



Convolutional Neural Networks

Machine Learning Decal

Hosted by Machine Learning at Berkeley



Agenda

Motivation

Convolutions

Convolutions in Neural Networks

Questions

Demo

Motivation

Applications of CNNs



- Computer vision
 - Face recognition
 - Scene labeling
 - Image segmentation
 - Image classification
 - Action recognition
 - Human pose estimation
 - Playing Games (Atari games, Pacman)
 - Medical image analysis



Image Segmentation

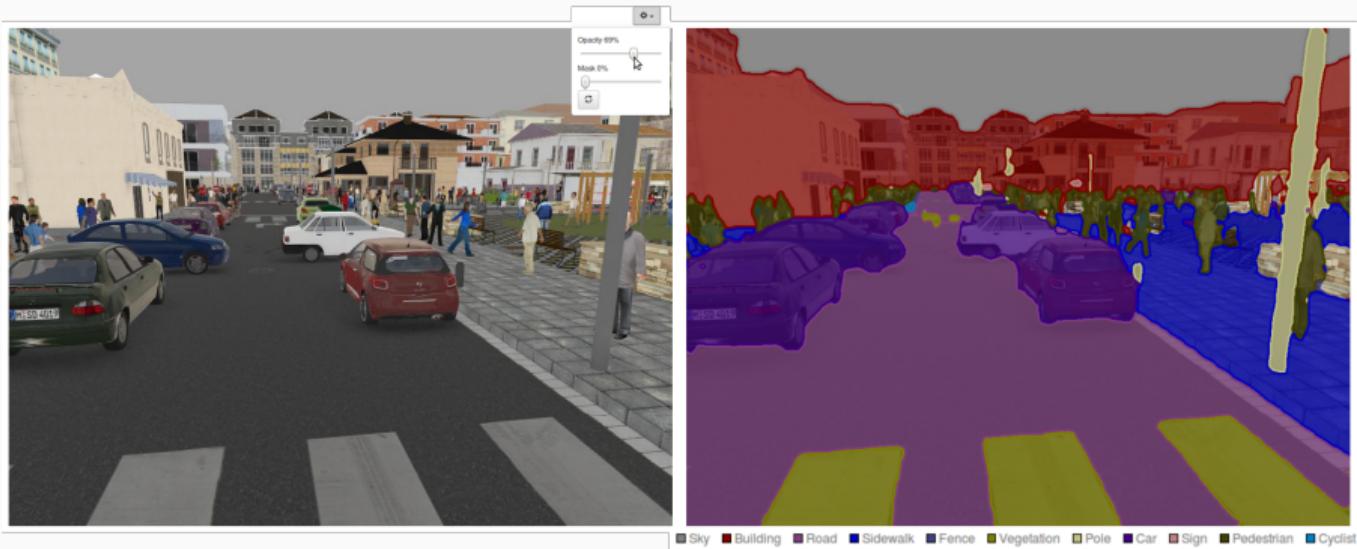


Image Generation

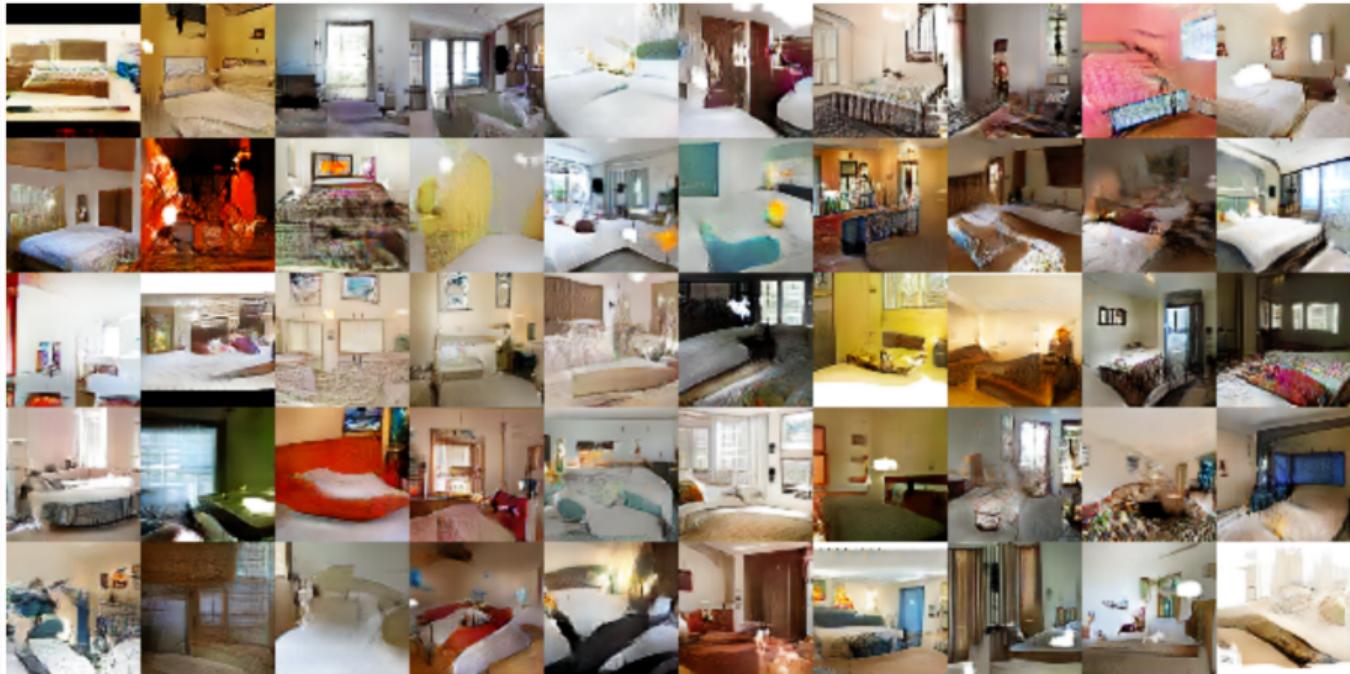


Image Style Transfer

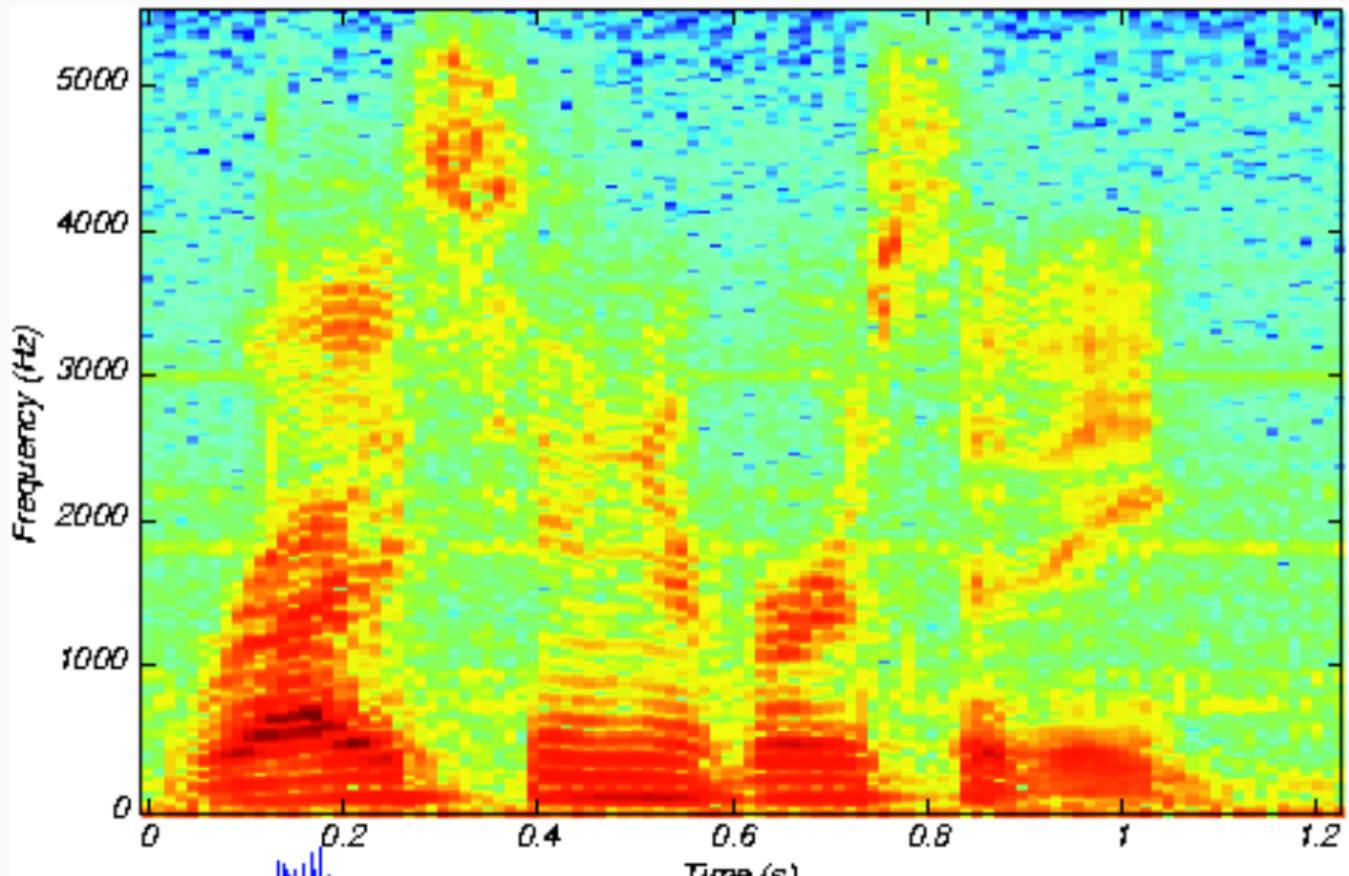


Another Style Transfer Example



Cycle GAN: <https://github.com/junyanz/CycleGAN>

Audio Processing

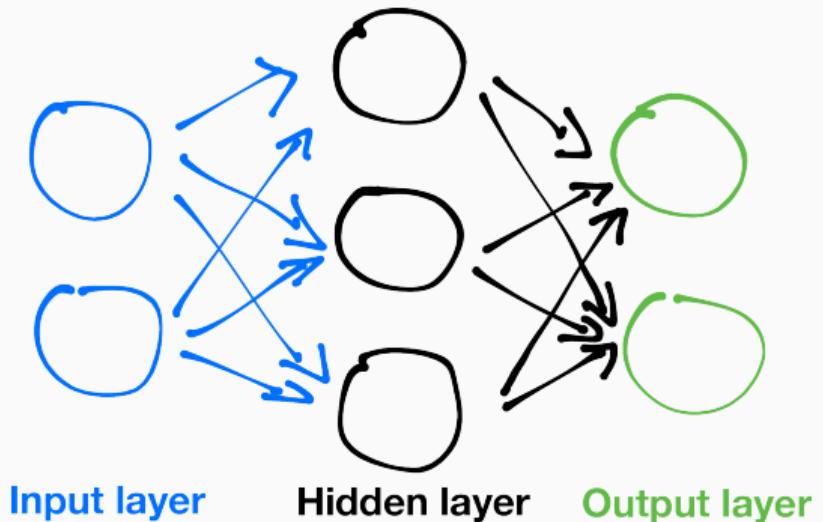


Industry use of CNNs?



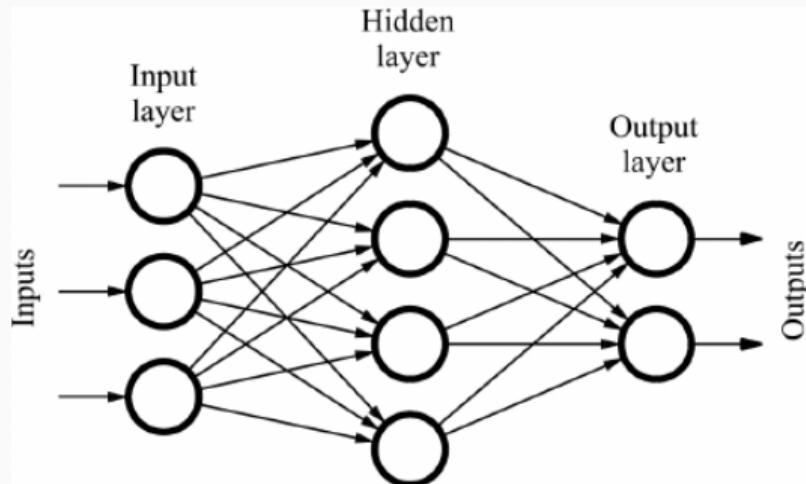
Genentech

Review: Feed Forward Neural Network



- Layered Architecture
- Three types of layers:
 - **Input Layer** - Data is passed into these neurons
