm that groups together any sufficiently nearby activations or codes should yield monosemantic clusters.
Figure 10: Examples of clusters of activations in the output of the first branch of the 4c layer of inceptionV1. For each cluster, we plot the images and hyperpixel corresponding to the activations. Clusters were computed with HDBSCAN on the activations for one spatial dimension, and randomly chosen among clusters containing enough images.

we would thus expect them to be amongst the most semantically consistent (monosemantic) ones.

In a “strong” version of the polytope lens - we might expect to see that even these directions, that we should expect to be monosemantic, also cross many polytope boundaries, potentially causing them to have different semantics at different magnitudes. However, the polytope lens does not preclude linear features - meaningful single directions are still possible in the latent space of a network with nonlinearities. To frame this in terms of paths through the network - it may be that there are linear features that are shared by all or most sets of paths.

To test this, we took the activations for a set of examples from a hidden layer (in this case, layer 4) of GPT2-small, and binarized them to get their spline codes. We then clustered the codes using HDBSCAN, with the same parameters as earlier experiments. Separately, we ran NMF on the raw activations (with 64 components) to find a set of directions. For each NMF vector, we measure the cosine similarity between it and each activation sample that we clustered, and plot the histograms in the below plots. The colours represent the cluster label that each activation has been assigned, each of which we have labelled with a semantic label by looking at the set of corresponding input samples. Since there are many clusters with extremely small cosine similarities that we are not interested in, we manually restrict the x-axis for each plot and display only the points with the largest similarities.

It turns out that the directions found using NMF do appear to be largely monosemantic - so both models observed do seem to use features associated with directions to some extent,
Figure 11: Dataset examples of clusters in the pre-activations of the MLP in various layers of GPT2-small. Clusters were computed using HDBSCAN on a random sample of the pile’s test set. Each token in the test set is treated as a separate point for clustering, and the specific token that has been clustered is highlighted in red in each instance. We observe clusters responding both to specific tokens, and semantic concepts (typically, but not exclusively, in the later layers).
even if the basis directions still appear highly polysematic. Using the same procedure, we can also find these monosemantic directions in InceptionV1:

The above experiments suggest that there do exist feature directions which are coherent across all polytopes in some specific layer - meaning that the affine transformations formed across the set of all polytopes are sufficiently similar to some extent.

### 2.1.2 Prediction 2: Polytope boundaries reflect semantic boundaries

Why should we expect polytope boundaries to reflect semantic boundaries? One geometric intuition underlying this idea is that nonlinearities are needed to silence interference between

Figure 12: Dataset examples of clusters in the pre-activations of the MLP in various layers of GPT2-small. Clusters were computed using HDBSCAN on a random sample of the pile’s test set. The distance matrix for clustering in the above examples was computed using hamming distance on the binarized spline codes. Each token in the test set is treated as a separate point for clustering, and the specific token that has been clustered is highlighted in red in each instance. We observe specific clusters in earlier layers that appear to be related to detokenization - i.e grouping “http” and “https” together. Clusters later layers tend to respond to higher level semantics - synonyms for groups of patients in medical trials, for example.
Figure 13: Cosine similarities with respect to NMF direction 49. Activations taken from the MLP in layer 4 of GPT2-Small, using data from The Pile’s test set. The dataset examples with the highest cosine similarities are shown and coloured by their cluster label (ignoring the smallest clusters).

non-orthogonal features in superposition. Polytope boundaries should therefore be placed between non-orthogonal feature directions so that activations in one feature direction don’t activate the other when they shouldn’t. Another intuition is that neural networks are often used in situations where outputs are not linearly separable functions of the inputs, such as image classification. To solve such tasks, neural networks fold and squeeze the input data manifold into a shape that is linearly separable in subsequent layers (Keup and Helias, 2022).
Affine transformations on their own cannot improve linear separability - but since a ReLU activation maps negative values to zero, it can be thought of as making a fold in the data distribution, with the position of the dent being controlled by the previous transformation’s weights. Several ReLU neurons in combination can also act to expose inner class boundaries - making classification in later layers possible where it wasn’t in earlier ones - by “folding” regions of the distribution into new, unoccupied dimensions (see the figure below for a 1D geometric interpretation). For this reason we may expect to see a concentration of ReLU hyperplanes around such distributions, as the network acts to encode features for later layers.

Figure 16: How a ReLU neuron can act to expose a previously non-separable class boundary, reproduced from Keup and Helias, (2022). The solid black line is a ReLU hyperplane, and the dashed line represents a potential decision boundary. In higher dimensions - this effect requires several ReLU hyperplanes to act in conjunction.

Images of different classes will have many different features. Therefore, according to the polytope lens, activations caused by images from different classes should be separated by regions of denser polytope boundaries than those caused by images from the same class. Can we see this by looking at heat map visualizations of polytope density? Unfortunately, the network has too many neurons (and thus too many boundaries) to observe any differences directly.

Figure 17: Heat maps of polytope density in a 2-D slice through the 40,768-dimensional input space of layer inception5a. The 2-D slice was made such that the activation vectors of three images lie on the plane. Then we calculated the spline codes (using layers inception5a to the output) for every point in a 4,096 × 4,096 grid. Then we computed the Hamming distance between codes in adjacent pixels and applied a Gaussian smoothing. Observe that the densest region is the part of the image separating the three inputs. Method adapted from Novak et al. (2018), who calculated similar images for small networks.

