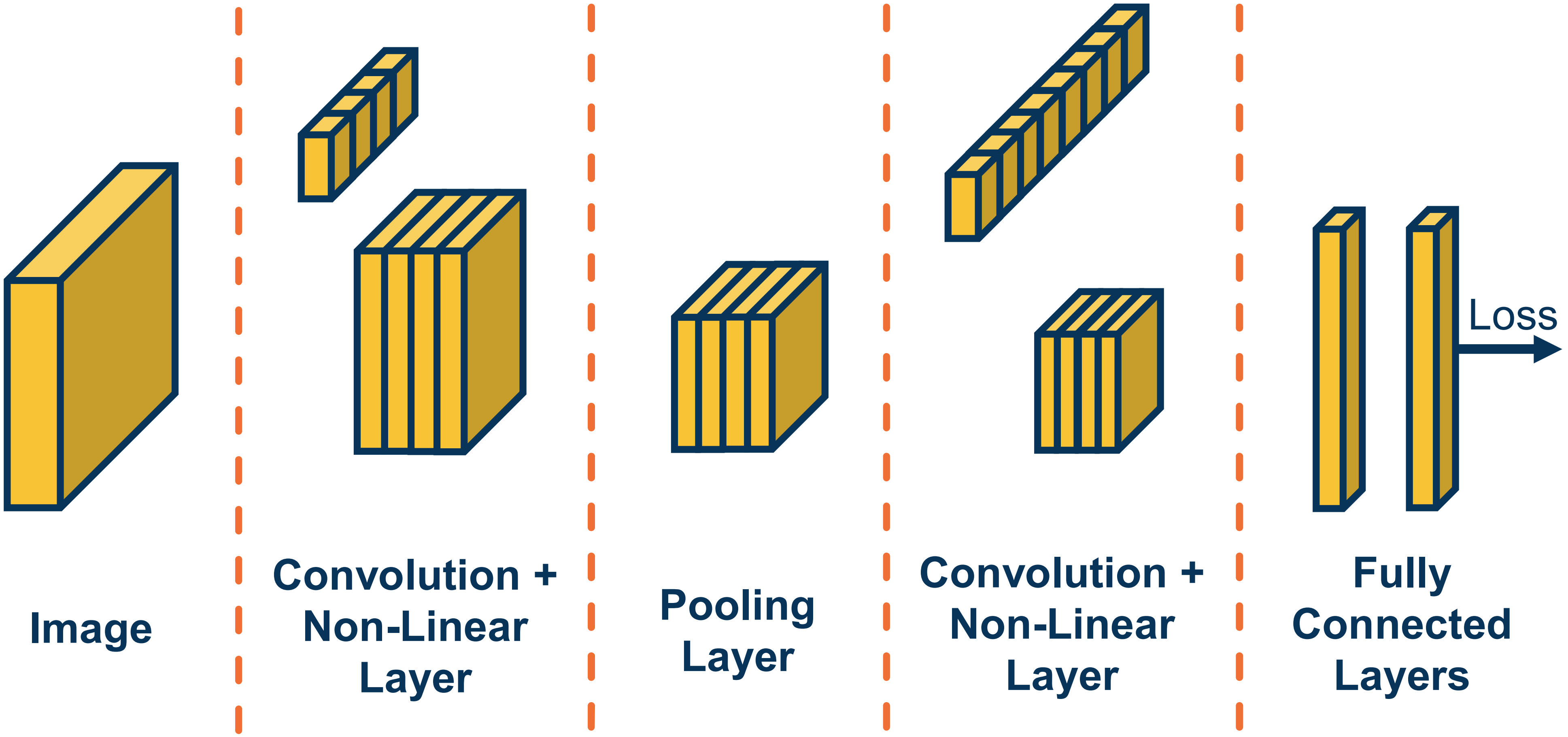
Topics:

- CNNs
- Transfer Learning
- Visualization

CS 4644-DL / 7643-A
ZSOLT KIRA

- **Assignment 2**
  - Due soon!
  - Resources (in addition to lectures):
    - [DL book: Convolutional Networks]
    - CNN notes https://www.cc.gatech.edu/classes/AY2022/cs7643_spring/assets/L10_cnns_notes.pdf
    - Backprop notes https://www.cc.gatech.edu/classes/AY2022/cs7643_spring/assets/L10_cnns_backprop_notes.pdf
    - **HW2 Tutorial @113, Conv @116, Focal Loss @117**
    - Slower OMSCS lectures on dropbox: Module 2 Lessons 5-6 (M2L5/M2L6) (https://www.dropbox.com/sh/iviro188gq0b4vs/AADdHxX_Uy1TkpF_yvIzX0nPa?dl=0)
- **Meta OH** right after this lecture (2pm ET)!
- **Projects**
  - Project proposal due **March 13th**
  - Some Meta project topics up.
  - March 8th – Class will be used for project planning session
  - Form teams and topic now!



Adding a Fully Connected Layer

These architectures have existed **since 1980s**

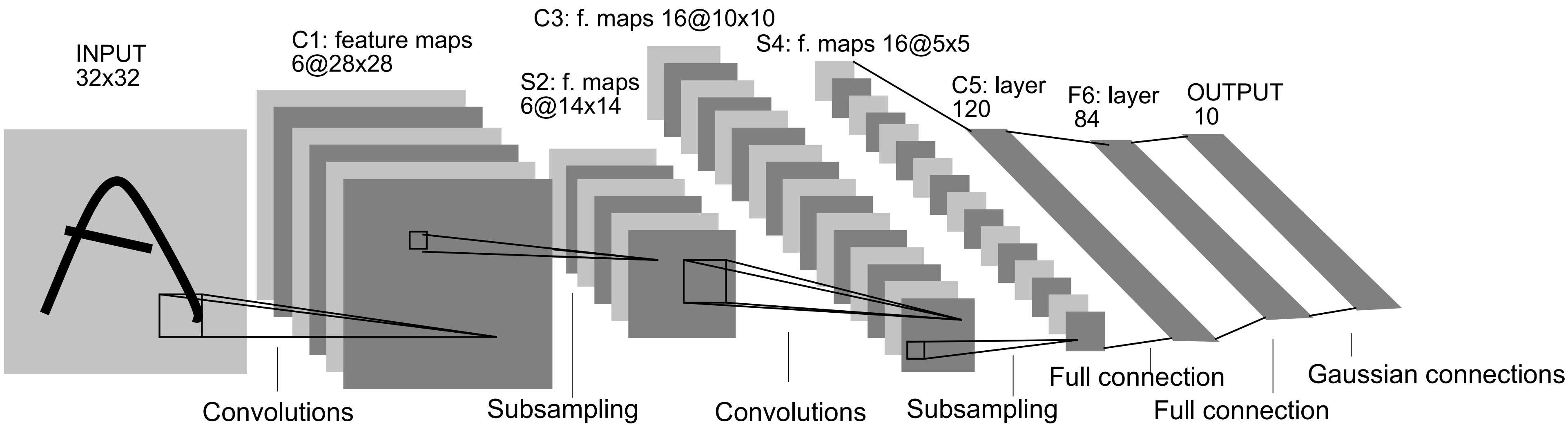
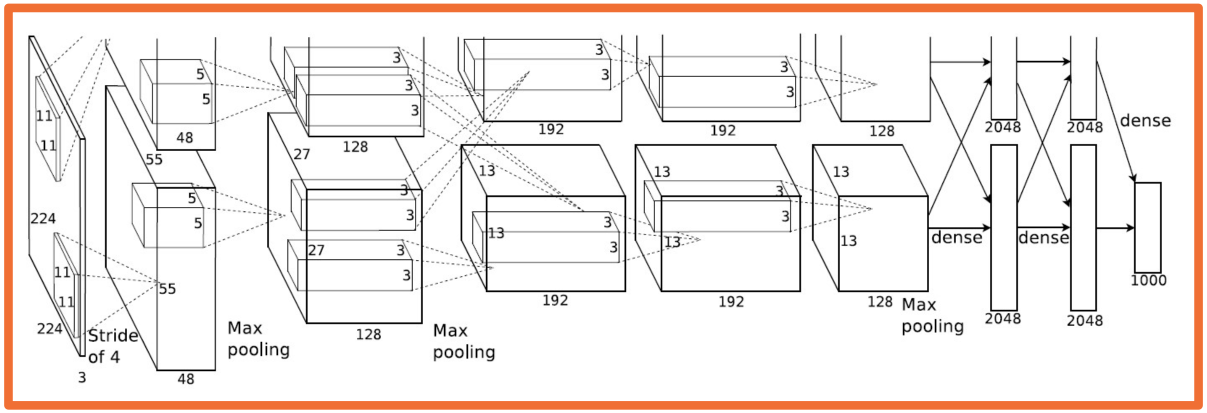


Image Credit: Yann LeCun, Kevin Murphy

LeNet Architecture

The Importance of Benchmarks

From: https://paperswithcode.com



Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Q: what is the output volume size? Hint: (227-11)/4+1 = 55

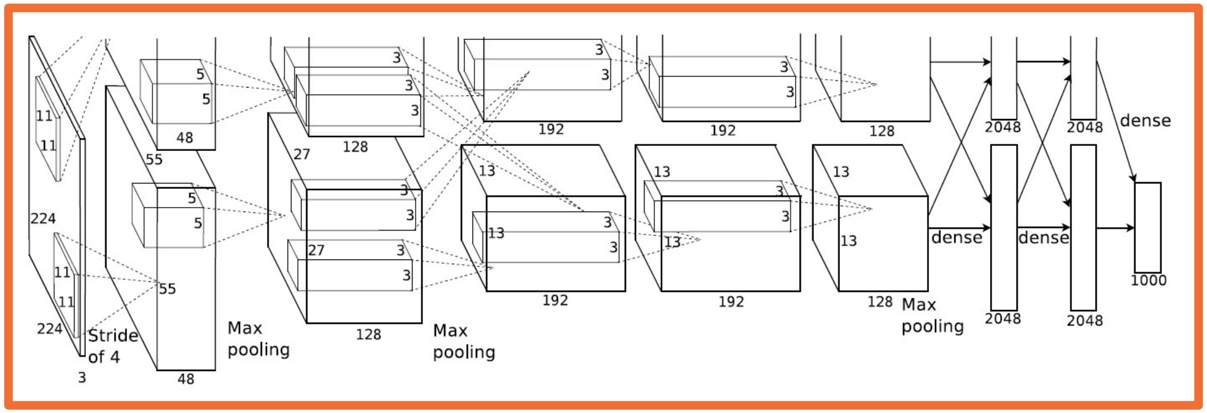
W' = (W - F + 2P) / S + 1

From: Slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

AlexNet – Layers and Key Aspects

Georgia Tech

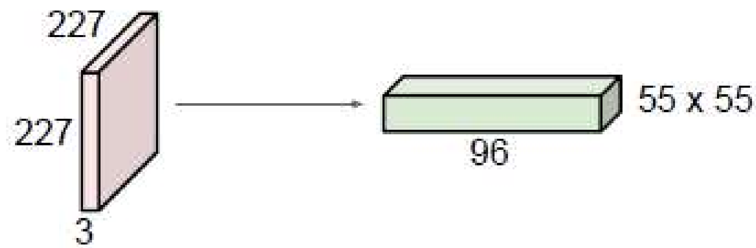
AlexNet – Layers and Key Aspects



Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**

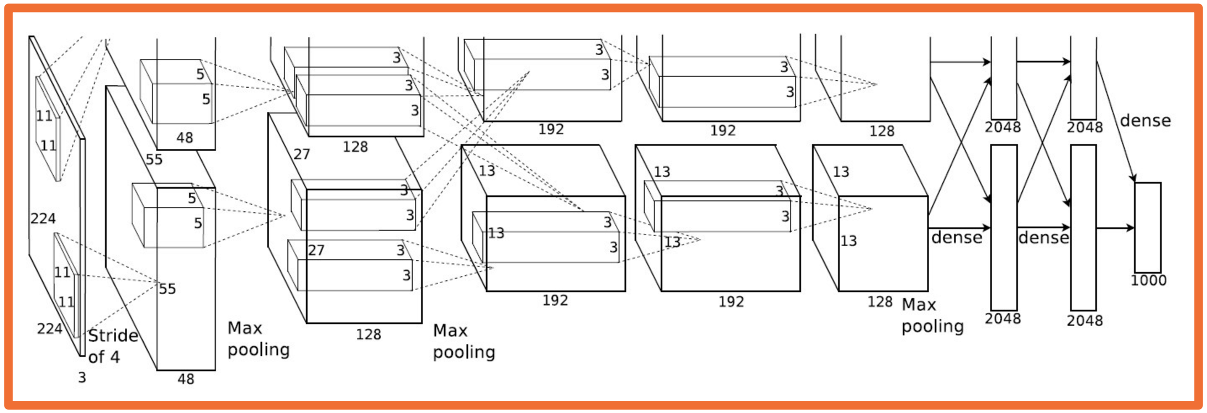
$$W' = (W - F + 2P) / S + 1$$



From: Slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Georgia Tech

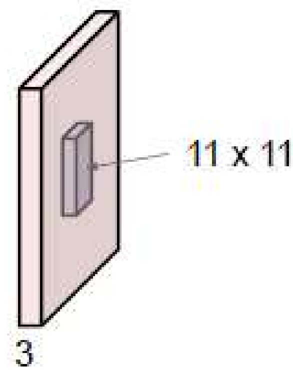
AlexNet – Layers and Key Aspects



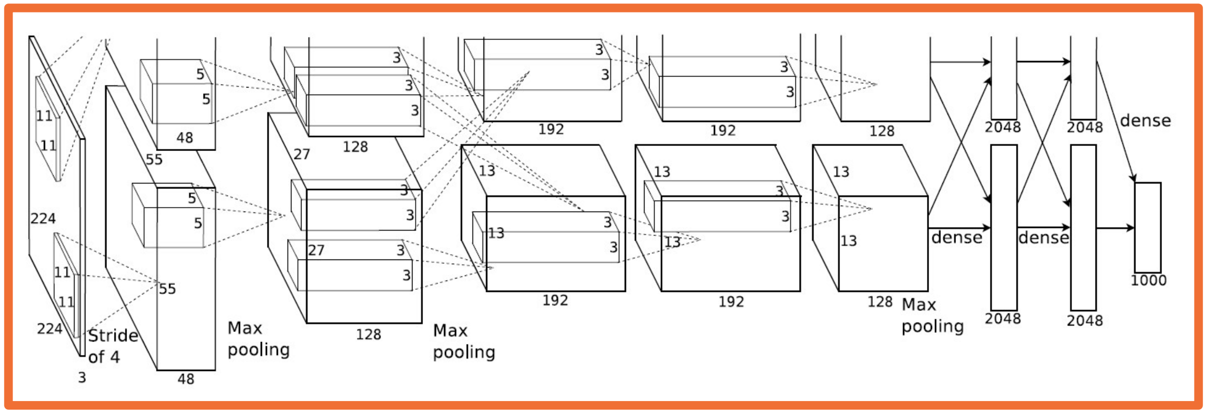
Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**

Q: What is the total number of parameters in this layer?