 - **Hidden Layer** - These neurons are "hidden from view"
 - **Output Layer** - These neurons output the result of the network

What's wrong with feed-forward NNs?



The curse of dimensionality.

Problem: high-dimensional image data



- If we want a neural network to take as input 256×256 images, the input layer will have 65,536 neurons
- ... if we have one hidden layers with 1,000 neurons, we're already at 65 million parameters
- ... if we have two hidden layers with 15,000 neurons each, we're at over a **billion** parameters

Solution: Convolutional Neural Networks



Basic Idea: Share parameters for different parts of input.

Convolutions

Convolution Parameters



- Kernel Size (number of people that are moving)
- Padding Type (edge behavior)
- Stride (how much the kernel moves at each step)

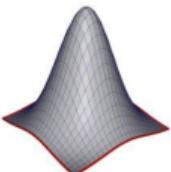
Convolutions as filters



*

1	2	1
2	4	2
1	2	1

=

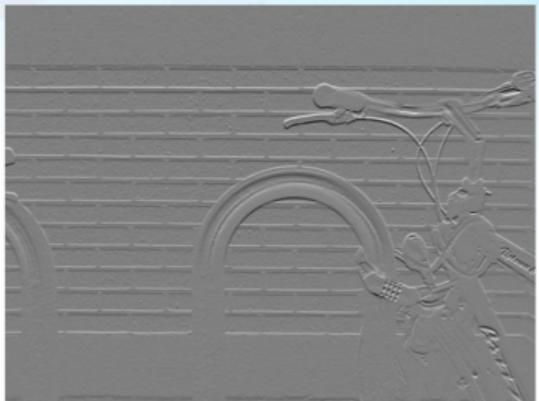




Convolutions as filters cont.

