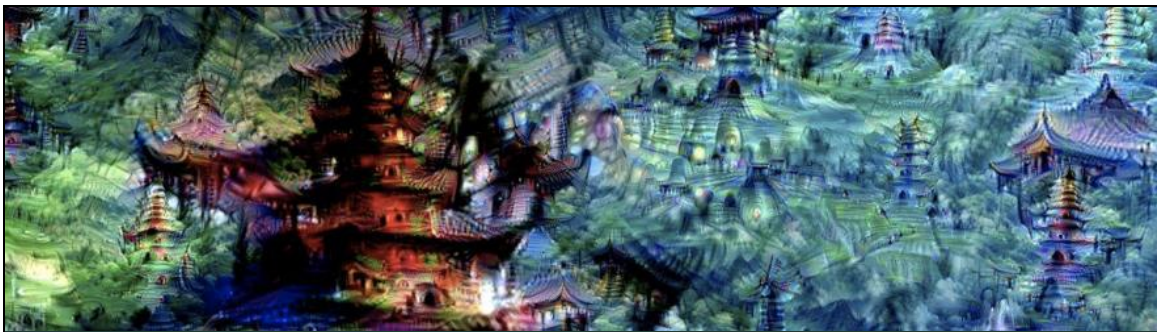


The background of the slide is a dark teal color. It features several concentric circles of varying shades of teal and grey, some of which have a grainy, textured appearance. On the left side, there is a large, semi-transparent teal shape that resembles a stylized letter 'L' or a corner of a square. The text is positioned on the right side of the slide.

ZOOM IN

AN
INTRODUCTION
TO
CIRCUITS



Why bother Zoom in?

Yes, it's deepdream in the background, we all remember the excitement and widespread interest this image generated when it came out, because, for the first time ever, it allowed the general public to understand how a neural network processed images.

And yet, even after 8 years, we haven't really understood the intricate details of its workings.

It's crazy how neural networks perform brilliantly and can detect cancer, predict climate, drive cars autonomously, and now, chat with us like a superhuman.

And even though all of the theory that we study makes sense, neural networks are still a black box to us when it comes to how they do the magic that they do.

Unraveling this mystery is now a new branch of machine learning known as interpretability.

And in an effort to interpret the Blackbox of neural networks, the authors of this paper opted to study individual neurons, their connections, and the weights that determine these connections.



Three Speculative Claims

Claim 1: Features

Claim 2: Circuits

Claim 3: Universality

Three empirical speculative claims: Features, Circuits, and Universality. These claims form the foundation of our exploration into neural networks and provide a framework for understanding their complex mechanisms.



Let's start with the first claim.

Features are the fundamental unit of neural networks.

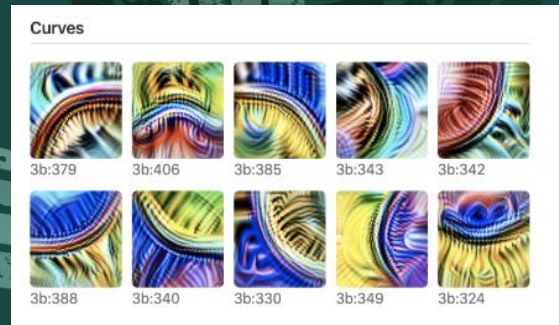
They correspond to directions.

These features can be rigorously studied and understood.

Features in neural networks can be understood as detectors.

For example, early layers contain features like edge or curve detectors, while later layers have features like floppy ear detectors or wheel detectors.

Example 1: Curve Detectors



Curve detectors are an example of these features.

Each unit in a neural network is responsible for detecting a specific feature in the input data. In the case of image recognition, these features might be lines, edges, textures, colors, shapes, etc.

Curve detectors are the units that have learned to recognize curves in the input data.

Groups of units that all detect the same type of feature (in this case, curves), but in different orientations are referred to as families of units in this article.

For example, one unit might detect vertical curves, another might detect horizontal curves, and another might detect diagonal curves. Together, these units form a "family" of curve detectors.

