



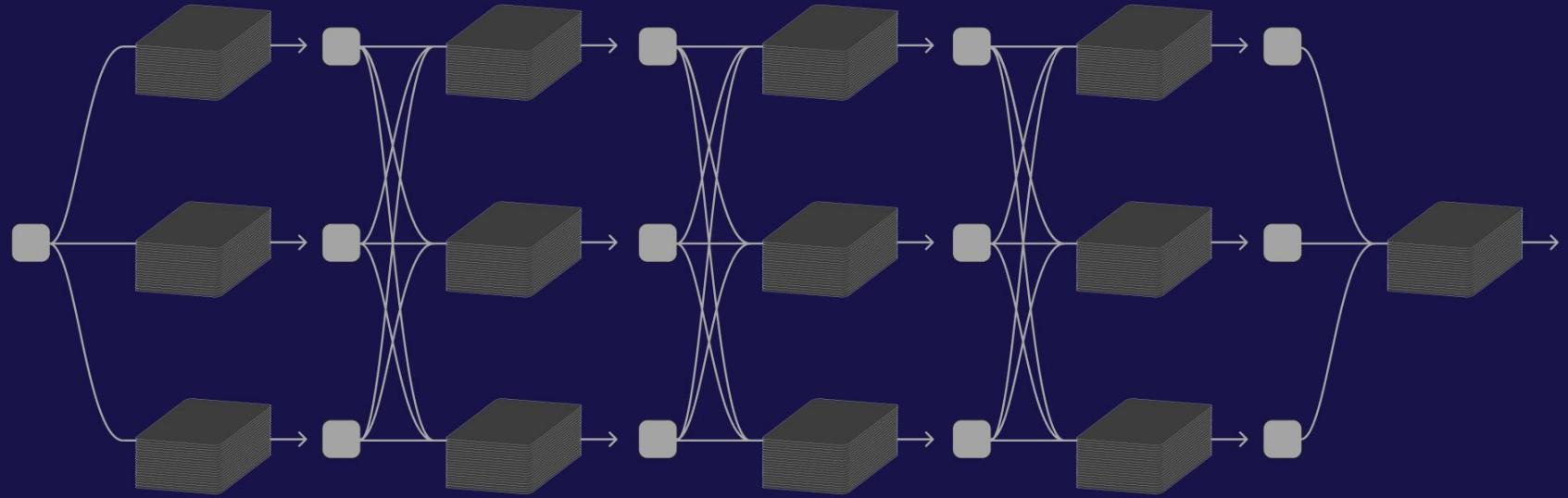
OpenAI

An Introduction to Circuits in CNNs

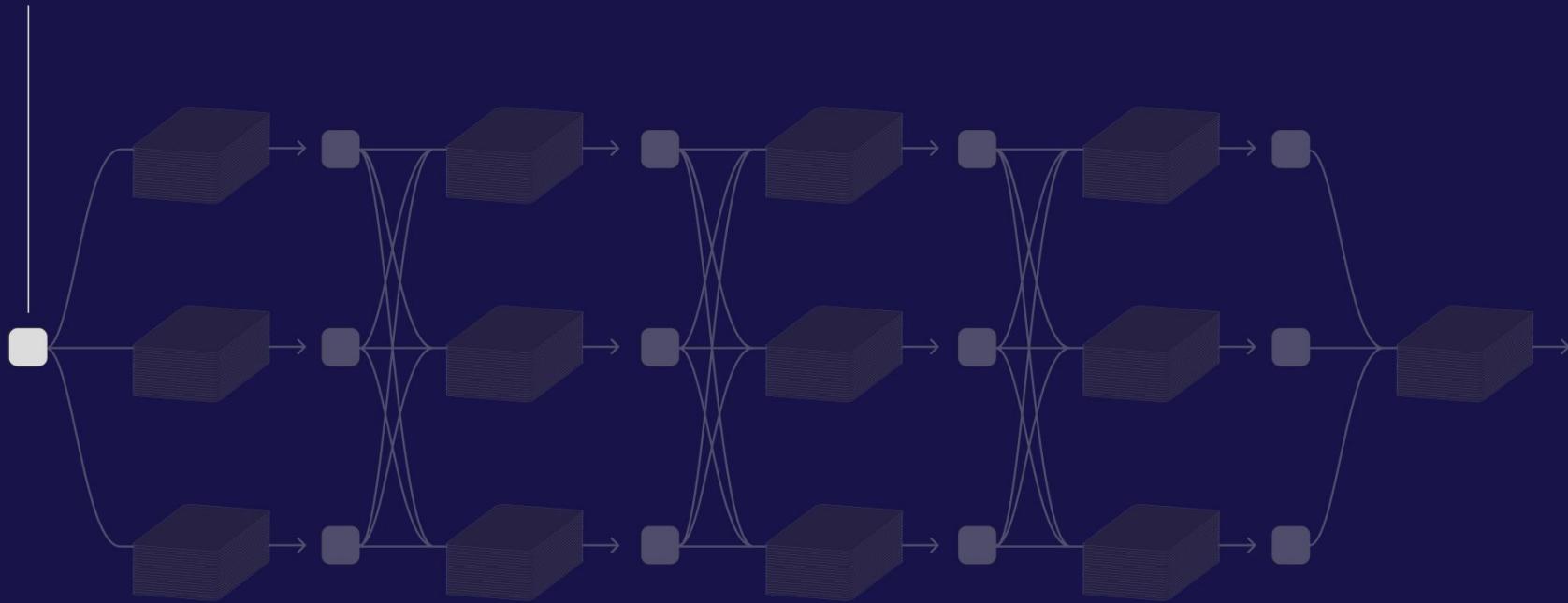
Chris Olah
OpenAI Clarity Team

Can we reverse engineer
neural networks?

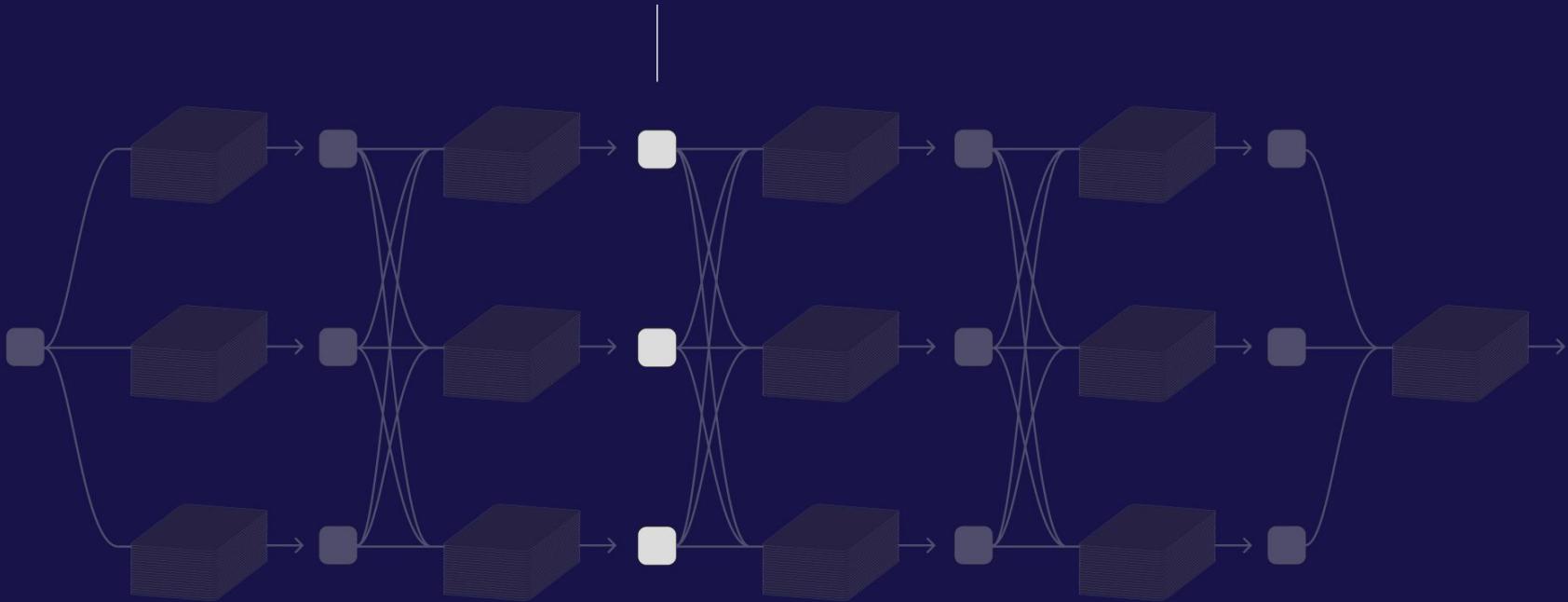
This generally isn't what interpretability
research aims to do