 $*$

1	2	1
0	0	0
-1	-2	-1

 $=$ 

Convolutions in Neural Networks

Convolution Neural Networks

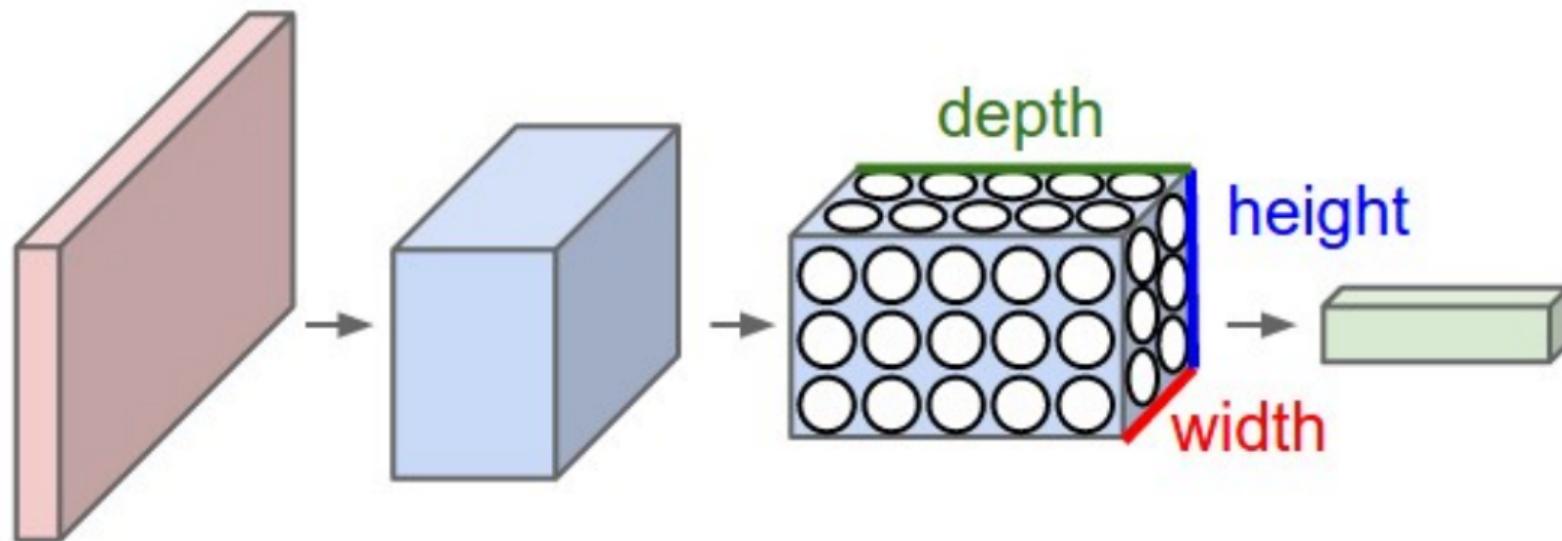


CNN = Learned convolutional filters.



What does our input look like?

Let's say we have a 32x32 color image. There are then 3 channels for the colors (RGB). So our input is a 3D volume of size 32x32x3



What do our filters look like?



3D filters !!!

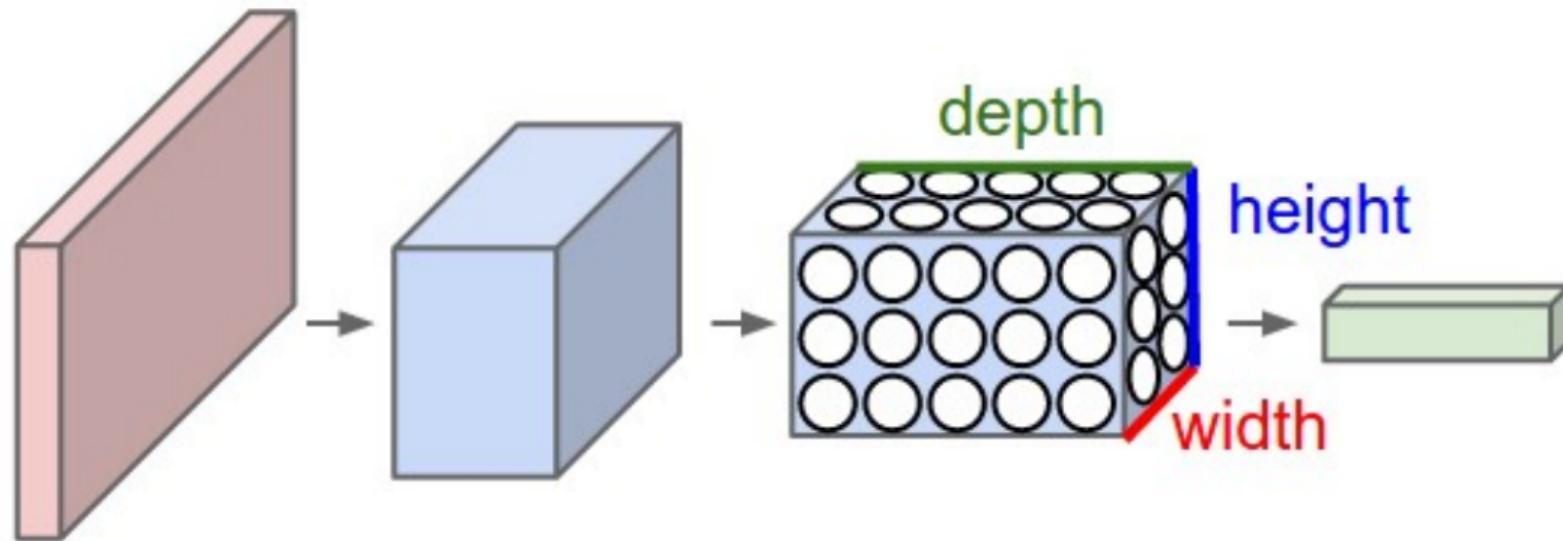
Our filters will then be $n \times n \times 3$, where n is a hyperparameter.

We can also have multiple filters per layer.

<https://i.stack.imgur.com/FjvuN.gif>

What will the output of a layer look like?

Our output will also be 3D of shape $(32 \times 32 \times m)$ where m is the number of filters in the layer.





How do we update filters?