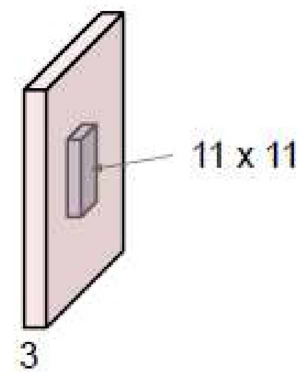


From: Slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



Input: 227x227x3 images

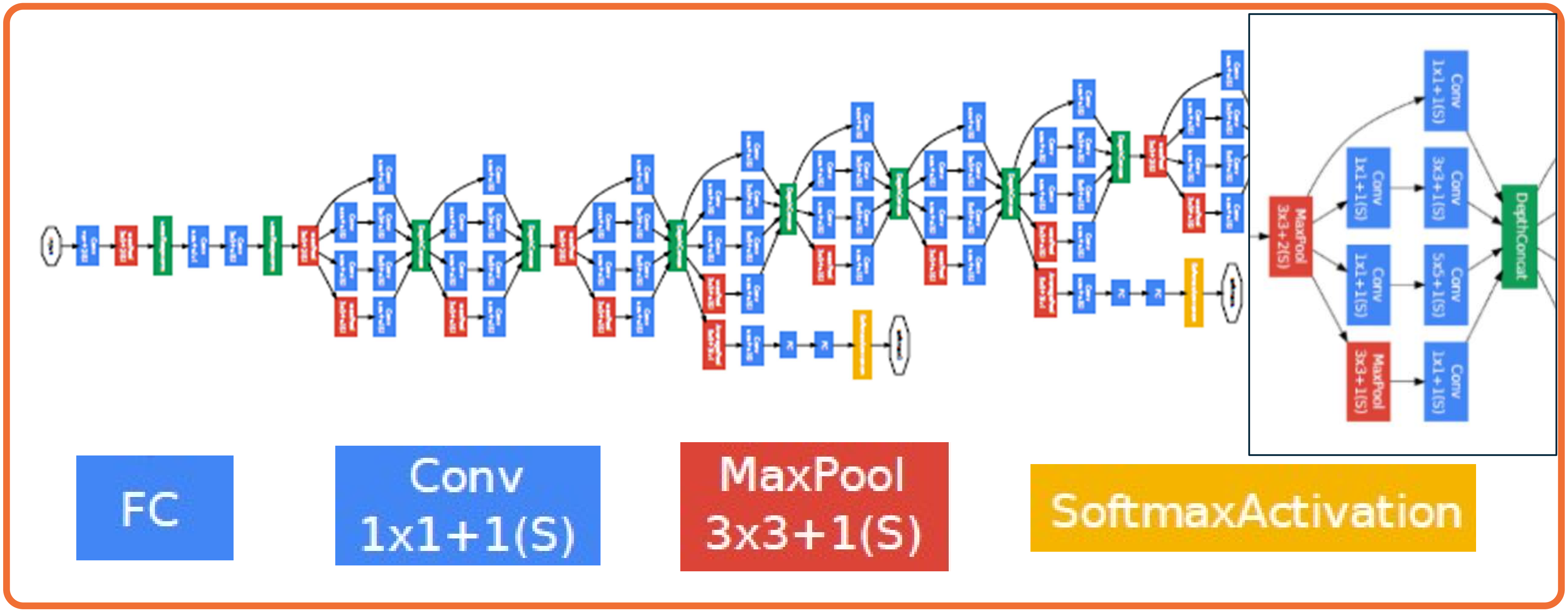
First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**
Parameters: (11*11*3 + 1)*96 = **35K**



From: Slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

AlexNet – Layers and Key Aspects

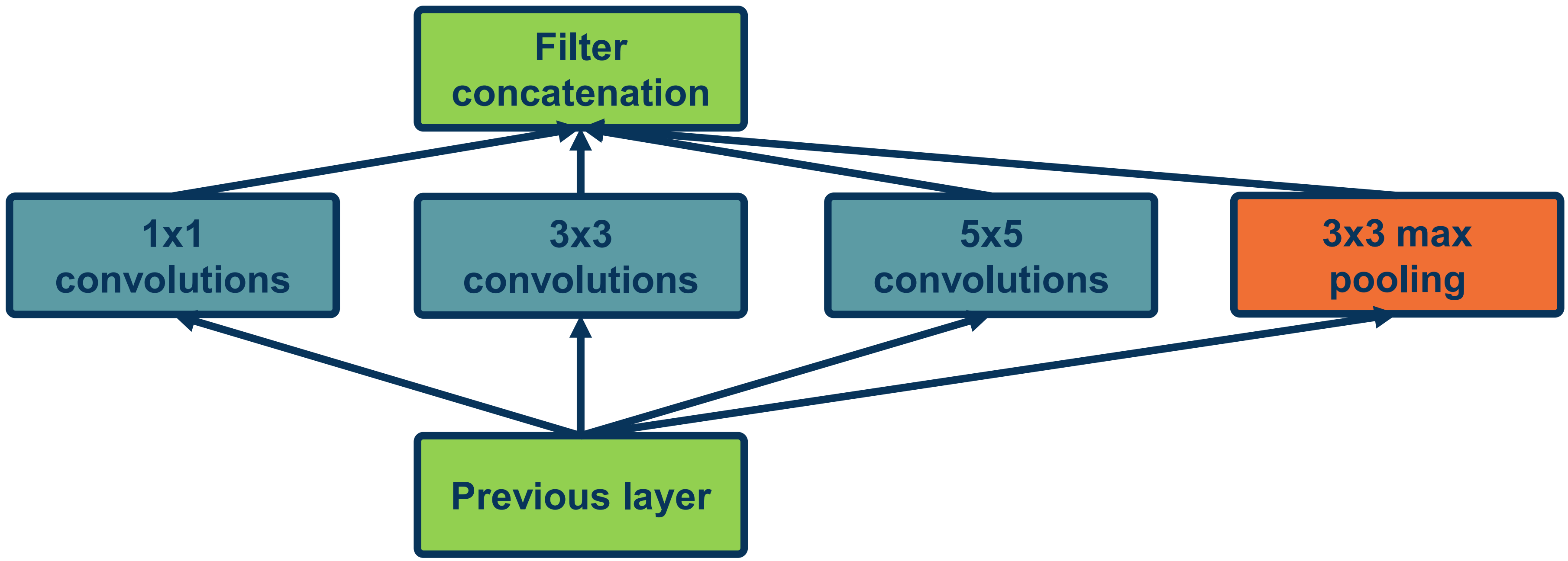
But have become **deeper and more complex**



From: Szegedy et al. Going deeper with convolutions

Inception Architecture

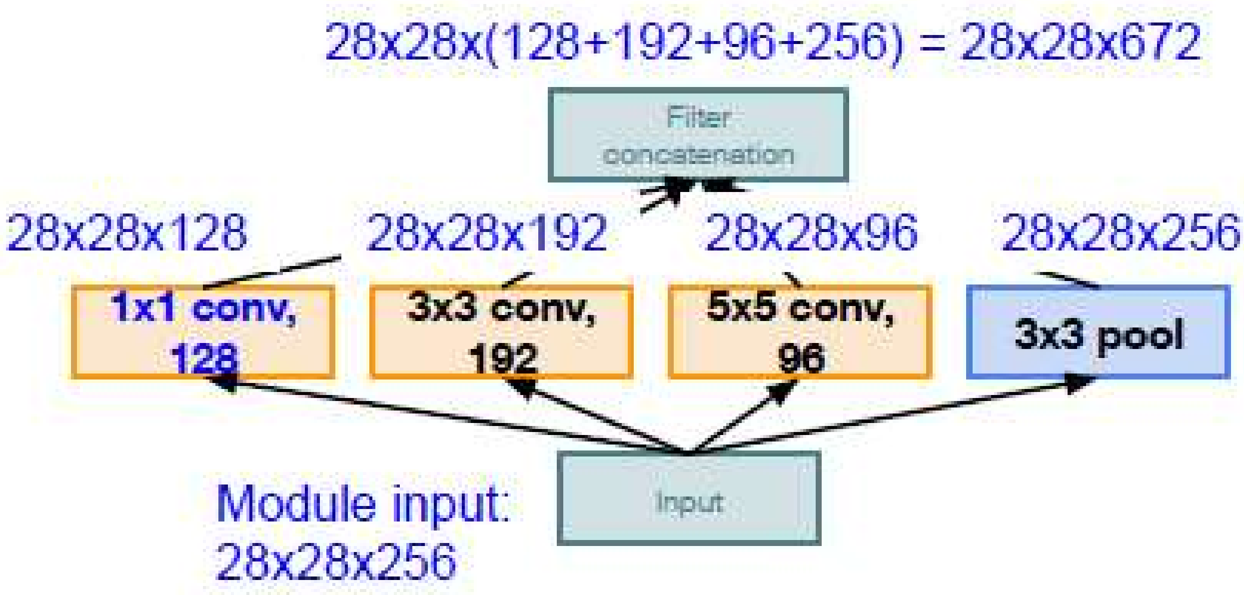
Key idea: Repeated blocks and multi-scale features



From: Szegedy et al. Going deeper with convolutions

Inception Module

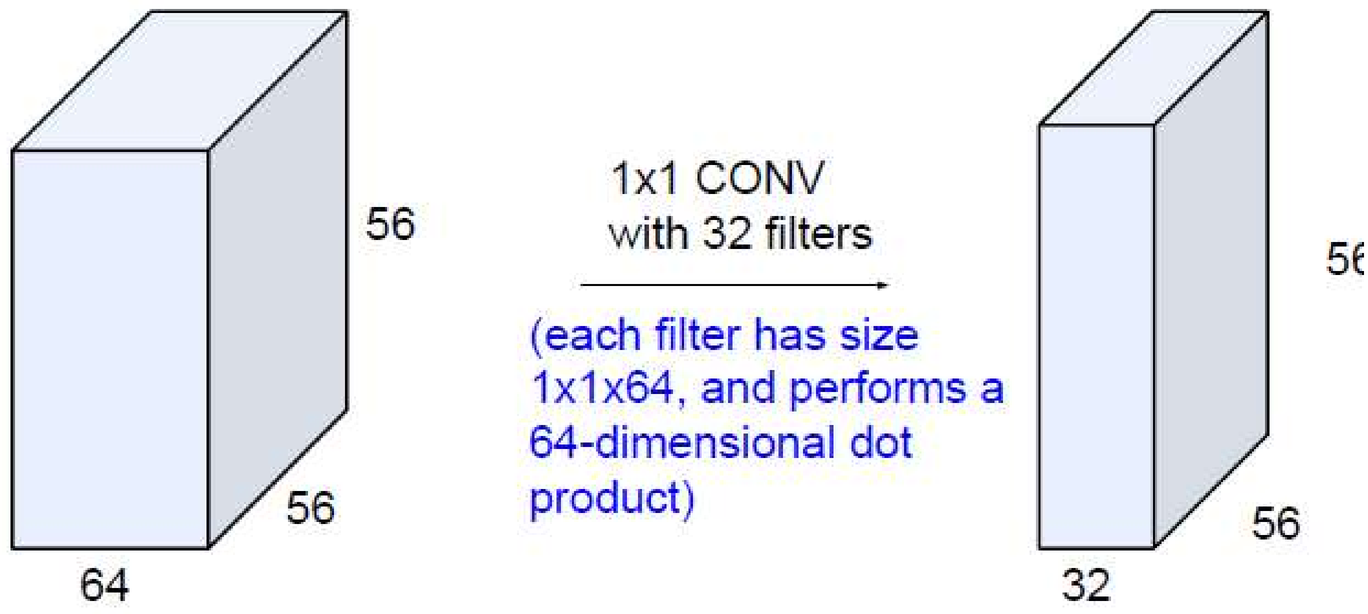
Key idea: Repeated blocks and multi-scale features



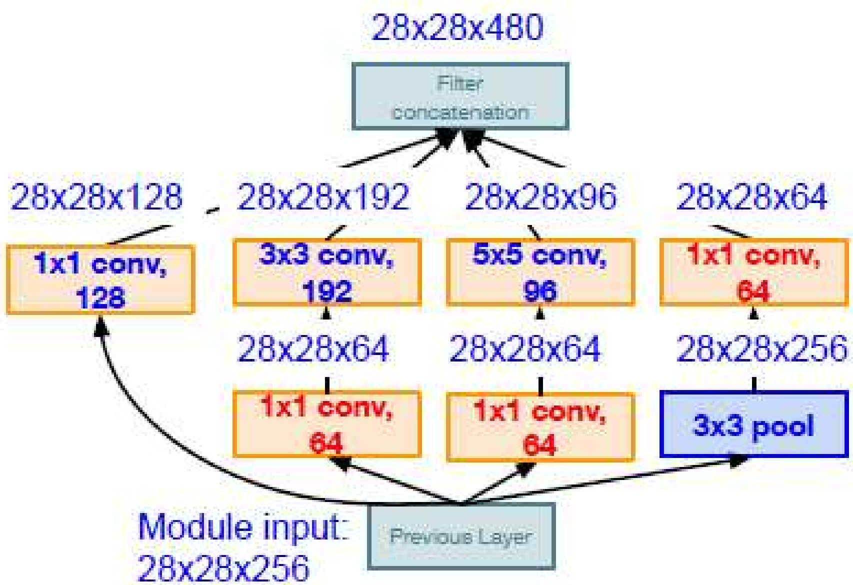
Naive Inception module

From: Slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Apply 1x1 convolutions as bottleneck layer (decrease number of channels!)