This is an important aspect of how neural networks process visual data.

By having different units specialize in detecting the same feature at different orientations, the network can build a more complete understanding of the patterns in the data.

But are these "curve detectors" really detecting curves?

This paper offers seven arguments, as a general toolkit for testing our understanding of features.



Argument 1

FEATURE VISUALIZATION

Optimizing the input to cause curve detectors to fire reliably produces curves.
This establishes a causal link since everything in the resulting image was added to cause the neuron to fire more.

Argument 2

DATASET EXAMPLES



The ImageNet images that cause these neurons to strongly fire are reliably curves in the expected orientation.

The images that cause them to fire moderately are generally less perfect curves or curves off orientation.

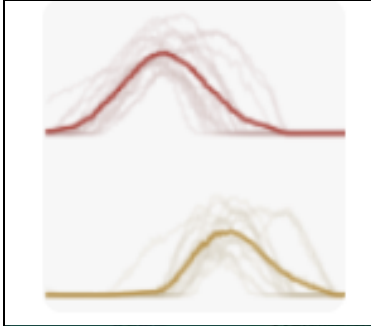
Argument 3

SYNTHETIC EXAMPLES



Curve detectors respond as expected to a range of synthetic curve images created with varying orientations, curvatures, and backgrounds.

They fire only near the expected orientation and do not fire strongly for straight lines or sharp corners.



Argument 4

JOINT TUNING

If we take dataset examples that cause a neuron to fire and rotate them, they gradually stop firing and the curve detectors in the next orientation begin firing. This shows that they detect rotated versions of the same thing. Together, they tile the full 360 degrees of potential orientations.

Argument 5

FEATURE IMPLEMENTATION



By looking at the circuit constructing the curve detectors, we can read a curve detection algorithm off of the weights.
We also don't see anything suggestive of a second alternative cause of the firing, although there are many smaller weights we don't understand the role of.

Argument 6

FEATURE USE



The downstream clients of curve detectors are features that naturally involve curves (e.g. circles, 3d curvature, spirals...).

The curve detectors are used by these clients in an expected manner.



Argument 7

HANDWRITTEN CIRCUITS

Based on our understanding of how curve detectors are implemented, we can do a cleanroom reimplementation, hand-setting all weights to reimplement curve detection. These weights are an understandable curve detection algorithm and significantly mimic the original curve detectors.

above arguments seem to establish that

- (1) curves cause these neurons to fire,**
- (2) each unit responds to curves at different angular orientations,**
- and**
- (3) if there are other stimuli that cause them to fire those stimuli are rare or cause weaker activations.**