X_{11}	X_{12}	X_{13}
X_{21}	X_{22}	X_{23}
X_{31}	X_{32}	X_{33}

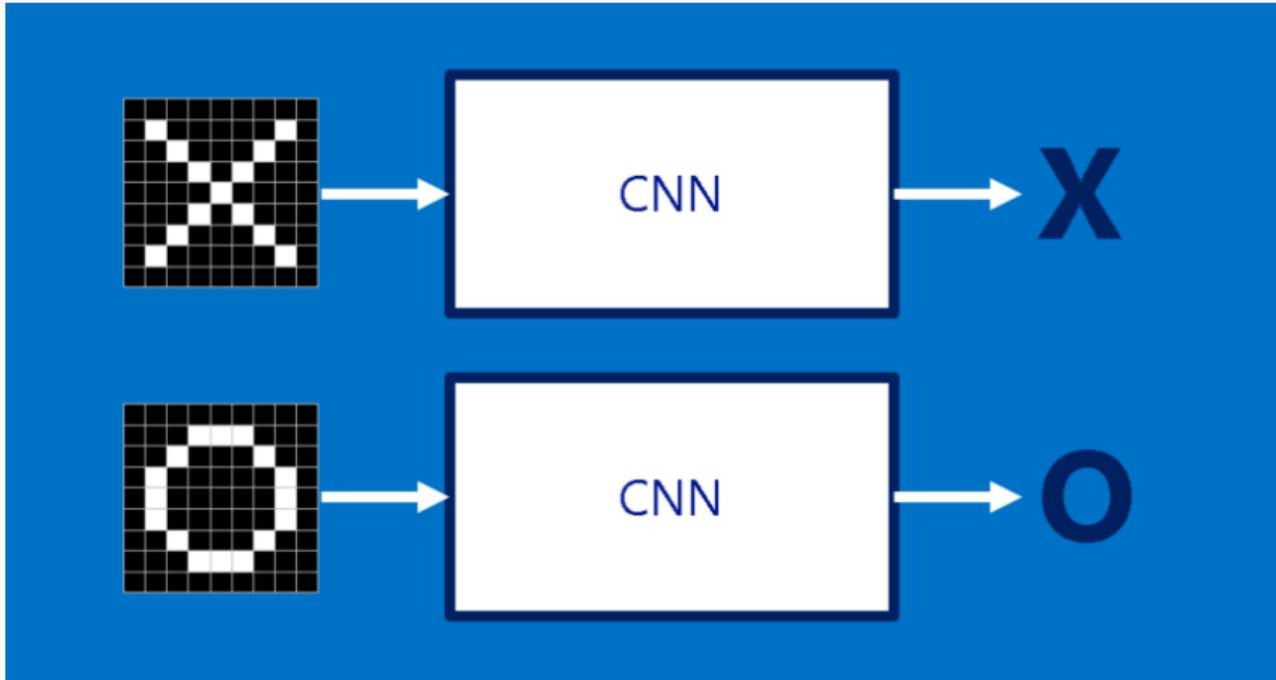
$X_{11} \partial h_{11}+$ $X_{12} \partial h_{12}+$ $X_{21} \partial h_{21}+$ $X_{22} \partial h_{22}$	$X_{12} \partial h_{11}+$ $X_{13} \partial h_{12}+$ $X_{22} \partial h_{21}+$ $X_{23} \partial h_{22}$
$X_{21} \partial h_{11}+$ $X_{22} \partial h_{12}+$ $X_{31} \partial h_{21}+$ $X_{32} \partial h_{22}$	$X_{22} \partial h_{11}+$ $X_{23} \partial h_{12}+$ $X_{32} \partial h_{21}+$ $X_{33} \partial h_{22}$

∂h_{11}	∂h_{12}
∂h_{21}	∂h_{22}

Derivative Computation (Backward pass) since pictures speak more than words



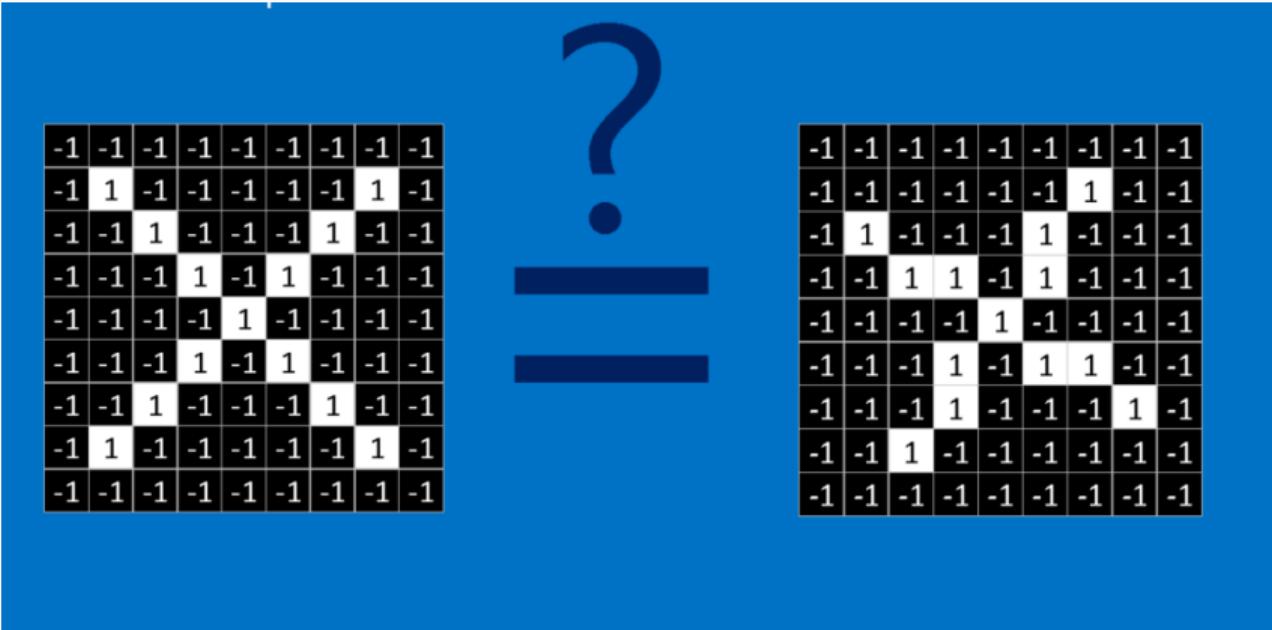
Let's look at a toy example!



- How would a CNN classify X's vs O's?



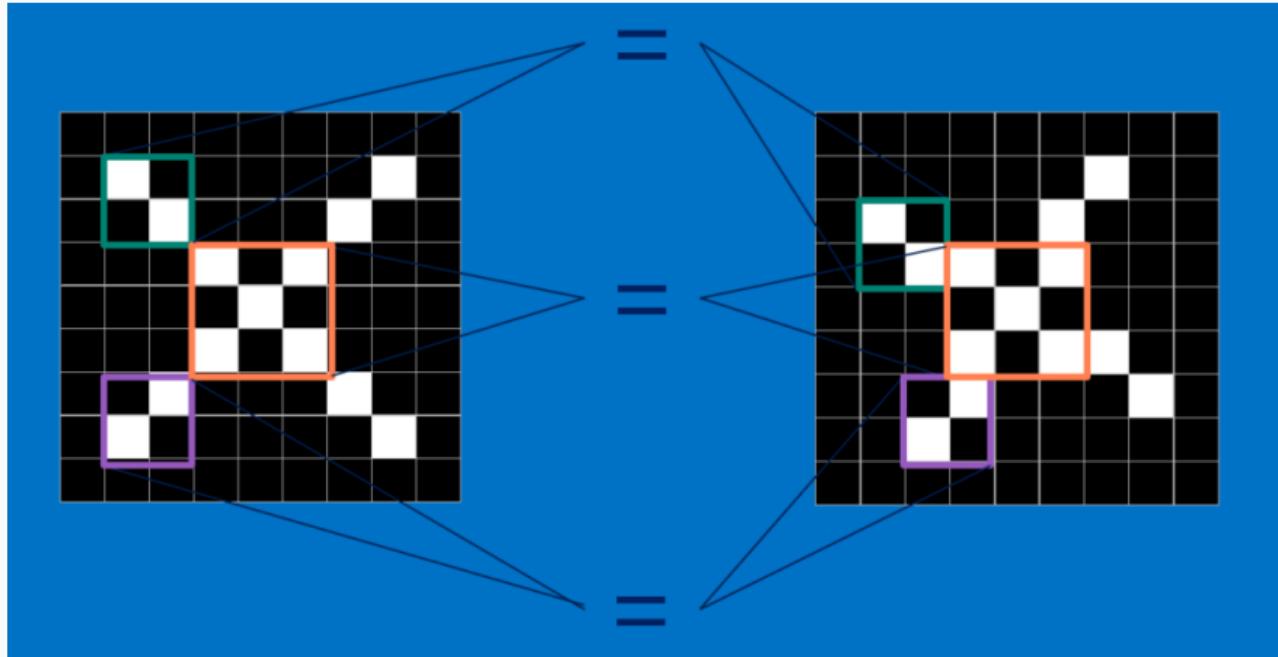
Would you classify this as an X?