From: Slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



Inception module with dimension reduction

Using same parallel layers as naive example, and adding "1x1 conv, 64 filter" bottlenecks:

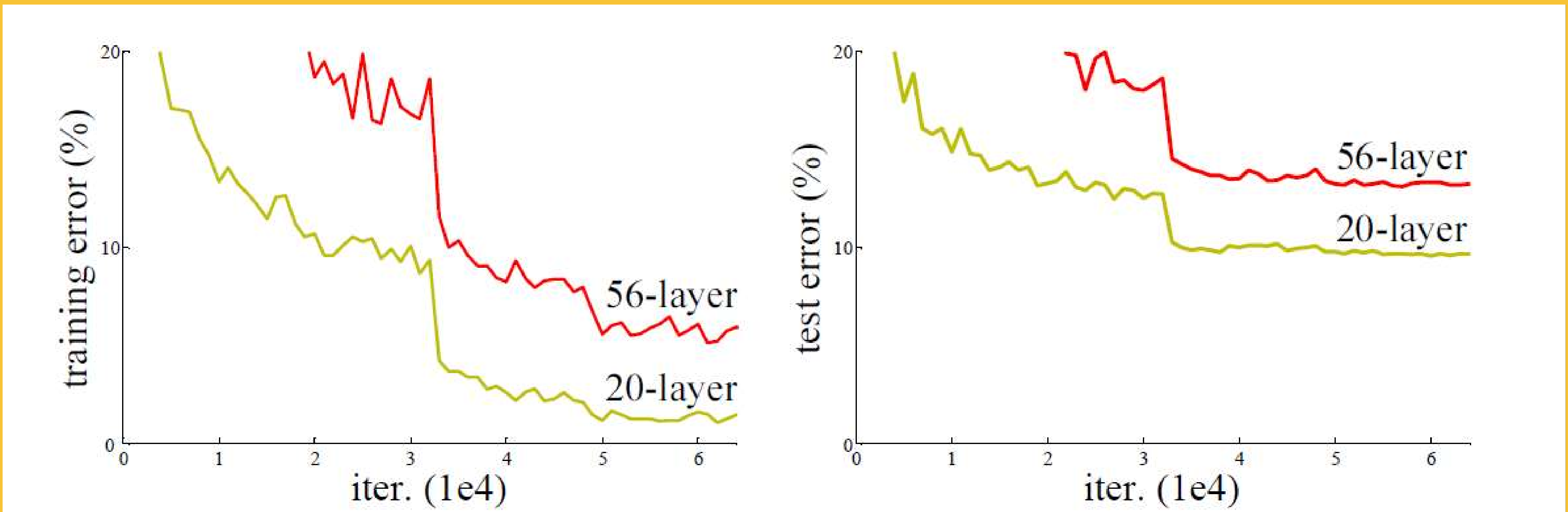
Conv Ops:
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x64
[5x5 conv, 96] 28x28x96x5x5x64
[1x1 conv, 64] 28x28x64x1x1x256
Total: 358M ops

Compared to 854M ops for naive version
Bottleneck can also reduce depth after pooling layer

From: Slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Inception Module

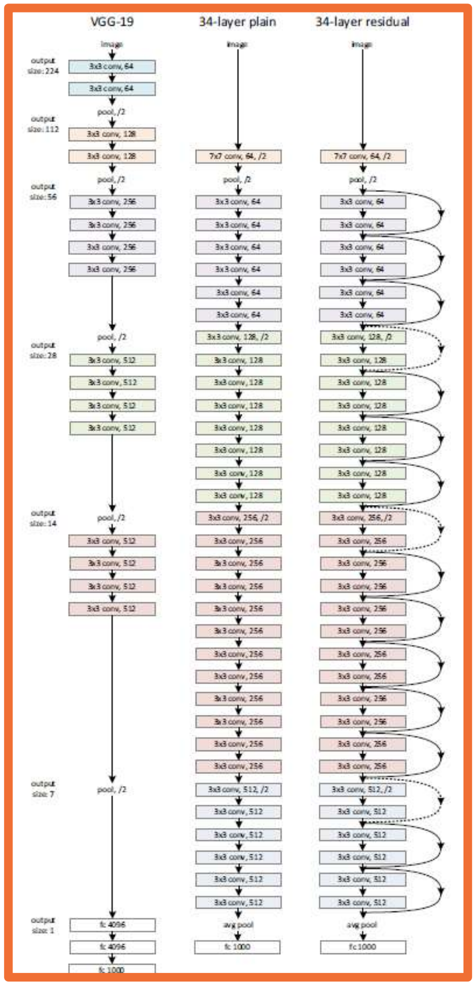
The Challenge of Depth

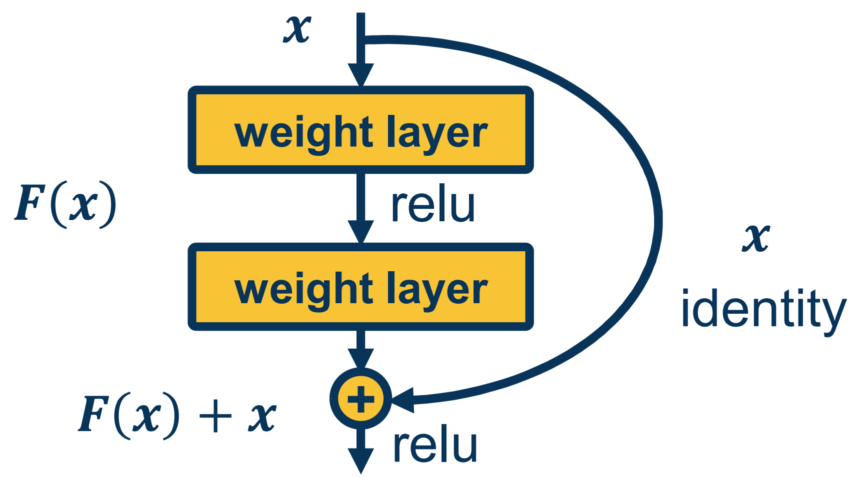


From: He et al., Deep Residual Learning for Image Recognition

Optimizing very deep networks is challenging!

Georgia Tech





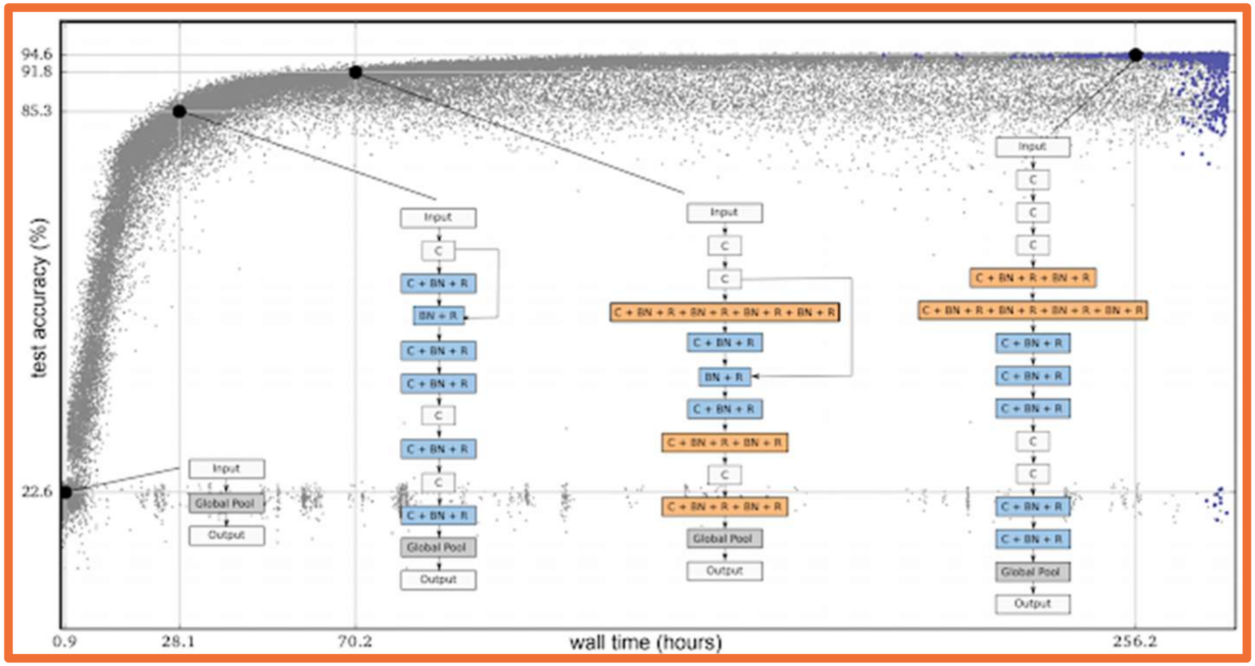
Key idea: Allow information from a layer to propagate to any future layer (forward)

Same is true for gradients!

From: He et al., Deep Residual Learning for Image Recognition

Several ways to *learn* architectures:

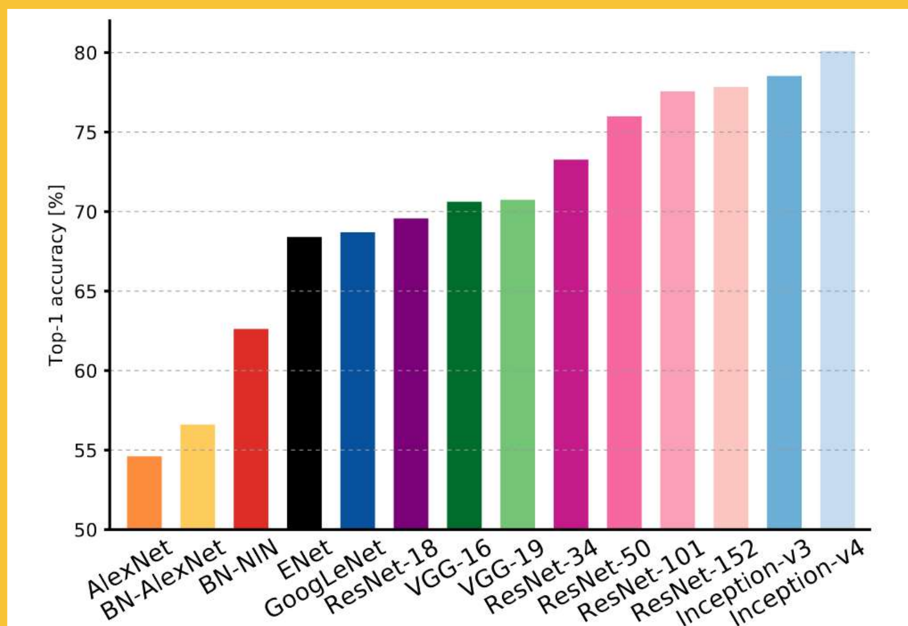
- Evolutionary learning and reinforcement learning
- Prune over-parameterized networks
- Learning of **repeated blocks** typical

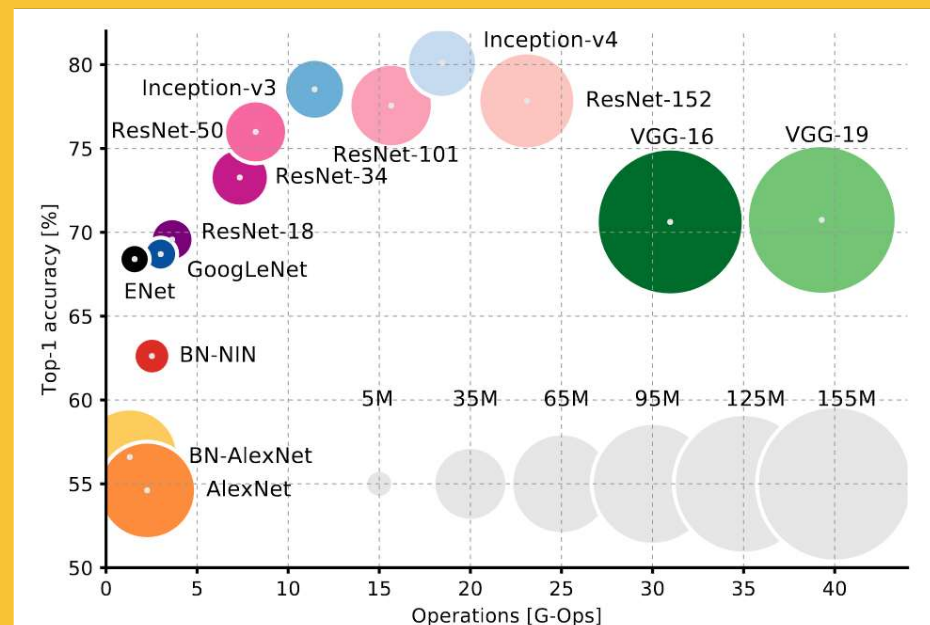


From: https://ai.googleblog.com/2018/03/using-evolutionary-automl-to-discover.html

