Topics:

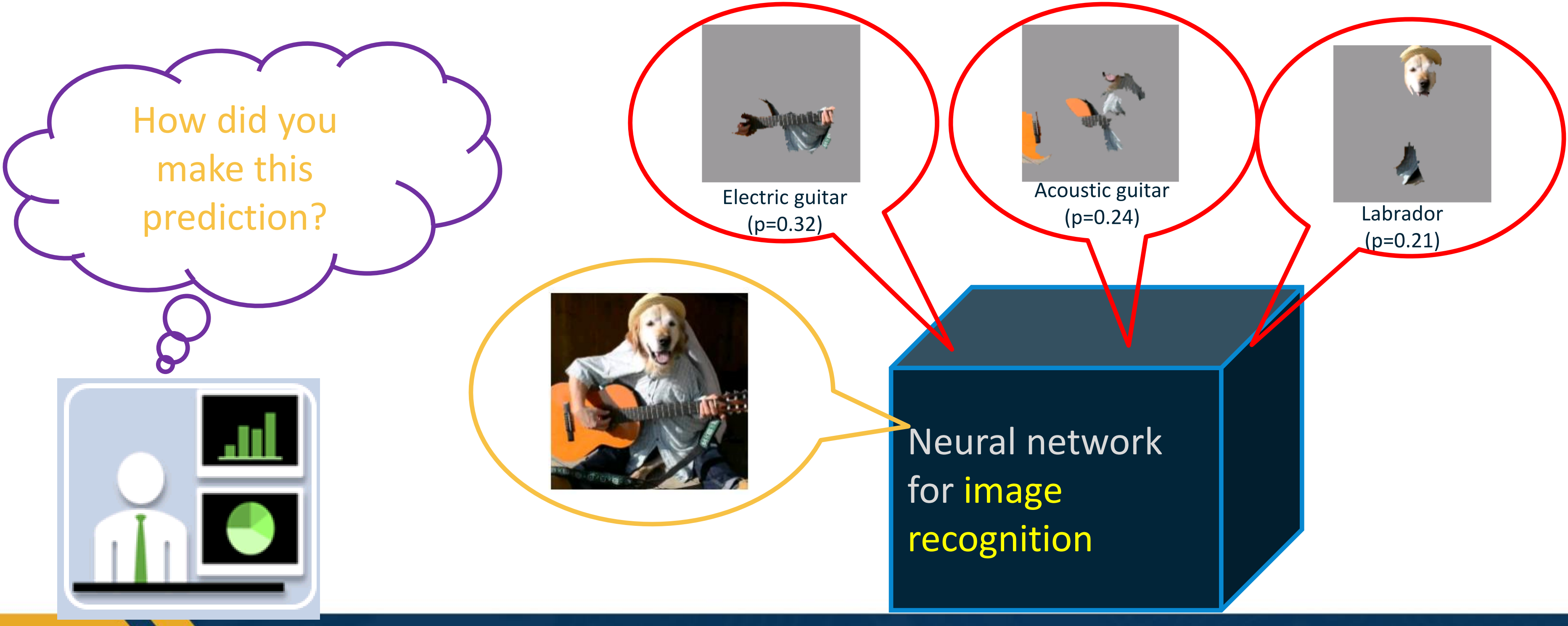
- Visualization

CS 4644-DL / 7643-A
ZSOLT KIRA

Visualization of Neural Networks

Interpretability Enables Trust in AI Models

Understand the reasons behind a prediction



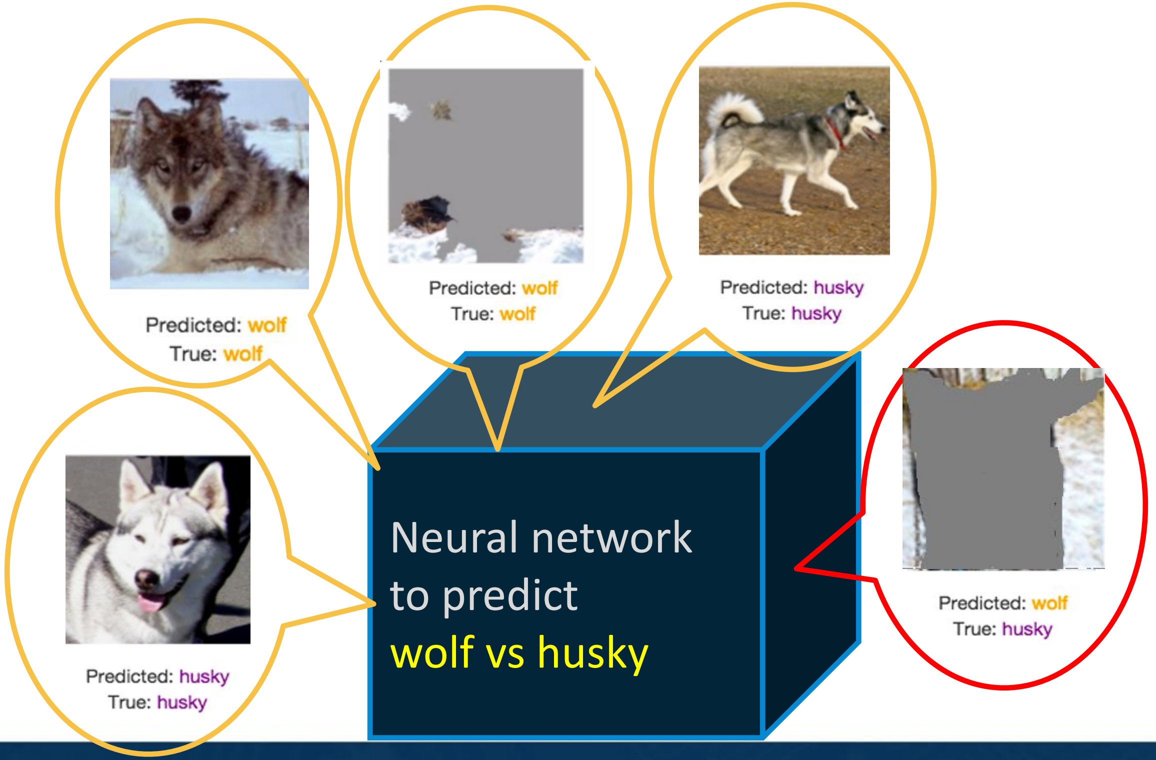
Data scientist

["Why Should I Trust You?": Explaining the Predictions of Any Classifier](#)
Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin
ACM SIGKDD, 2016

Interpretability Enables Trust in AI Models

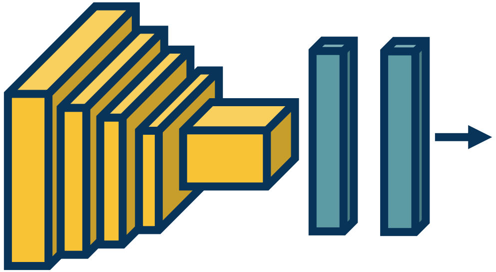
Figure out when NOT to trust a model

Data scientist

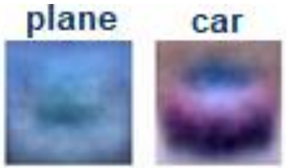


"Why Should I Trust You?": Explaining the Predictions of Any Classifier
Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin
ACM SIGKDD, 2016

Given a **trained** model, we'd like to understand what it learned.



Weights

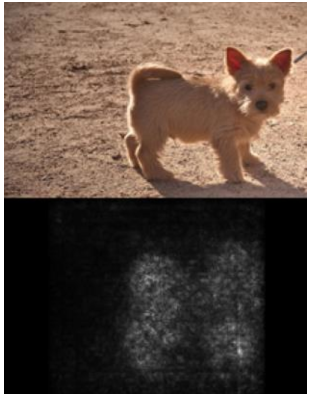


Fei-Fei Li, Justin Johnson, Serena Yeung, from CS 231n

Zeiler & Fergus, 2014

Activations

Gradients



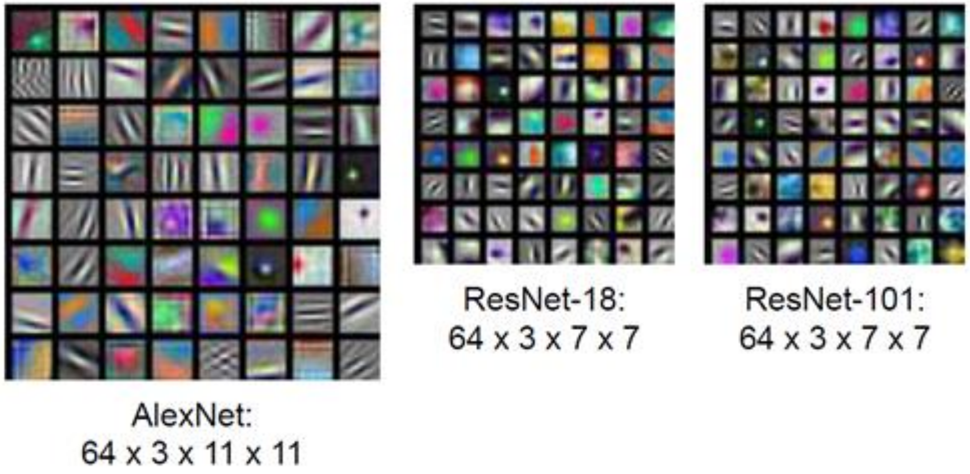
Simonyan et al, 2013

Robustness

Hendrycks & Dietterich, 2019

FC Layer: Reshape weights for a node back into size of image, scale 0-255

Conv layers:
For each kernel, scale values from 0-255 and visualize

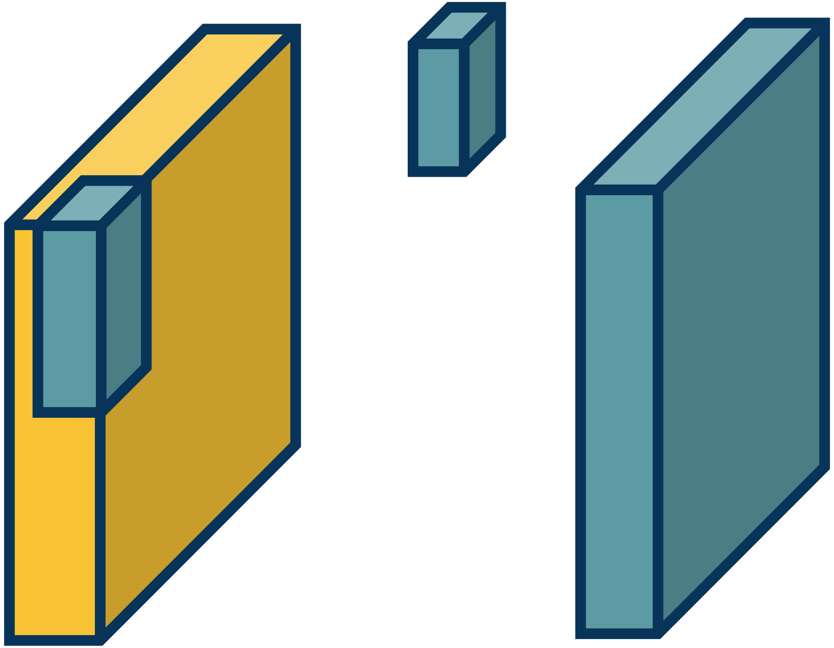


Problem: 3x3 filters difficult to interpret!