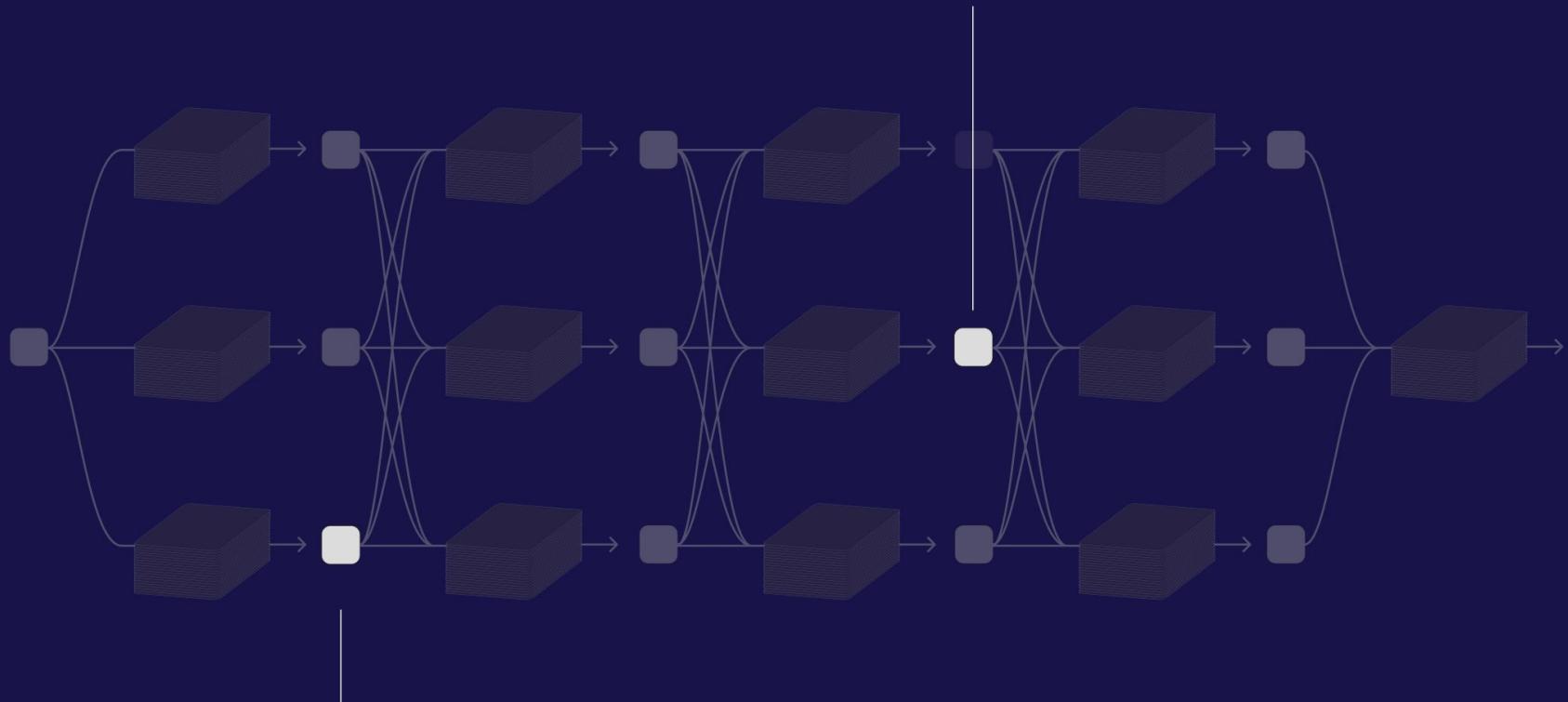
This part of the input influenced the output the most



We can predict X from this layer?



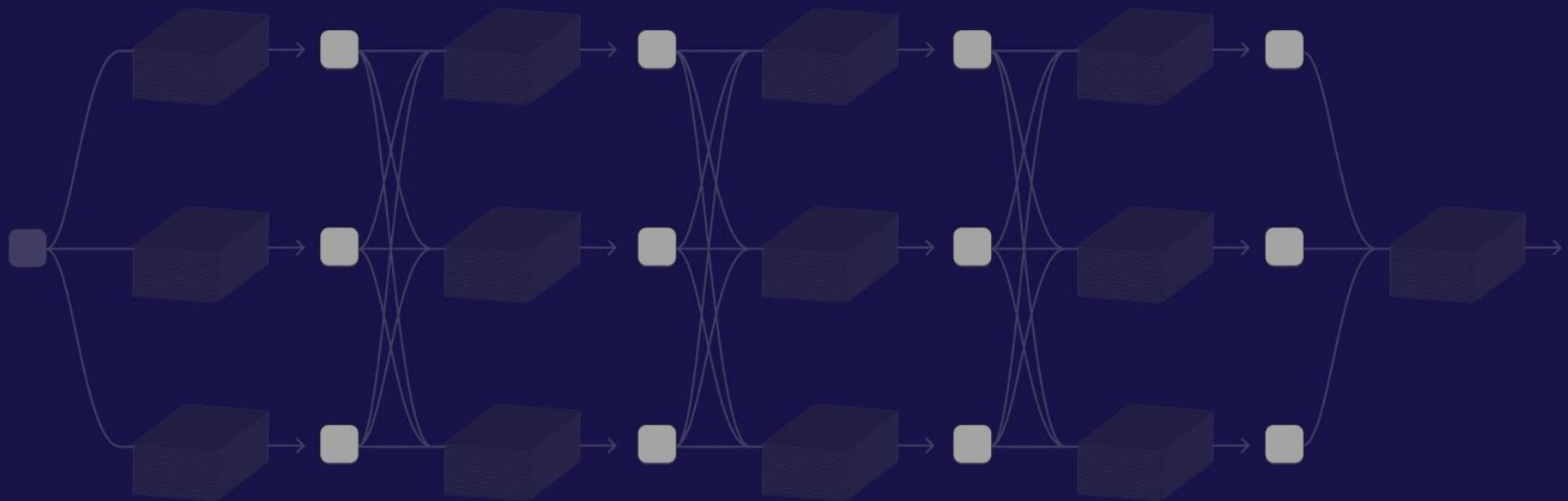
This seems to be a car detector?



But this one doesn't seem meaningful

What would it even mean to “reverse engineer”
a neural network?

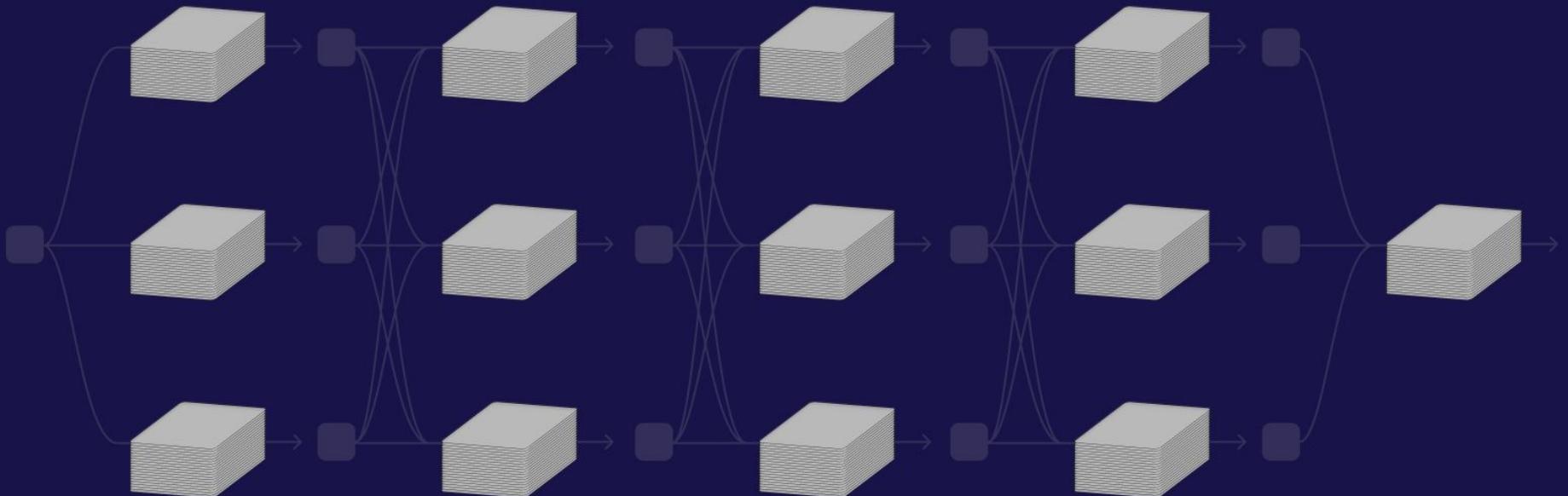
Understanding neurons is like understanding the variables in a computer program



Imagine looking at code like this:

```
is_line = ???  
is_curve = ???  
is_dog = ???  
is_high_low = ???  
is_window = ???  
is_wheel = ???  
is_dog_neck_head_right = ???  
is_car = ???
```

The weights are the actual “assembly code” of our model



To reverse engineer a neural network,
we need to understand the weights.

But the weights can only be understood if
we understand the neurons.

Reverse engineer a neural network
in two not at all easy steps:

1. Correctly understand all the neurons.
2. Understand the weights connecting them.

Step 1: Understanding Neurons

Step 1: analyze and understand every neuron.



Significant literature reporting on neurons or combinations of neurons which seem to track meaningful latent variables.

- eg. Mikolov et al 2013, Zhou et al 2014, Karpathy et al 2015, Radford et al 2017, Olah et al 2017.

Also a skeptical literature.

- eg. Donnelly & Roegiest 2017, Morcos et al 2018, Geirhos et al 2018, Ilyas et al 2019.

Machine learning doesn't have a tradition of evaluating these kinds of claims, so it isn't surprising there's disagreement.