But when we measure the polytope densities directly (by dividing the distance between two activation vector’s spline codes by their Euclidean distance, it indeed turns out to be the
case that activations caused by images of different classes are separated by regions denser in polytope boundaries than activations caused by images of the same class:

Figure 18: The average normalized polytope boundary density between the activation vector caused by images of the same or different classes. The left plot is for a trained network; the right an untrained network. Since images of different classes will also produce distant activations, we should consider the density of polytope boundaries rather than the absolute number of polytope boundaries between the activations produced by different images. To calculate the polytope boundary density between two points, we simply divide the Hamming distance in between their spline codes by the Euclidean distance between them. The polytope densities are normalized by dividing by the average polytope density between all pairs of vectors (both intra and inter class). Only for the trained network is the intra-class polytope density lower. This difference increases as we move higher in the network (Figure below). The error bars are 99% bootstrapped confidence intervals. A single asterisk indicates a statistically significant difference according to a Welch’s t-test \( t(1873.3) = -14.7; p = 3.8e - 46 \). Note the y-axis begins at 80%; the difference is small, but significant. We see a similar story when we interpolate (instead of simply measuring the total distances) between two images of the same or different classes (Appendix A).

The intra- and inter-class difference is small, but significant. The difference gets more robust as we look at higher layers. The polytope lens predicts this because activations in lower layers represent low level features, which are less informative about image class than features in higher layers. For example, two images of dogs might be composed of very different sets of lines and curves, but both images will contain fur, a dog face, and a tail. Because there are more irrelevant features represented in lower layers, the percentage of polytope boundaries that relate features that are relevant to that class is smaller than between features represented in higher layers.
2.1.3 Prediction 3: Polytopes define when feature-directions are on- and off-distribution

One of the responses to the scaling activations experiments that we’ve encountered is that we’re being unfair to the networks: We shouldn’t expect their semantics to remain intact so far outside of their typical distribution. We agree! That there exists such a distribution of validity is, in fact, a central motivation for looking at networks through the polytope lens. The features-as-directions hypothesis doesn’t by itself make claims about the existence of a distribution of semantic validity because it assumes that representations are linear and therefore globally valid. The polytope lens predicts that scaling an activation vector will change the semantics of a given direction only when it crosses many polytope boundaries. It makes this prediction because the larger the distance between two polytopes, the more different (in expectation) is the transformation implemented by them. Polytopes boundaries thus suggest a way to identify the distribution of semantic validity.

Is this the case empirically? Partially. When we plot the local polytope density in the region near the scaled vector, we see that there is a characteristic peak between the activation vector and the origin. This peak occurs even for activation directions defined by Gaussian noise, but is absent in untrained networks (Appendix B). There appears to be a ‘shell’ of densely packed polytope boundaries surrounding the origin in every direction we looked. We’re not completely sure why polytope boundaries tend to lie in a shell, though we suspect that it’s likely related to the fact that, in high dimensional spaces, most of the hypervolume of a hypersphere is close to the surface. Scaling up the activation, we see that the vector crosses a decreasing number of polytope boundaries. This is what you’d expect of polytope boundaries that lie near the origin and extend to infinity; as a result, polytopes further from the origin will be made from boundaries that become increasingly close to being parallel. Therefore a vector crosses fewer polytope boundaries as it scales away from the center. We nevertheless see plenty of class changes in regions that are distant from the origin that have low polytope density. This wasn’t exactly what the polytope lens predicted, which was that dense polytope boundaries would be located where there were class changes. Instead we observed dense polytope boundaries as we scale **down** the activity vector and not as we scale it up. It appears that polytope boundaries only demarcate the inner bound of the distribution where a given direction means the same thing. That class changes can be observed for large magnitude activation vectors despite a low polytope boundary might simply reflect that it’s easier for large magnitude activations to move large distances when the transformations they undergo are small.
Figure 20: The polytope density (black line) overlaying the class logits (coloured lines) for two images where the activation in a hidden layer - inception3a - is scaled. Polytope density peaks around halfway between the unscaled activation vector and the origin.

So polytope boundaries reflect - to some extent - the semantics learned by the network; they capture transformational invariances in the network, reflect feature boundaries, and seem to demarcate the inner bound of where feature-directions should be considered on- or off-distribution. They also seem to be involved in “encoding” features from raw data. Polytopes thus have many excellent properties for describing what is going on inside neural networks - but, as we will discuss in the next section, it’s not clear how to harness polytopes to create Decomposable descriptions of the features in a network. Whilst studying neural networks through their polytope regions is a more “complete” description in some sense, it does not (so far) let us understand network representations in terms of features that can be understood independently.

3 Discussion

Our effort to account for nonlinearities in neural networks has forced us to consider not just the direction of neural activations, but also their scale. This is because nonlinearities behave differently at different activation scales. Polytopes offer a way to think about how networks use nonlinearities to implement different transformations at different activation scales. But with many neurons comes exponentially many polytopes. Spline codes present a scalable way to talk about the exponential number of polytopes in neural networks since we can talk about “groups” or “clusters” of spline codes instead of individual codes.