Evolving Architectures and AutoML

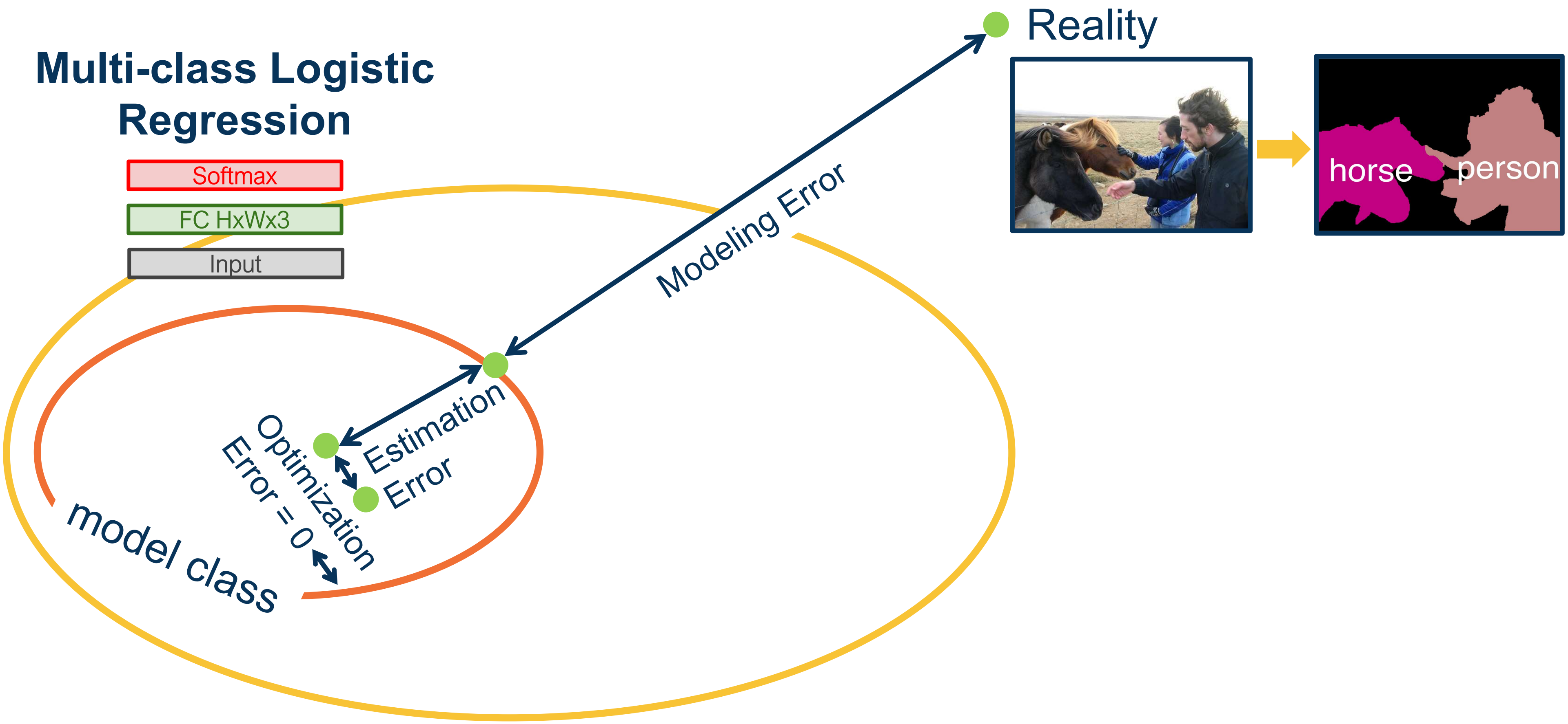
Computational Complexity





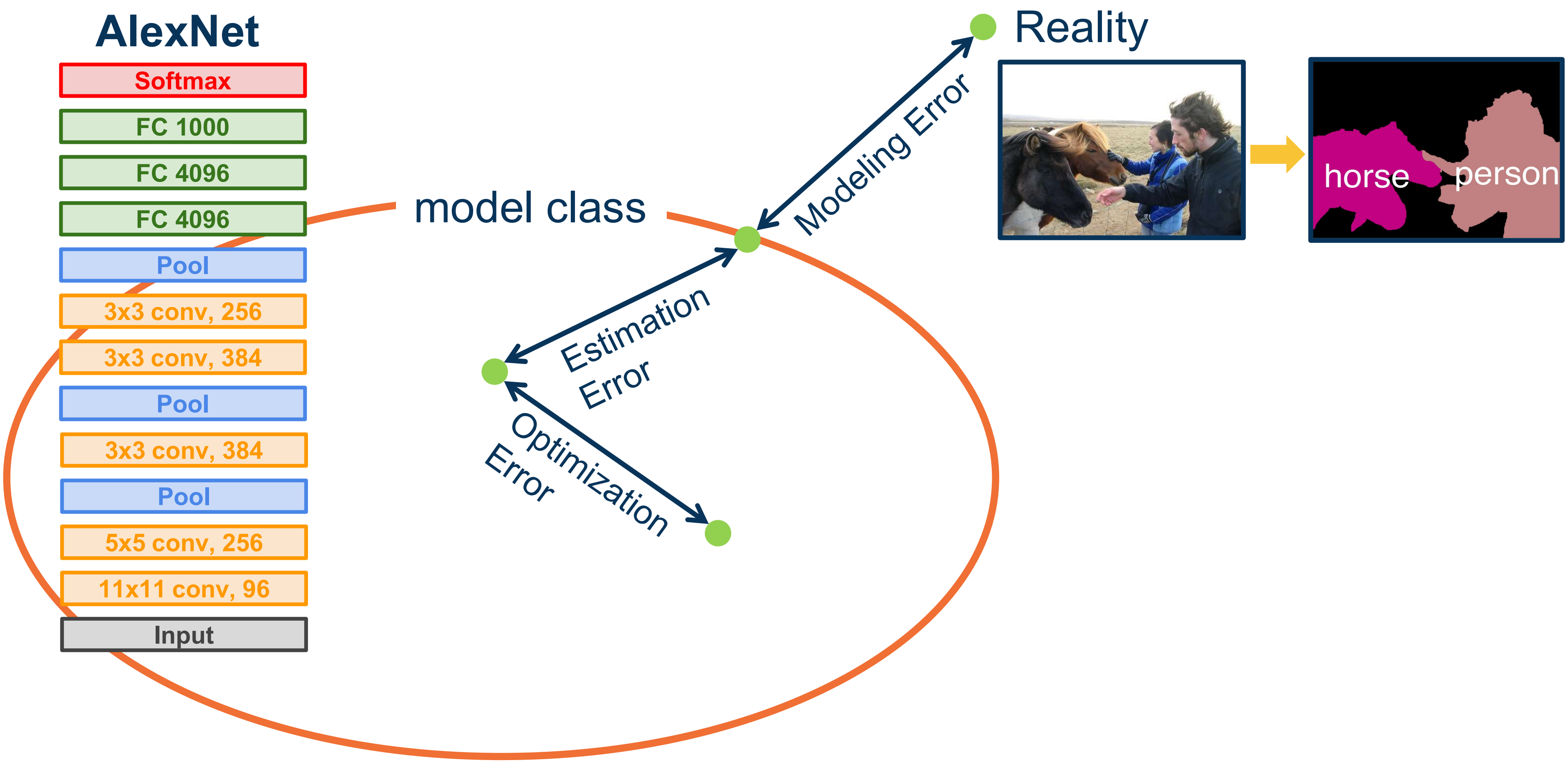
From: An Analysis Of Deep Neural Network Models For Practical Applications

Transfer Learning & Generalization



From: slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

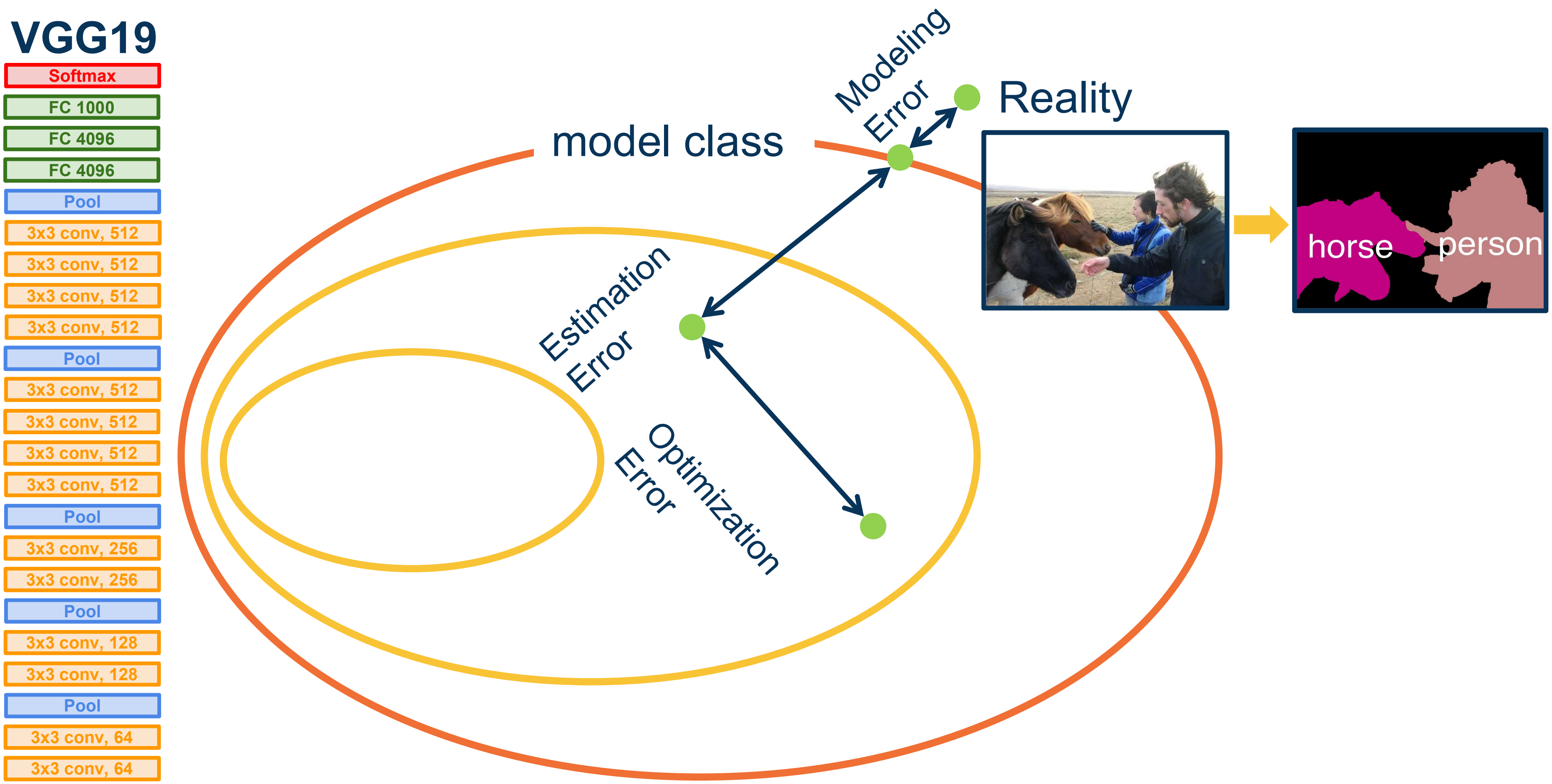
Generalization



From: slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Generalization

Generalization

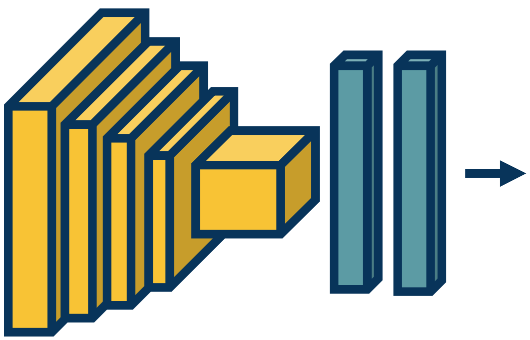


From: slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

What if we don’t have enough data?