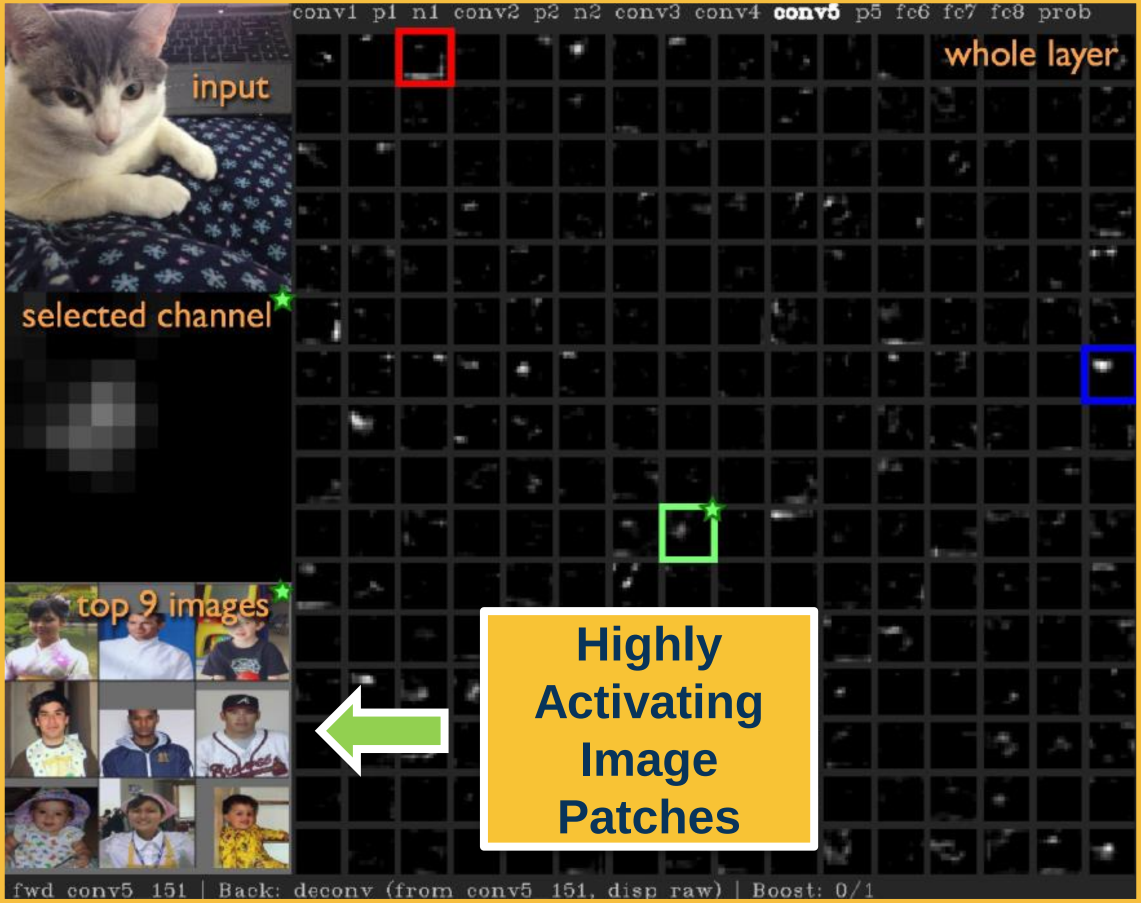
Adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, from CS 231n

We can also produce **visualization output (aka activation/filter) maps**

These are **larger** early in the network.



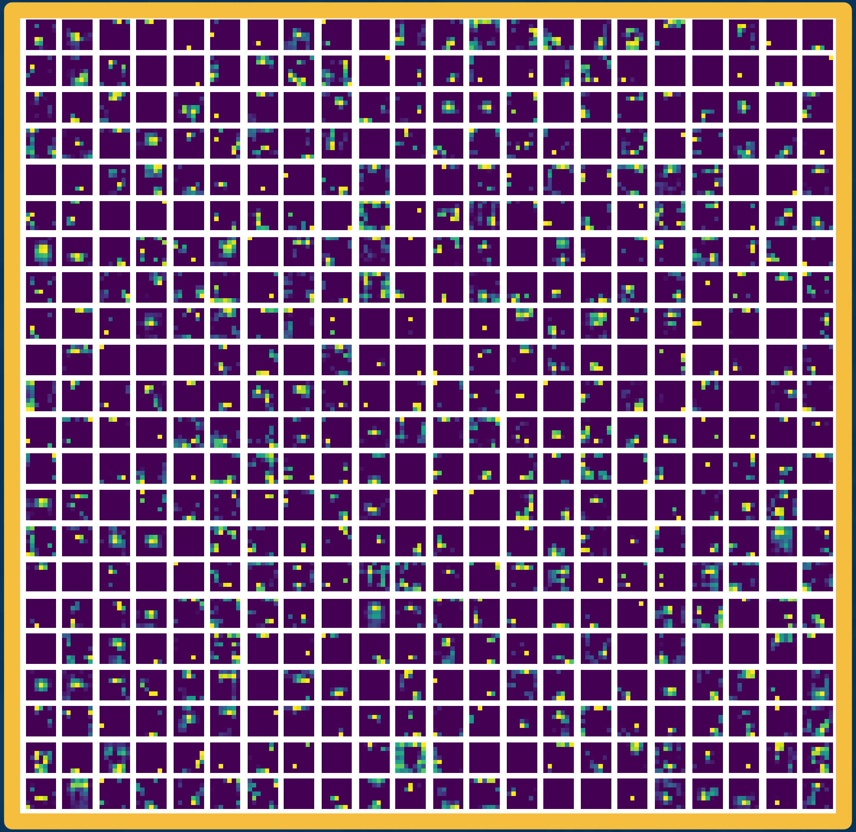
Visualizing Output Maps



Highly Activating Image Patches

From: Yosinski et al., "Understanding Neural Networks Through Deep Visualization", 2015

Activations – Small Output Sizes

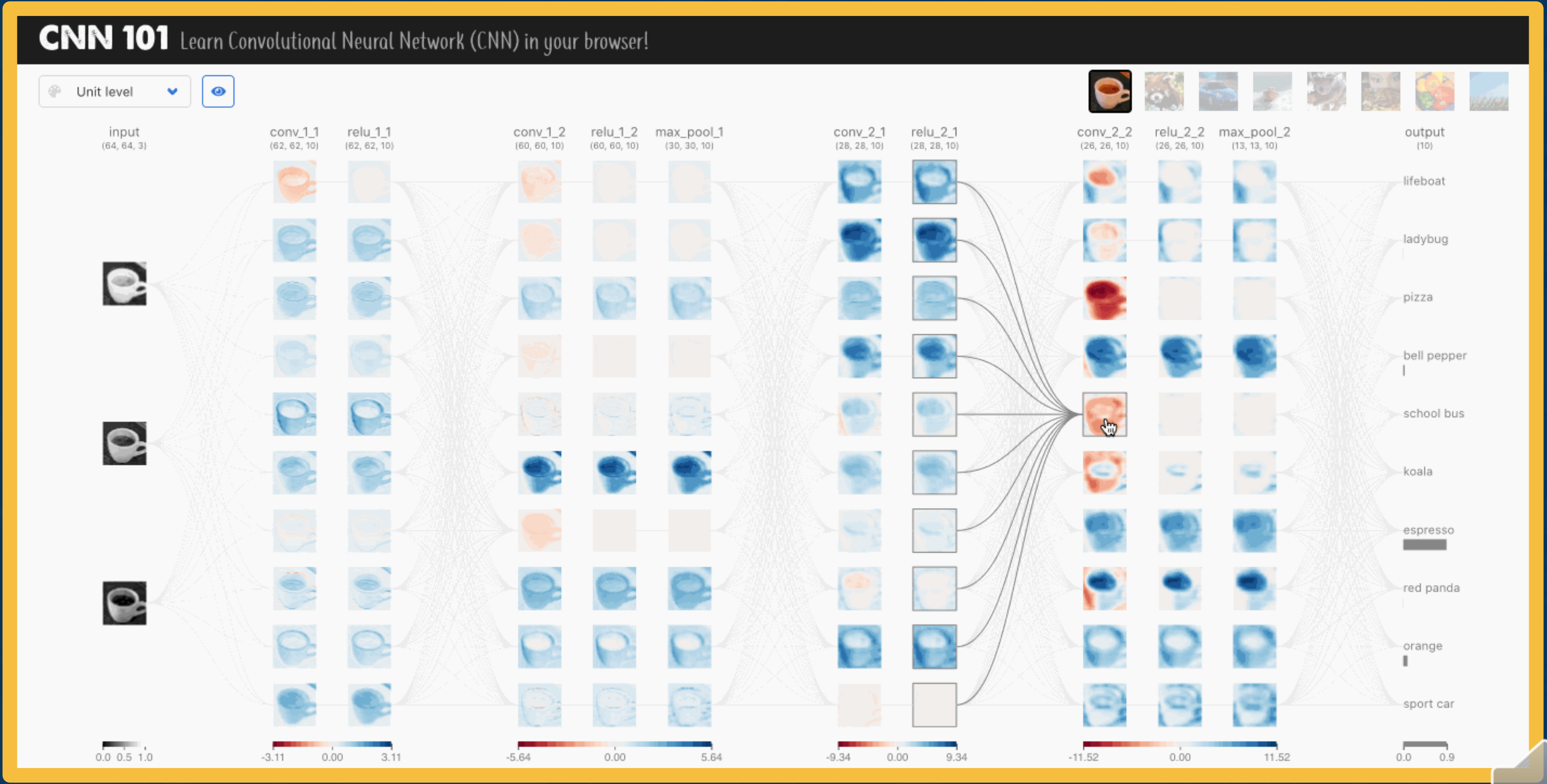


Activations of last conv layer in VGG network

Problem: Small conv outputs also hard to interpret

Georgia Tech

CNN101 and CNN Explainer



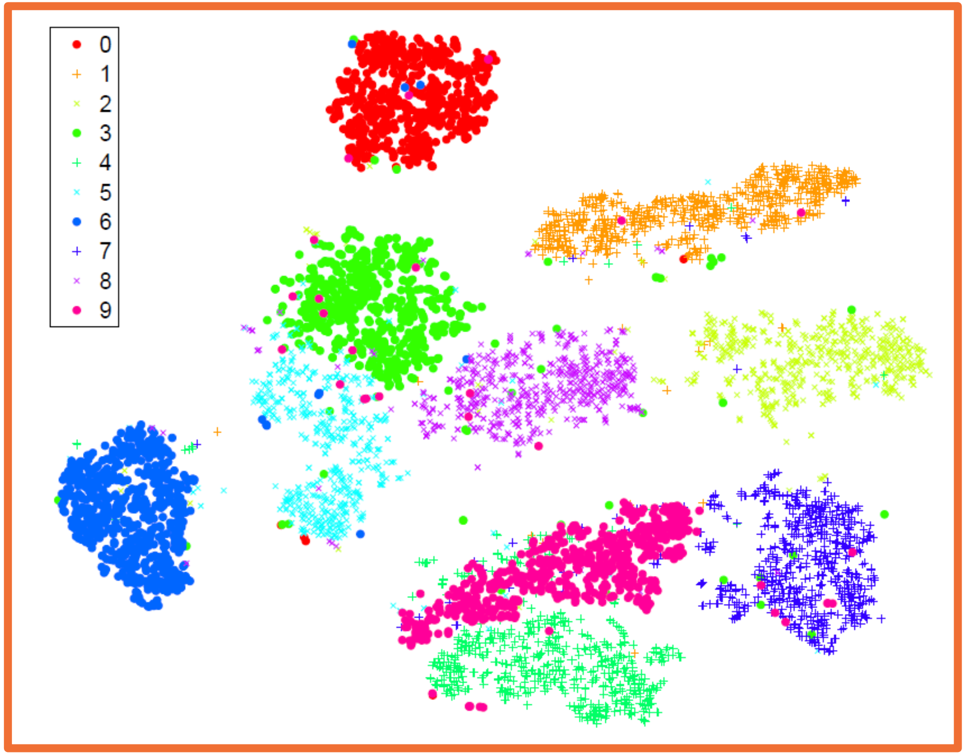
https://poloclub.github.io/cnn-explainer/ *https://fredhohman.com/papers/cnn101*

We can take the activations of any layer (FC, conv, etc.) and **perform dimensionality reduction**

- Often reduce to two dimensions for plotting
- E.g. using Principle Component Analysis (PCA)

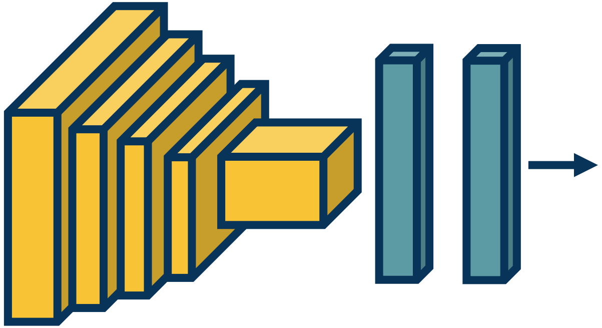
t-SNE is most common

- Performs non-linear mapping to preserve pair-wise distances

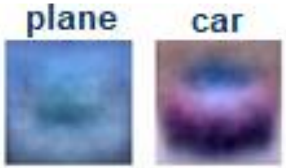


Van der Maaten & Hinton, "Visualizing Data using t-SNE", 2008.