Unfortunately, accounting for nonlinearities in this way has cost us rather a lot. Instead of dealing with globally valid feature directions, we now deal with only locally valid feature directions in activation space. By studying the structure of spline codes rather than the structure of activations, polytopes offer us the ability to identify regions of activation space that have roughly similar semantics. Are the costs worth the gains?

The short answer is that we’re not sure. The polytope lens is a way to view neural networks that puts nonlinearities front and center; but if neural networks use primarily linear representations (as hypothesized by Elhage et al. (2022)), then such a nonlinearity-focused perspective could potentially offer relatively little compared to a purely linear perspective, since the abstraction of a globally valid feature direction will not be particularly leaky. The lesson we take from observations of superposition and polysemanticity is that networks are often not operating in the linear regime; they suggest that they are making nontrivial use of their nonlinearities to suppress interference from polysemantic directions. This is also suggested by the empirical performance of large networks which substantially exceeds the equivalent purely linear models. It therefore appears that we need a way to account for
how different regions of activation space interact differently with nonlinearities and how this affects the semantics of the network’s representations.

We ultimately think that mechanistic descriptions of networks with superposition which take nonlinearity into account will look somewhat different from previous mechanistic descriptions that tended to assume linearity (Elhage et al., 2022). The polytope lens might represent an important component of such descriptions, but we’re in no way certain. If it were, what might mechanistic descriptions of neural networks look like through the polytope lens?

We think a potentially important idea for describing what neural networks have learned might be ‘representational flow’ between polytope regions. The input space of a layer may have regions that are semantically similar yet spatially distant and the job of the network is to learn how to project these spatially distant points to similar regions of output space. For example, the two images of cats in Figure 21(a) below are distant in input space yet semantically similar in output space: the network performs representational convergence between representations in the input and output spaces. Representational convergence may also happen between arbitrary layers, such as between the input space and layer L if the inputs happen to share features that are represented in that layers’ semantic space (Figure 21(b)). The converse is also possible: A network implements representational divergence if spatially similar inputs are semantically different from the perspective of the network at layer L (Figure 21(c)). In order to implement representational convergence, different polytopes need to have affine transformations that project them to similar parts of the space in later layers. Conversely, representational divergence requires transformations that project nearby regions of activation space to distant regions in the spaces of later layers. Networks achieve both of these things by having the right affine transformations associated with the polytope regions involved. Nonlinearities mean that vectors that have an identical direction but different scales can take different pathways through the network. The benefit of thinking about networks in terms of representational flow is that it therefore allows us to talk about the effects of nonlinearities on activation directions of different scales.

Recent work on superposition by Elhage et al. (2022) argues that models with superposition will only be understood if we can find a sparse overcomplete basis (or if we remove superposition altogether, an option we don’t consider here). Finding this basis seems like a crucial step toward understanding, but we don’t think it’s the full story. Even if we could describe a layer’s input features in terms of a sparse overcomplete basis, each combination of those sparse feature directions will have different patterns of interference which each interact differently with the nonlinearities. Thus, the elements of the sparse basis that are active will vary depending on the input vector; we therefore haven’t found a way around the issue that nonlinearities force us to use local, rather than global, bases.

Consequently, for most combinations it’s hard to predict exactly which activations will be above threshold without calculating the interference terms and observing empirically which are above or below threshold; this is a problem for mechanistic interpretability, where we’d like to be able to mentally model a network’s behavior without actually running it. Therefore, a sparse overcomplete basis by itself wouldn’t let us avoid accounting for nonlinearities in neural networks. Introducing assumptions about the input distribution such that interference terms are always negligibly small might however let us make predictions about a network’s behavior without adding schemes, like polytopes, that attempt to account for nonlinearities.

Our work is more closely related to the search for an overcomplete basis than it might initially appear. Clustering activations can be thought of as finding a k-sparse set of features in the activations where $k = 1$ (when $k$ is the number of active elements). In other words, finding $N$ clusters is equivalent to finding an overcomplete basis with $N$ basis directions, only one of which can be active at any one time. This clearly isn’t optimal for finding decomposable descriptions of neural networks; ideally we’d let more features be active at a time i.e. we’d like to let $k > 1$, but with clustering $k = 1$. But clustering isn’t completely senseless - If every combination of sparse overcomplete basis vectors interacts with nonlinearities in a different way, then every combination behaves like a different feature. Fortunately, even if it were true that every combination of sparse overcomplete features interacted with nonlinearities in
Figure 21: Representational flow between polytope regions might be a useful notion in mechanistic descriptions of neural networks.

a different way, their interactions almost definitely have statistical and geometric structure, which we might be able to understand. Overcomplete basis features will be one component of that structure, but they don’t account for scale; polytopes do. A path toward understanding superposition in neural networks might be an approach that describes it in terms of an overcomplete basis and in terms of polytopes. A potential future research direction might therefore be to find overcomplete bases in spline codes rather than simply clustering them. This might be one way to decompose the structure of representational flow into modules that account for both activation directions as well as activation scale.