Step 1: Train on large-scale dataset

Input Image

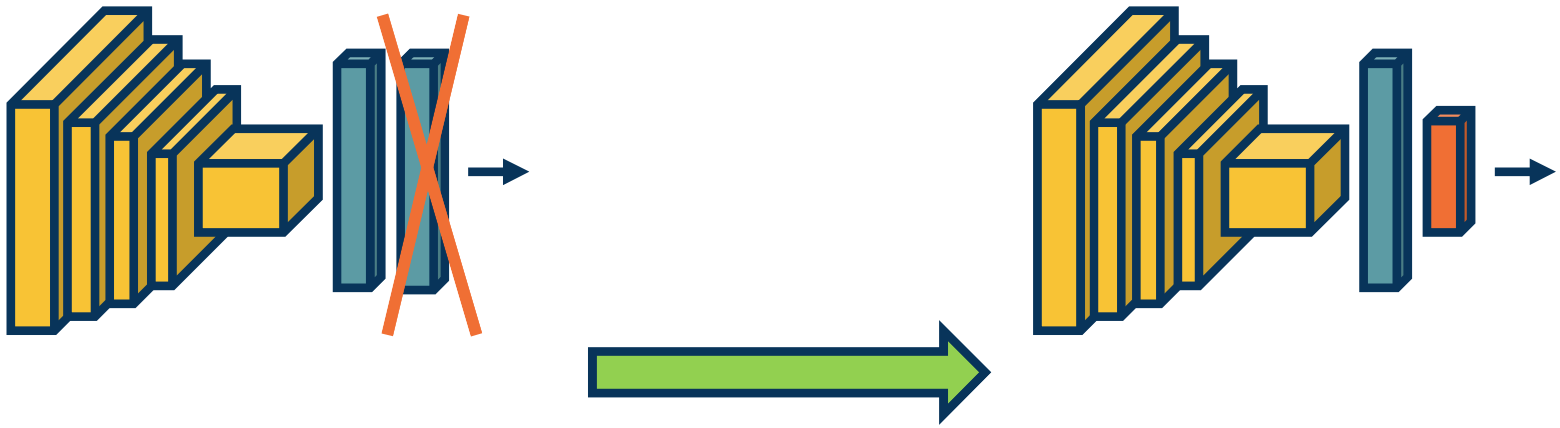


Predictions

Convolutional Neural Networks

Transfer Learning – Training on Large Dataset

Step 2: Take your custom data and **initialize** the network with weights trained in Step 1



Replace last layer with new fully-connected for output nodes per new category

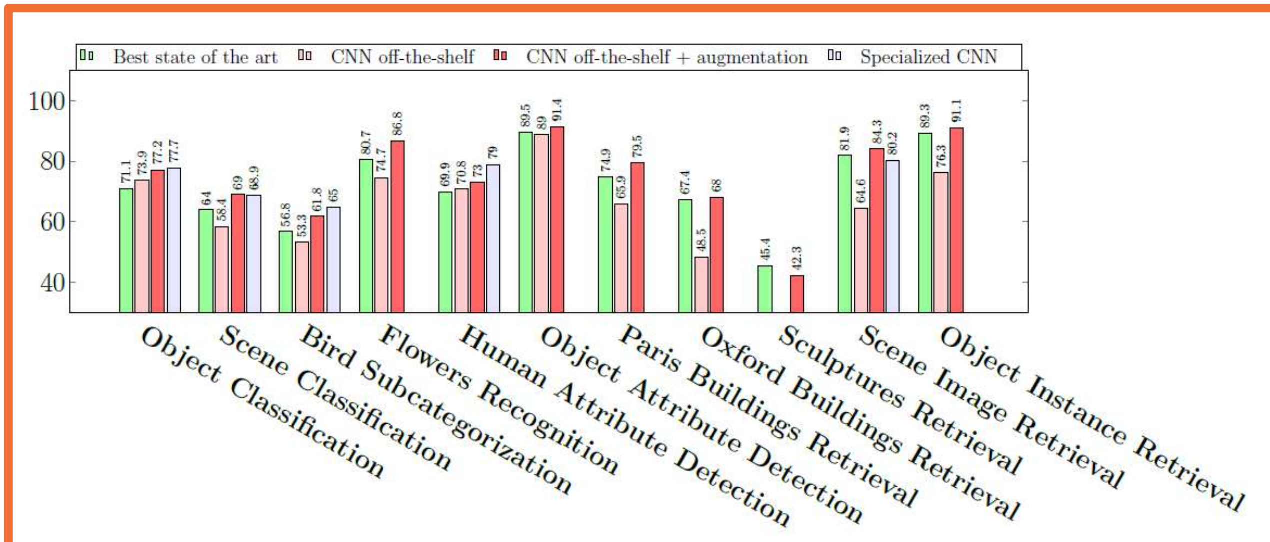
Step 3: (Continue to) train on new dataset

- **Finetune:** Update all parameters
- **Freeze** feature layer: Update only last layer weights (used when not enough data)

Replace last layer with new fully-connected for output nodes per new category

This works extremely well! It was surprising upon discovery.

- Features learned for 1000 object categories will work well for 1001st!
- Generalizes even across tasks (classification to object detection)



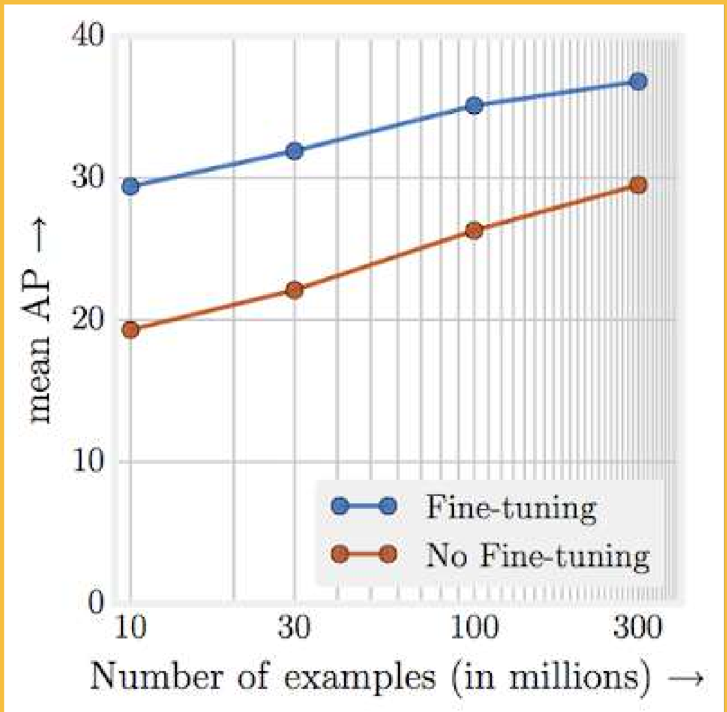
From: Razavian et al., CNN Features off-the-shelf: an Astounding Baseline for Recognition

Learning with Less Labels

But it doesn't always work that well!

- If the **source** dataset you train on is very different from the **target** dataset, transfer learning is not as effective
- If you have enough data for the target domain, it just results in faster convergence
  - See He et al., "Rethinking ImageNet Pre-training"

Effectiveness of More Data



From: Revisiting the Unreasonable Effectiveness of Data https://ai.googleblog.com/2017/07/revisiting-unreasonable-effectiveness.html

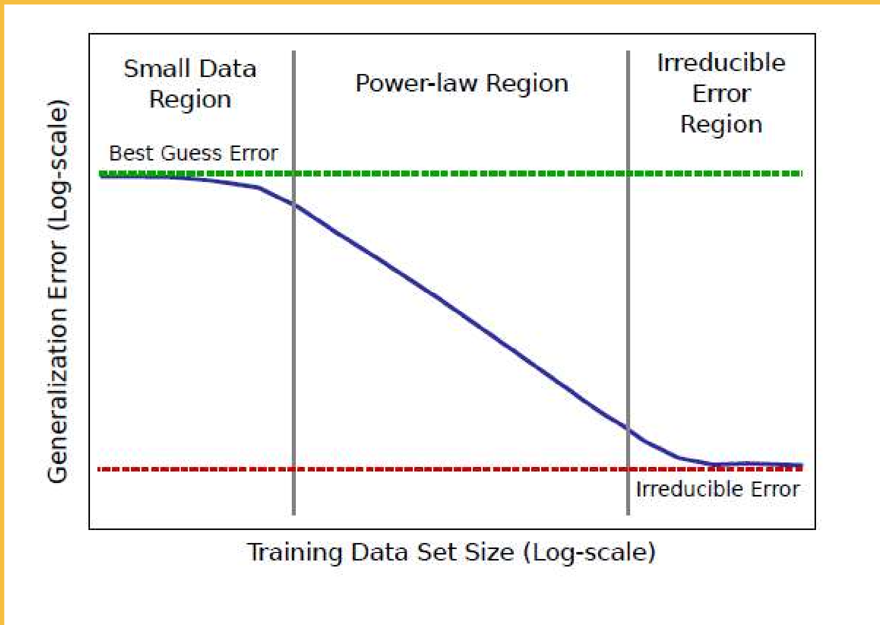


Figure 6: Sketch of power-law learning curves

From: Hestness et al., Deep Learning Scaling Is Predictable

Georgia Tech

There is a large number of different low-labeled settings in DL research

Setting	Source	Target	Shift Type
Semi-supervised	Single labeled	Single unlabeled	None
Domain Adaptation	Single labeled	Single unlabeled	Non-semantic
Domain Generalization	Multiple labeled	Unknown	Non-semantic
Cross-Task Transfer	Single labeled	Single unlabeled	Semantic
Few-Shot Learning	Single labeled	Single few-labeled	Semantic
Un/Self-Supervised	Single unlabeled	Many labeled	Both/Task

Non-Semantic Shift

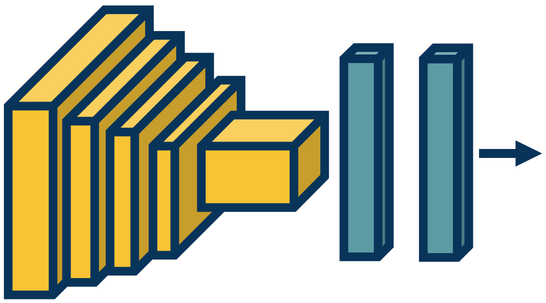
Semantic Shift

Dealing with Low-Labeled Situations

Georgia Tech

Visualization of Neural Networks

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



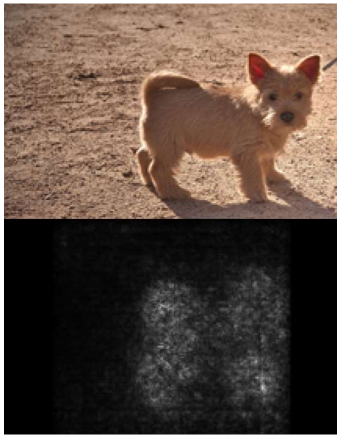
Weights

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

Zeiler & Fergus, 2014

Activations

Gradients



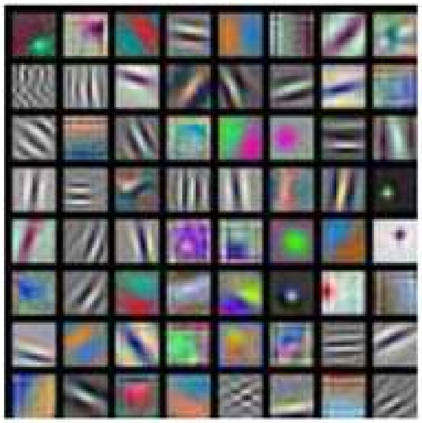
Simonyan et al, 2013

Robustness

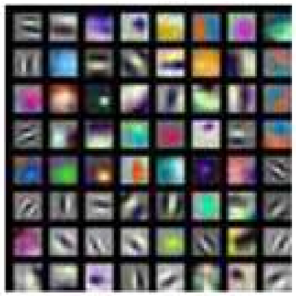
Hendrycks & Dietterich, 2019

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

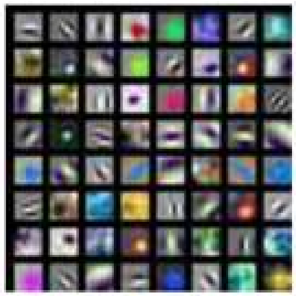
Conv layers:
For each kernel, scale values from 0-255 and visualize



AlexNet:
64 x 3 x 11 x 11



ResNet-18:
64 x 3 x 7 x 7



ResNet-101:
64 x 3 x 7 x 7

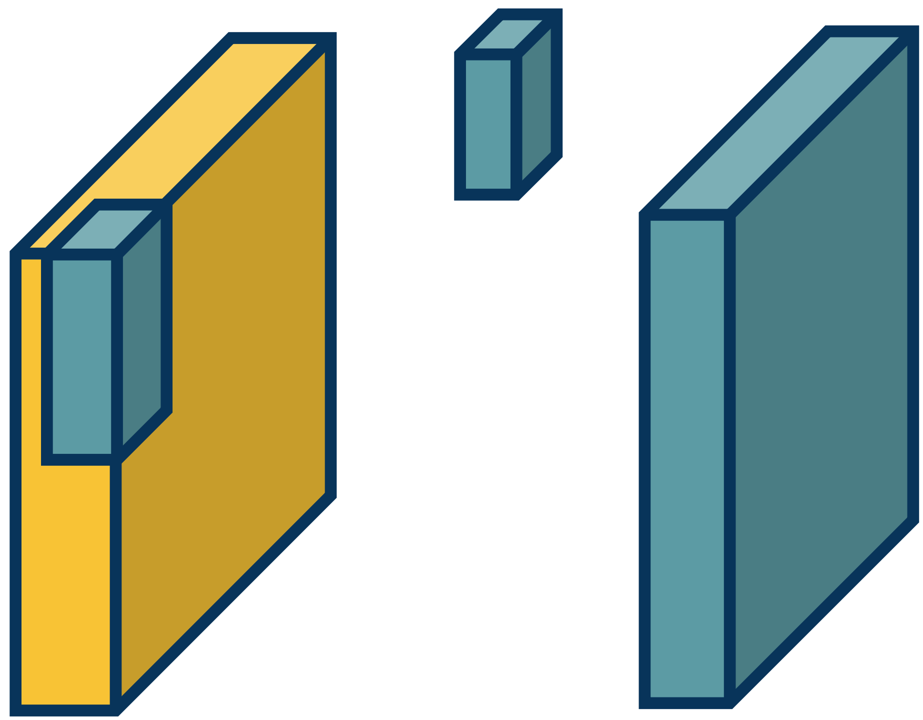
Problem: 3x3 filters difficult to interpret!