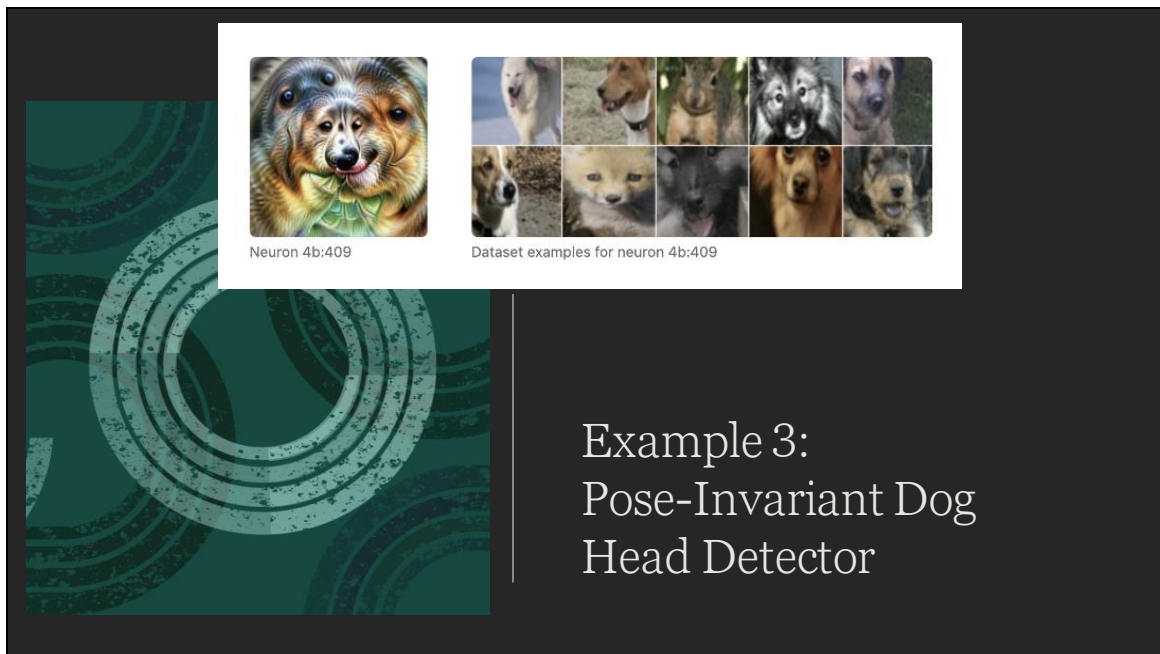
Example 2: High-Low Frequency Detectors

High-low frequency detectors are another type of feature. They look for low-frequency patterns on one side of their receptive field, and high-frequency patterns on the other side.

They seem to be one of several heuristics for detecting the boundaries of objects, especially when the background is out of focus."

All seven of the arguments described earlier can also be used to study high-low frequency neurons with some tweaking.

Authors believe these arguments collectively provide strong support for the idea that these really are a family of high-low frequency contrast detectors.



Moving to more complex, high-level features, we have the pose-invariant dog head detector.

the neuron is designed to recognize dog heads regardless of their orientation or pose. The image is a visualization of what the neuron is looking for when it's analyzing an input.

It's important to note that the geometry in the image is not possible in the real world, but it's very informative about what the neuron is looking for.

The image shows a dog head in various orientations, indicating that the neuron is looking for dog heads in any pose.

It's a great example of how neural networks can learn to recognize complex patterns and objects.