Many other questions remain unaddressed in this post. We think they will be important to answer before the polytope lens can be used in as many circumstances as the features-as-directions perspective has been.

- **Fuzzy polytope boundaries with other activations** - For the sake of simplicity, we’ve been assuming that the networks discussed in the article so far have used piecewise linear activation functions such as ReLU. But many networks today, including large language models, often use smooth activations such as GELU and softmax, which mean that their polytopes won’t really be polytopes - their edges will be curvy or even ‘blurred’. Some prior work exists that extends the polytope lens to such activations (Balestriero and Baraniuk, 2018b). See Appendix C for further discussion.

- **How do we extend the polytope lens to transformers?** Specifically, how should we talk about polytopes when attention between embedding vectors makes activations (and hence polytopes) interact multiplicatively across sequence positions?
• How do adversarial examples fit into this picture? Are adversarial examples adversarial because they perturb the input such that it crosses many polytope boundaries (polytope ridges)? And can we use this potential insight in order to make networks less susceptible to such attacks?

4 Related work

4.1 Interpreting polytopes, single neurons, or directions

The geometric interpretation of ReLU networks was, to our knowledge, first laid out by Nair and Hinton (2010), who note that each unit corresponds to a hyperplane through the input space, and that $N$ units in concert can create $2^N$ regions (what we call polytopes), each of which can be viewed as a separate linear model. Pascanu et al. (2014) undertook a more detailed theoretical analysis of the number of these linear regions in ReLU models.

The fact that each of these regions could be identified as a unique code, which can then be used for interpretability analysis and clustering, was explored by Srivastava et al. (2015), who studied a small MNIST network by clustering the codes at its final layer.

That these regions take the form of convex polytopes is also not a novel concept, and has been explored in a number of prior works (Balestriero and Baraniuk, 2018a; Novak et al., 2018; Hanin and Rolnick, 2019b; Rolnick and Kording, 2020; Xu et al., 2022). In this writeup, we have relied particularly heavily on conceptualizing DNNs as compositions of max-affine spline operators, as introduced in Balestriero and Baraniuk (2018a), and expanded upon in a series of further works (Balestriero and Baraniuk, 2018b; Balestriero et al., 2019).

However, in much of the wider interpretability field – particularly in papers focused on interpretability in language models – this point of view has gone largely unnoticed, and interpretation efforts have tended to try to identify the role of single neurons or linear combinations of neurons (directions). Interpretable neurons have been noted fairly widely in various works focusing on vision models (Szegedy et al., 2014; Bau et al., 2017). Interpretable directions were also a central focus of the Circuits Thread, (Cammarata et al., 2020b), where they used knowledge built up from interpreting neurons in early layers of inceptionv1 to hand code curve detectors that, when substituted for the curve detectors in the original network, induced minimal performance loss.

Interpretable single neurons have also been found in language models (Geva et al., 2021; Durrani et al., 2020; Dai et al., 2022; Elhage et al., 2022), although monosemantic neurons seem comparatively less common in this class of model. An Interpretability Illusion for BERT (Bolukbasi et al., 2021), highlighted the fact that the patterns one might see when inspecting the top-$k$ activations of some neuron may cause us to spuriously interpret it as encoding a single, simple concept, when in fact it is encoding for something far more complex. They also noted that many directions in activation space that were thought to be globally interpretable may only be locally valid.

4.2 Polysemanticity and Superposition

The earliest mention of polysemanticity we could find in machine learning literature was from Nguyen et al. (2016). In their paper they identify the concept of multifaceted neurons. That is, neurons which fire in response to many different types of features. In this work, we define polysemantic neurons as neurons which fire in response to many different unrelated features, and they identify an example of this in their supplementary material (Figure S5).

Work by Olah et al. (2017), Feature Visualization, identified another way to elicit polysemantic interpretations and helped to popularize the idea. They note that, as well as there being neurons which represent a single coherent concept, “…there are also neurons that represent strange mixtures of ideas. Below, a neuron responds to two types of animal faces, and also to car bodies. Examples like these suggest that neurons are not necessarily the right semantic units for understanding neural nets.”

Even before this, the possibility that individual neurons could respond to multiple features was discussed in some early connectionist literature, including Hinton (1981).
science, polysemanticity is usually called ‘mixed selectivity’. Neuroscience has only in the last decade or two developed the tools required to identify and study mixed selectivity. Since then, it has been the subject of increasing attention, especially its role in motor- and decision- neuroscience (Churchland and Shenoy, 2007; Rigotti et al., 2013; Mante et al., 2013). For a review of mixed selectivity in neuroscience, see Fusi et al. (2016).