- A computer can't easily distinguish this example as an X.. so what can we do?

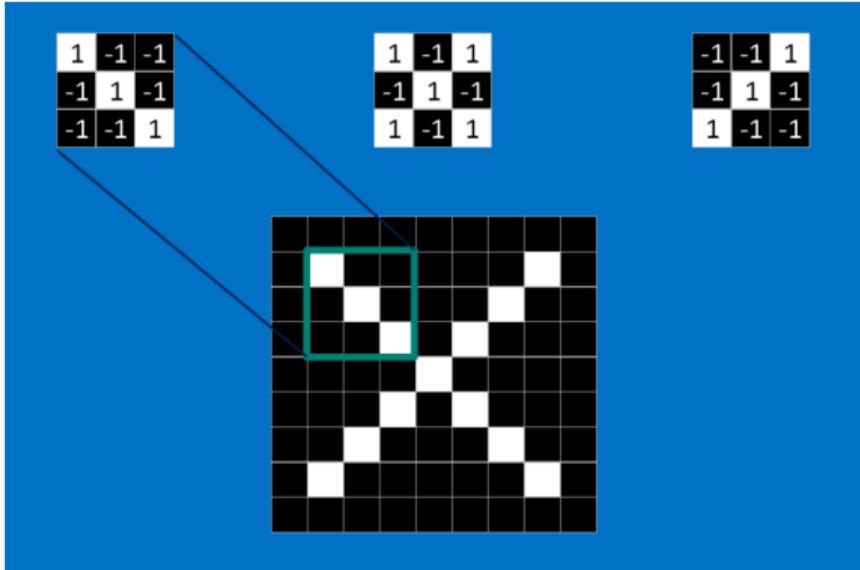


What is a feature?



- We can break down the image into smaller images and look for patterns!
- Can we see here the similar patterns?
- These diagonals and crosses can be thought of as features!

Convolutional Filters



- Let's break this up into smaller problems... what about smaller patterns!



CNN Layers: Convolutions

1	-1	-1
-1	1	-1
-1	-1	1

$$1 \times 1 = 1$$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

1	1	1
1	1	1
1	1	1

CNN Layers: Convolutions



-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1



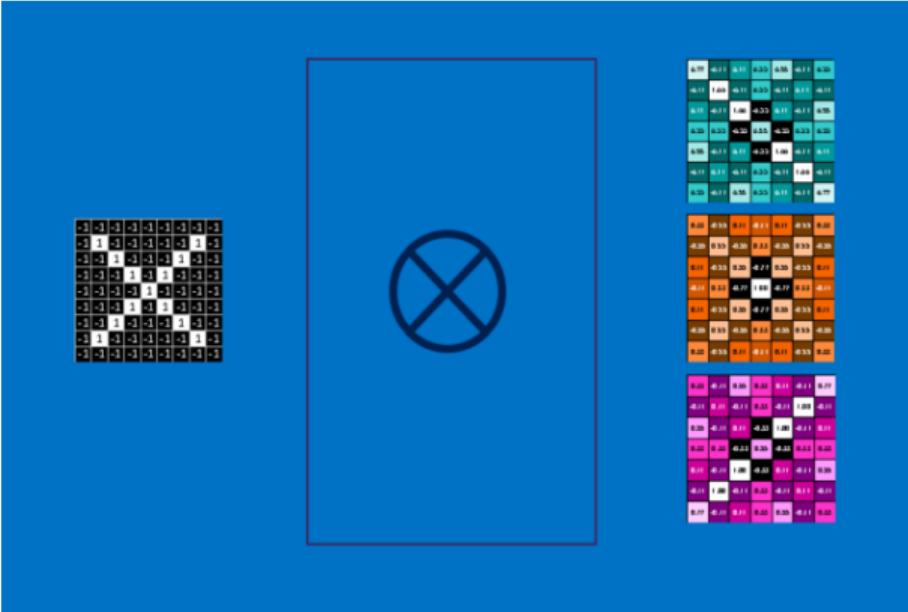
1	-1	-1
-1	1	-1
-1	-1	1

=

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

- Now lets apply this process to every part of the board.
- We can look at the new board and call it a filtered version of this specific feature
- Shows us where on the image the feature matches the most

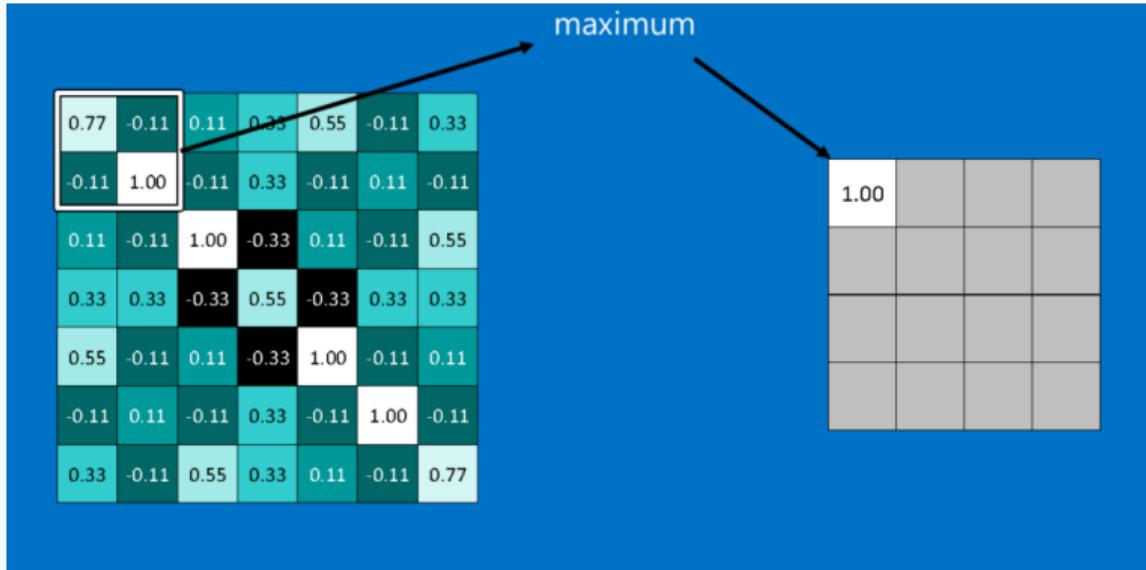
CNN Layers: Convolutions



- Now lets apply this process to every part of the image for different features.