Dimensionality Reduction: t-SNE



Weights

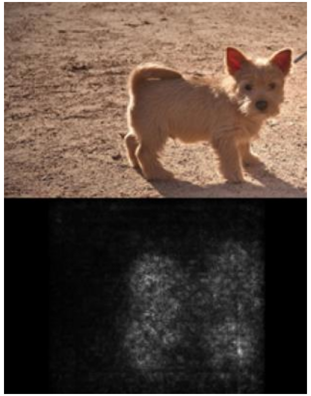


Fei-Fei Li, Justin Johnson, Serena Yeung, from CS 231n

Zeiler & Fergus, 2014

Activations

Gradients



Simonyan et al, 2013

Robustness

Hendrycks & Dietterich, 2019

Visualizing Neural Networks

Summary & Caveats

While these methods provide **some** visually interpretable representations, they can be misleading or uninformative (Adebayo et al., 2018)

Assessing interpretability is difficult

- Requires **user studies** to show **usefulness**
- E.g. they allow a user to predict mistakes beforehand

Neural networks learn **distributed representation**

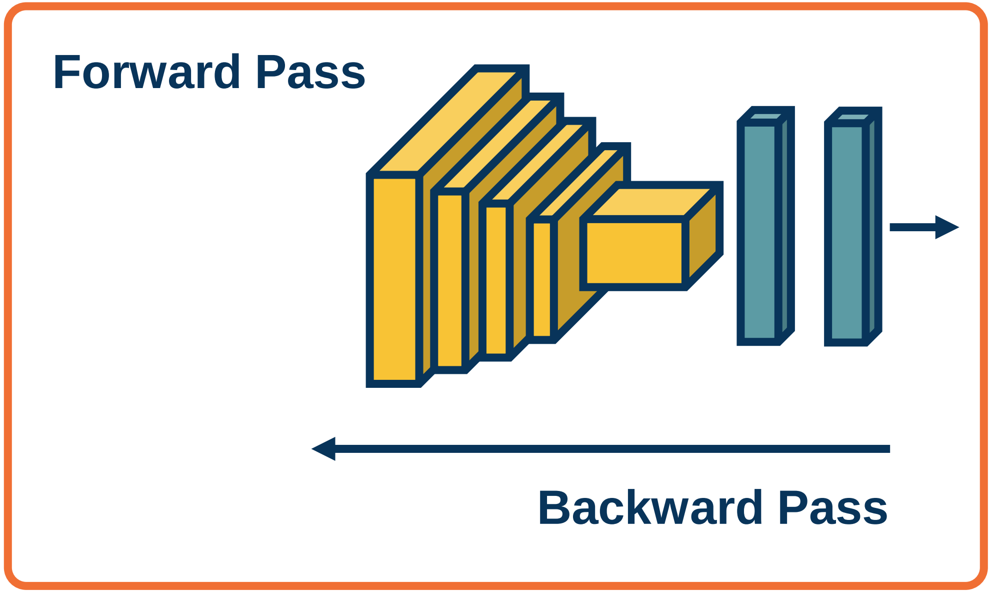
- (no one node represents a particular feature)
- This makes interpretation difficult

Adebayo et al., "Sanity Checks for Saliency Maps", 2018.

Georgia Tech

Gradient-Based Visualizations

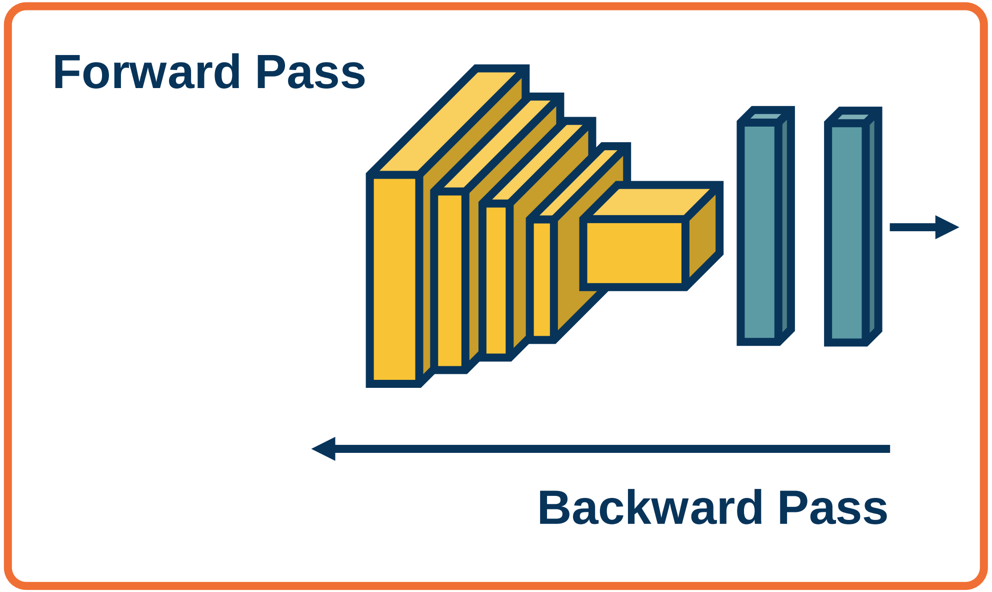
Given a **trained** model, we can perform forward pass given an input to get scores, softmax probabilities, loss and then backwards pass to get gradients



- Note: We are keeping parameters/weights **frozen**
  - Do not use gradients w.r.t. weights to perform updates

Backwards pass gives us **gradients** for all layers: How the loss changes as we change different parts of the input

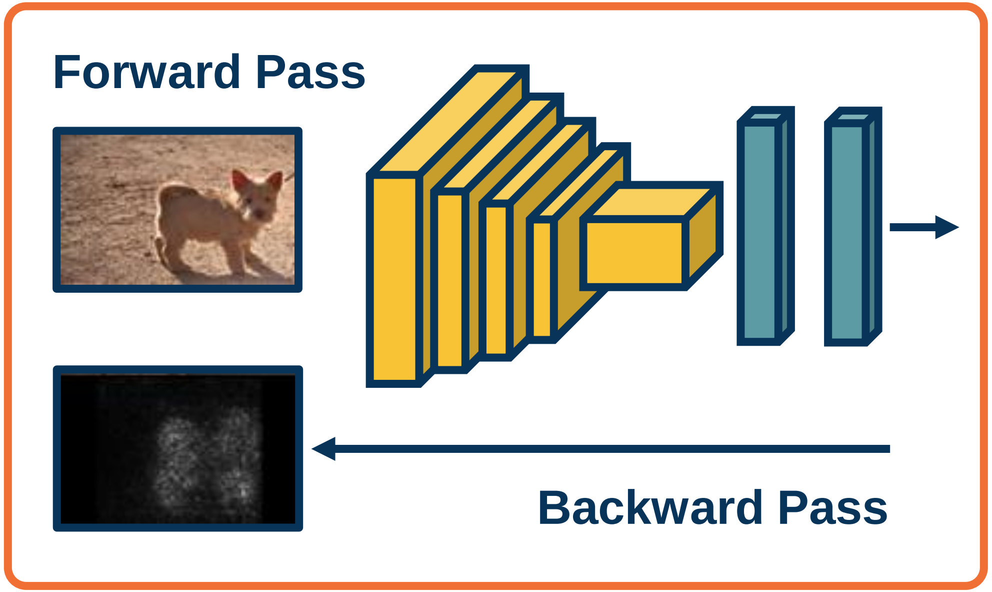
This can be **useful not just for optimization**, but also to understand what was learned



- Gradient of **loss** with respect to **all layers** (including input!)
- Gradient of **any layer** with respect to **input** (by cutting off computation graph)

Idea: We can backprop to the image

- Sensitivity of loss to individual pixel changes
- Large sensitivity implies important pixels
- Called **Saliency Maps**



In practice:

- Instead of loss, find gradient of classifier **scores** (pre-softmax)
- Take absolute value of gradient
- Sum across all channels

From: Simonyan et al., "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", 2013

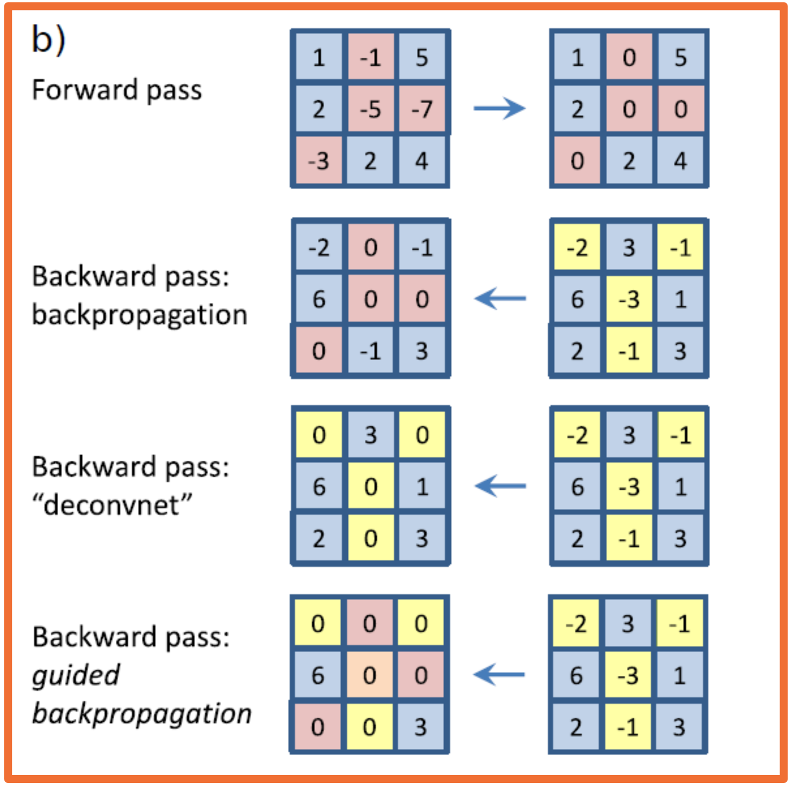
Gradient of Loss w.r.t. Image

Normal backprop not always best choice

Example: You may get parts of image that **decrease** the feature activation

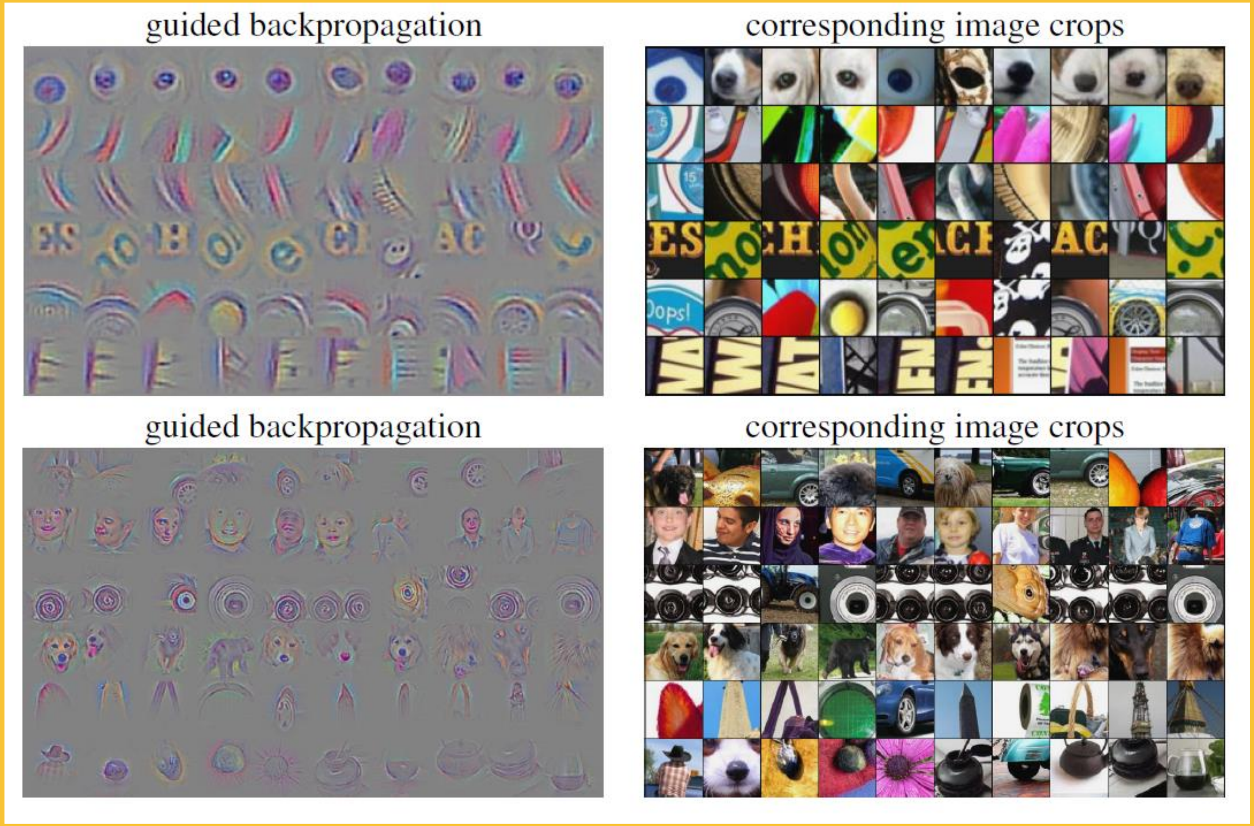
- There are probably lots of such input pixels

Guided backprop can be used to improve visualizations



From: Springenberg et al., "Striving For Simplicity: The All Convolutional Net"