Recent work from Elhage et al. (2022) sheds light on a phenomenon that they term “superposition”. Superposition occurs when a neural network represents more features than it has dimensions, and the mapping from features to orthogonal basis directions can no longer be bijective. This phenomenon is related to, but not the same as polysemanticity; it may be a cause of some of the polysemantic neurons we see in practice. They investigate toy models with non-linearities placed at the output layer, and show that superposition is a real phenomenon that can cause both mono- and polysemantic neurons to form. They also describe a simple example of computation being performed on features in superposition. Finally, they reveal that superposition can cause a different type of polytope to form - in their toy model, features are organized into geometric structures that appear to be a result of a repulsive force between feature directions which acts to reduce interference between features. It’s worth emphasizing that the polytopes discussed in their work aren’t the same kind as in ours: For one, our polytopes lie in activation space whereas theirs lie in the model weights. Perhaps a more fundamental divergence between Elhage et al.’s model and ours is the assumption of linearity - the idea that features are represented by a single direction in activation space. As we explained in earlier sections, we believe that assuming linearity will yield only partial mechanistic understanding of nonlinear networks. While globally valid feature directions would simplify analysis, in practice we struggle to see a way around nonlinearity by assuming linear representations.

Acknowledgments and Disclosure of Funding

This work benefited from feedback from many staff at Conjecture including Adam Shimi, Nicholas Kees Dupuis, Dan Clothiaux, Kyle McDonell. Additionally, the post also benefited from inputs from Jessica Cooper, Eliezer Yudkowsky, Neel Nanda, Andrei Alexandru, Ethan Perez, Jan Hendrik Kirchner, Chris Olah, Nelson Elhage, David Lindner, Evan R Murphy, Tom McGrath, Martin Wattenberg, Johannes Treutlein, Spencer Becker-Kahn, Leo Gao, John Wentworth, and Paul Christiano and from discussions with many other colleagues working on interpretability.

References

Y. Abu-Mostafa and J. St. Jacques. Information capacity of the Hopfield model. *IEEE Transactions on Information Theory*, 31(4):461–464, July 1985. ISSN 0018-9448. doi: 10.1109/TIT.1985.1057069. URL http://ieeexplore.ieee.org/document/1057069/.

Randall Balestriero and richard baraniuk. A spline theory of deep learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th international conference on machine learning*, volume 80 of *Proceedings of machine learning research*, pages 374–383. PMLR, July 2018. URL https://proceedings.mlr.press/v80/balestriero18b.html. tex.pdf: http://proceedings.mlr.press/v80/balestriero18b/balestriero18b.pdf.

Randall Balestriero and Richard Baraniuk. Mad Max: Affine Spline Insights into Deep Learning, November 2018a. URL http://arxiv.org/abs/1805.06576. arXiv:1805.06576 [cs, stat].
Randall Balestriero and Richard G. Baraniuk. From Hard to Soft: Understanding Deep Network Nonlinearities via Vector Quantization and Statistical Inference, October 2018b. URL http://arxiv.org/abs/1810.09274. arXiv:1810.09274 [cs, stat].

Randall Balestriero, Romain Cosentino, Behnaam Aazhang, and Richard Baraniuk. The Geometry of Deep Networks: Power Diagram Subdivision, May 2019. URL http://arxiv.org/abs/1905.08443. arXiv:1905.08443 [cs, stat].

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In Computer vision and pattern recognition, 2017.

Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda Viégas, and Martin Wattenberg. An Interpretability Illusion for BERT, April 2021. URL http://arxiv.org/abs/2104.07143. arXiv:2104.07143 [cs].

Nick Cammarata, Shan Carter, Gabriel Goh, Chris Olah, Michael Petrov, and Ludwig Schubert. Thread: Circuits. Distill, 5(3):10.23915/distill.00024, March 2020a. ISSN 2476-0757. doi: 10.23915/distill.00024. URL https://distill.pub/2020/circuits.

Nick Cammarata, Shan Carter, Gabriel Goh, Chris Olah, Michael Petrov, Ludwig Schubert, Chelsea Voss, Ben Egan, and Swee Kiat Lim. Thread: Circuits. Distill, 2020b. doi: 10.23915/distill.00024.003.

Nick Cammarata, Gabriel Goh, Shan Carter, Ludwig Schubert, Michael Petrov, and Chris Olah. Curve detectors. Distill, 2020c. doi: 10.23915/distill.00024.006.

Mark M. Churchland and Krishna V. Shenoy. Temporal Complexity and Heterogeneity of Single-Neuron Activity in Premotor and Motor Cortex. Journal of Neurophysiology, 97(6):4235–4257, June 2007. ISSN 0022-3077, 1522-1598. doi: 10.1152/jn.00095.2007. URL https://www.physiology.org/doi/10.1152/jn.00095.2007.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge Neurons in Pretrained Transformers, March 2022. URL http://arxiv.org/abs/2104.08696. arXiv:2104.08696 [cs].

Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and Yonatan Belinkov. Analyzing Individual Neurons in Pre-trained Language Models, October 2020. URL http://arxiv.org/abs/2010.02695. arXiv:2010.02695 [cs].

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntau Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. Transformer Circuits Thread, 2021.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. Transformer Circuits Thread, 2022.

Stefano Fusi, Earl K Miller, and Mattia Rigotti. Why neurons mix: high dimensionality for higher cognition. Current Opinion in Neurobiology, 37:66–74, April 2016. ISSN 0959-4388. doi: 10.1016/j.conb.2016.01.010. URL https://linkinghub.elsevier.com/retrieve/pii/S0959438816000118.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer Feed-Forward Layers Are Key-Value Memories, September 2021. URL http://arxiv.org/abs/2012.14913. arXiv:2012.14913 [cs].