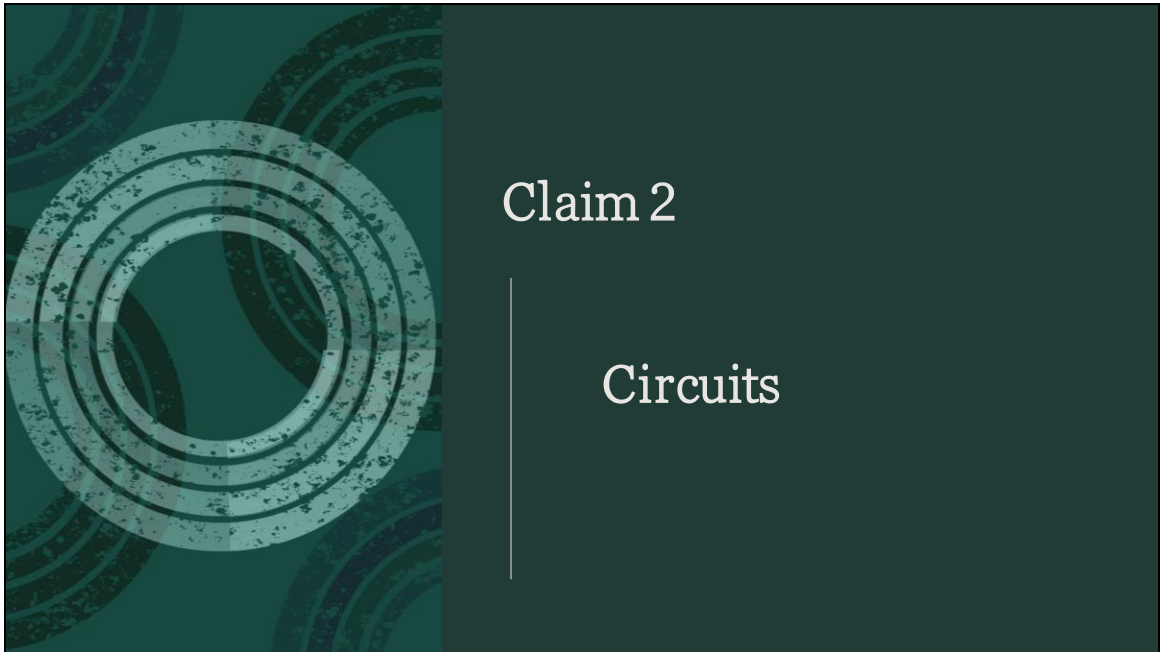
Polysemantic Neurons

So far understanding the neurons has been simple and obvious. However, not all neurons are so straightforward.

Some neurons, called polysemantic neurons, respond to multiple unrelated inputs. For example, InceptionV1 contains one neuron that responds to cat faces, fronts of cars, and cat legs.

To be clear, this neuron isn't responding to some commonality of cars and cat faces. Feature visualization shows us that it's looking for the eyes and whiskers of a cat, for furry legs, and for shiny fronts of cars — not some subtle shared feature.

One natural question to ask is why do polysemantic neurons form? They seem to result from a phenomenon we call “superposition” where a circuit spreads a feature across many neurons, presumably to pack more features into the limited number of neurons it has available.



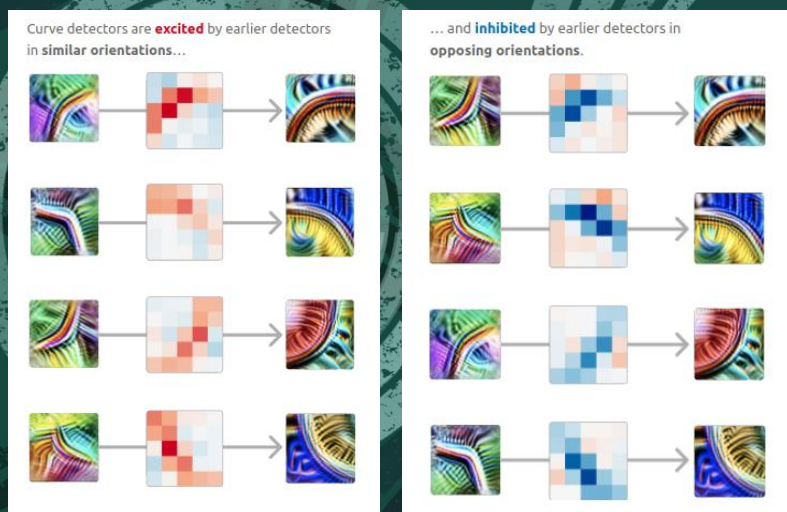
The second claim is that these features connect to form circuits.

A 'circuit' is a computational subgraph of a neural network.

It consists of a set of features and the weighted edges that go between them in the original network.

These circuits can also be rigorously studied and understood.

Circuit 1: Curve Detectors



Let's see how curve detectors are implemented from earlier features and connect to the rest of the model.

"Let's look at an example. Curve detectors are primarily implemented from earlier, less sophisticated curve detectors and line detectors. These curve detectors are used in the next layer to create 3D geometry and complex shape detectors."

In this case, the model is implementing a 5x5 convolution, so the weights linking these two neurons are a 5x5 set of weights, which can be positive or negative. A positive weight means that if the earlier neuron fires in that position, it excites the late neuron. Conversely, a negative weight would mean that it inhibits it.

The curve detectors are more likely to activate (or "fire") when they receive input from other detectors that are recognizing similar features.

For example, if a curve detector is designed to recognize curves oriented at a 45-degree angle,

it might be excited by input from other detectors that are recognizing lines or edges that are also oriented at a 45-degree angle.

The curve detectors are less likely to activate when they receive input from detectors

that are recognizing features in opposing orientations.