Guided Backprop Results



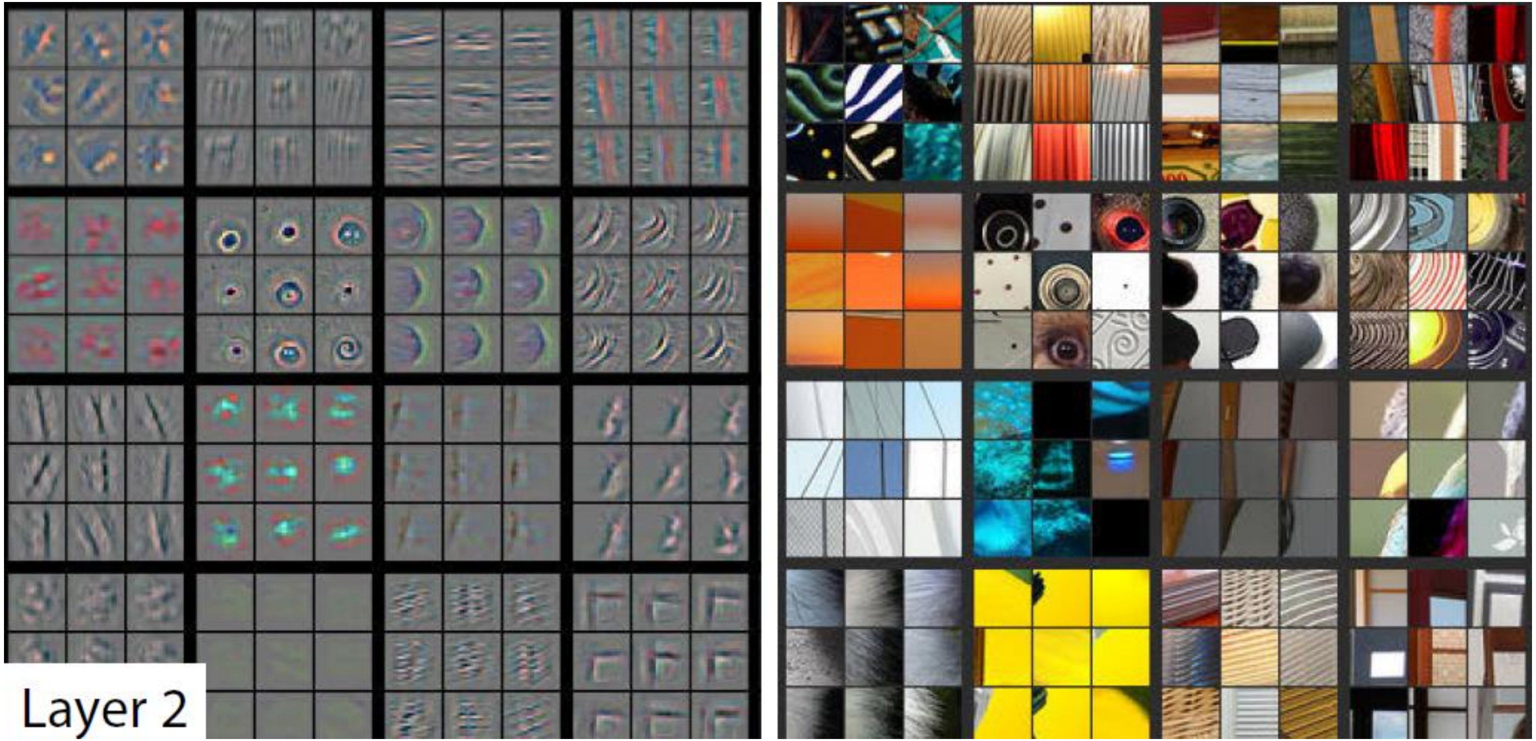
From: Springenberg et al., "Striving For Simplicity: The All Convolutional Net"

VGG Layer-by-Layer Visualization

Note: These images were created by a slightly different method called **deconvolution,** which ends up being similar to guided backprop

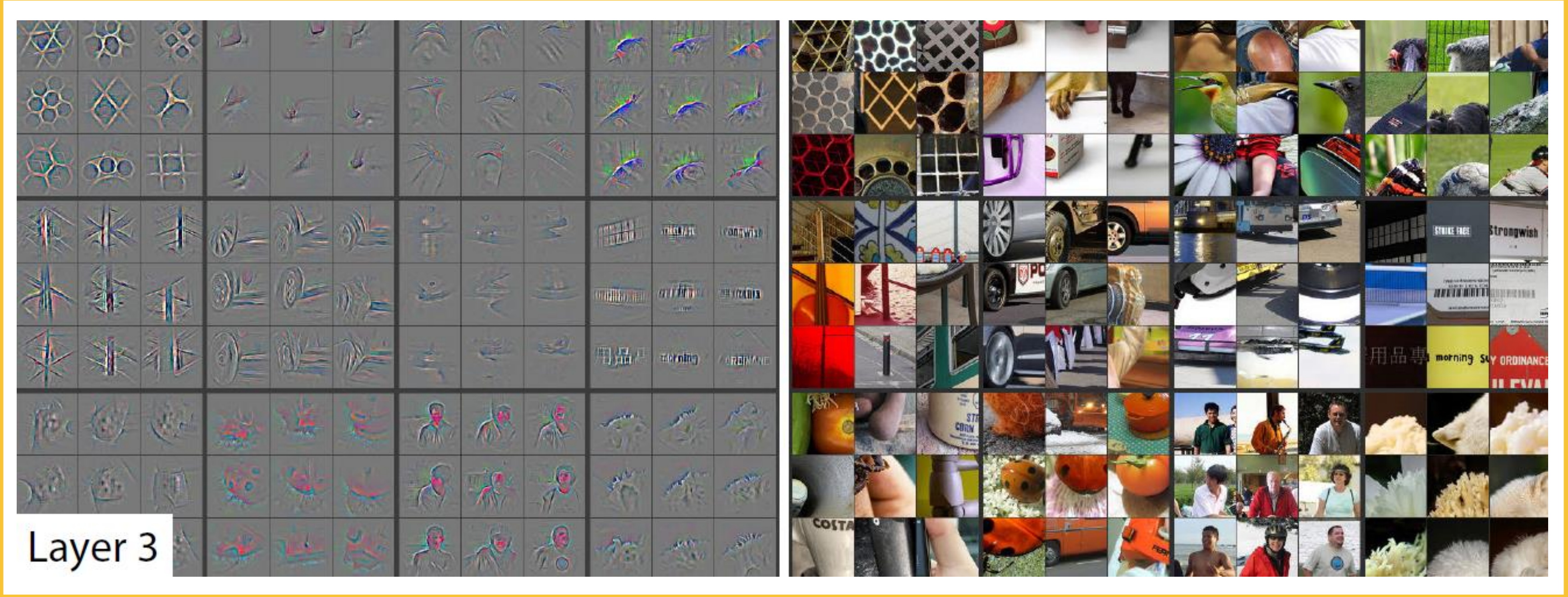
From: "Visualizing and Understanding Convolutional Networks, Zeiler & Fergus, 2014.

VGG Layer-by-Layer Visualization



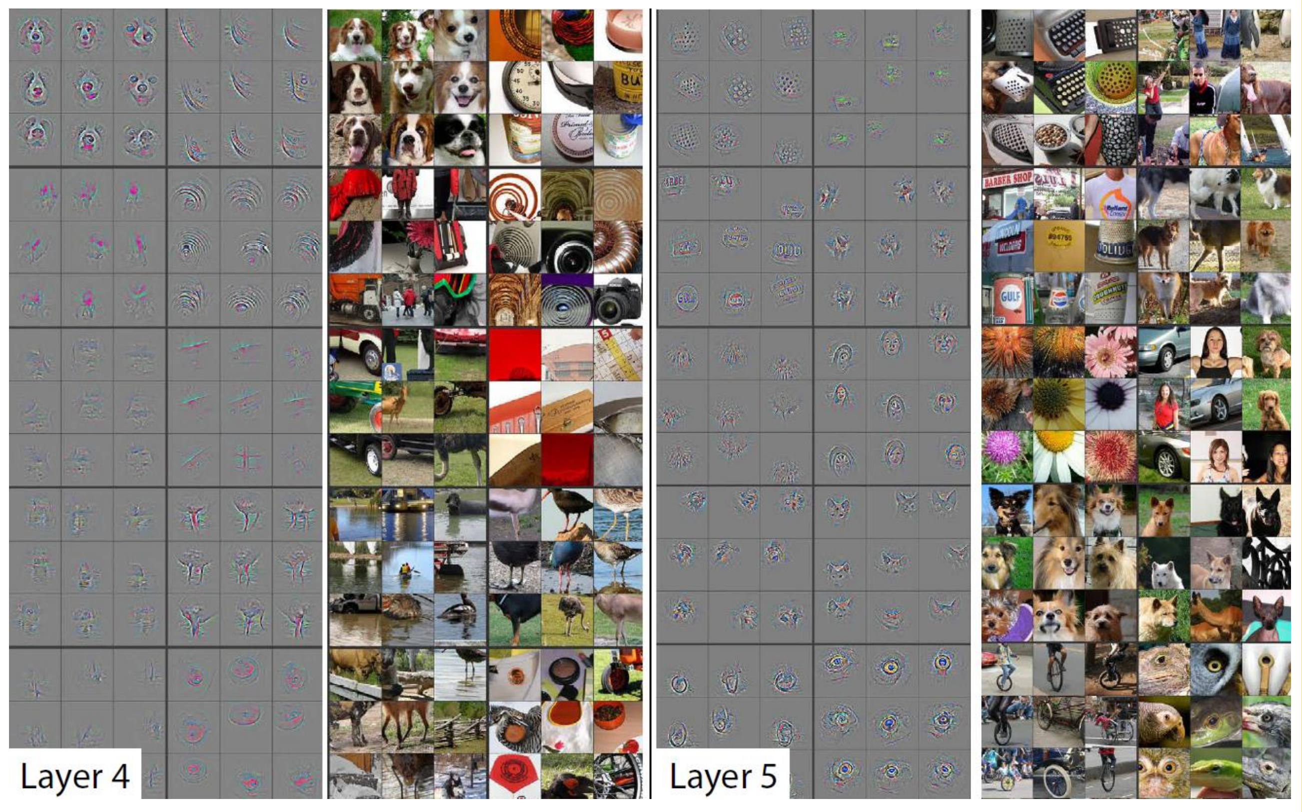
From: "Visualizing and Understanding Convolutional Networks, Zeiler & Fergus, 2014.

VGG Layer-by-Layer Visualization

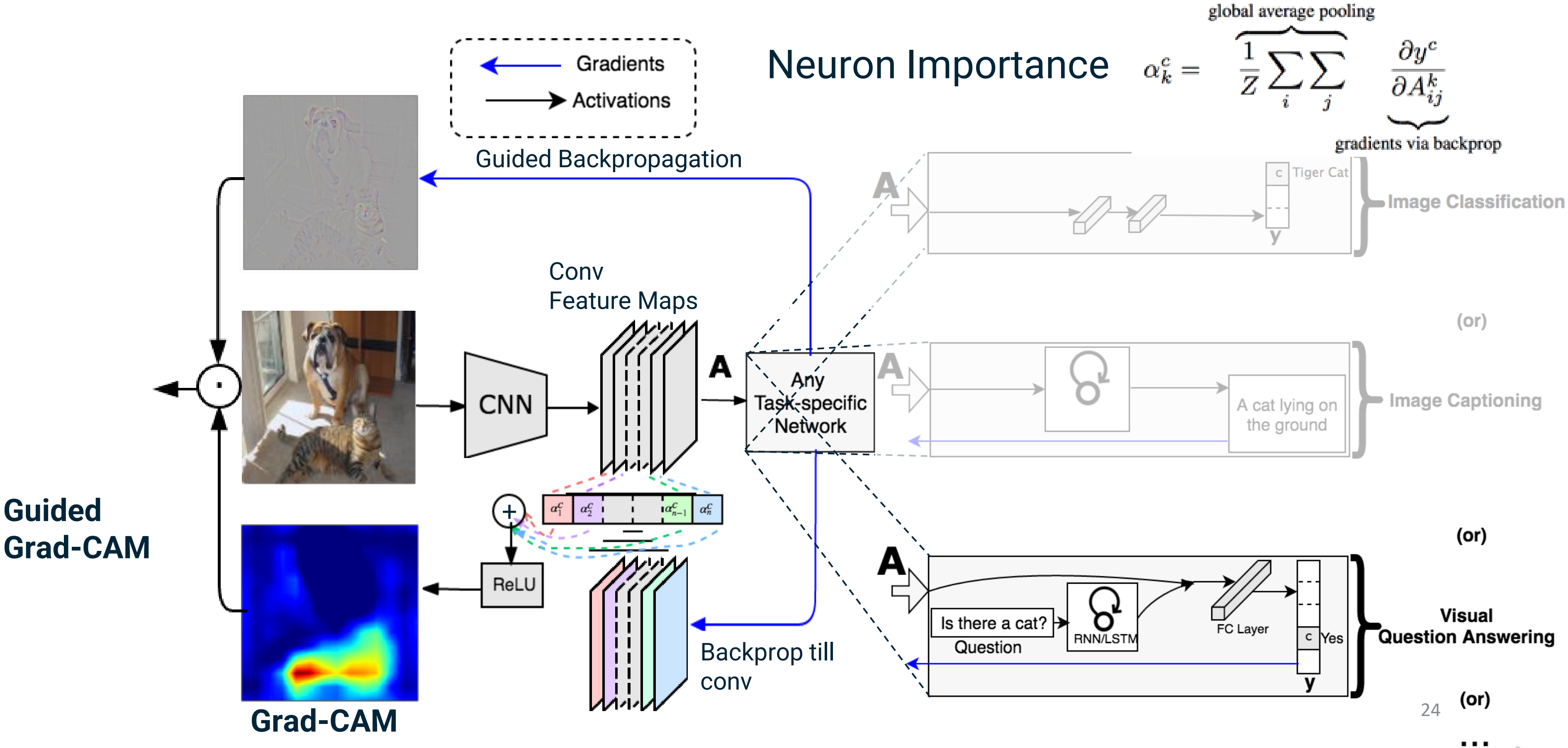


From: "Visualizing and Understanding Convolutional Networks, Zeiler & Fergus, 2014.