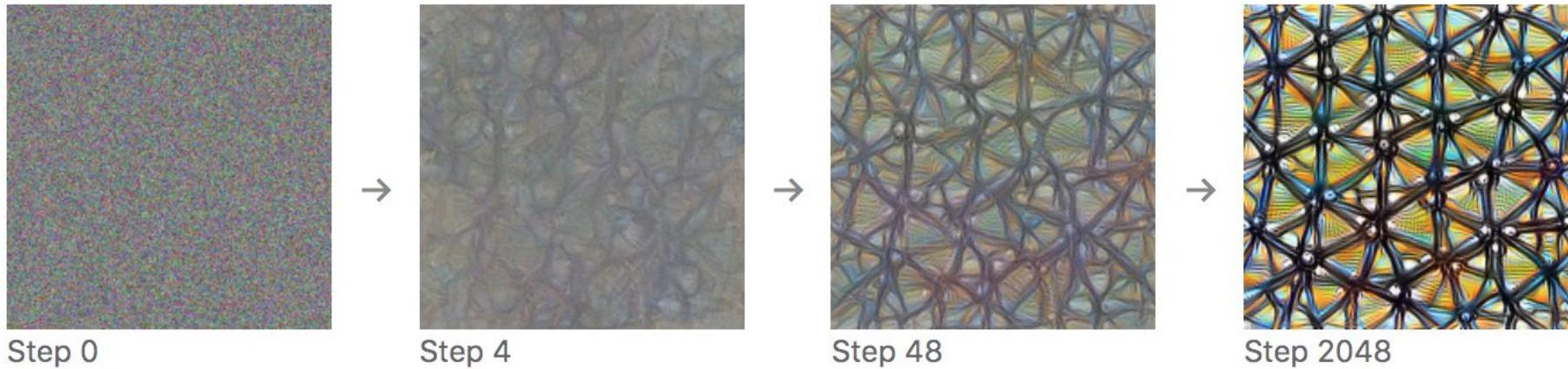
Example: Is neuron 4b:409 a dog head detector?

Dataset examples it fires for:



How do we know it isn't a snout detector with a large receptive field?
Or detecting a special head-only fur texture?

Feature Visualization (or activation maximization) studies features by doing gradient descent to create maximal stimuli.



See tutorial on Distill. Developed by many including:

- eg. Erhan et al., 2009; Mahendran & Vedaldi, 2015; Nguyen, et al., 2015; Mordvintsev, et al., 2015; Olah et al, 2017

Example: Is neuron 4b:409 a dog head detector?



Feature Visualization



Dataset Examples

Using feature visualization confirms that the neuron cares about a combination of snout, eyes and fur in a global structure: something more like a dog head.

We often use feature visualizations as “variable names” for neurons because



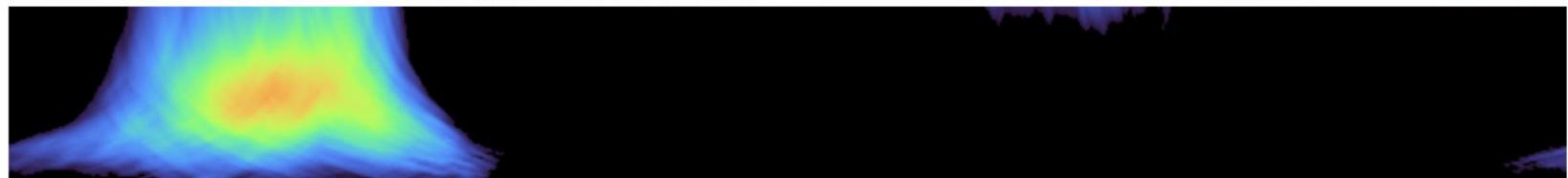
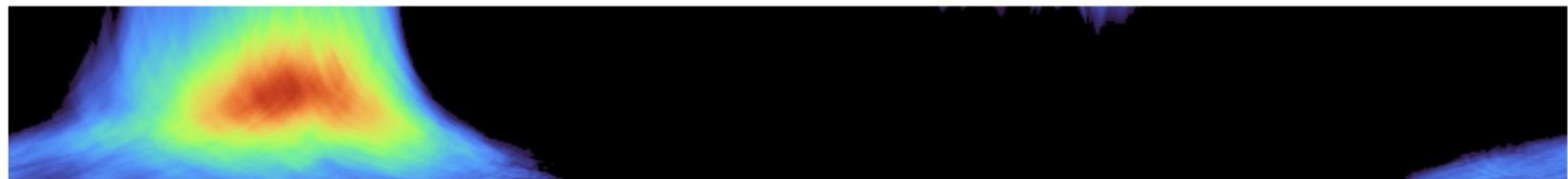
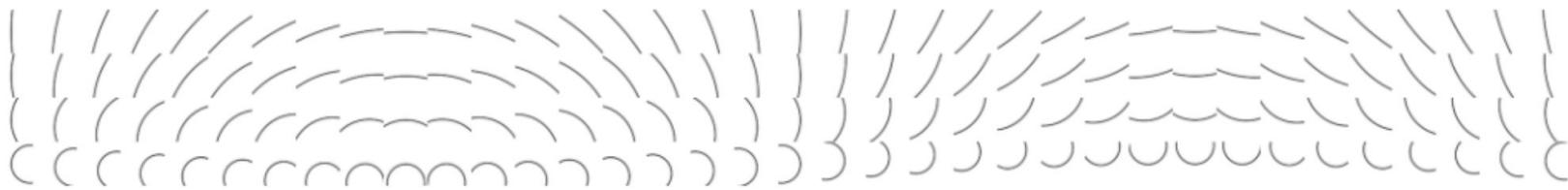
is more memorable and
Informative than **4b:409**

If we want to interrogate neurons more carefully, we can learn a lot from neuroscience. For example, how does this “curve detector” respond to changes in orientation?



3b:379 Activations by Orientation

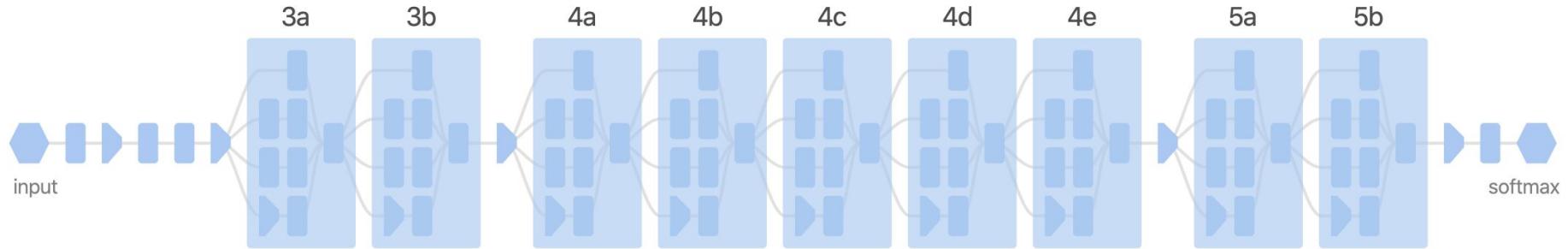




0 σ 25 σ
Standard Deviations

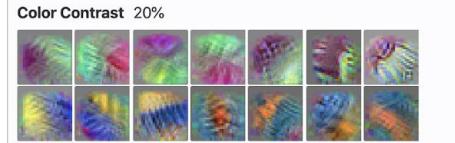
We could have a whole talk on
characterizing features!

Mostly focused on one model: InceptionV1



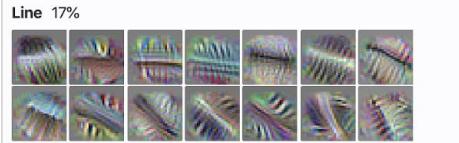
Systematically analyzed thousands of neurons,
using a battery of techniques.

Of ~10,000, we presently understand ~3,000
quite well, with ~80% coverage in early vision.



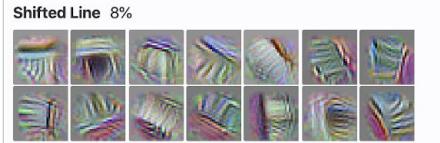
Show all 40 neurons.

These units detect a color on one side of the receptive field, and a different color on the opposite side. Composed of lower-level color contrast detectors, they often respond to color transitions in a range of translation and orientation variations. Compare to earlier color contrast ([conv2d0](#), [conv2d1](#)) and later color contrast ([mixed3a](#), [mixed3b](#)).



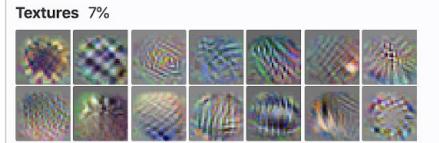
Show all 33 neurons.

These units are beginning to look for a single primary line. Some look for different colors on each side. Many exhibit "combing" (small perpendicular lines along the main one), a very common but not presently understood phenomenon in line-like features across vision models. Compare to [shifted lines](#) and later [lines](#) ([mixed3a](#)).



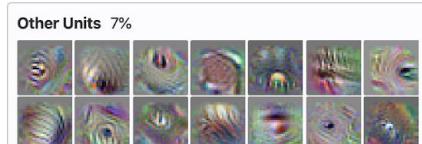
Show all 16 neurons.

These units look for edges "shifted" to the side of the receptive field instead of the middle. This may be linked to the many 1×1 convs in the next layer. Compare to [lines](#) (non-shifted) and later [lines](#) ([mixed3a](#)).



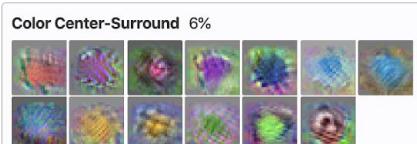
Show all 15 neurons.

A broad category of units detecting repeating local structure.



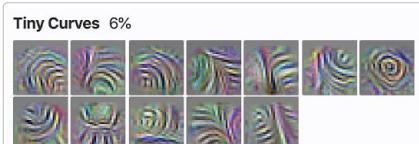
Show all 14 neurons.

Catch-all category for all other units.



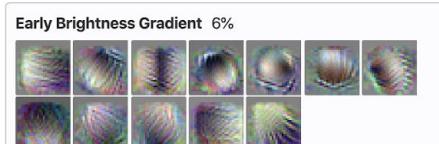
Show all 13 neurons.