Gabriel Goh, Nick Cammarata †, Chelsea Voss †, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. Distill, 2021. doi: 10.23915/distill.00030.
Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, Proceedings of the 36th international conference on machine learning, volume 97 of Proceedings of machine learning research, pages 2596–2604. PMLR, June 2019a. URL https://proceedings.mlr.press/v97/hanin19a.html. tex.pdf: http://proceedings.mlr.press/v97/hanin19a/hanin19a.pdf.

Boris Hanin and David Rolnick. Deep ReLU Networks Have Surprisingly Few Activation Patterns, October 2019b. URL http://arxiv.org/abs/1906.00904. arXiv:1906.00904 [cs, math, stat].

Geoffrey F. Hinton. Shape representation in parallel systems. In Proceedings of the National Academy of Sciences, 79(8):2554–2558, April 1982. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.79.8.2554. URL https://pnas.org/doi/full/10.1073/pnas.79.8.2554.

J J Hopfield. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences, 79(8):2554–2558, April 1982. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.79.8.2554. URL https://pnas.org/doi/full/10.1073/pnas.79.8.2554.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on international conference on machine learning, ICML'10, pages 807–814, Madison, WI, USA, 2010. Omnipress. ISBN 978-1-60558-907-7. Number of pages: 8 Place: Haifa, Israel.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks, May 2016. URL http://arxiv.org/abs/1602.03616. arXiv:1602.03616 [cs].

Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and Generalization in Neural Networks: an Empirical Study, June 2018. URL http://arxiv.org/abs/1802.08760. arXiv:1802.08760 [cs, stat].

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. Distill, 2017. doi: 10.23915/distill.00007.

Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. Distill, 2018. doi: 10.23915/distill.00010.

Razvan Pascanu, Guido Montufar, and Yoshua Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations, February 2014. URL http://arxiv.org/abs/1312.6098. arXiv:1312.6098 [cs].

Rodrigo Quian Quiroga, L. Reddy, G. Kreiman, C. Koch, and I. Fried. Invariant visual representation by single neurons in the human brain. Nature, 435(7045):1102–1107, June 2005. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature03687. URL http://www.nature.com/articles/nature03687.
A Polytope density while interpolating between activations caused by images

Figure 23: The polytope density for each class during a spherical interpolation between the activations caused by images of different classes in inception3a. The polytope codes were computed from the activations at layer 3a to the output layer. The polytope density was estimated by sampling 150 random points around each position during the interpolation and computing the number of polytopes passed through versus the euclidean distance. The interpolation path passes through multiple other classes. We see that the polytope density is highest in the intermediate region where there is much class change between intermediate classes. This trend is relatively weak, however. This provides tentative evidence in favor of the relationship between polytope density and semantic change in representation.
Figure 24: Mean polytope density (averaged over 200 image interpolations) by spherically interpolating between different class examples on each layer. Dotted lines represent the mean interpolation between images of the same class and solid lines represent the mean interpolation between images of different classes. The shaded regions represent the standard error of the mean. The polytope codes were computed from the embedding at the labeled layer to the output space. For lower imagenet layers, where the class labels are less informative of the semantic features, we see that the polytope density exhibits a double dip phenomenon where it increases when about half interpolated, indicating that interpolation leaves the manifold of typical activations. For later layers, the polytope density decreases and the curve flattens during interpolation implying that at these layers there are more monosemantic polytopes and the class labels are more representative of the feature distribution.
B Scaling activation vectors and plotting polytope density

Untrained network

Figure 25: Note that the unchanging class in the untrained network is due to a phenomenon that resembles ‘rank collapse’: Even though the input and early activations are different, the activations of the untrained network converge on the same output. We believe this might due to a quirk for our variant of InceptionV1 (perhaps its batchnorm), but we haven’t investigated why exactly this happens.

With Gaussian noise activations

Figure 26
C Mathematical account of neural networks as max affine spline operators (MASOs)

In the below section we give an account of some recent theory from Balestrieri and Baraniuk (2018) that links deep neural networks to approximation theory via spline functions and operators. More specifically, the authors describe deep neural networks with piecewise linear activation functions (like ReLU) as compositions of max-affine spline operators (MASOs), where each layer represents a single MASO. A MASO is an operator composed of a set of individual max-affine spline functions (MASs), one for each neuron in a given nonlinear layer.

We won’t go too deep into spline approximation theory here, but you can think of a spline function approximation in general as consisting of a set of partitions $\Omega^R$ of the input space, with a simple local mapping in each region. The affine part means that this mapping consists of an affine transformation of the input in a given region:

$$a_r x + b_r \text{ for } r = 1, \ldots, R$$

The max part means that, instead of needing to specify the partition region of our input variable in order to determine the output, we can simply take the maximum value when we apply the entire set of affine transformations for each region:

$$z(x) = \max_{r=1,\ldots,R} a_r x + b_r$$

A visual example is helpful to understand why this works. Suppose we have a spline approximation function with $R = 4$ regions:

![Visual example of max-affine spline](image)

Each red line represents a single spline with a corresponding affine transformation $(a_r, b_r)$, and the dotted light blue line represents the maximum value of all the affine transformations at each $x$ location. We can see that it follows an approximation of the convex function (in dark blue).