For example, if a curve detector is designed to recognize curves oriented at a 45-degree angle,

it might be inhibited by input from detectors that are recognizing lines or edges that are oriented at a 135-degree angle.

This behavior allows the curve detectors to specialize in recognizing curves in specific orientations.

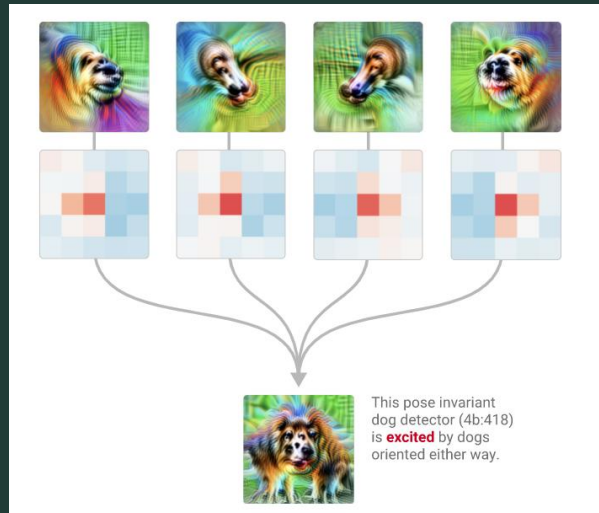
By being excited by similar features and inhibited by opposing features, they can more accurately and reliably detect the presence of curves in their preferred orientation.

It's also worth noting how the weights rotate with the orientation of the curve detector.

The symmetry of the problem is reflected as a symmetry in the weights.

Authors call circuits with exhibiting this phenomenon an "equivariant circuit"

Circuit 2: Oriented Dog Head Detection



A huge part of what an ImageNet model has to do is tell apart different animals. In particular, it has to distinguish between a hundred different species of dogs! And so, unsurprisingly, it develops a large number of neurons dedicated to recognizing dog-related features, including heads. This circuit uses a combination of features that respond to different parts of a dog's head, such as the eyes, ears, and snout. By combining these features, the network can recognize a dog's head from different angles.

As can be seen in the image, within the network, there are two parallel paths of neurons that are processing the input data. These paths are "mirrored," meaning they are doing similar computations but in opposite directions, i.e. detecting left-facing heads and right-facing heads.

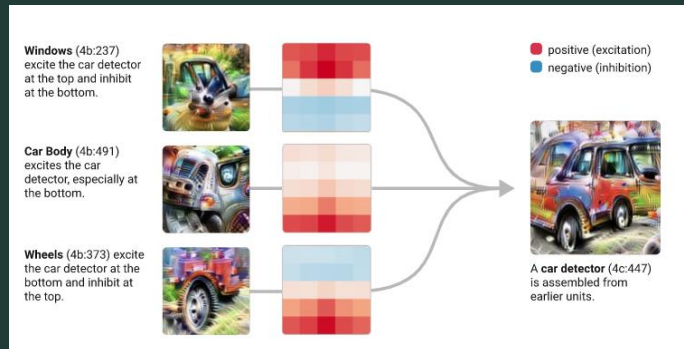
At each step, these pathways try to inhibit each other, sharpening the contrast meaning that the two paths are not just processing the data independently, they are also interacting with each other. Specifically, they are inhibiting each other, so that when one path activates, it suppresses the activity of the other path. This can help to sharpen the contrast between the features that each path is detecting, making it easier for the network to distinguish between them.

Authors call this pattern “unioning over cases”. The network separately detects two cases (left and right) and then takes a union over them to create invariant “multifaceted” units.

And because the two pathways inhibit each other, this circuit actually has some XOR-like properties.

As can be seen in the image, the union step is very interesting as the regions of excitation extend from the center in different directions depending on orientation, allowing snouts to converge into the same point.