These units look for one color in the middle and another (typically opposite) on the boundary. Generally more sensitive to the center than boundary. Compare to later [Color Center-Surround](#) ([mixed3a](#)) and [Color Center-Surround](#) ([mixed3b](#)).



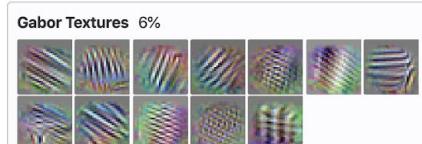
Show all 12 neurons.

Very small curve (and one circle) detectors. Many of these units respond to a range of curvatures all the way from a flat line to a curve. Compare to [later curves](#) ([mixed3a](#)) and [curves](#) ([mixed3b](#)). See also circuit example and discussion of use in forming small circles/eyes ([mixed3a](#)).



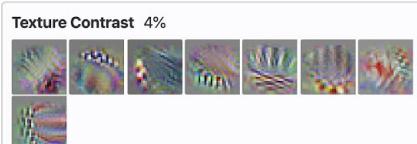
Show all 12 neurons.

These units detect oriented gradients in brightness. They support a variety of similar units in the next layer. Compare to [later brightness gradients](#) ([mixed3a](#)) and [brightness gradients](#) ([mixed3b](#)).



Show all 12 neurons.

Like complex Gabor units from the previous layer, but larger. They're probably starting to be better described as a texture.



These units look for different textures on opposite sides of their receptive field. One side is typically a Gabor pattern.



These units detect Gabor patterns in two orthogonal directions, selecting for a "hatch" pattern.



Several units look for mixtures of colors but seem indifferent to their organization.

Texture 25%



Show all 65 neurons.

This is a broad, not very well defined category for units that seem to look for simple local structures over a wide receptive field, including mixtures of colors. Many live in a branch consisting of a maxpool followed by a 1x1 conv, which structurally encourages this.⁸

Color Center-Surround 11%



Show all 30 neurons.

These units look for one color in the center, and another (usually opposite) color surrounding it. They are typically much more sensitive to the center color than the surrounding one. In visual neuroscience, center-surround units are classically an extremely low-level feature, but we see them in the later parts of early vision. Compare to earlier Color Center-Surround (`conv2d2`) and later Color Center-Surround (`mixed3b`).

High-Low Frequency 5%



Show all 15 neurons.

These units look for transitions from high-frequency texture to low-frequency. They are primarily used by boundary detectors (`mixed3b`) as an additional cue for a boundary between objects. (Larger scale high-low frequency detectors can be found in `mixed4a` (245, 93, 392, 301), but are not discussed in this article.)

A detailed article on these is forthcoming.

Brightness Gradient 5%



Show all 15 neurons.

These units detect brightness gradients. Among other things they will help detect specularity (shininess), curved surfaces, and the boundary of objects. Compare to earlier brightness gradients (`conv2d2`) and later brightness gradients (`mixed3b`).

Color Contrast 5%



Show all 14 neurons.

These units look for one color on one side of their receptive field, and another (usually opposite) color on the opposing side. They typically don't care about the exact position or orientation of the transition. Compare to earlier color contrast (`conv2d0`, `conv2d1`, `conv2d2`) and later color contrast (`mixed3b`).

Complex Center-Surround 5%



Show all 14 neurons.

This is a broad, not very well defined category for center-surround units that detect a pattern or complex texture in their center.

Line Misc. 5%



Show all 14 neurons.

Broad, low confidence organizational category.

Lines 5%



Show all 14 neurons.

Units used to detect extended lines, often further excited by different colors on each side. A few are highly combed line detectors that aren't obviously such at first glance. The decision to include a unit was often decided by whether it seems to be used by downstream client units as a line detector.

Other Units 5%



Show all 14 neurons.

Catch-all category for all other units.

Repeating patterns 4%



Show all 12 neurons.

This is broad, catch-all category for units that seem to look for repeating local patterns that seem more complex than textures.

Curves 4%



Show all 11 neurons.

These curve detectors detect significantly larger radii curves than their predecessors. They will be refined into more specific, larger curve detectors in the next layer. Compare to earlier curves (`conv2d2`) and later curves (`mixed3b`).

A detailed article on these is forthcoming.

BW vs Color 3%



These "black and white" detectors respond to absences of color. Prior to this, color detectors contrast to the opposite hue, but from this point on we'll see many compare to the absence of color. See also BW circuit example and discussion.

Angles 3%



Fur Precursors 2%



Eyes / Small Circles 1%



Crosses / Diverging Lines 1%



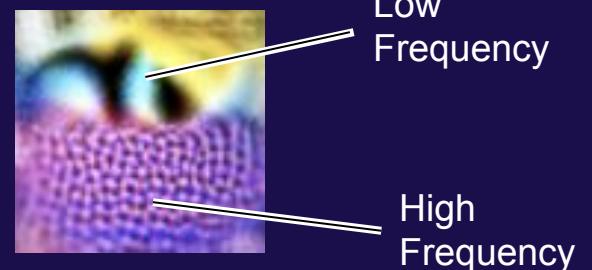
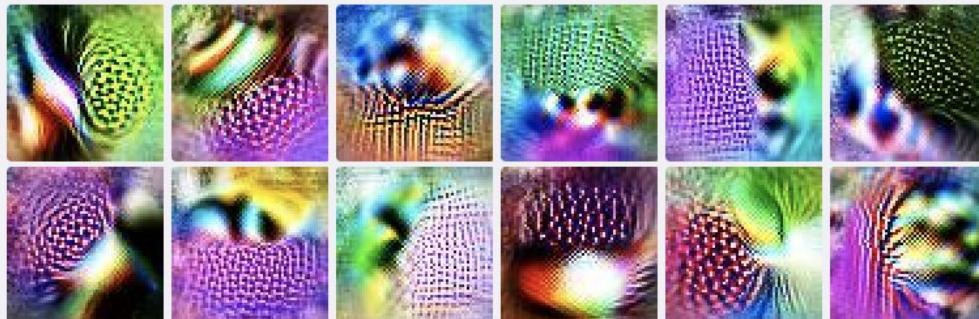
Some neurons are natural features
you might expect.

Curves 2%



Some neurons are mysterious at first
but turn out to be meaningful.

High-Low Frequency 6%

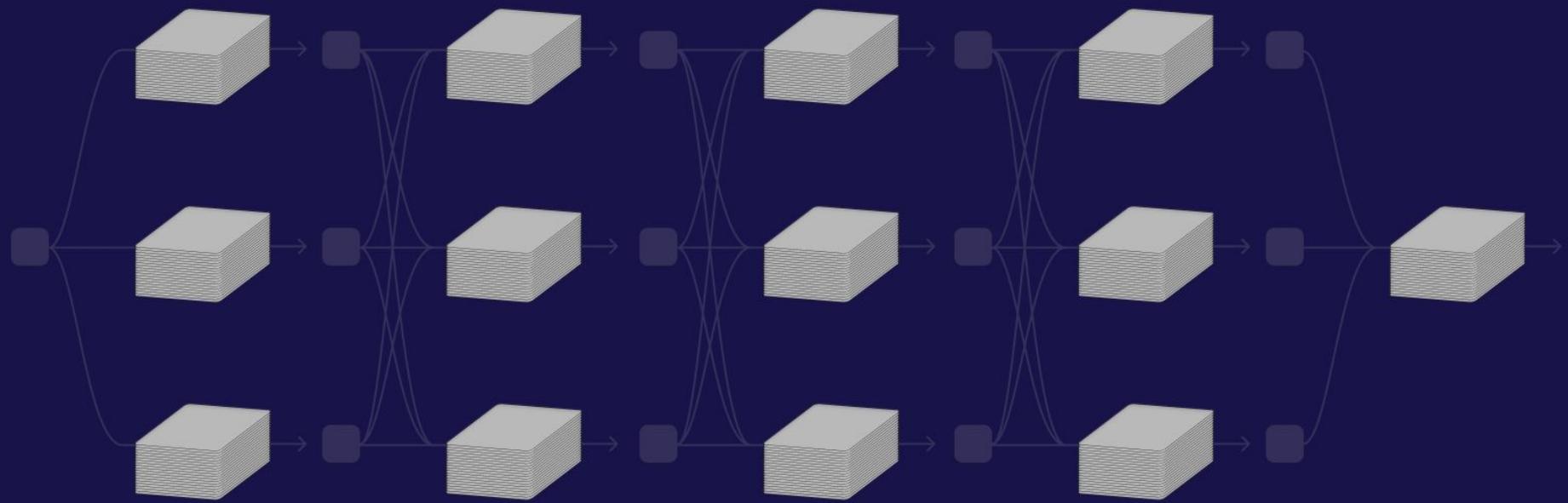


And some “polysemantic” neurons just turn out to respond to surprising mixtures of multiple things