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

Visualizing Weights

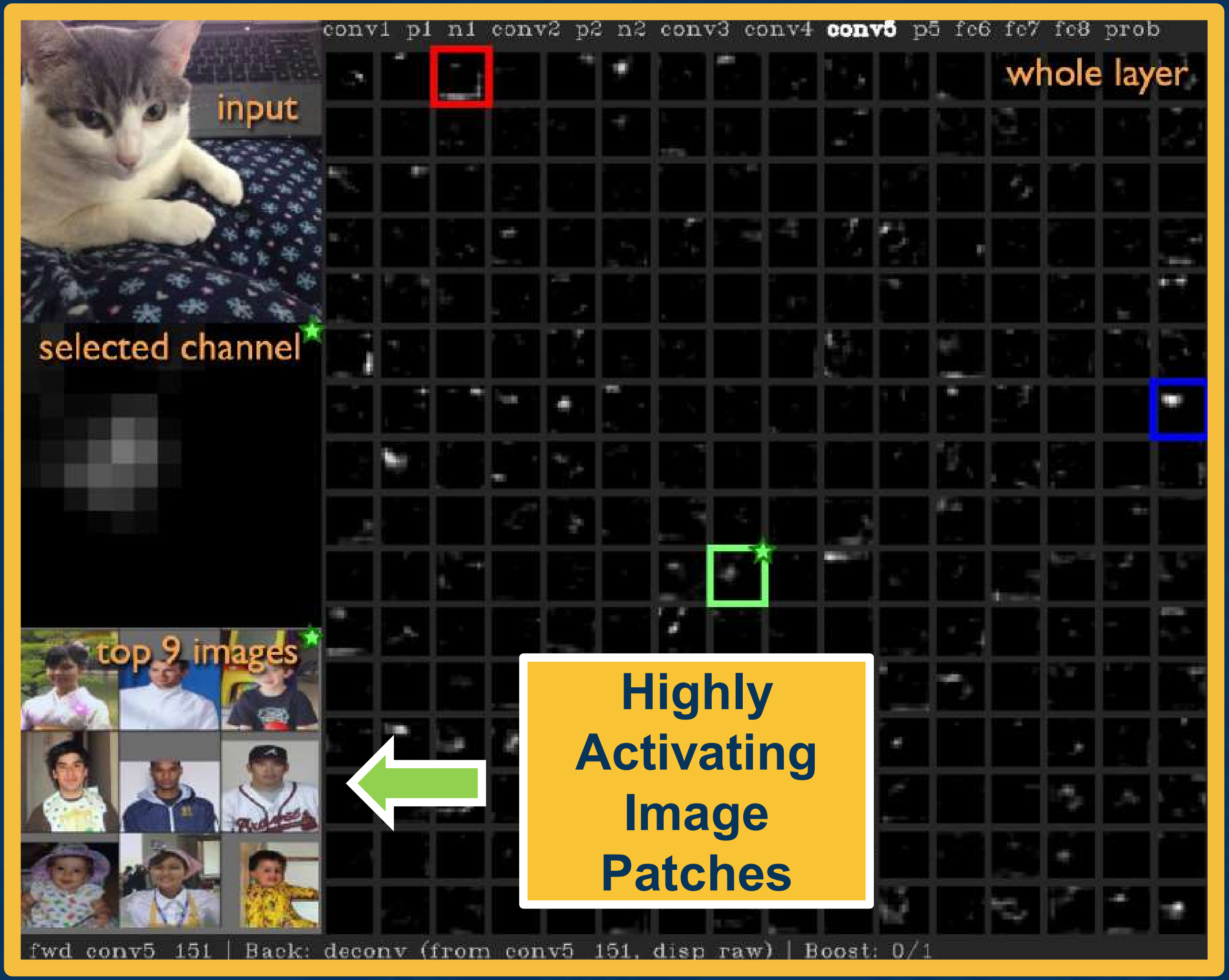
We can also produce **visualization output (aka activation/filter) maps**

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



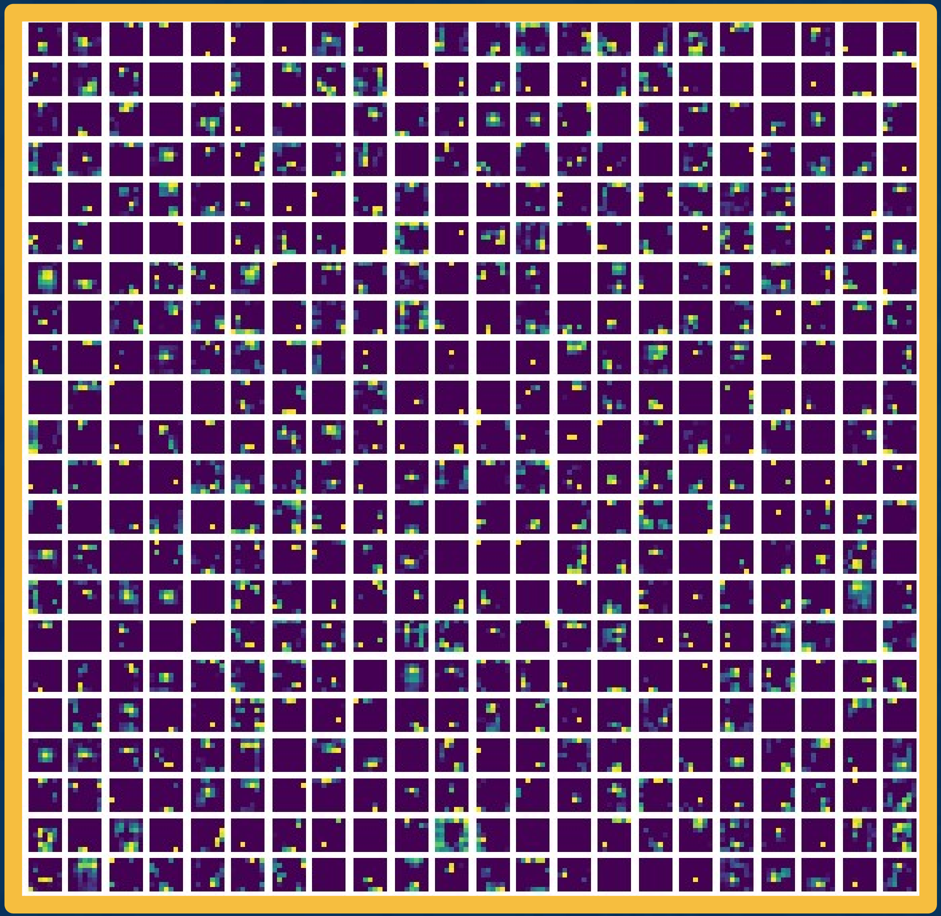
Visualizing Output Maps

Visualizing Output Maps



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

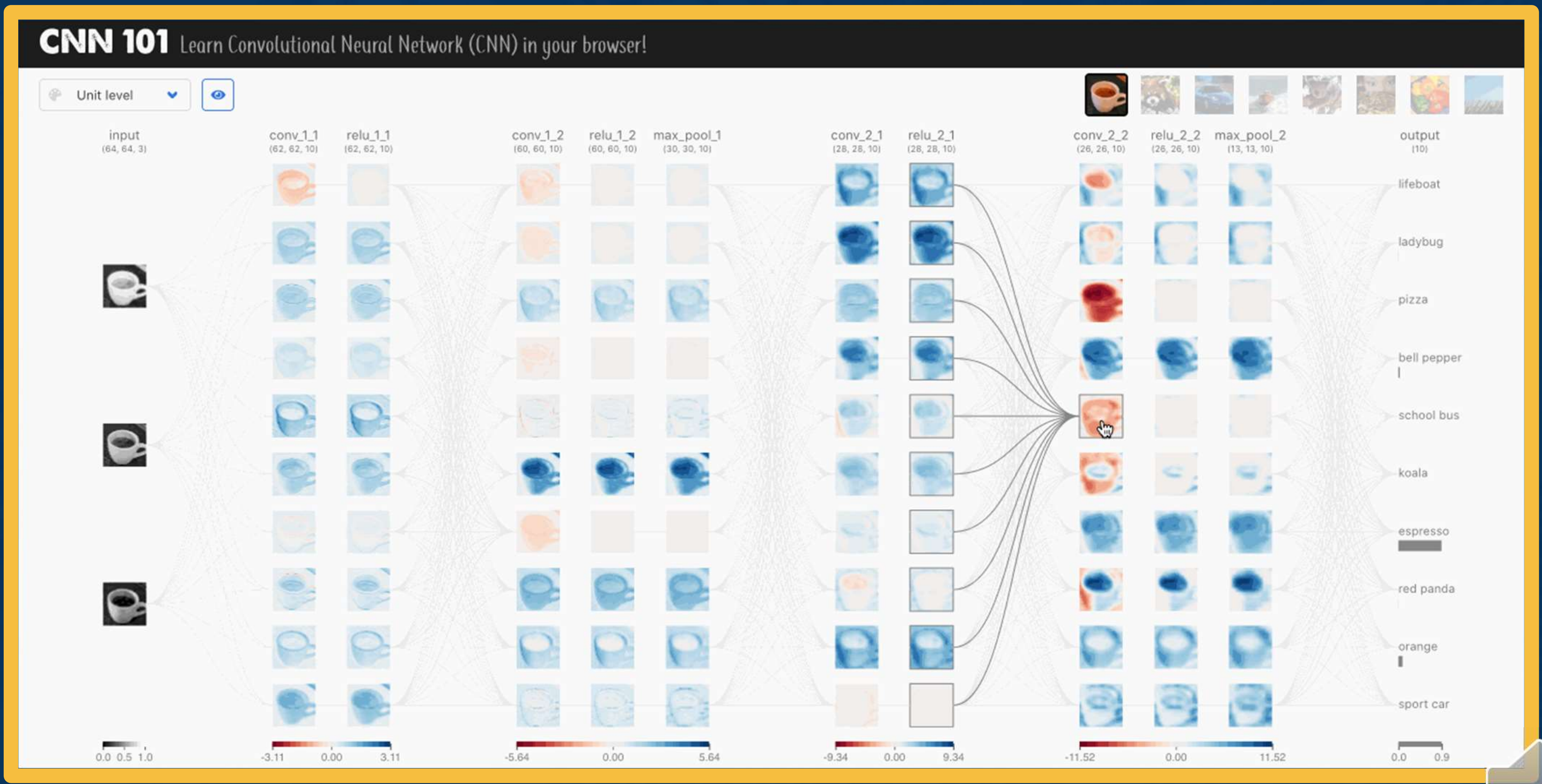
Activations – Small Output Sizes



Activations of last conv layer in VGG network

Problem: Small conv outputs also hard to interpret

CNN101 and CNN Explainer



https://poloclub.github.io/cnn-explainer/

https://fredhohman.com/papers/cnn101

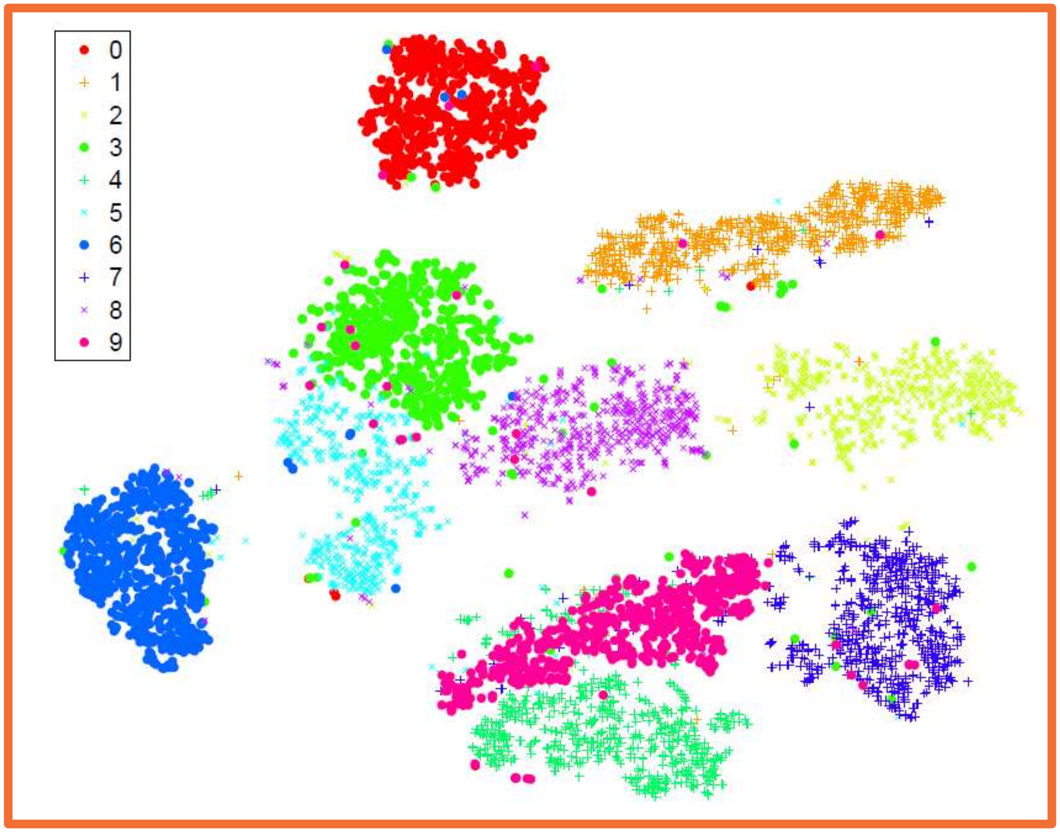
Georgia Tech

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)