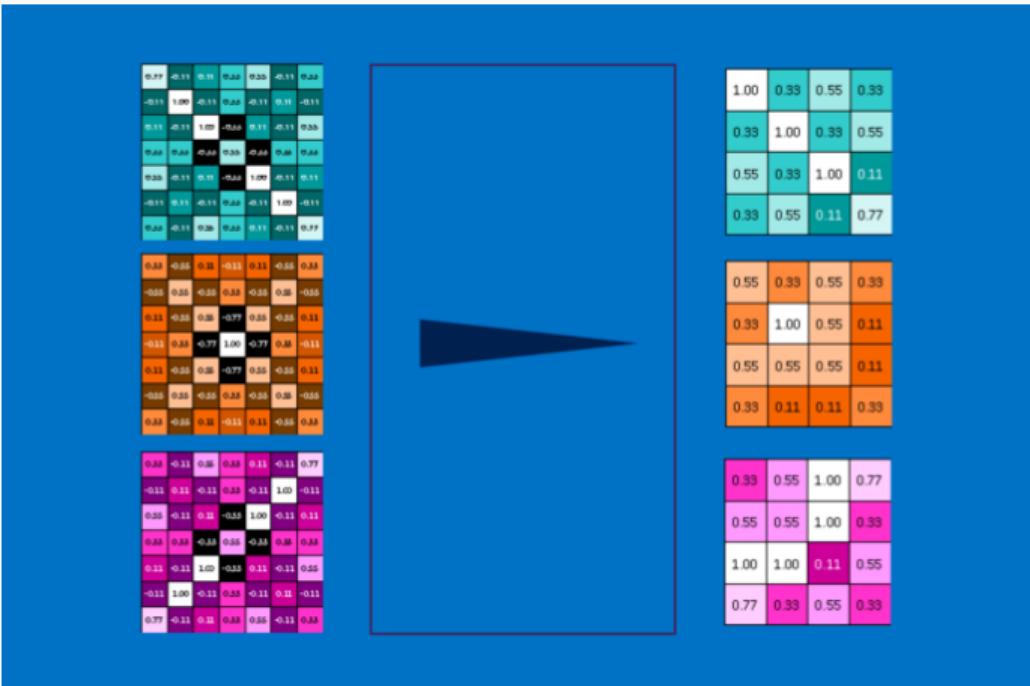


CNN Layers: Pooling



- If CNN's are really deep, doing that math for every possible pixel gets to be quite computationally expensive
- That's why we have pooling layers, that down sample our image!

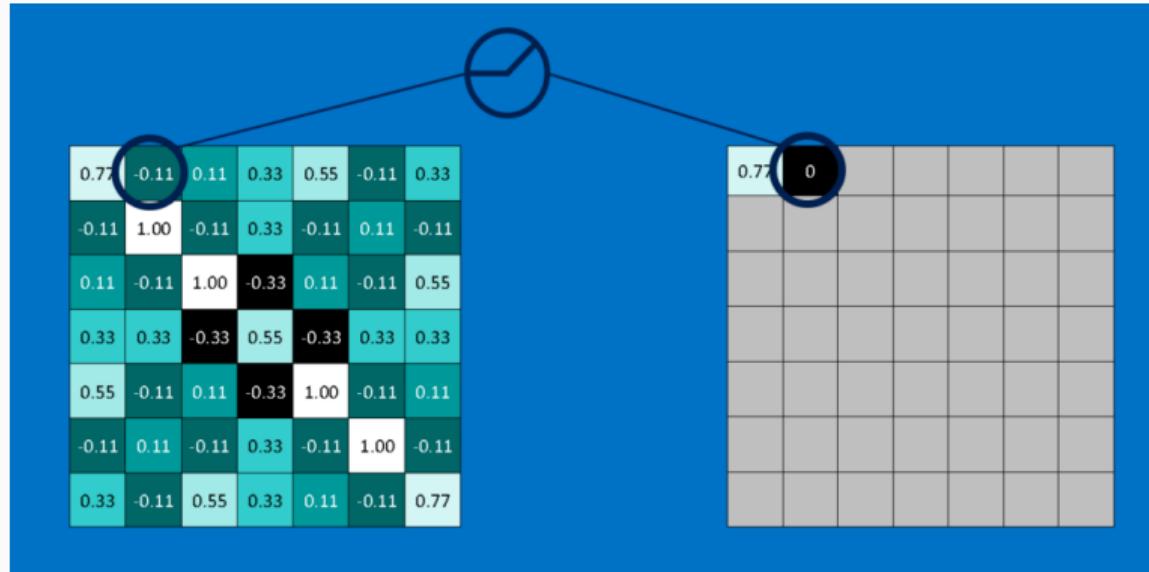
CNN Architecture



- By applying these pooling layers, we try to capture the patterns and get rid of extra unneeded information.