Dataset examples

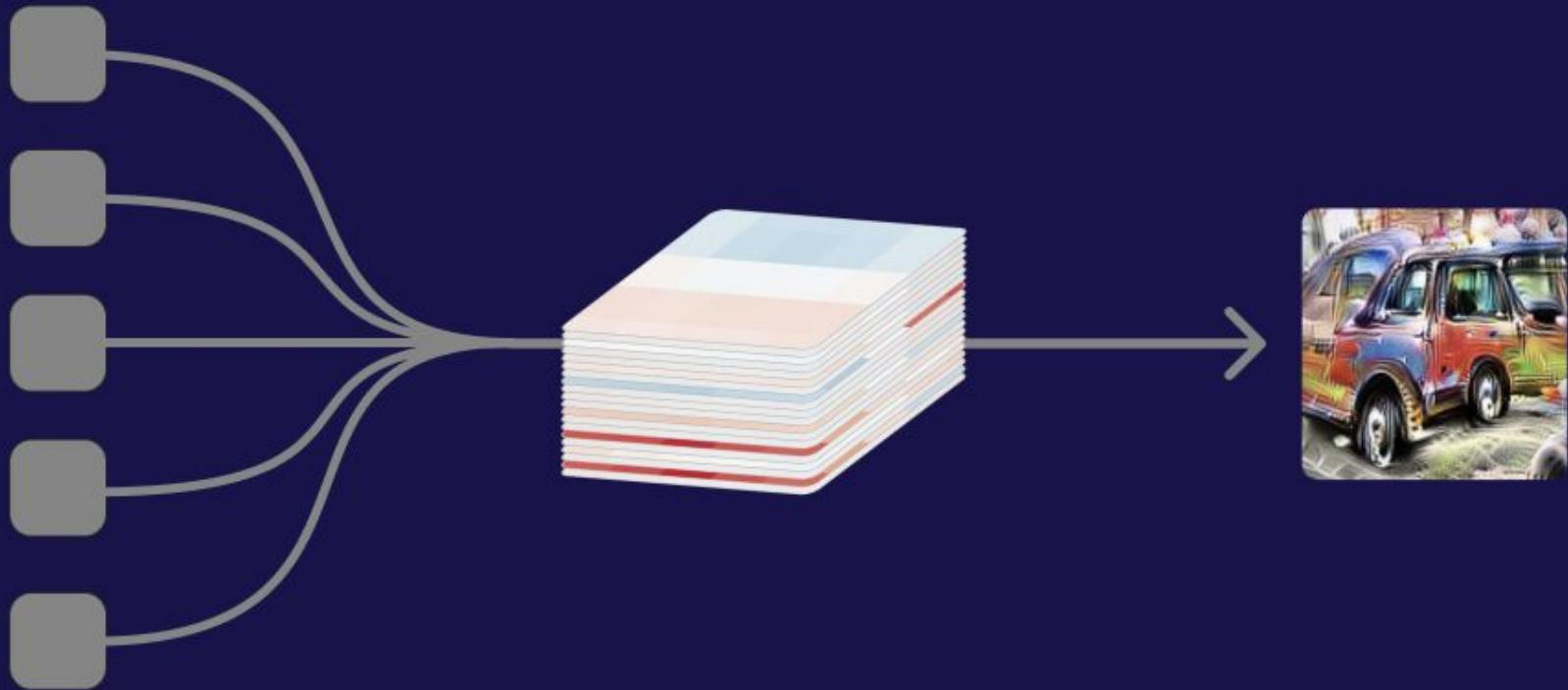
But how do we go from
understanding features
to understanding weights?

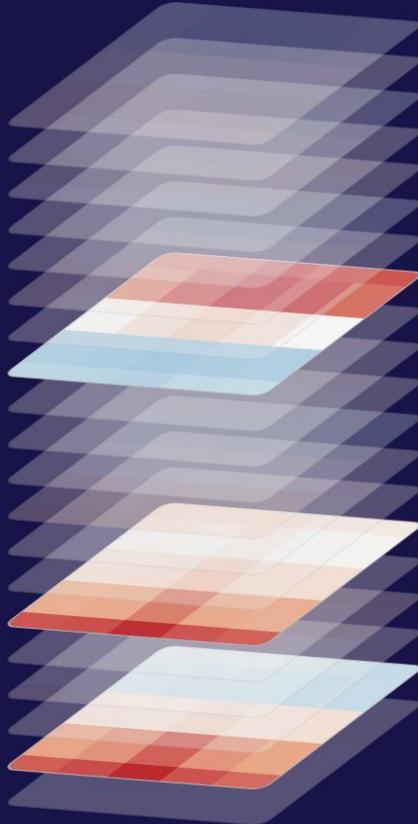


We often look at weights in the first conv layer,
but not others. Why is that?

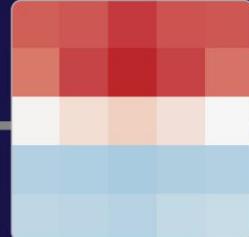


Weights in the first layer are *contextualized* automatically.
For other layers, we need to understand the neurons.



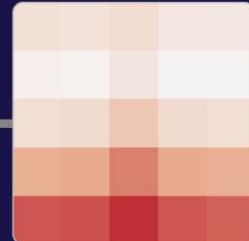


Windows (4b:237)
excite the car detector
at the top and inhibit
at the bottom.

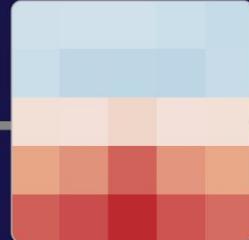


- positive (excitation)
- negative (inhibition)

Car Body (4b:491)
excites the car
detector, especially at
the bottom.

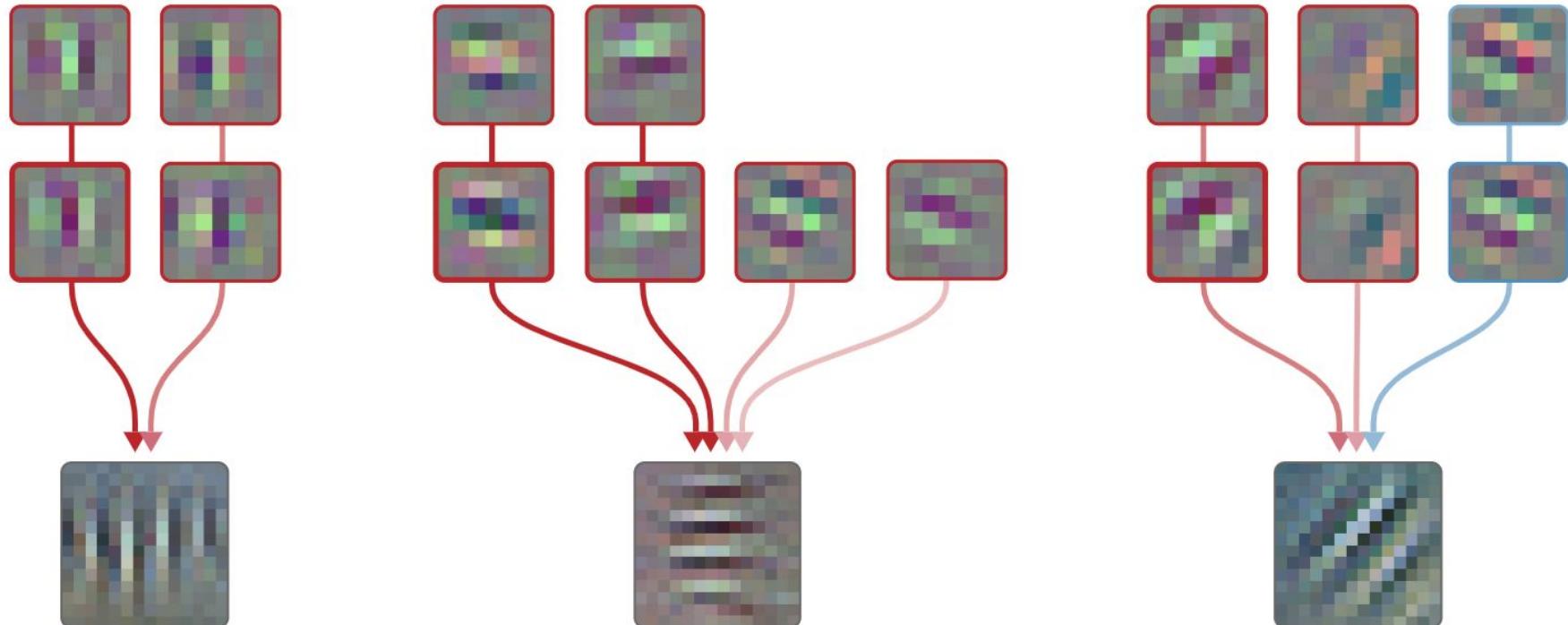


Wheels (4b:373) excite
the car detector at the
bottom and inhibit at
the top.