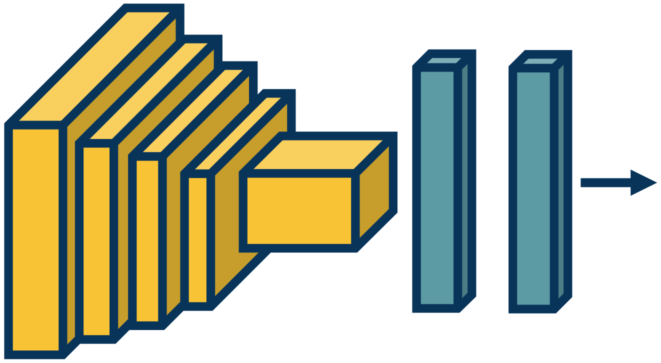
t-SNE is most common

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



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

Dimensionality Reduction: t-SNE



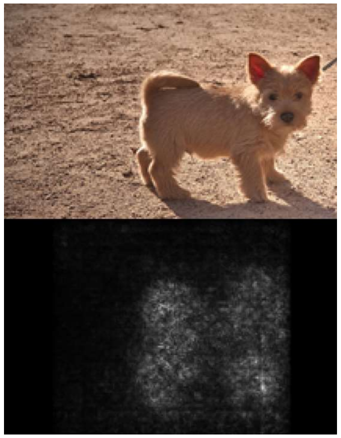
Weights

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

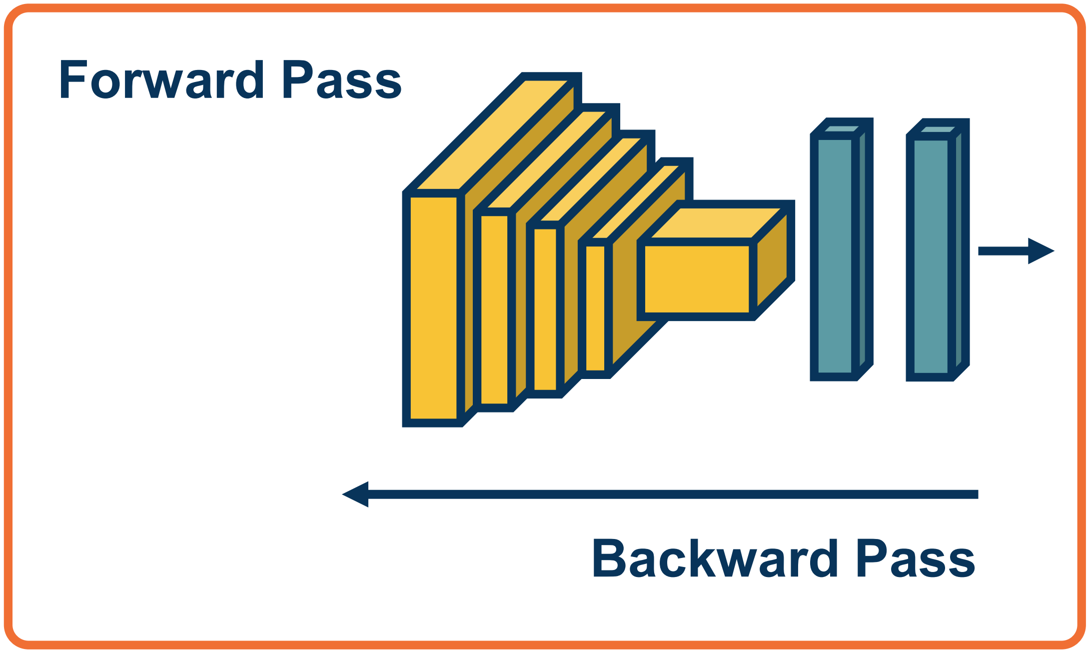
Adebayo et al., "Sanity Checks for Saliency Maps", 2018.

Georgia Tech

Gradient-Based Visualizations

Georgia Tech

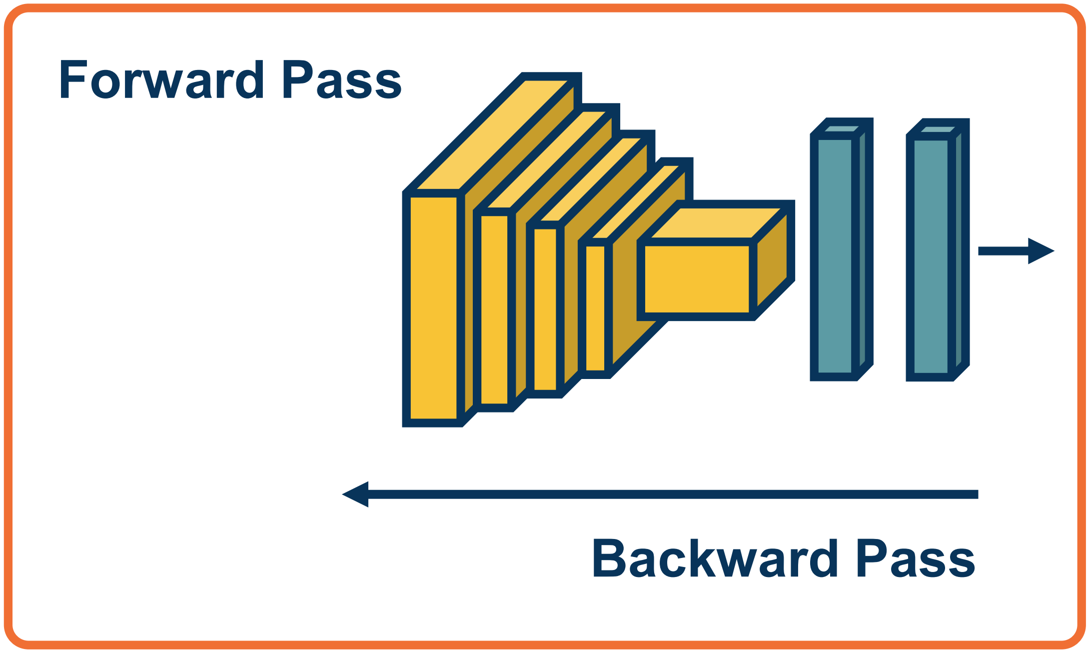
Given a **trained** model, we can perform forward pass given an input to get scores, softmax probabilities, loss and then backwards pass to get gradients



- 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

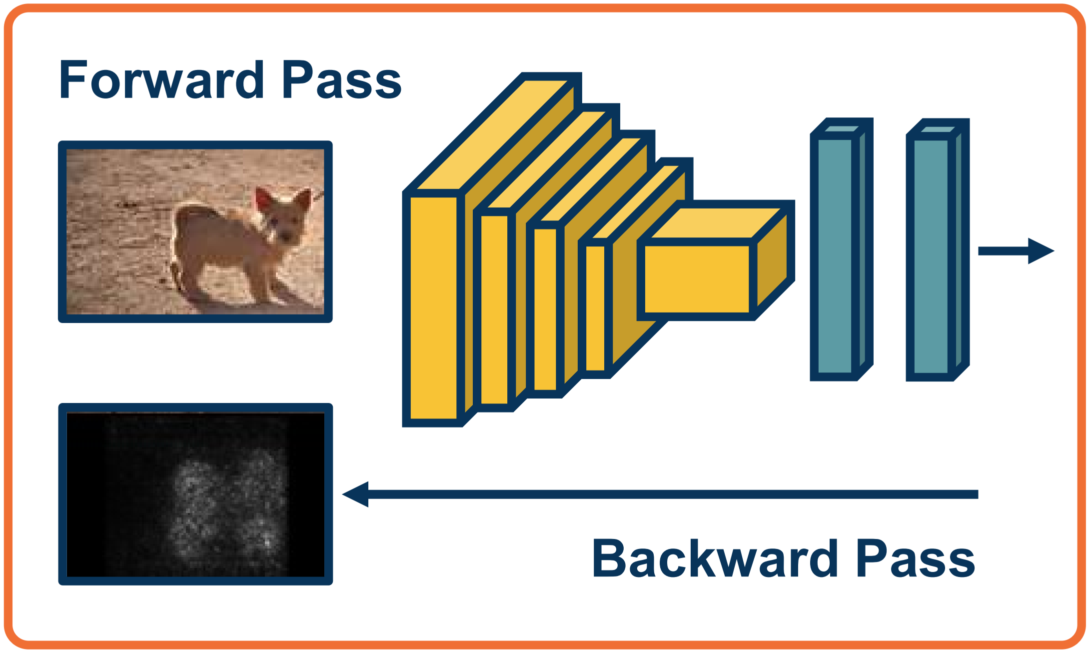
This can be **useful not just for optimization**, but also to understand what was learned



- 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:

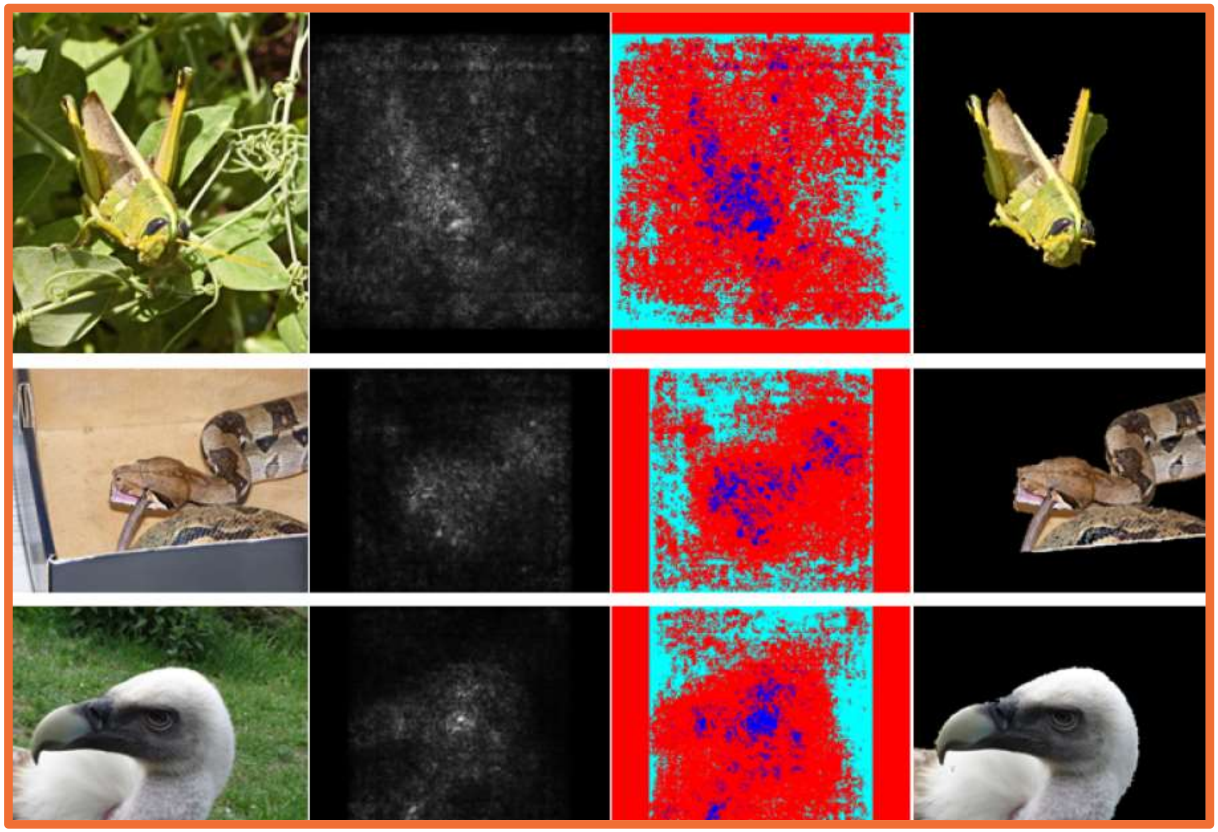
- Instead of loss, find gradient of classifier **scores** (pre-softmax)
- Take absolute value of gradient
- Sum across all channels

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

Gradient of Loss w.r.t. Image

Applying traditional (non-learned) computer vision segmentation algorithms on gradients gets us **object segmentation for free**!

Surprising because **not part of supervision**



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

Object Segmentation for Free!

Georgia Tech

Can be used to **detect dataset bias**

- E.g. snow used to misclassify as wolf

Incorrect predictions also informative

(a) Husky classified as wolf (b) Explanation

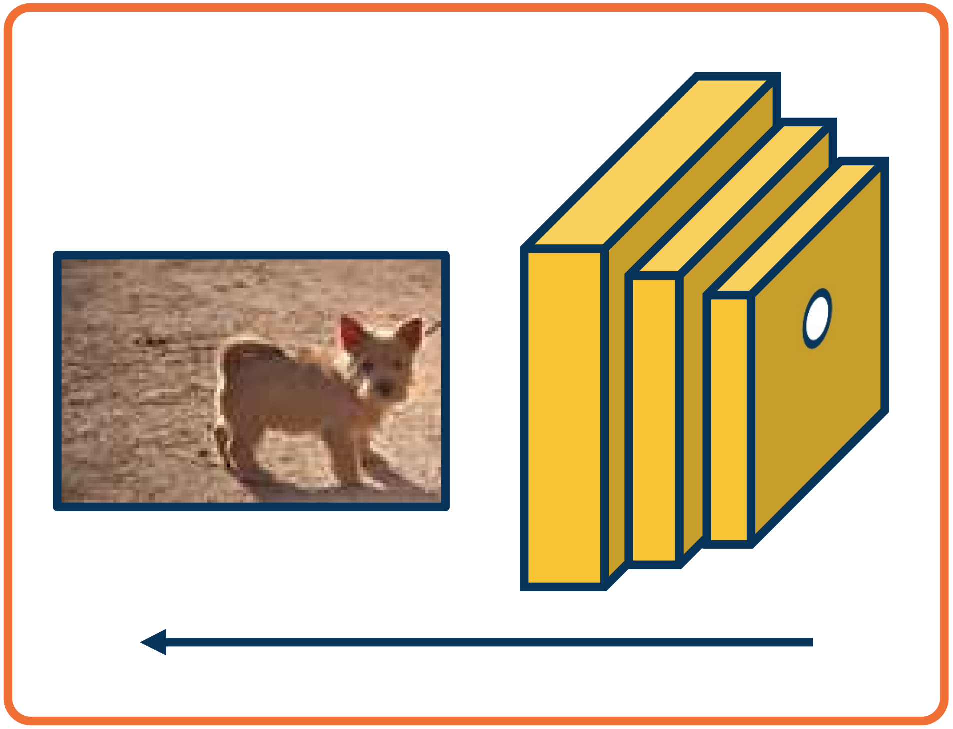
From: Ribeiro et al., "Why Should I Trust You?": Explaining the Predictions of Any Classifier