VGG Layer-by-Layer Visualization



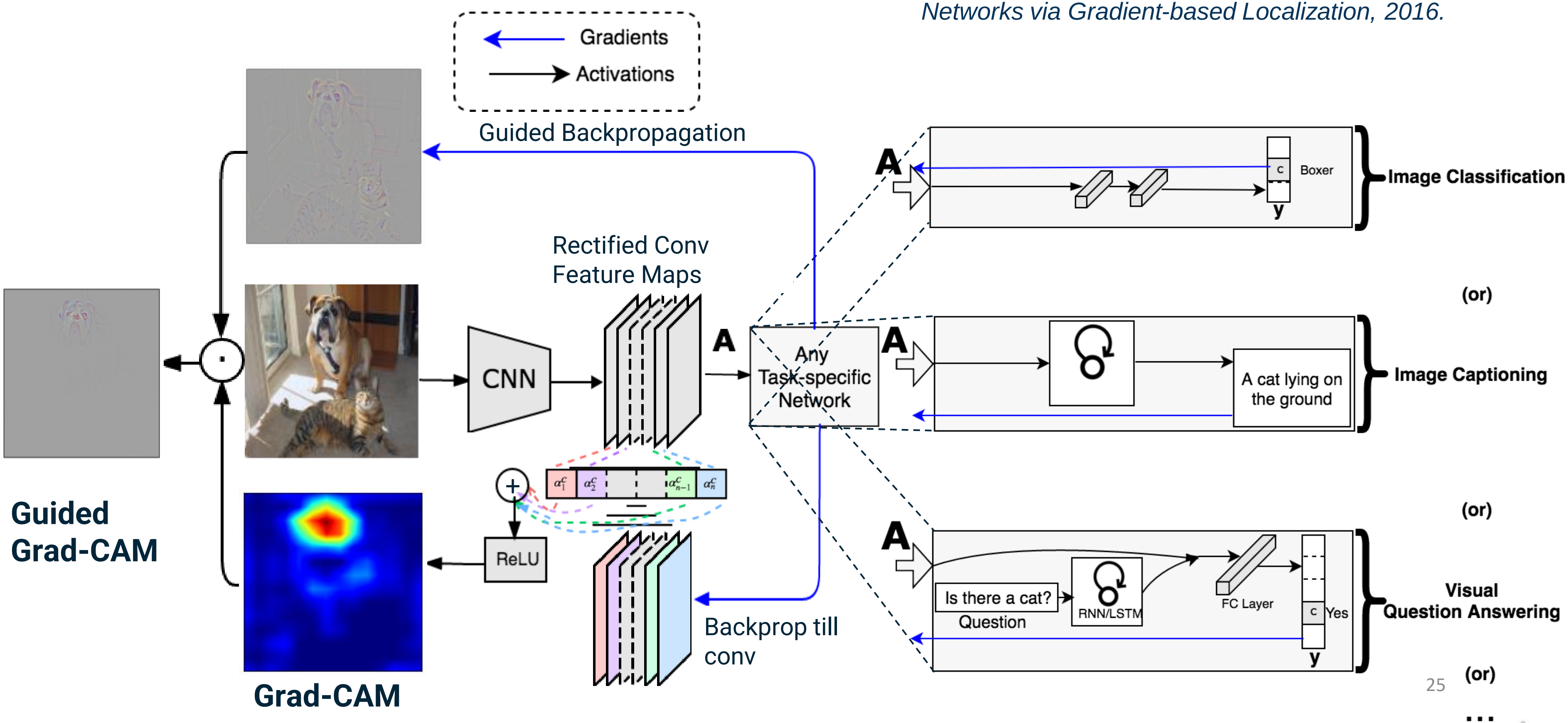
From: "Visualizing and Understanding Convolutional Networks, Zeiler & Fergus, 2014.



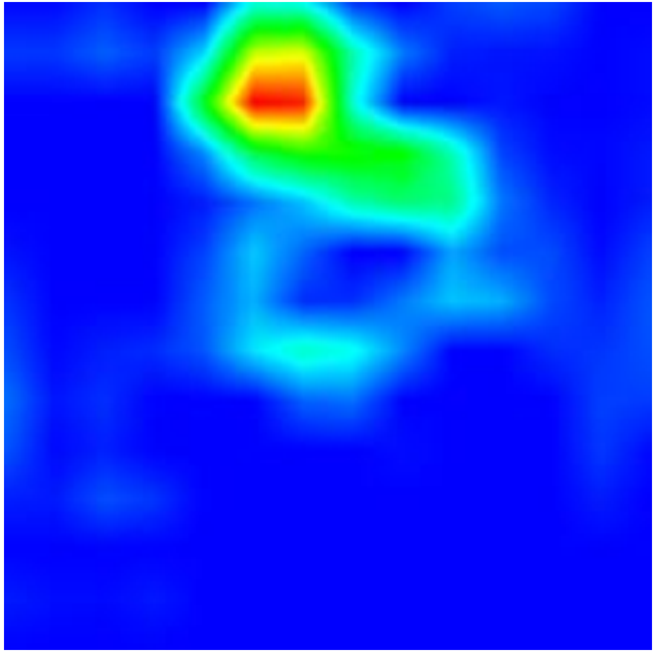
Selfvaraju et al., Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization, 2016.

GradCAM

Selfvaraju et al., *Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization, 2016.*



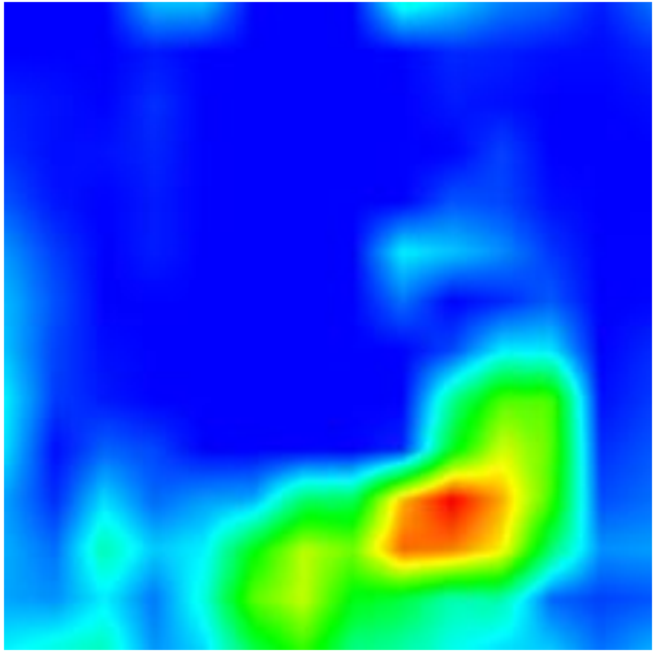
Grad-CAM



What animal is in this picture? Dog

Selfvaraju et al., Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization, 2016.

Grad-CAM



What animal is in this picture? Cat

Selfvaraju et al., Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization, 2016.

Summary

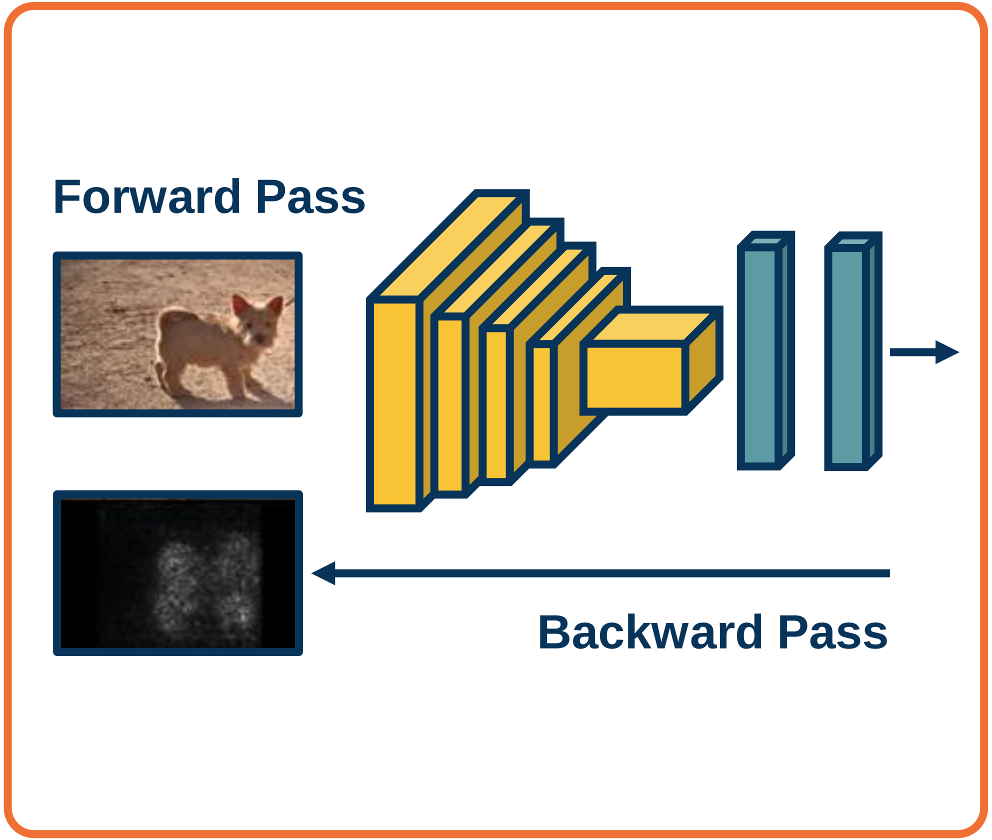
- Gradients are important **not just for optimization**, but also for **analyzing** what neural networks **have learned**
- Standard backprop **not always the most informative** for visualization purposes
- Several ways to **modify the gradient flow** to improve visualization results

Optimizing the Input Images

Idea: Since we have the gradient of scores w.r.t. inputs, can we *optimize* the image itself to maximize the score?

Why?

- Generate images from scratch!
- Adversarial examples



From: Simonyan et al., "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", 2013

Gradient Ascent on the Scores

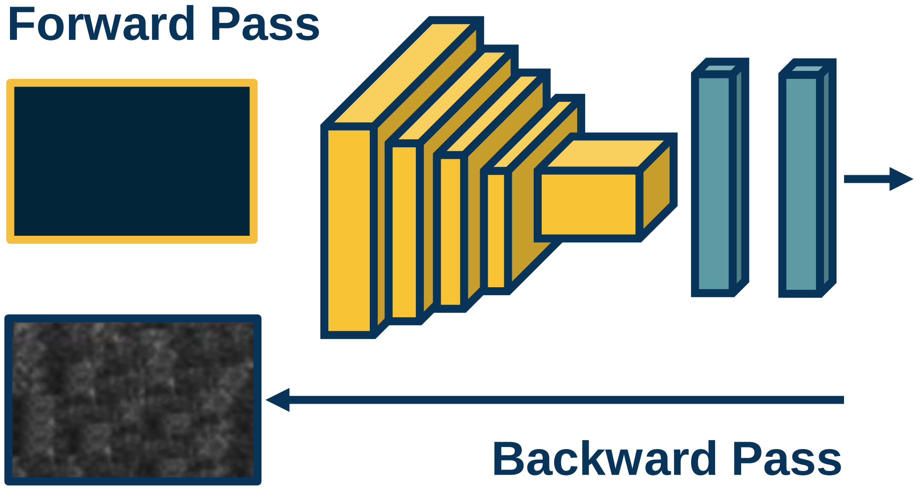
We can perform **gradient ascent** on image

- Start from random/zero image
- Use scores to avoid minimizing other class scores instead

Often need **regularization term** to induce statistics of natural imagery

- E.g. small pixel values, spatial smoothness

$$\arg\max S_c(I) - \lambda \, ||I||_2^2$$



$$I = I + \alpha \frac{\partial S_c}{\partial I}$$

From: Simonyan et al., "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", 2013

Example Images

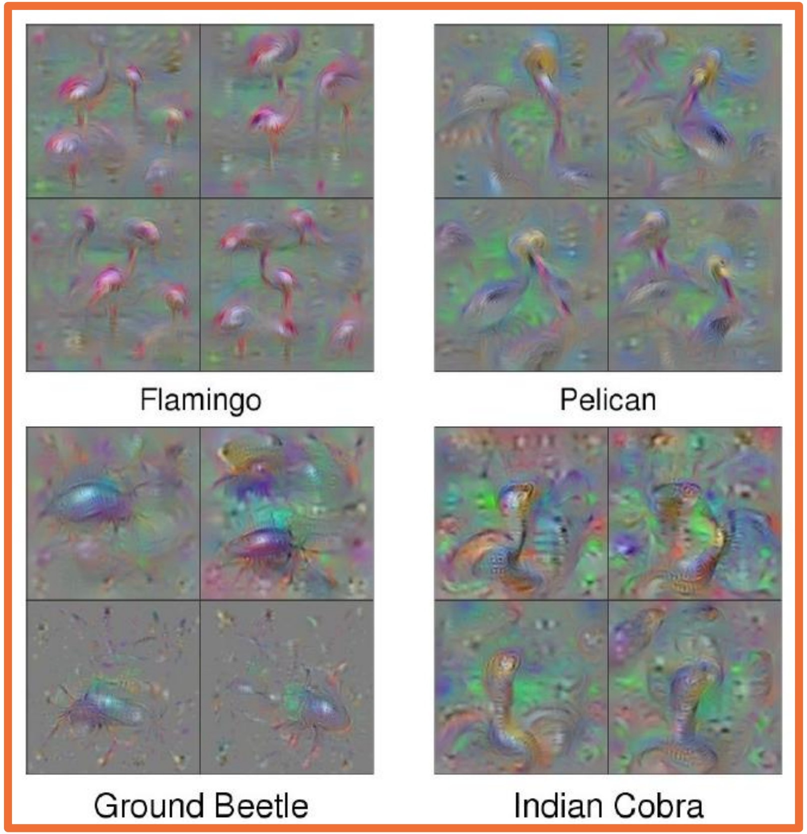
dumbbell cup dalmatian

Note: You might have to squint!