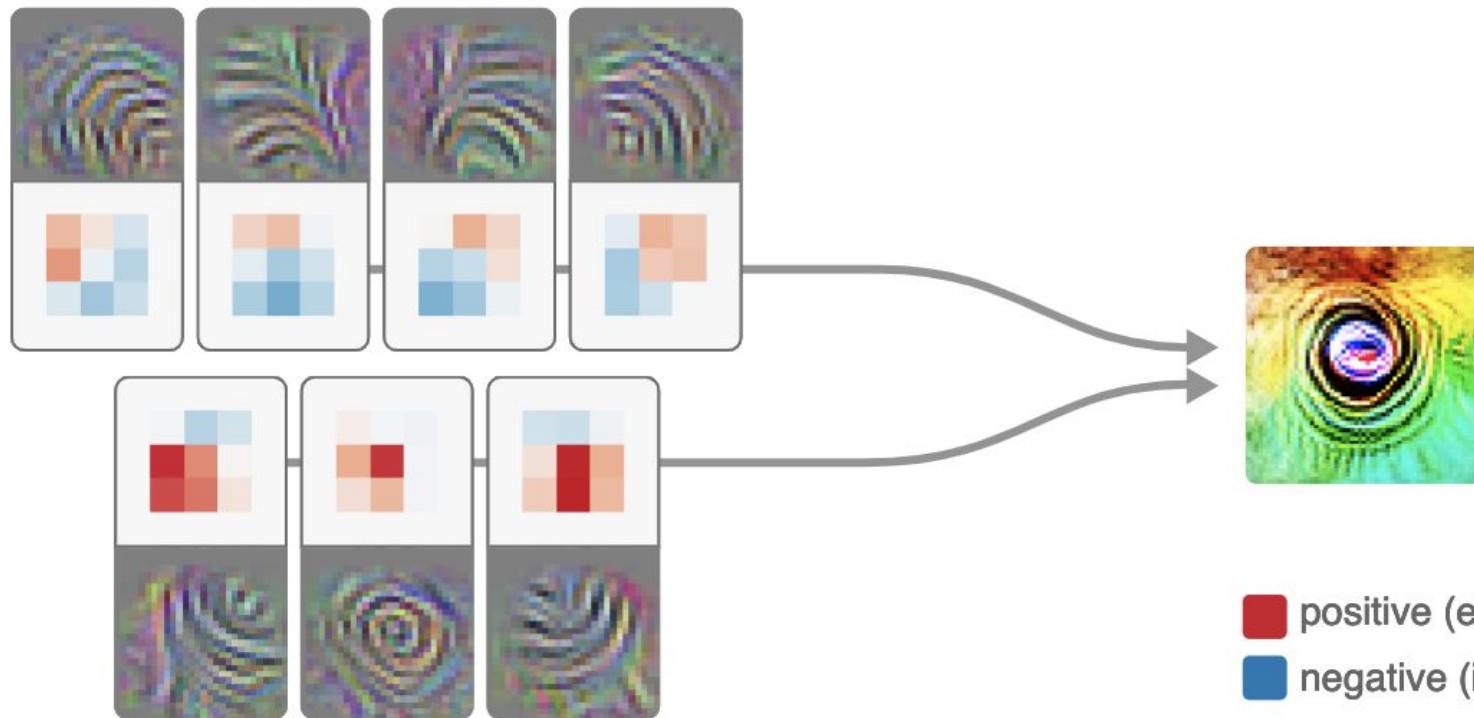


A car detector (4c:447)
is assembled from
earlier units.

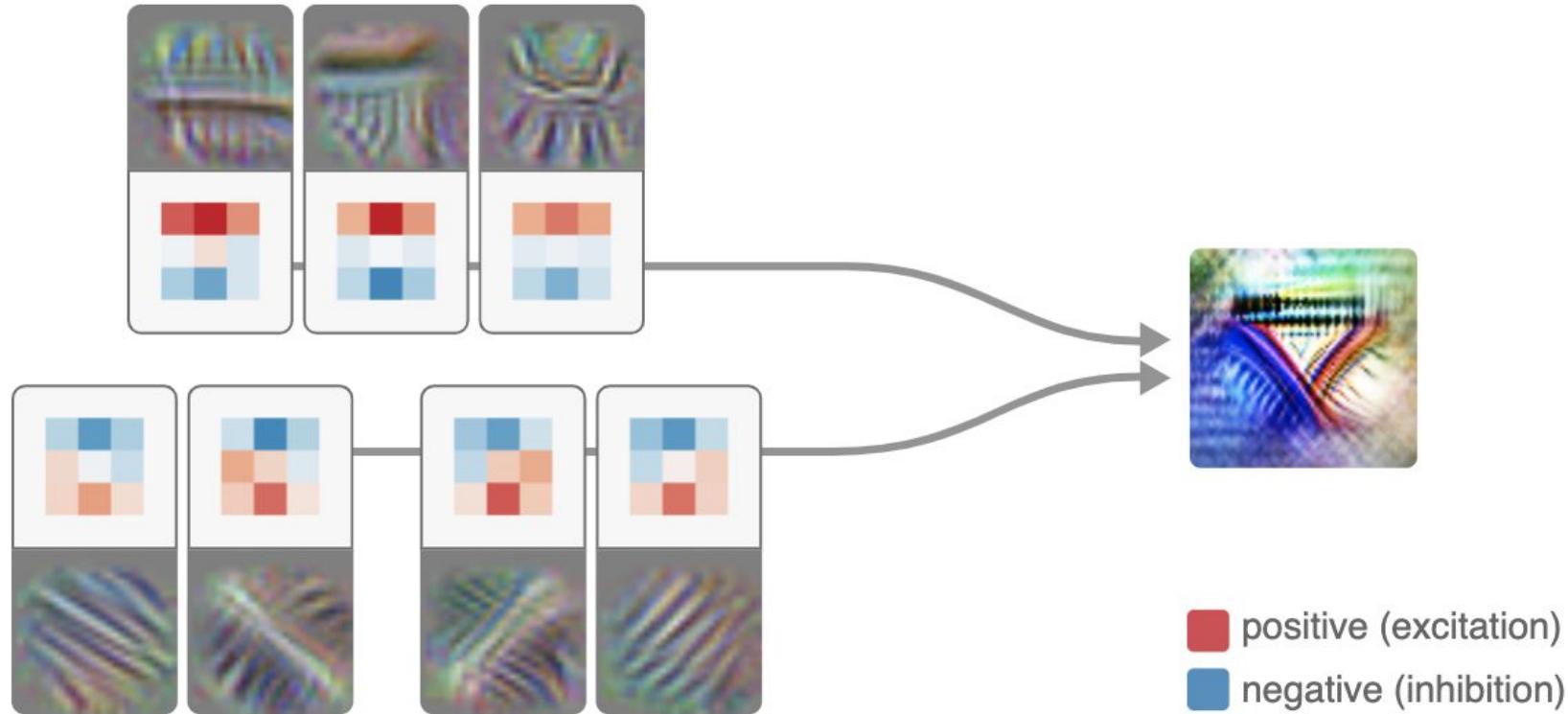
A couple interesting circuits



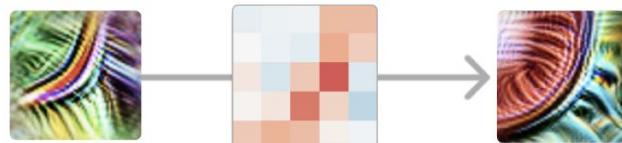
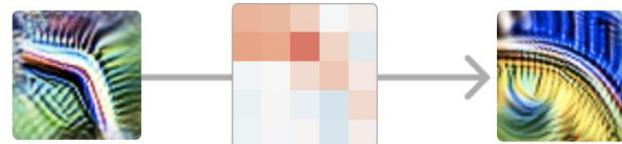
All neurons in the previous layer with at least 30% of the max weight magnitude are shown, both **positive (excitation)** and **negative (inhibition)**. Click on a neuron to see its forwards and backwards weights.



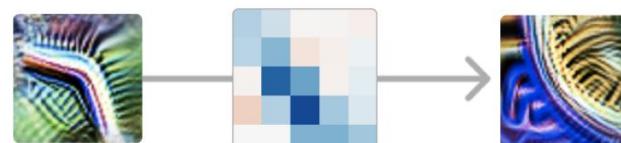
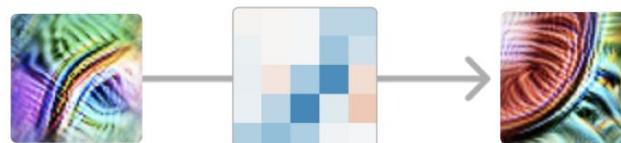
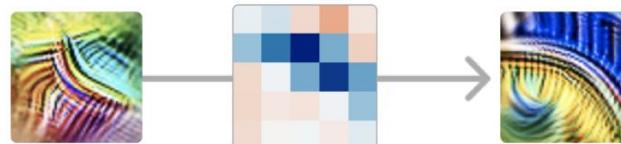
positive (excitation)
negative (inhibition)

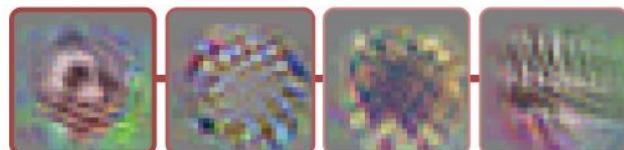
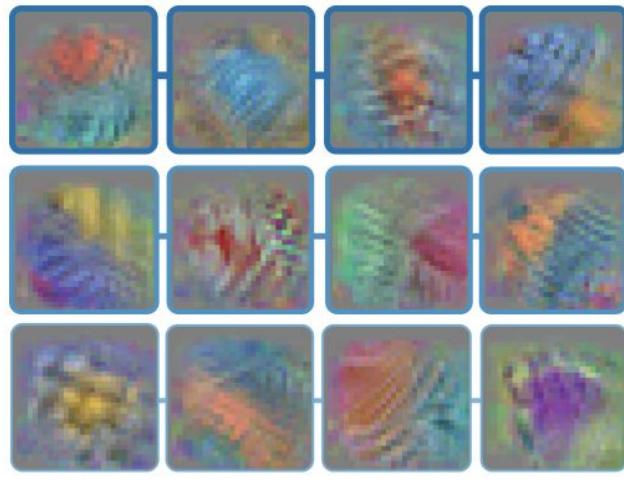


Curve detectors are excited by earlier detectors in **similar orientations**...

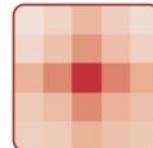
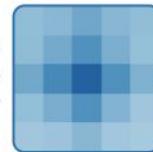


... and inhibited by earlier detectors in **opposing orientations**.



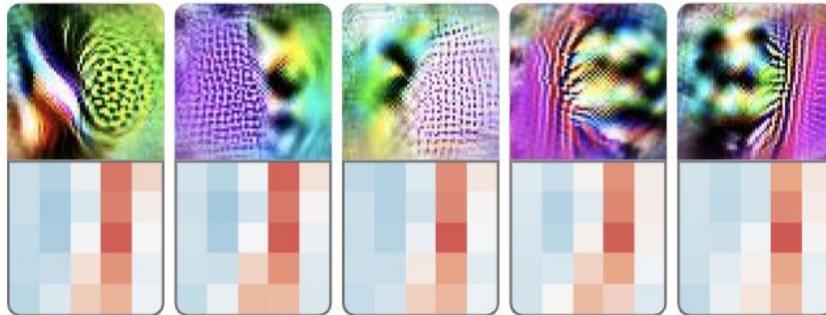


Black and white detectors are created by **inhibiting** against a wide variety of color detectors.