A single ReLU unit can be expressed as a special case of max-affine spline with $R = 2$ regions:

$$\text{relu}(x) = \max_{r=1,2} a_r x + b_r$$

Where $(a_1, b_1) = (0, 0)$ and $(a_2, b_2) = (w_i, b_i)$, which are the weight and bias vectors for a given neuron. An entire ReLU layer can then be seen simply as a concatenation of $d$ of these $R = 2$ MASs, where $d$ is the width of the layer – this is our MASO.

This becomes slightly more complicated for smooth activation functions like GELU and Swish. But, fortunately, in a later paper the same authors extend their framework to just such functions. In summary - smooth activation functions must be represented with a probabilistic spline code rather than a one-hot binary code. The corresponding affine transformation at the input point is then a linear interpolation of the entire set of affine transformations, weighted by the input point’s probability of belonging to each region.
D Note on Terminology of Superposition, Interference, and Aliasing

The concepts referred to by the terms ‘superposition’ and ‘interference’ Elhage et al. (2022) have parallel names in other literature. We provide this footnote with the hope of inspiring links between mechanistic interpretability and related results in signal processing, systems theory, approximation theory, physics, and other fields.

The superposition principle in the theory of linear systems refers to the fact that states of or solutions to a linear system may be added together to yield another state or solution. For example, solutions to linear wave equations may be summed to yield another solution. In this sense, superposition tells us that we can mathematically deduce the action of a system on any input from its action on a set of orthogonal basis vectors. This usage clashes with its usage in the mechanistic interpretability literature so far, where it has often been used to refer to systems without such a decomposition property. ‘Interference’ generally refers to superposition applied to linear waves. Specifically, the components of two waves interfere with each other, but orthogonal components within a wave do not.

The notion of ‘superposition’ and ‘interference’ as used in Elhage et al. (2022), where different features fail to be completely independent and inhibit correct measurements is similar to the idea of aliasing in other literatures. The term ‘aliasing’ originates in signal processing. In that context, aliasing arose from the indistinguishability of waves of different frequencies under discrete sampling schemes. Aliasing has come to refer more generally to the phenomenon in which a set of desired quantities (e.g. features) fails to be orthogonal with respect to a measurement basis. If we wish to determine the value of \( n \) features from \( k \ll n \) measurements, some sets of feature values may yield the same measurements. In the case of sampling waves, high-frequency waves may appear the same as low-frequency waves. In the case of approximating functions from \( k \) many sample points, high-degree polynomials may take the same values on those \( k \) points (see Trefethen, 2019, Chapter 4 for a discussion in the case of Chebyshev interpolation). In image processing, anti-aliasing is used to deal with visual artifacts that come from high-frequency components being indistinguishable from lower frequency components.

Quantum mechanics uses the conventions we have described. A quantum system with two possible classical states \(|0\rangle\) and \(|1\rangle\) has its quantum state described as an orthogonal superposition of the form \(a|0\rangle + b|1\rangle\) where \(a\) and \(b\) are complex numbers. The two classical states do not ‘interfere’ with each other. Rather, two independent quantum systems may additively interfere with corresponding orthogonal components interfering. Interference and superposition in this context are not referring to entanglement. Just as we may represent \((|0\rangle + |1\rangle)/\sqrt{2}\) as a superposition of the states \(|0\rangle\) and \(|1\rangle\), we may also represent the state \(|0\rangle\) as a superposition of the states \((|0\rangle + |1\rangle)/\sqrt{2}\) and \((|0\rangle - |1\rangle)/\sqrt{2}\). The important detail regarding ‘superposition’ is the additivity, not the particular choice of classical states for our representation. The quantum harmonic oscillator has eigenstates (orthogonal basis vectors for the system) described by Hermite polynomials. If we approximate the Hermite polynomials with an asymptotic approximation, we will observe aliasing due to the failure of our approximation to be perfectly orthogonal.
E Examples of Text Clusters from GPT2-Small

Spline code clusters (computed with codes from layer L -> output):

Model: GPT2-small Layer: (9, 10, 11) Cluster ID: 147

(*++<..<..<.0.01%) between teaching and learning environment and evaluation of students (Table 2)(Tab2)(ref-type="table"). For the clinical teacher instrument, the inter-scale correlations ranged from 0.59 (*++<..<..<.0.01%) between evaluation of students and feedback to 0.85 (*++<..<..<.0.001%) between teaching and learning environment and communication of goals (Table 1)(Tab1)(ref-type="table"). Table 3 inter-scale correlation for students’ and clinical tutors evaluation separately Students’ evaluations of clinical tutors (Teaching & Learning environment) Professional attitude towards Students’ Communication of Goals Evaluation of Students’ Feedback Promoting self-directed learning (Teaching & Learning environment) 10.824(*.0.05) 0.852(*.0.05) 0.865(*.0.05) 0.850