Rather than loss or scores, we can pick a neuron somewhere deep in the network and compute gradient of **activation** with respect to input

Steps:

- Pick a neuron
- Find gradient of its activation w.r.t. input image

- Can also first find highest activated image patches using its corresponding neuron (based on receptive field)



From: Ribeiro et al., "Why Should I Trust You?": Explaining the Predictions of Any Classifier

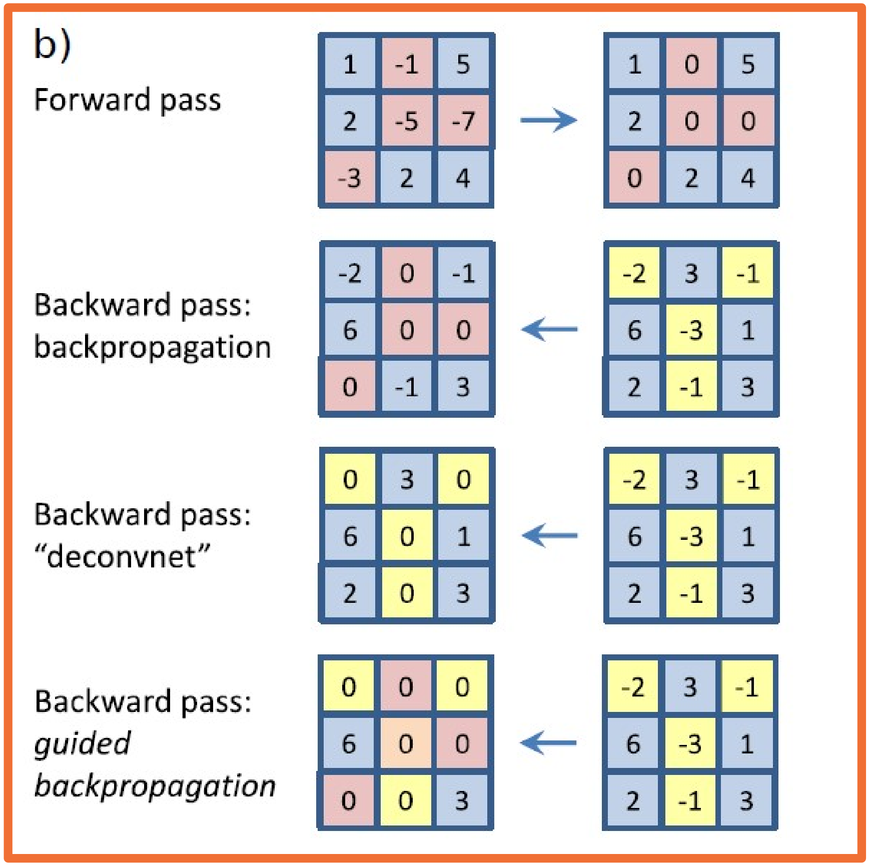
Gradient of Activation with respect to Input

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

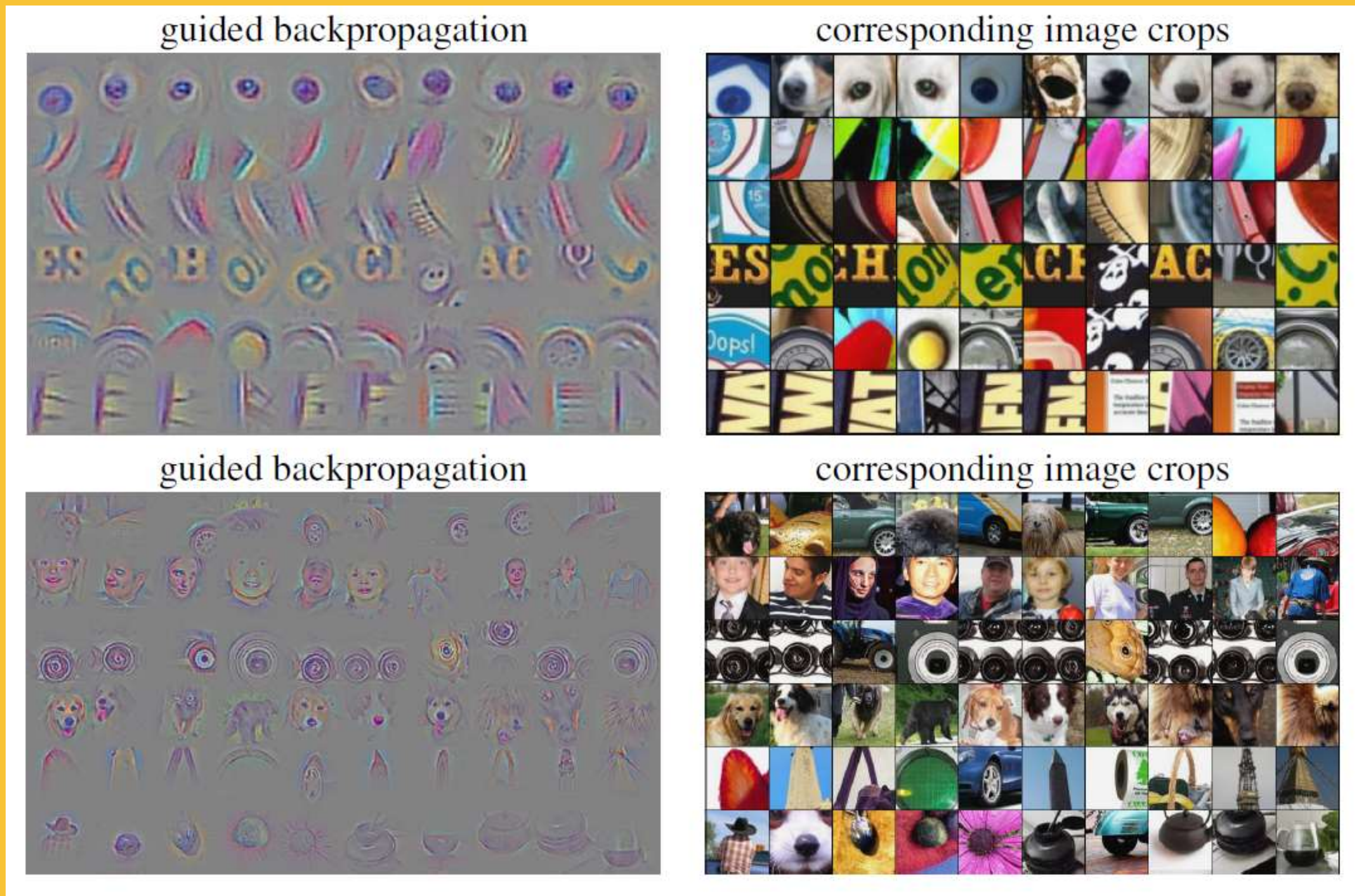
Guided backprop can be used to improve visualizations



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

Guided Backprop

Guided Backprop Results



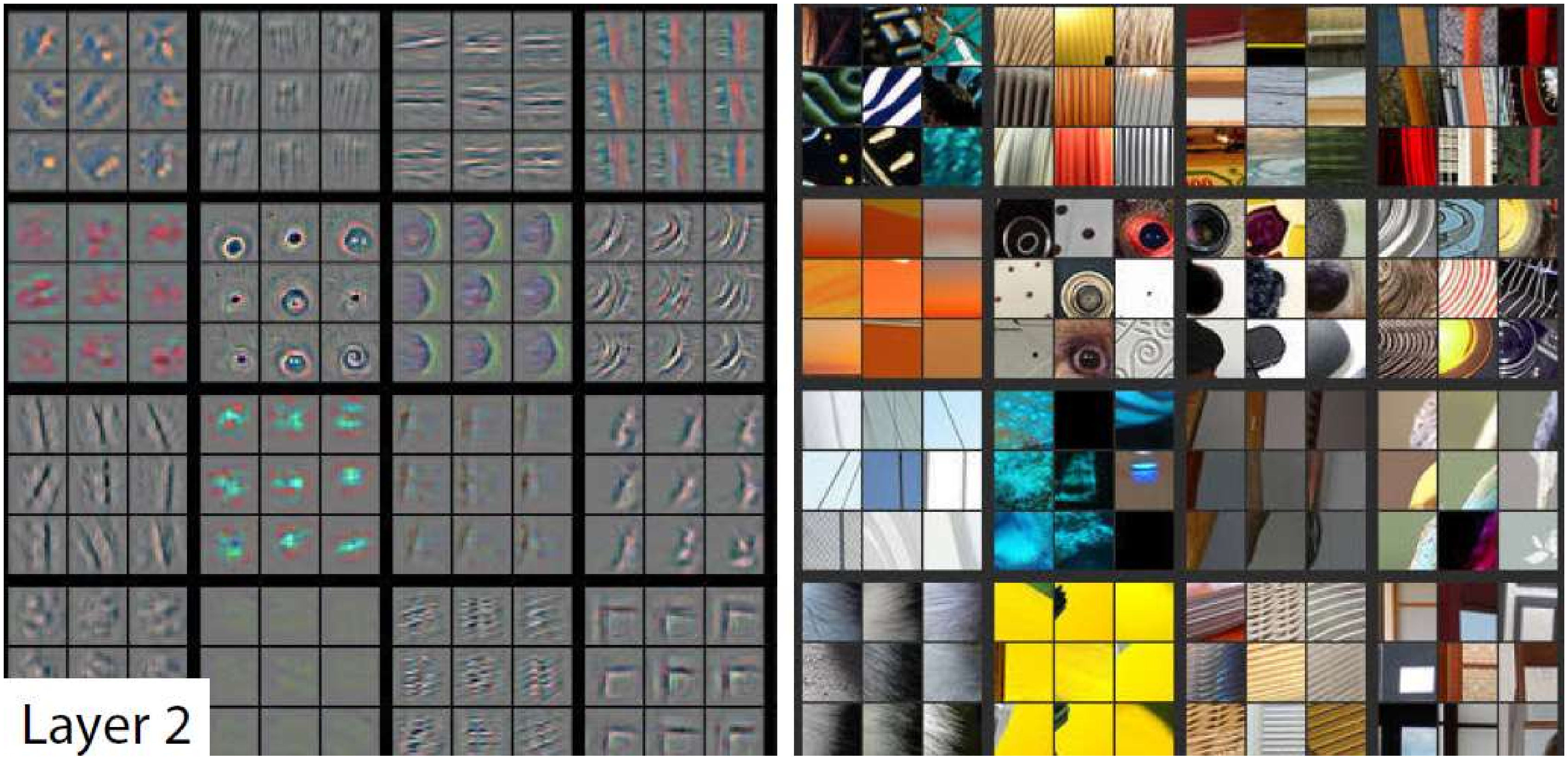
From: Springenberg et al., "Striving For Simplicity: The All Convolutional Net"

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

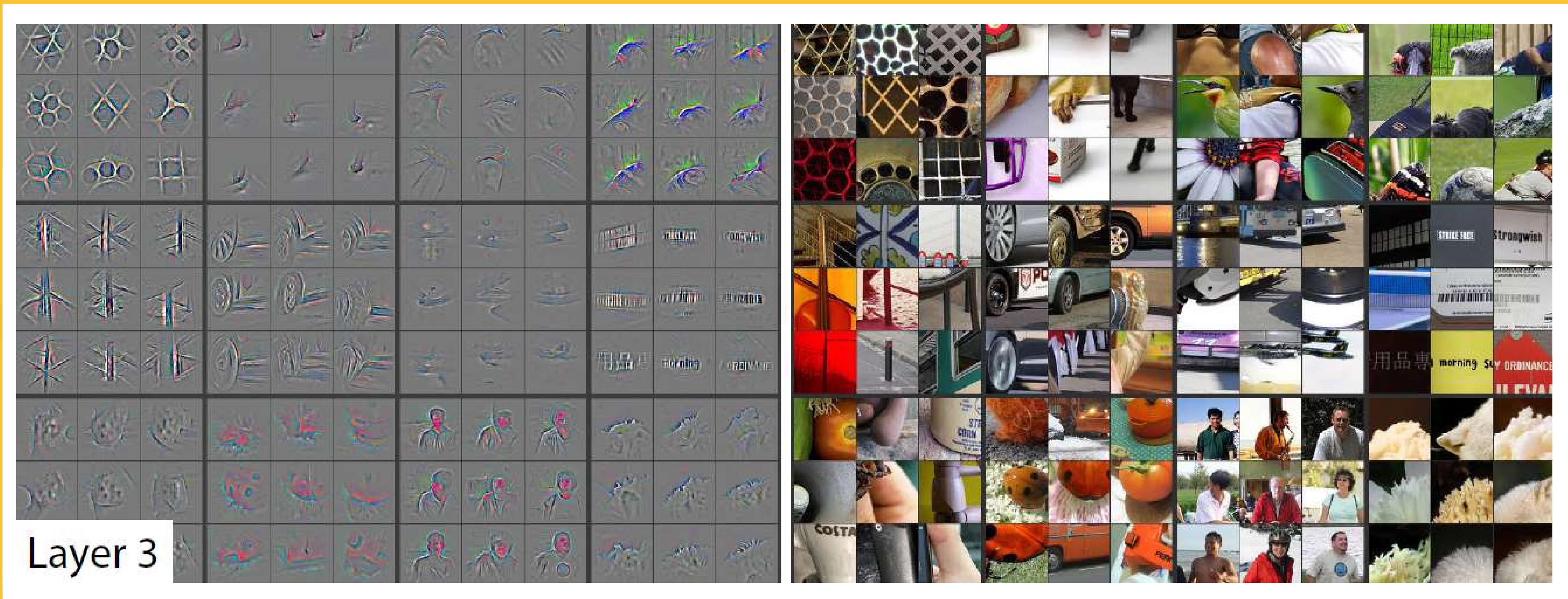
From: "Visualizing and Understanding Convolutional Networks, Zeiler & Fergus, 2014.

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.