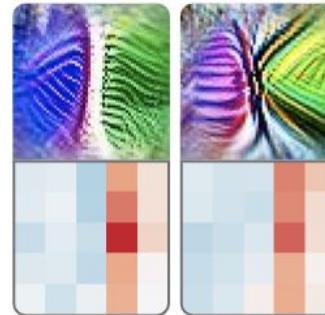


- positive (excitation)
- negative (inhibition)

High-low frequency detectors



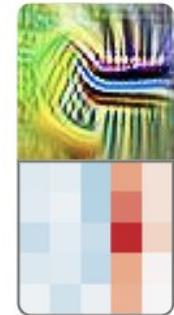
Edges



Color Contrasts



End of Line



mixed3b creates boundary detectors that rely on many cues, including changes in frequency, changes in color, and actual edges.

These detectors vary in orientation, preferring concave vs convex boundaries, and type of foreground.

- positive (excitation)
- negative (inhibition)

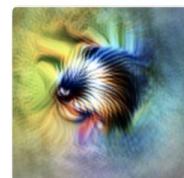
InceptionV1 has a **left-oriented** pathway detecting dogs facing left...

... and a symmetric **right-oriented** pathway detecting dogs facing right. At each step, the two pathways **inhibit** each other and **excite** the next stage.

Oriented Fur (3b)



Oriented Heads (4a)



Union over
left and right
cases.

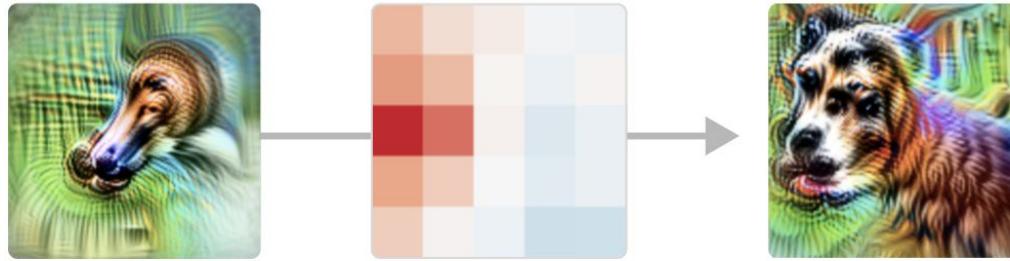
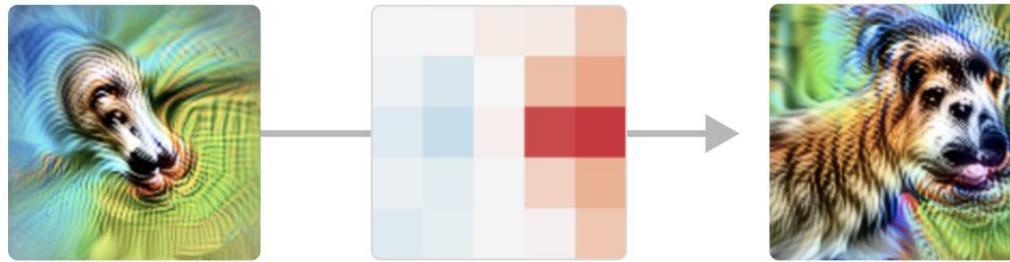
Union over
left and right
cases.

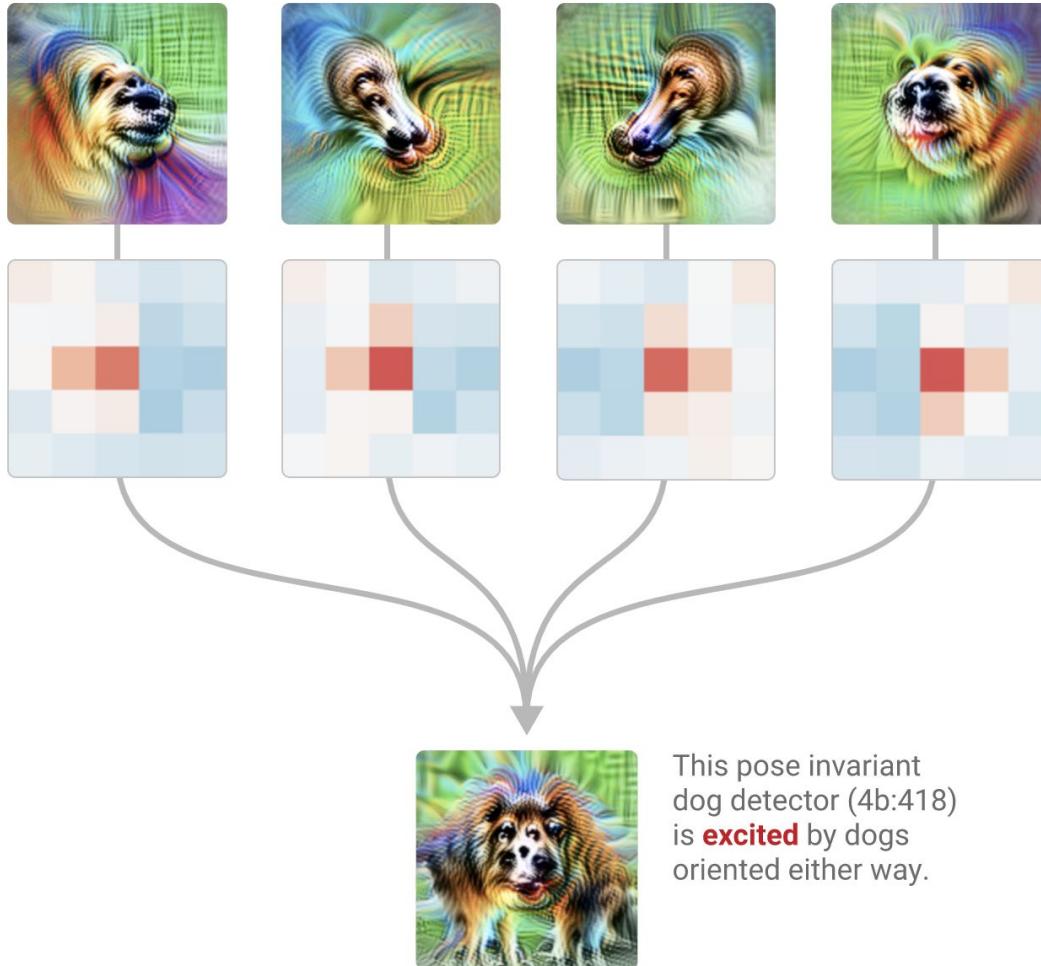
Orientation-Invariant Head (4b)



Orientation-Invariant Head+Neck (4c)

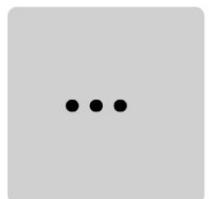








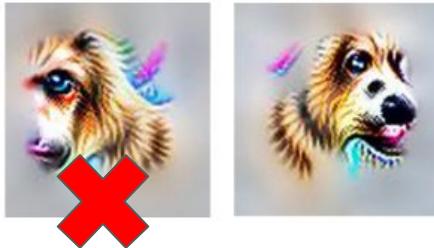
Car feature is spread across many polysemantic neurons.



Neural networks are full of beautiful structure
if you take the time to look!

How do we know we aren't fooling
ourselves?

Edit circuits to change model behavior

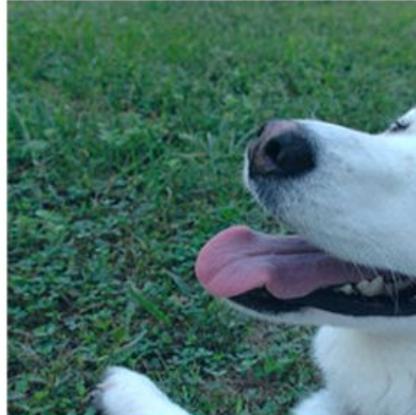


Dropping the right-facing head+neck detector
Causes animals facing right to be misclassified.

prediction: hippopotamus
edited prediction: chimpanzee



prediction: kuvasz
edited prediction: Great Pyrenees



prediction: carousel
edited prediction: cuirass (armor)



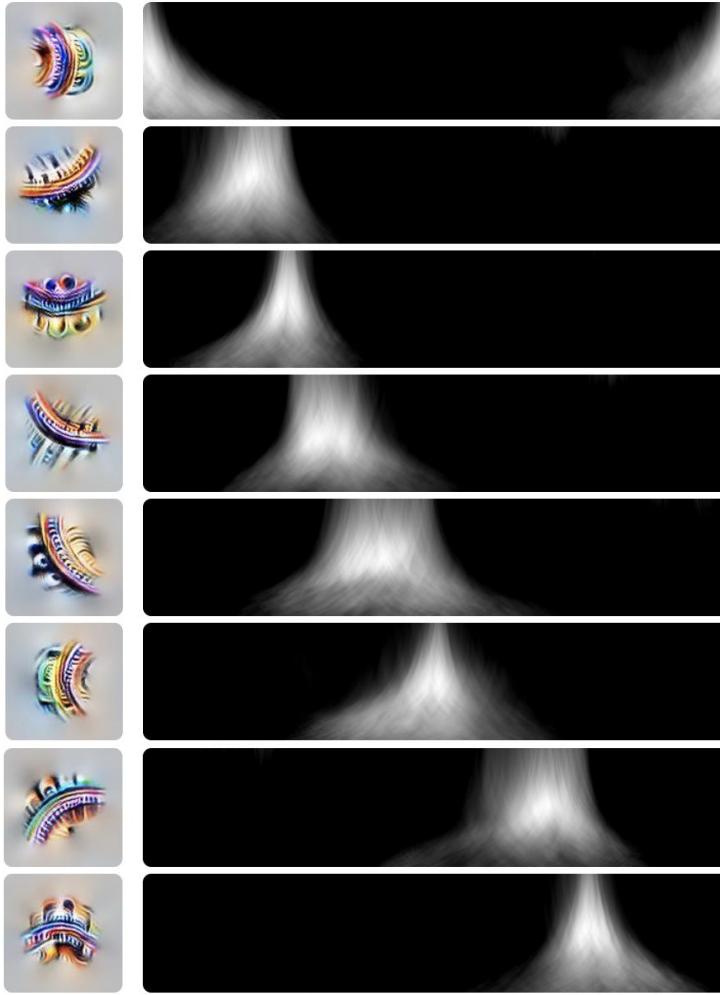
prediction: bighorn
edited prediction: warthog