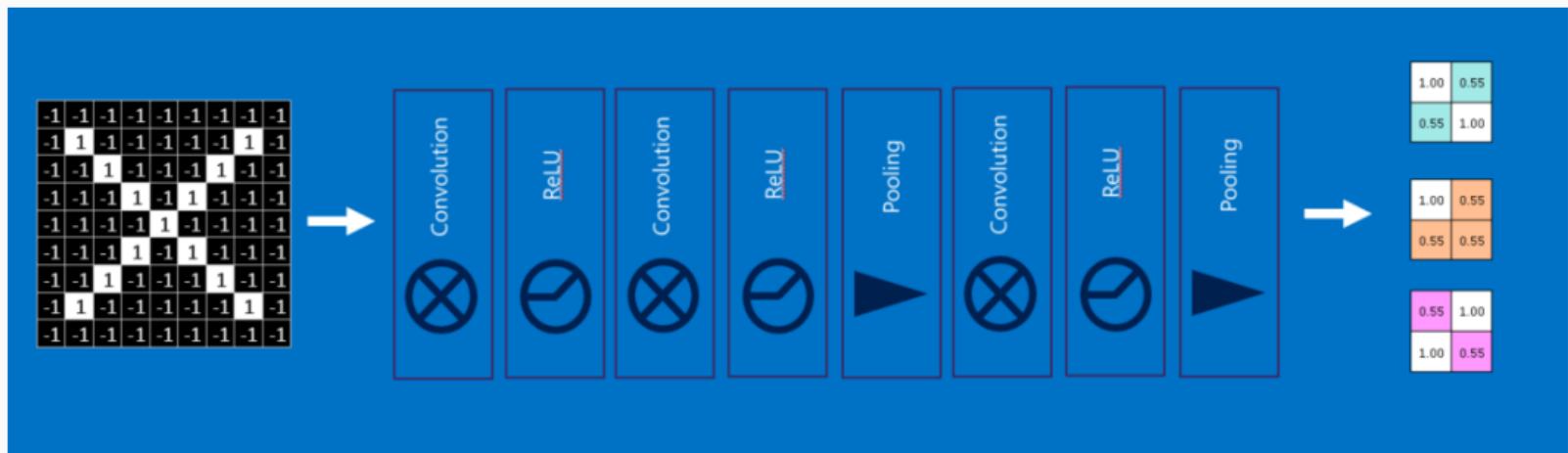
CNN Layers: Activation



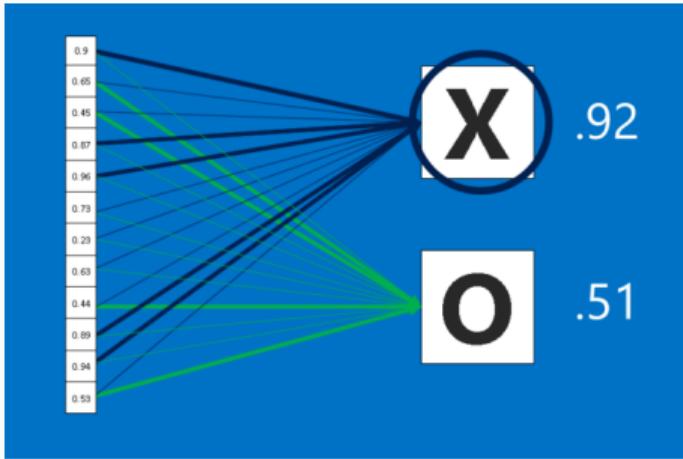
- Remember those activation functions you learned from the last lecture? We use them here too!
- Applying RELU we simply get rid of all the negatives!



Let's put it all together!



CNN: Fully Connected Layer

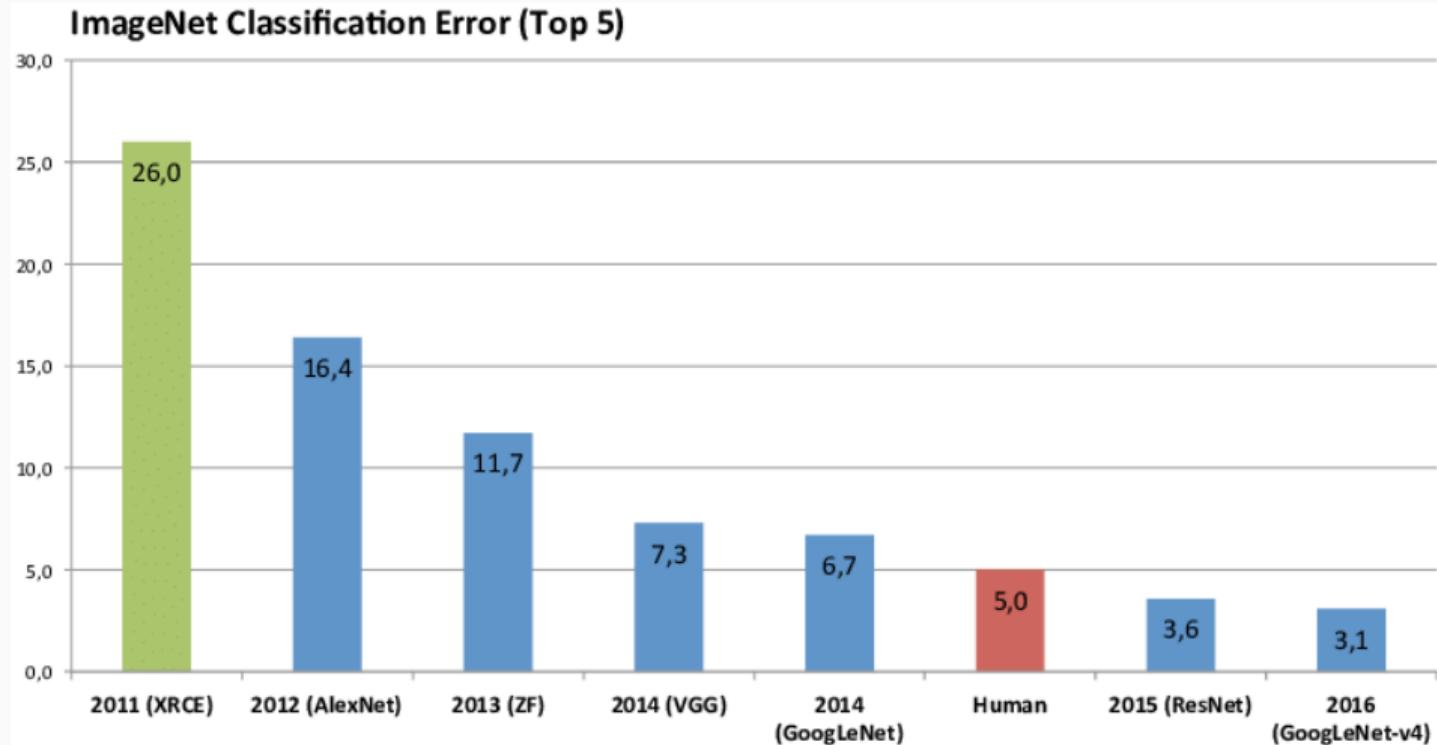


- At the end we use a fully connected layer to hold a vote if the image is either an X or O
- Each value has a weight though, so not all values contribute equally
- This goes back to the ideas you learned last week!





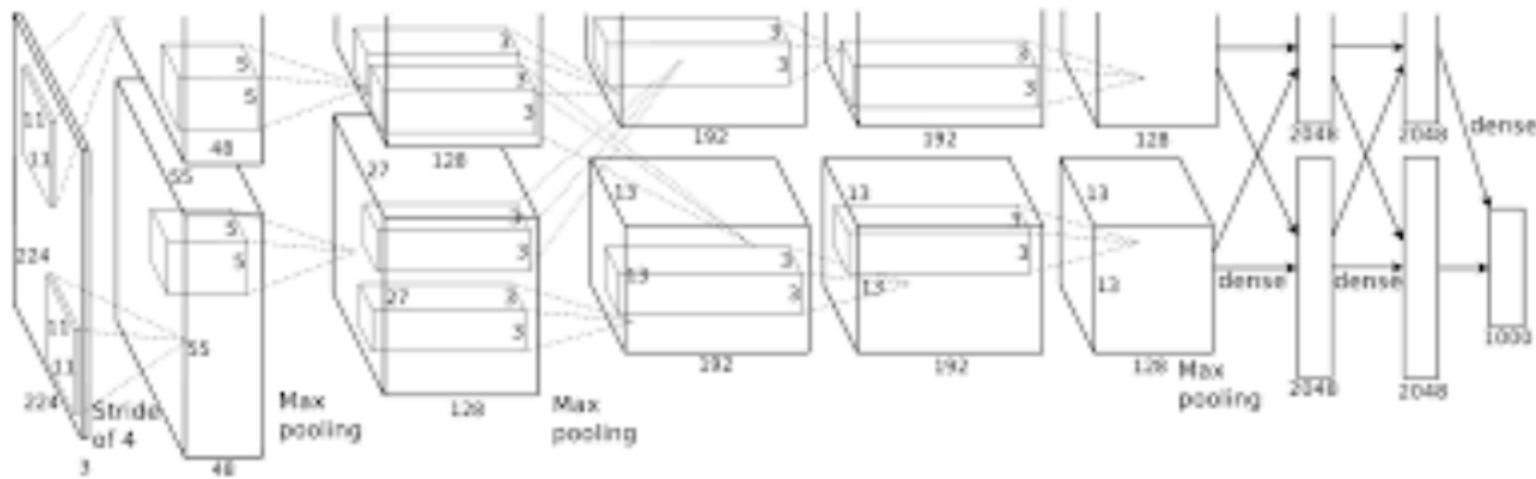
Imagenet: Sparked Beginning of DL Image Renaissance