Circuit 3: Cars in Superposition



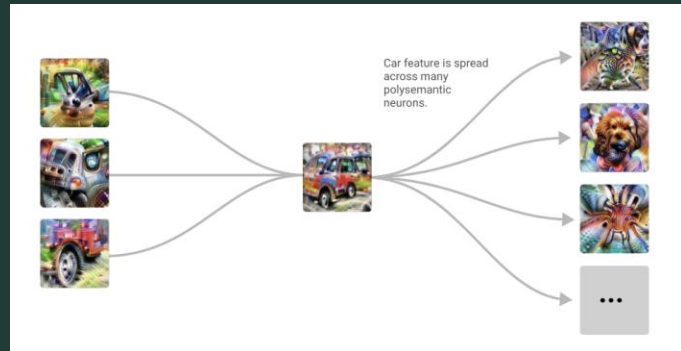
Now let's consider a specific example from the mid-late layer of the InceptionV1 model.

This layer contains a neuron that's designed to detect cars. It does this by looking for specific features associated with cars, such as wheels at the bottom and windows at the top.

However, something interesting happens in the next layer. Instead of creating another pure car detector, the model spreads the car feature over several neurons that seem to be primarily focused on detecting something else, in this case, dogs.

This might seem counterintuitive at first. Why would a model mix up car and dog features? This phenomenon is what the authors call 'superposition' and believe efficiency is the reason for it.

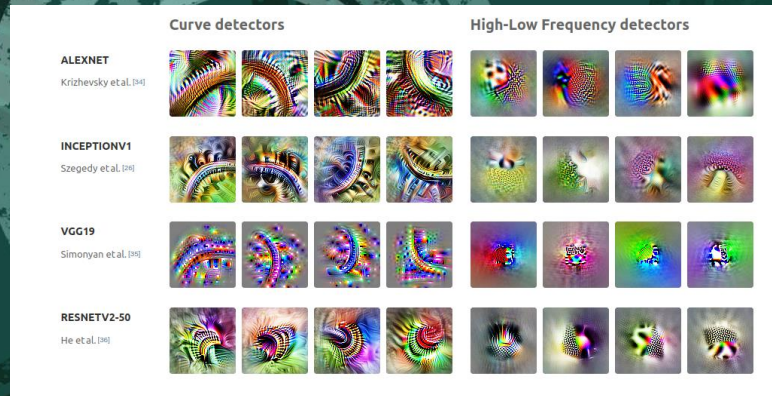
Circuit 3: Cars in Superposition



By superimposing the car feature onto the dog-detecting neurons, the model can save on resources. It doesn't need to dedicate a whole new neuron to the car feature. Instead, it can retrieve the car feature later on, as long as cars and dogs don't appear together in the same image.

This suggests that the occurrence of polysemantic neurons - i.e the neurons that respond to multiple, unrelated inputs - is not just a random event. Instead, it's a deliberate strategy used by the model to conserve resources and improve efficiency."

Claim 3 Universality



The third claim is that similar features and circuits form across different models and tasks. As can be seen in the image, the low level low-level features seem to form across a variety of vision model architectures (including AlexNet, InceptionV1, InceptionV3, and residual networks).

However, this claim is more speculative, and more research is needed to confirm it.




Experiments

Category 1:
Feature Visualization

Category 2:
Arguments Validation

Category 3:
Universality

These are the broad categories of experiments we can perform to implement this paper.



Category 1
Feature Visualization

Experiments

How many Low level features can you visualize?

Curves	Edges	Colors
Shapes	Textures	Gradients
	High-Low frequency	

The idea is to explore all these different low level features and visualize them. We can do this for different subsets of ImageNet or any other dataset as well. The aim is to get insightful visualizations that can help us understand what is exactly happening in all these different filters.



Another interesting experiment would be to validate the paper's hypothesis about these arguments that the authors have made.
These arguments make up a general toolkit for testing our understanding of other features.

Category 3
Universality

Experiments

MODELS

ALEXNET

GOOGLNET

RESNET

VGG-16

DATASETS



HYPER- PARAMETERS



Finally, we can try to validate the universality claim on our end with small experiments that work on different combinations of datasets, models and hyperparameter tuning.

You can even use your custom dataset and custom model.



THANK YOU