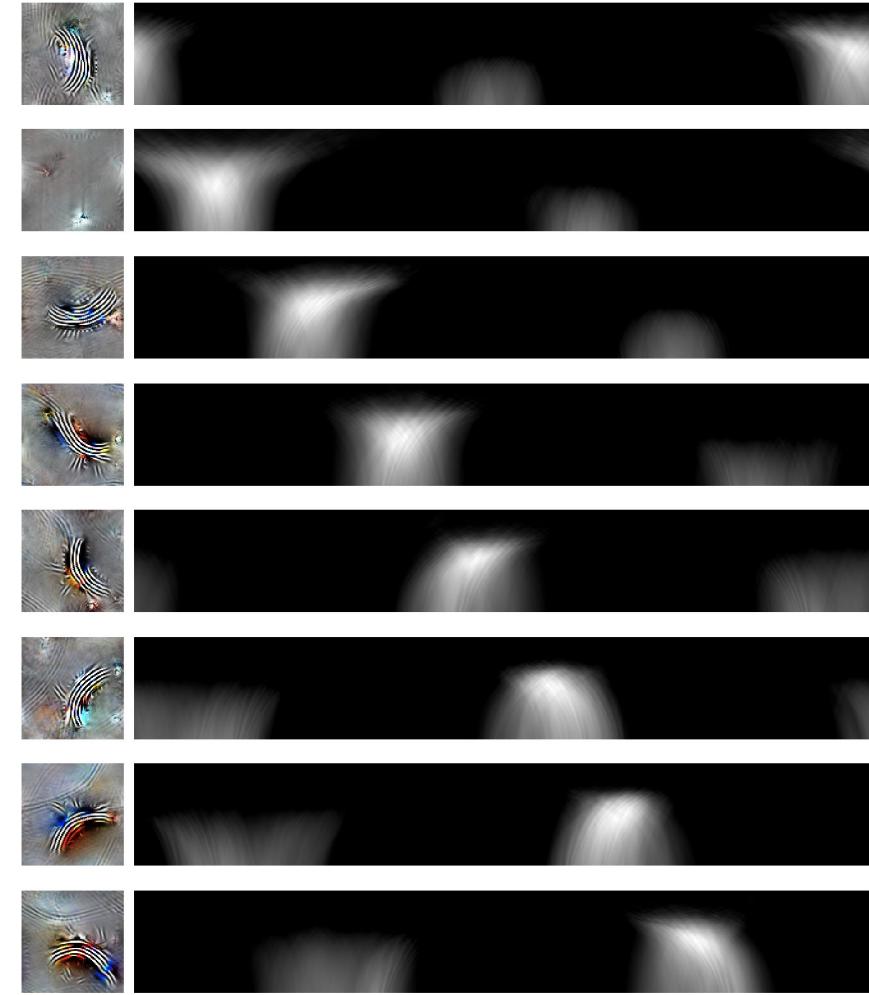
Clean room reimplementation of hundreds of neurons
over five layers, building up to curve detectors.

(Wrote a small python program that filled in the weights of a neural network)

“Natural” Curve Detectors



“Hand-written” Curve Detectors



Directions for Further Research

Challenge: Fully understand a single neural network

Presently, no human being understands any non-trivial neural network. Can we "fully understand" just a single neural network? If anyone can demonstrate this is truly possible, even if it takes immense effort, it seems like this would profoundly change dialogue around neural networks.

How would you even formalize this? InceptionV1 seems like the obvious target given our investment -- what would we need to do to fully understand InceptionV1?

Open Questions

When do "meaningful" features form?

Some models (eg. atari RL models) seem to largely not learn features that track meaningful latent variables, in contrast to typical features in ImageNet models which seem to. When and why do “meaningful features” form? Jacob Hilton has a forthcoming paper on this.

What about "polysemantic" neurons?

Polysemanticity is when a neuron responds to multiple, unrelated features. How widespread is polysemanticity? Why does it form? Is there a way to detect it without a human inspecting a neuron? Are there "hidden features" we never see because they're always small parts of many neurons? How can we remove polysemanticity as a barrier to completely understanding neural networks?

To what extent (and when) do the same features/circuits form across models? Is there a "periodic table" of visual features?

ALEXNET

Krizhevsky et al. [34]



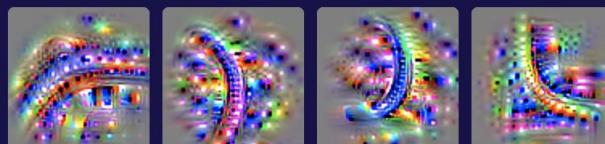
INCEPTIONV1

Szegedy et al. [26]



VGG19

Simonyan et al. [35]



How does network architecture influence the features/circuits that form, or how they are organized? What about sparsity, or adversarial training?

How do features and circuits evolve during training?

To what extent are there abstract recurring patterns -- "circuit motifs" -- in neural networks? Traditional study of circuit motifs relies on highly sparse graphs to systematically analyze them. Is there something analogous?

What kind of larger scale structure or organization exists among features and circuits?

How can we scale circuits-style analysis to very large models or large numbers of models? Candidate approaches:

- (1) Find a way to hierarchically understand circuits (Filan et al's modularity work both speak to this a little).
- (2) Find a way to study "diffs" between networks so that you don't need to start from scratch with new models (Gabriel Goh has unpublished work on this).

Building a research community

The content of this talk is from the Circuits thread on Distill:

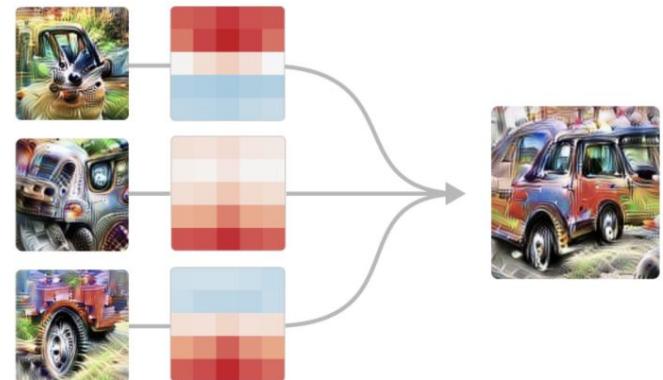
<https://distill.pub/2020/circuits/>

March 10, 2020

Thread: Circuits

Nick Cammarata, Shan Carter, Gabriel Goh, Chris Olah,
Michael Petrov, and Ludwig Schubert

What can we learn if we invest heavily in reverse engineering a single neural network?



Circuits slack channel:

Go to <http://slack.distill.pub> and join #circuits.

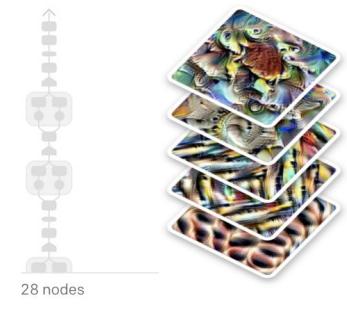
- >300 members
- ~20 active participants
- ~10 researchers from multiple institutions working on circuits projects
- Multiple unpublished drafts on circuits.

Multiple open source libraries implementing the underlying techniques.

The OpenAI Microscope
is a collection of
visualizations of every
significant layer and
neuron of eight
important vision models.

AlexNet

A landmark in computer vision, this 2012 winner of ImageNet has over 50,000 citations.



AlexNet (Places)

The same architecture as the classic AlexNet model, but trained on the Places365 dataset.



Inception v1

Also known as GoogLeNet, this network set the state of the art in ImageNet classification in 2014.



Inception v1 (Places)

The same architecture as the classic Inception v1 model, but trained on the Places365 dataset.



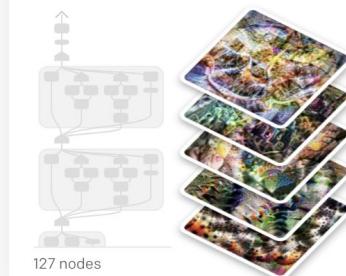
VGG 19

Introduced in 2014, this network is simpler than Inception variants, using only 3x3 convolutions and no branches.



Inception v3

Released in 2015, this iteration of the Inception architecture improved performance and efficiency.



*All the good stuff is my collaborators;
If something seems horribly off it's probably me.*

Clarity Team

Ludwig Schubert
Nick Cammarata
Gabriel Goh
Michael Petrov

And our friends at other orgs :)

Shan Carter
Arvind Satyanarayan
Ian Johnson
Katherine Ye
Alex Mordvintsev
Zan Armstrong
Ria Kalluri
Brice Menard