From: Simonyan et al., "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", 2013

Can improve results with **various tricks**:

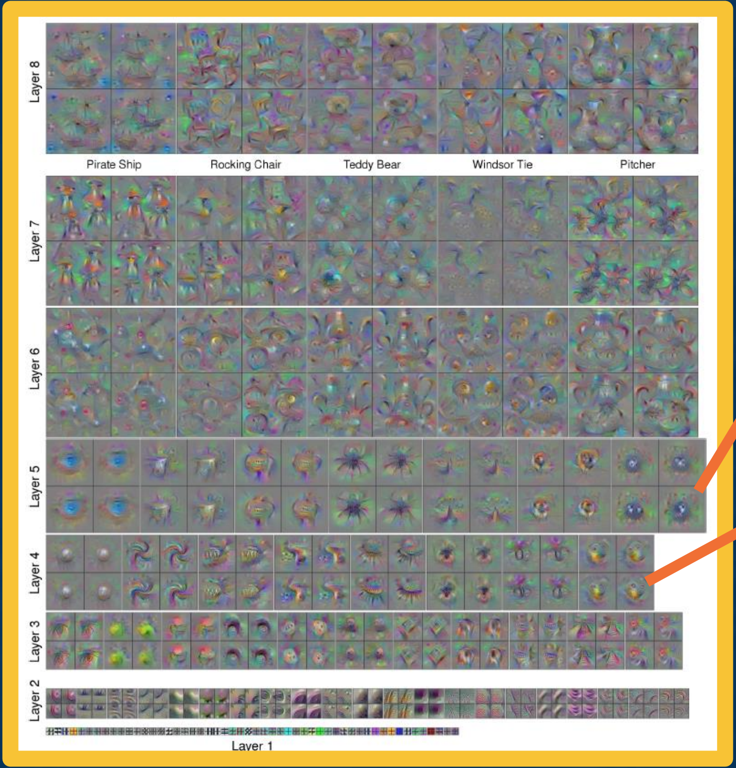
- Clipping or normalization of small values & gradients
- Gaussian blurring



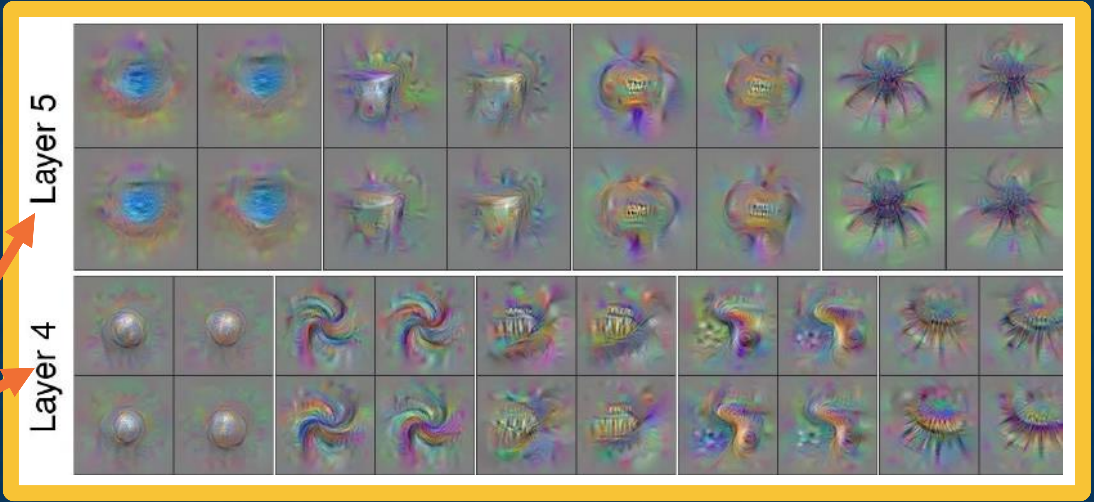
From: Yosinski et al., "Understanding Neural Networks Through Deep Visualization", 2015

Example Images

Improved Results



Note: Can generate input images to maximize any arbitrary activation!



From: Yosinski et al., "Understanding Neural Networks Through Deep Visualization", 2015

Summary

We can optimize the input image to **generate** examples to increase class scores or activations

This can show us a great deal about what examples (not in the training set) **activate the network**

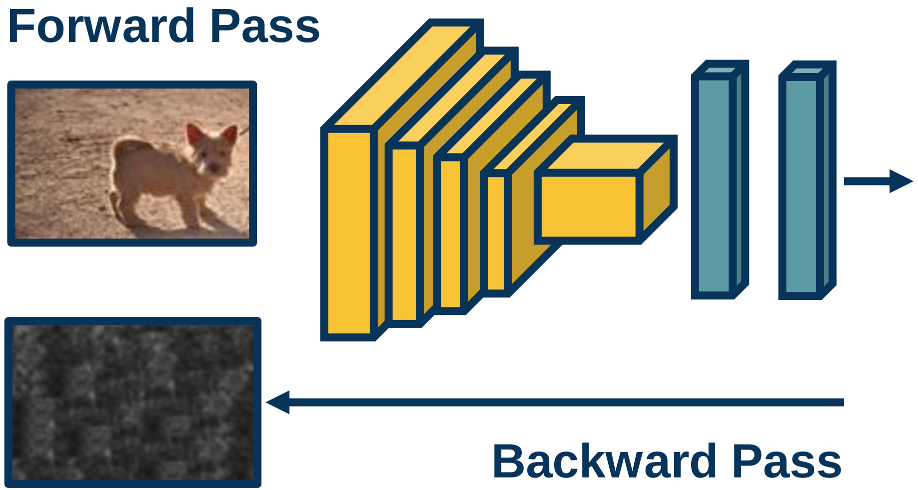
Testing Robustness

- We can perform **gradient ascent** on image
- Rather than start from zero image, why not real image?
- And why not optimize the score of an **arbitrary** (incorrect!) class

Surprising result: You need very small amount of pixel changes to make the network confidently wrong!

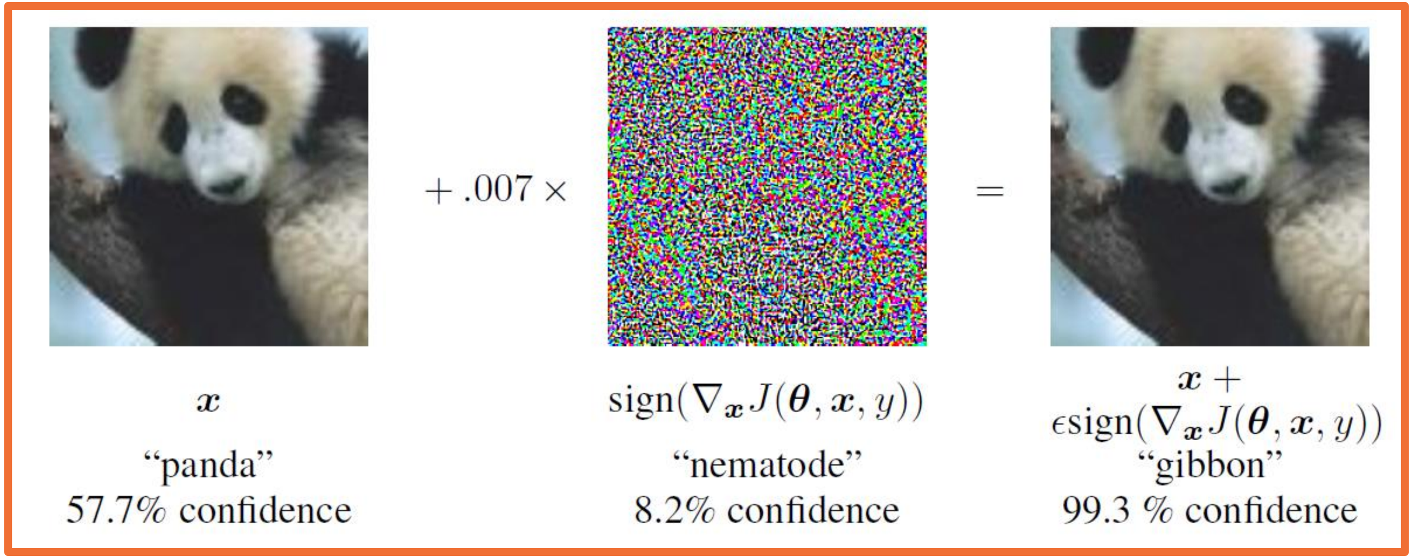
$$\arg\max S_c(I) - \lambda \left|\left|I\right|\right|_2^2$$

where $c = cat$



From: Simonyan et al., "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", 2013

Gradient Ascent on the Scores



Note this problem is not specific to deep learning!

- Other methods also suffer from it
- Can show how **linearity** (even at the end) can bring this about
  - Can add many small values that add up in right direction

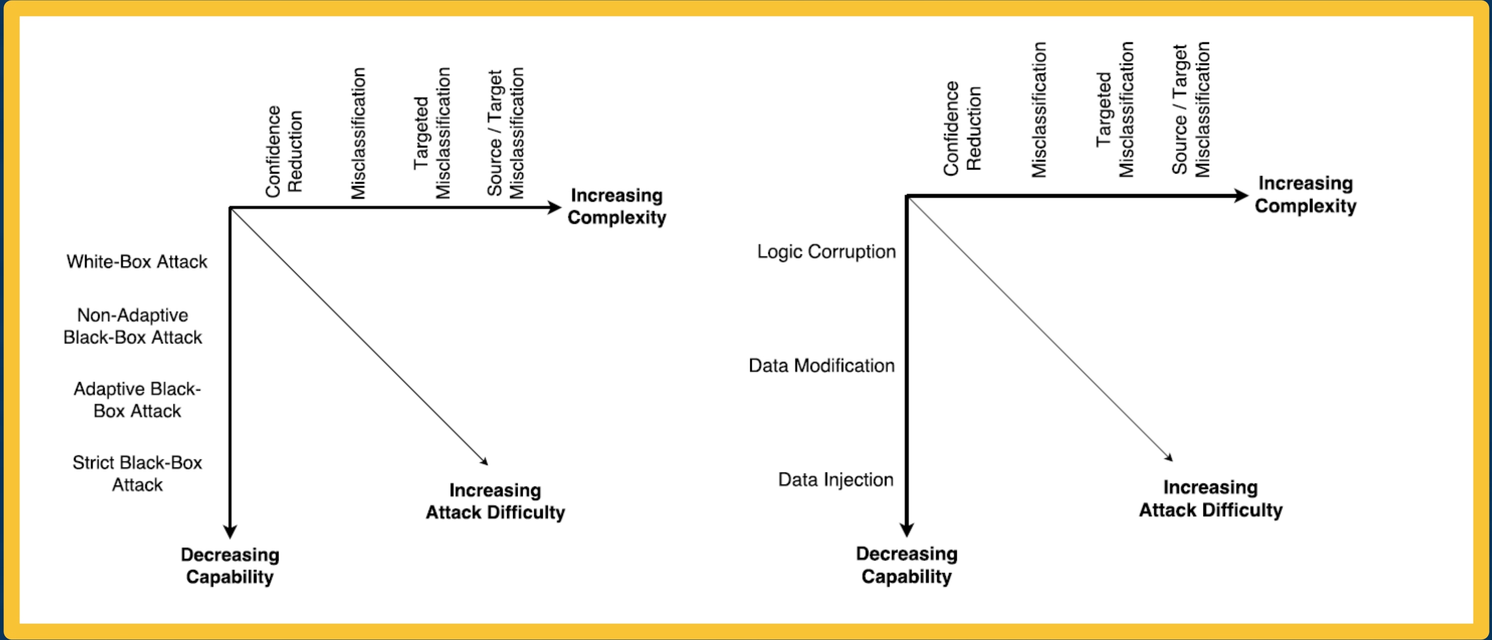
From: Goodfellow et al., “Explaining and Harnessing Adversarial Examples”, 2015

Example of Adversarial Noise

Variations of Attacks

Single-Pixel Attacks!