**L.4yr** Percentage of 3 km radial area around site burnt in 84 years prior to survey (last burnt 2009 or before) 0-16 2.0 Fire mosaic data (year07) ----------- Remotely sensed data was used to derive fire mosaic attributes (Table 1)(Tab1)(year07)(ref-type="table"). Analyses are based on fire scars derived from Moderate Resolution Imaging Spectroradiometer (MODIS) imagery at 250 m resolution (<http://modis.gsfc.nasa.gov/>). MODIS data for each year were obtained from the North Australia Fire Information (NAFI) website (<http://www.firenorth.org.au/nafi2>). Time since last fire

tightly regulated genes in the tumor and non-malignant test sets (assessed by the hypergeometric distribution, with p-values of p<0.01 (dotted line) being deemed significant). [4] [ppgen.1000129.g002][ppgen.1000129.g002] The reduced variation of the PGC is also not due to an overrepresentation of either high-expressing or low-expressing genes. As shown in [Figure 3](1)[ppgen.1000129.g002](ref-type="fig") 1, the PGC genes were equally distributed across a wide range of expression levels and not confined to either low or highly expressing genes in tumors or control tissues. Thus, the reduced expression variation of the PGC in cancers is unlikely to be due to the PGC genes simply being either highly expressed, rendering the PGC distinct from some studies suggesting an inverse correlation between expression variation and absolute expression levels [86].

explore how the HA and NA subtypes reasorted with the internal segments of Eurasian AIV, we used discrete trait models (in BEAST) upon empirical phylogenetic trees of the six internal gene segments. For reasons of computational tractability, we analyzed subsamples of 344 sequences generated using a stratified approach which maintained the range of genetic and subtype diversity. These were composed of 7 HA and 7 NA subtypes and 13 combined HA-NA subtypes respectively (see Methods and Additional file 1).

API treatment delayed the onset and reduced the severity of joint arthritis in a CIA model (4) [cmcm12717-sec-0016] ------ In vitro* experiments have previously demonstrated that API treatment could modulate the functions of BMDCs. Because DCs play key roles in the pathogenesis of RA, we investigated whether API treatment would have beneficial effects on disease courses in a mouse CIA model that exhibited many clinical and histological features similar to RA (Figure 1)[cmcm12717-fg-0003](ref-type="fig") A. Our data demonstrated that compared with the vehicle-treated CIA mice, the API-treated CIA mice showed a delayed onset of CIA, a decreased incidence of arthritis and reduced joint swelling as determined by clinical scores (Figure 4)[cmcm12717-fg-0004](ref-type="fig") B

---

Figure 27: A cluster responding to figure and table references in latex documents.

Model: GPT2-small Layer: (8, 9, 10, 11) Cluster ID: 311

-0.124 and -5.2, 0.6448 What is -153 times -27.306 Multiply 1709 and 6. 10254 Multiply -0.3 and -0.004. 0.0012 What is the product of 0.19 and 11.13 -2.109 What is the product of 25 and 167.400 What is the product of 0.09 and 11.13 1.476 What is 0.025 times 0.227 -0.0583 Multiply -0.2 and -5.44. 1.088 -374 times -0.3 11.22 -2 times 3.57 77 Calculate 0.3 * 193. 57.9 -0.3561 -30.5 0.1 times 1.53

4 (e) -139 What is the third biggest value in :10 -4.1, 0.108, 8.070 Which is the second biggest value? (a) -10 (b) -107 (c) 1/0 (d) -1 (e) -1/0.15 (f) -0.1 If What is the second biggest value in -541 -0.07 , 0.15, -27 -3.5 Which is the fifth smallest value? (a) -2 (b) 3/4 (c) -2/3/2007 (d) -3/2 (e) 2/9 (f) 2/5 If Which is the biggest value? (a) 0 (b) 0.05 (c) 2907

8 (3) C1 -N1 -C9 119.4 (3) C18-C13 -C14 117.9 (3) C1 -N1 -C13 131.2 (2) C12-C13 -C14 116.3 (3) C9 -N1-C11 109.3 (2) N2-C14 -C13 116.6 (3) C10-N2-C14 119.6 (3) N2-C14 -C13

temperature overnight. Filtration, concentration and flash column chromatography (toluene/EtOAc: 18:1) provided **Sat** (1.40 g, 77%) as a colorless oil. [0] (D) -32.5 +7.0 (t * 0.1, MeOH); [1] (H NMR (300 MHz, CDCl3) 8 2.12 (ddd, J = 7.7)

in -57985.5, -5.46847 -5 What is the smallest value in 1/14, -0.39, -191, 19, -0.57 -191 Which is the smallest value? (a) -1/20 (b) -0.4 (c) -1107131 (d) -0.2 What is the biggest value in -26475.8 -5747.2/2 Which is the fourth smallest value? (a) 3/8 (b) -4 (c) -0.2 203.3 (d) -619 A Which is the seventh smallest value? (a) 0.1 (b) 1/9 (c) -1/00 (d) -0.4 (e) 0.1 (f) 2/
Figure 29: A cluster responding to words followed by commas (or conjunctive pronouns?).

Figure 30: A cluster responding to spans of time.
Activation clusters:

Model: GPT2-small Layer: 4 Cluster ID: 673

Figure 31: A 'detokenization' cluster that responds both to the word “is” and its contraction.
Figure 33: A cluster responding to forward slashes in file paths.

Figure 34: A cluster responding to multiples of ten (verbal and numeric).