Su et al., “One Pixel Attack for Fooling Deep Neural Networks”, 2019.



White vs. Black-Box Attacks of Increasing Complexity

Chakraborty et al., Adversarial Attacks and Defences: A Survey, 2018

Summary of dversarial Attacks/Defenses

Similar to other security-related areas, it’s an active **cat-and-mouse game**

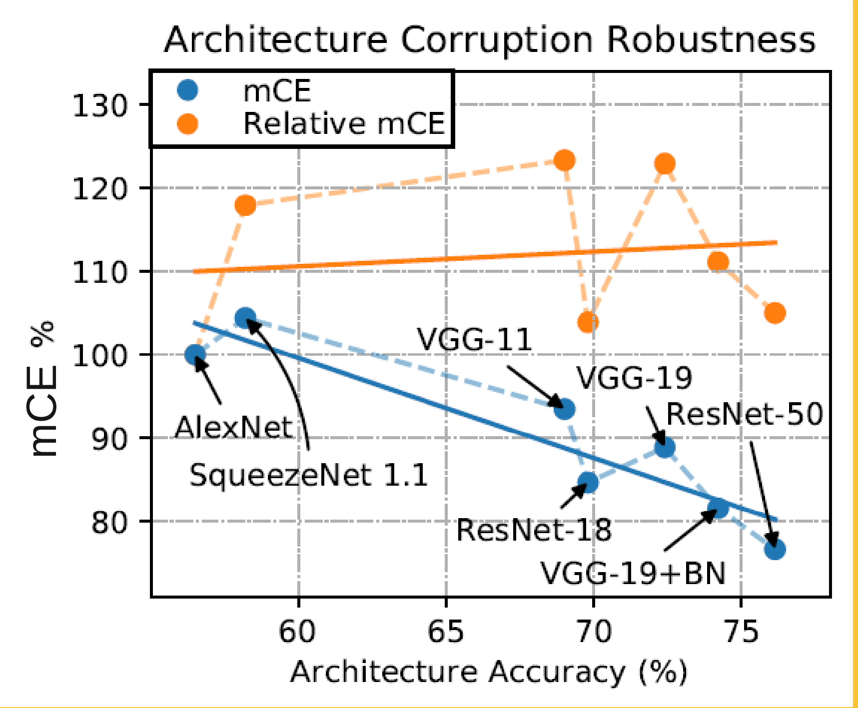
Several defenses such as:

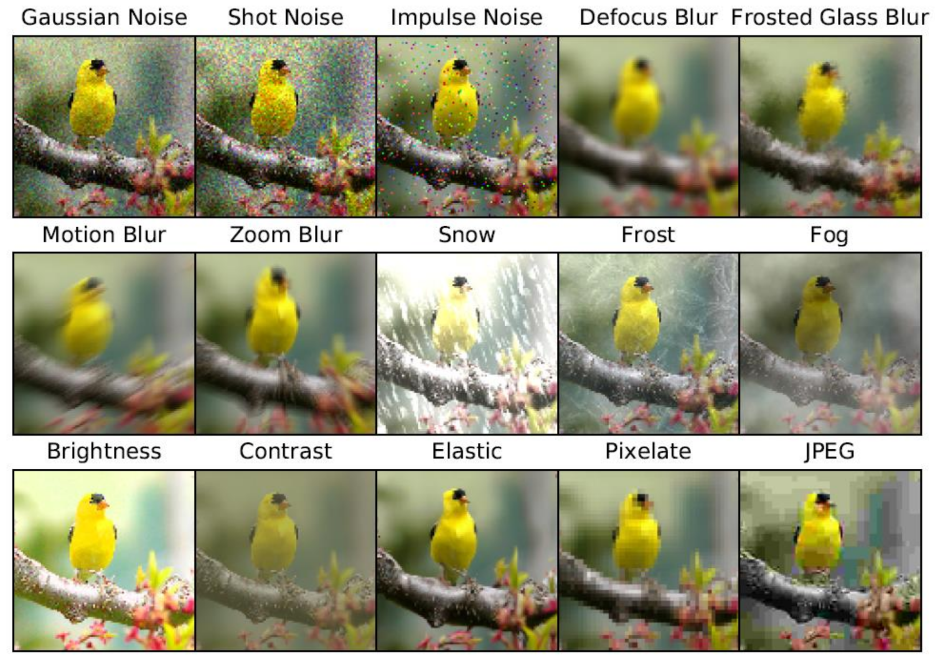
- Training with adversarial examples
- Perturbations, noise, or re-encoding of inputs

There are **not universal methods** that are robust to all types of attacks

Georgia Tech

Other Forms of Robustness Testing





$$\mathrm{CE}_c^f = \left(\sum_{s=1}^{5} E_{s,c}^f\right) \Big/ \left(\sum_{s=1}^{5} E_{s,c}^{\mathrm{AlexNet}}\right).$$

Hendrycks & Dietterich, "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations", 2019.

We can try to understand the **biases of CNNs**

- Can compare to those of humans

Example: Shape vs. Texture Bias

Geirhos, “ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness”, 2018.

(a) Texture image

81.4%	**Indian elephant**
10.3%	indri
8.2%	black swan

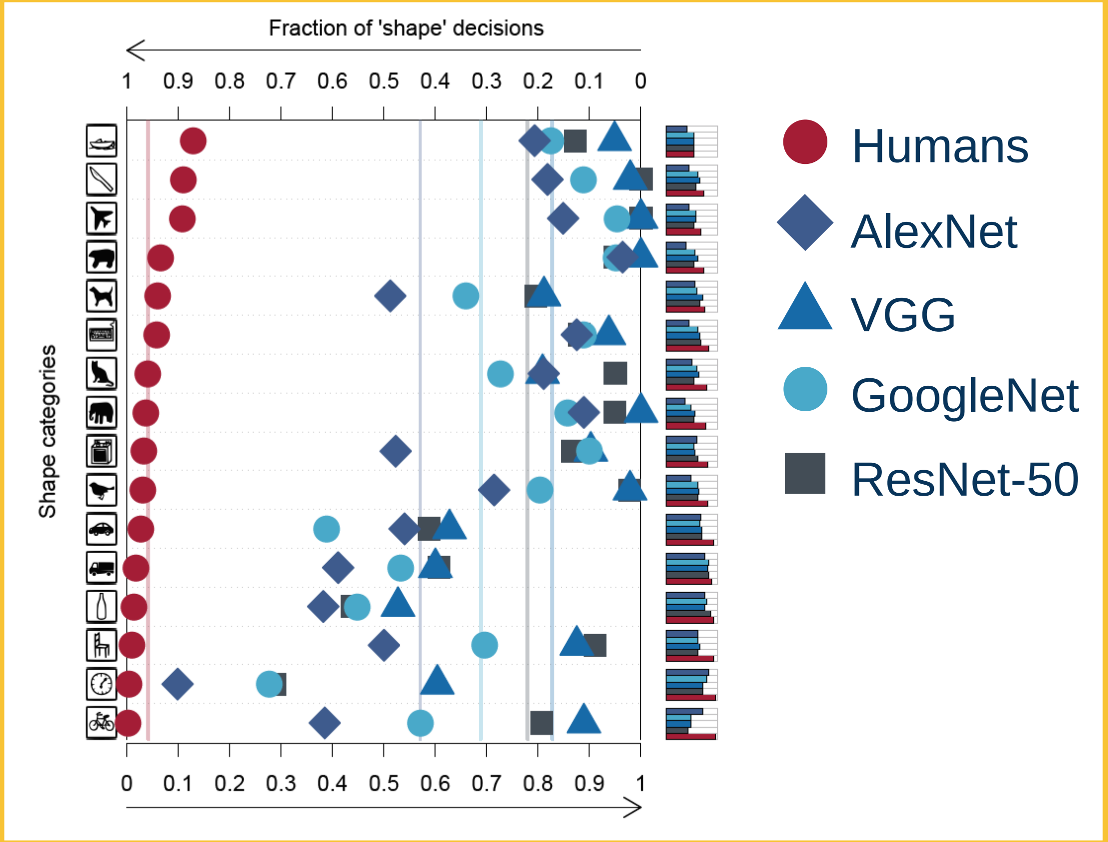
(b) Content image

71.1%	**tabby cat**
17.3%	grey fox
3.3%	Siamese cat

(c) Texture-shape cue conflict

63.9%	**Indian elephant**
26.4%	indri
9.6%	black swan

Shape vs. Texture Bias



Geirhos, "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness", 2018.

Summary

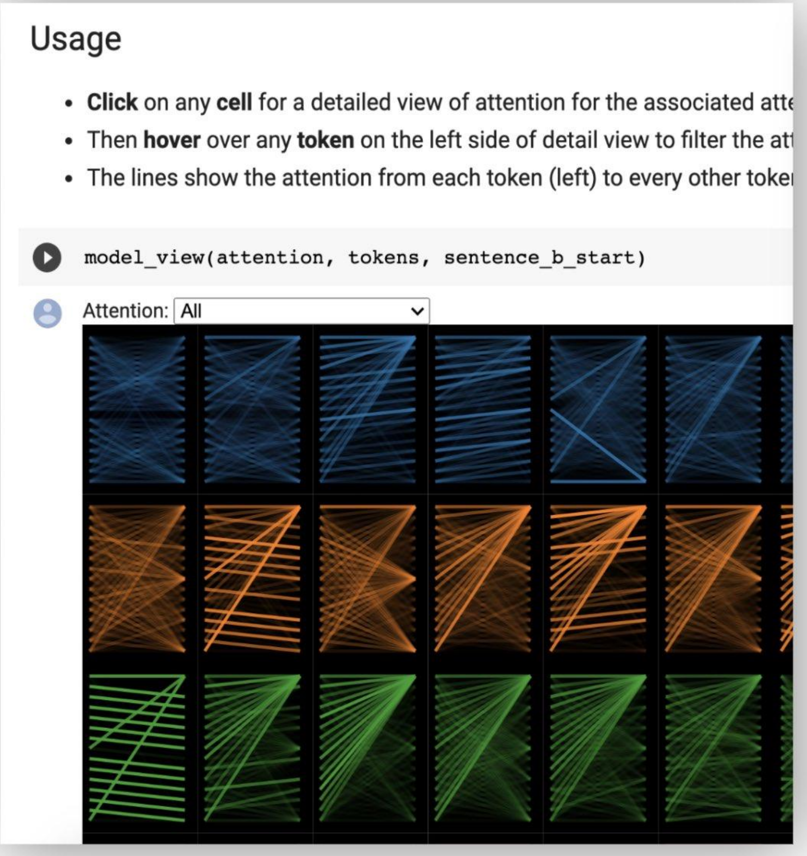
- Various ways to test the **robustness** and **biases** of neural networks
- Adversarial examples have **implications** for understanding and trusting them
- Exploring the **gain of different architectures** in terms of robustness and biases can also be used to understand what has been learned

Georgia Tech

What about Transformers, LLMs, etc.

- Large models, especially transformers, can be seen as implementing **algorithms**

- Several ways to try to understand:
  - Visualization – Often attention
  - Distill into more interpretable model
  - Reverse engineer
  - Forward engineer: Algorithm → compiler → Weights!



https://github.com/jessevig/bertviz

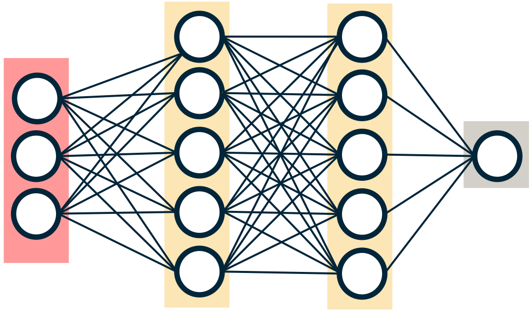
Visualization

Georgia Tech

Engineering Predictors for Interpretability

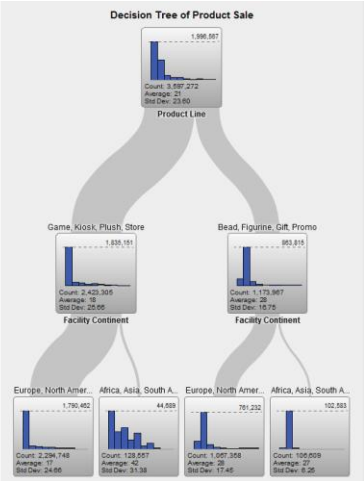
Using Shallow Decision Tree to Simulate Neural Network Prediction

Data (X, y)



Predict $\hat{y}$

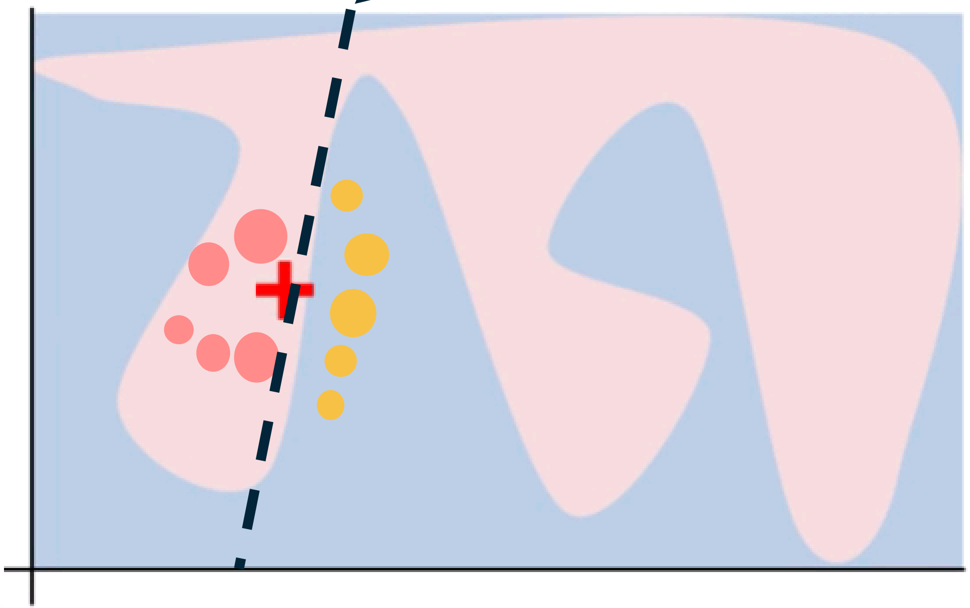
Train Tree on $(X, \hat{y})$



[Beyond Sparsity: Tree Regularization of Deep Models for Interpretability]
Mike Wu , Michael C. Hughes , Sonali Parbhoo , Maurizio Zazzi , Volker Roth , and Finale Doshi-Velez

Local Interpretable Model-agnostic Explanations (LIME)

Gives explanations for individuals predictions from a classifier



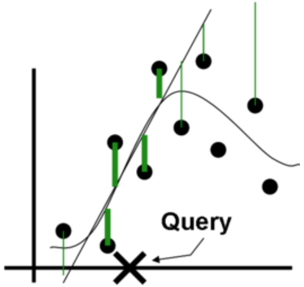
LIME builds an interpretable model of explanatory data samples at local areas in the analyzed data.

["Why Should I Trust You?": Explaining the Predictions of Any Classifier]
Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin.
ACM SIGKDD, 2016

Original Image

P(labrador) = 0.21

Perturbed Instances	P(Labrador)
	0.92
	0.001
	0.34



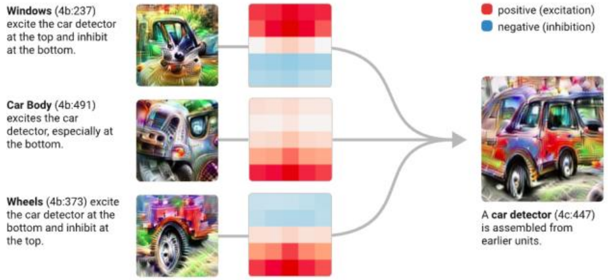
Locally weighted regression

Explanation

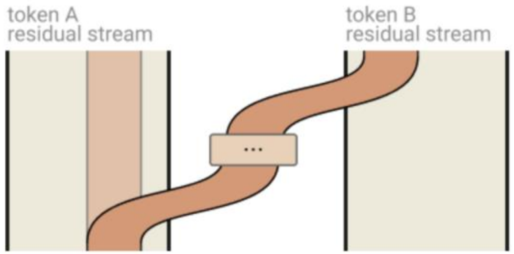
Image Source: https://drive.google.com/file/d/0BybIrZgHugfYZ0ZCSWNPWFNONEU/view

What is Mechanistic Interpretability?

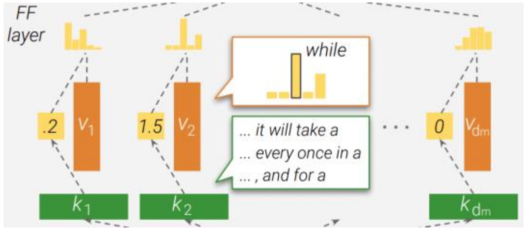
- **Goal:** Reverse engineer neural networks
  - Like reverse-engineering a compiled program binary to source code
- **Hypothesis**: Models learn human-comprehensible algorithms and can be understood, if we learn how to make it legible
- Understanding **features** - the variables inside the model
- Understanding **circuits** - the algorithms learned to compute features
- **Key property**: Distinguishes between cognition with identical output
- A deep knowledge of circuits is crucial to understand, predict and align model behaviour



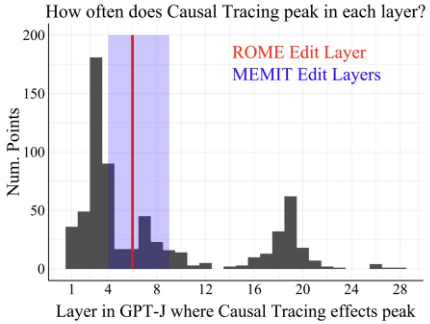
Slide by Neel Nanda



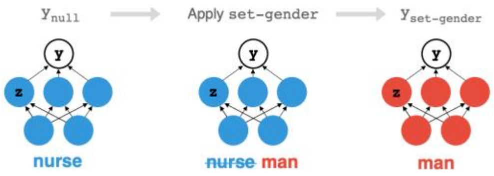
[A Mathematical Framework for Transformer Circuits (Elhage et al, Anthropic 2021)]



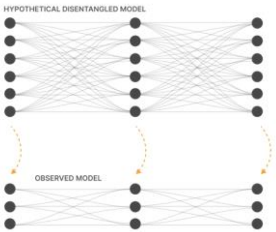
[Transformer Feed-Forward Layers Are Key-Value Memories (Geva et al, EMNLP 2021)]



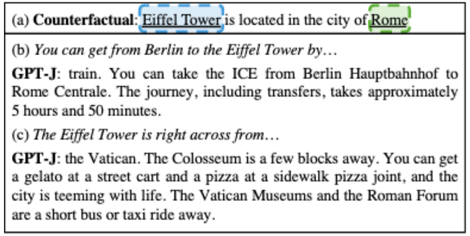
[Does Localization Inform Editing? (Hase et al, 2023)]



[Investigating Gender Bias in Language Models Using Causal Mediation Analysis (Vig et al, NeurIPS 2020)]



[Toy Models of Superposition (Elhage, Anthropic 2022)]


(a) **Counterfactual**: Eiffel Tower is located in the city of Rome

(b) *You can get from Berlin to the Eiffel Tower by…*

GPT-J: train. You can take the ICE from Berlin Hauptbahnhof to Rome Centrale. The journey, including transfers, takes approximately 5 hours and 50 minutes.

(c) *The Eiffel Tower is right across from…*

GPT-J: the Vatican. The Colosseum is a few blocks away. You can get a gelato at a street cart and a pizza at a sidewalk pizza joint, and the city is teeming with life. The Vatican Museums and the Roman Forum are a short bus or taxi ride away.

[Locating and Editing Factual Associations in GPT (Meng et al, NeurIPS 2022)]

A Growing Area of Research

Slide by Neel Nanda

Circuits = Functions: How does the model think?

- **Zero layer transformers model bigram statistics.** The bigram table can be accessed directly from the weights.
- **One layer attention-only transformers are an ensemble of bigram and "skip-trigram"** (sequences of the form "A… B C") models. The bigram and skip-trigram tables can be accessed directly from the weights, without running the model.
- **Two layer attention-only transformers can implement much more complex algorithms using** compositions of attention heads. These compositional algorithms can also be detected directly from the weights. Notably, two layer models use attention head composition to create "induction heads", a very general in-context learning algorithm.

A Mathematical Framework (Elhage et al)

Open Problems: Analysing Toy Language Models

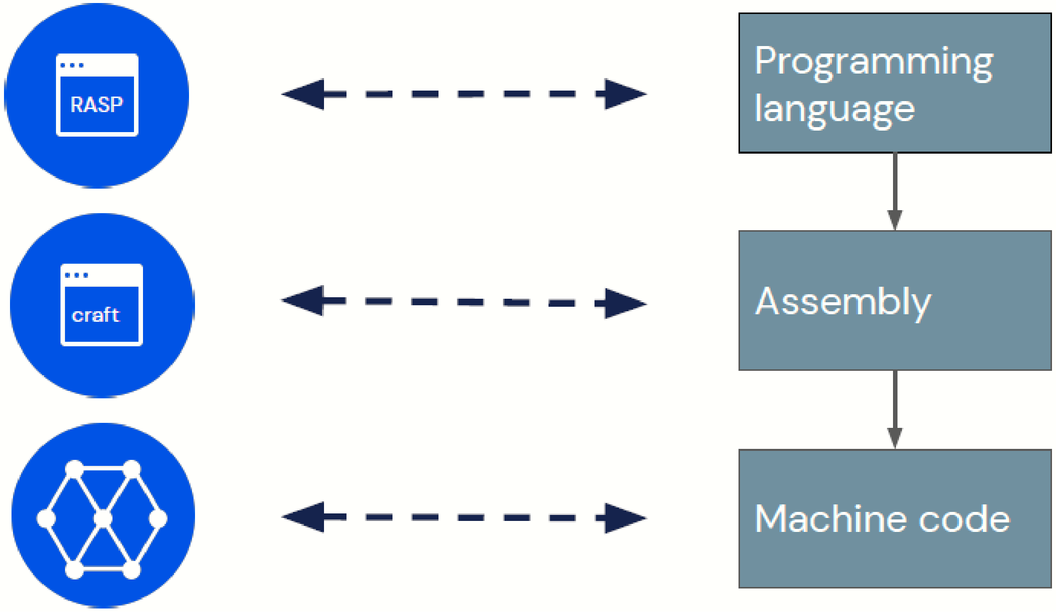
Circuits = Functions: How does the model think?

- **Attention heads can be understood as independent operations**, each outputting a result which is added into the residual stream. Attention heads are often described in an alternate "concatenate and multiply" formulation for computational efficiency, but this is mathematically equivalent.
- **Attention-only models can be written as a sum of interpretable end-to-end functions** mapping tokens to changes in logits. These functions correspond to "paths" through the model, and are linear if one freezes the attention patterns.
- **Transformers have an enormous amount of linear structure**. One can learn a lot simply by breaking apart sums and multiplying together chains of matrices.

[A Mathematical Framework (Elhage et al)](#)

Open Problems: [Analysing Toy Language Models](#)

Tracr works analogously to how we would translate a programming language into executable code



Lindler et al., Tracr: Compiled Transformers as a Laboratory for Interpretability, https://arxiv.org/abs/2301.05062

Forward Engineering (Compiler)

Tracr translates human readable code into transformer model weights in three steps

RASP

Human readable code in domain-specific language

craft

Basis independent representation of vector spaces and transformers

Neural network weights

Lindler et al., Tracr: Compiled Transformers as a Laboratory for Interpretability, https://arxiv.org/abs/2301.05062

RASP is a symbolic programming language to describe transformer computations

RASP

Arbitrary element-wise functions f( ) ⟵ - - - ⟶ **MLP layers**

"Select-aggregate" operations ⟵ - - - ⟶ (w/ some limitations) **Attention layers**

RASP = "Restricted Access Sequence Programming"

Weiss, Gail, Yoav Goldberg, and Eran Yahav. "*Thinking like transformers.*" ICML 2021.

Lindler et al., Tracr: Compiled Transformers as a Laboratory for Interpretability, https://arxiv.org/abs/2301.05062

Forward Engineering (Compiler)

Slide by David Lindler

Georgia Tech

Tracr can compile a range of meaningful programs, but it is not fully general

We can implement programs to:
- Count tokens and compute histograms
- Detect all occurrences of a patterns
- Sort the input sequence
- Check balanced parentheses (Dyck-n)
- ...

Limitations of RASP

- Binary attention patterns
- Designed to model algorithmic tasks and not probabilistic tasks
- Programs still relatively close to transformer architecture

Limitations of Tracr

- Resulting models are large and inefficient
- Many possible optimization missing
- Some advanced RASP features not supported

Lindler et al., Tracr: Compiled Transformers as a Laboratory for Interpretability, https://arxiv.org/abs/2301.05062

- Large models, especially transformers, can be seen as implementing **algorithms**

- Several ways to try to understand:
  - Visualization – Often attention
  - Distill into more interpretable model
  - Reverse engineer
  - Forward engineer: Algorithm → compiler → Weights!

Summary