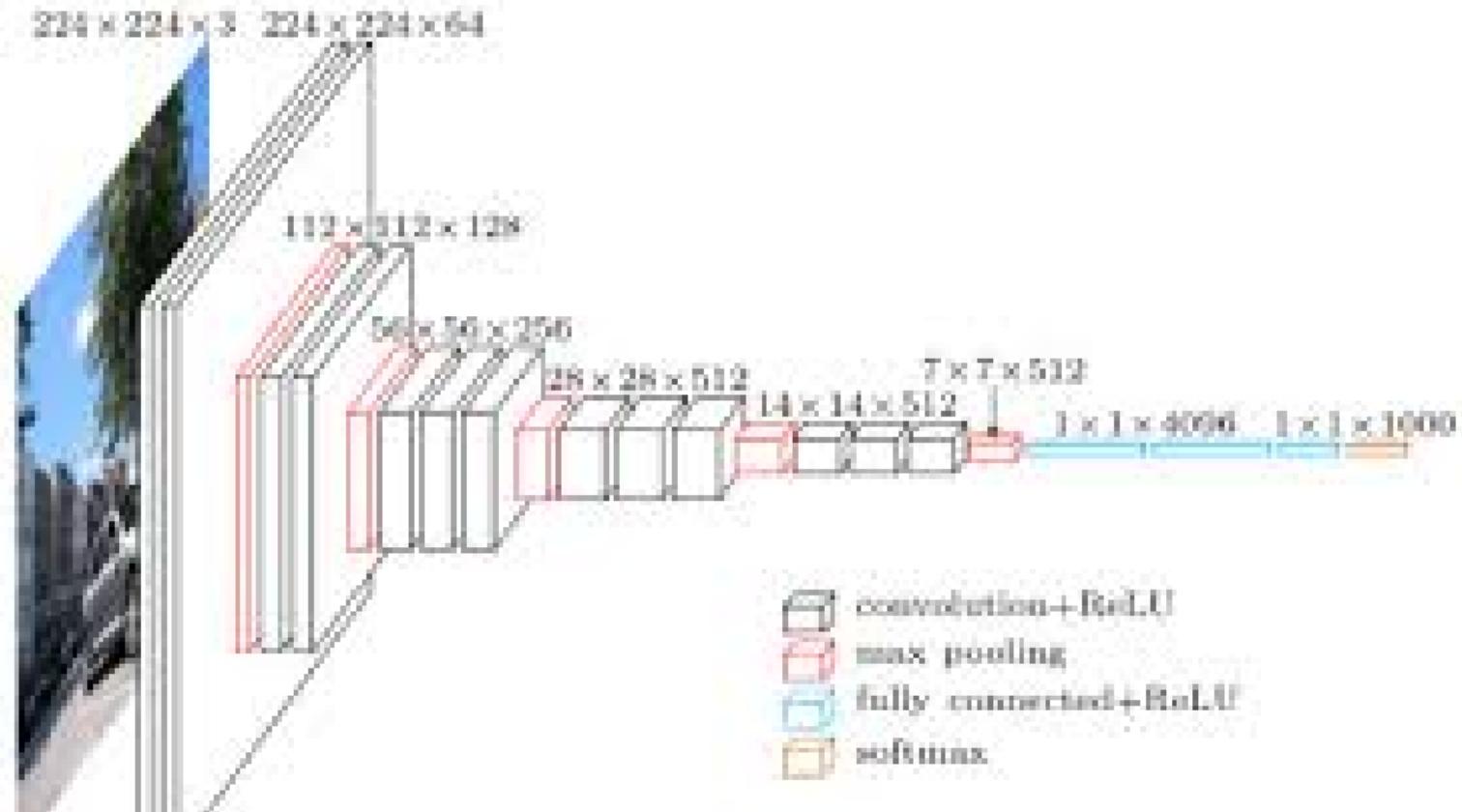


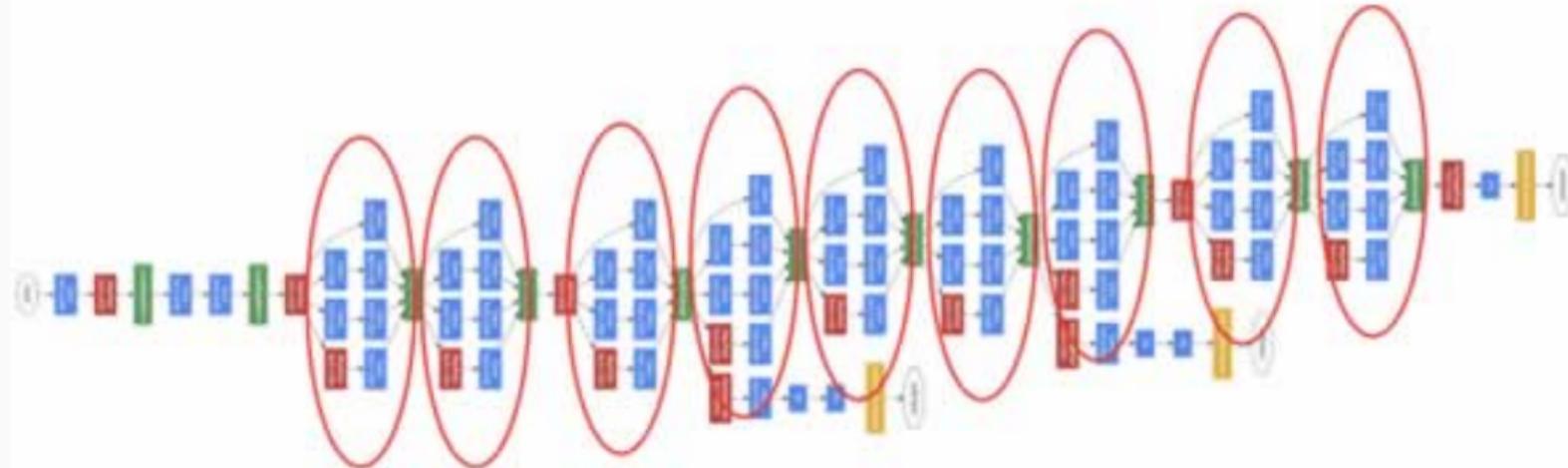
- "The 9 Deep Learning Papers You Need to Know About"
(<https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html>)
- Many of these models are used as benchmarks in deep learning
- There exist many pre-trained versions of the above models that can be leveraged for a new task (this is called **transfer learning**)
- Let's look at a few...

AlexNet (2012)



VGG-16 (2014)





9 Inception modules

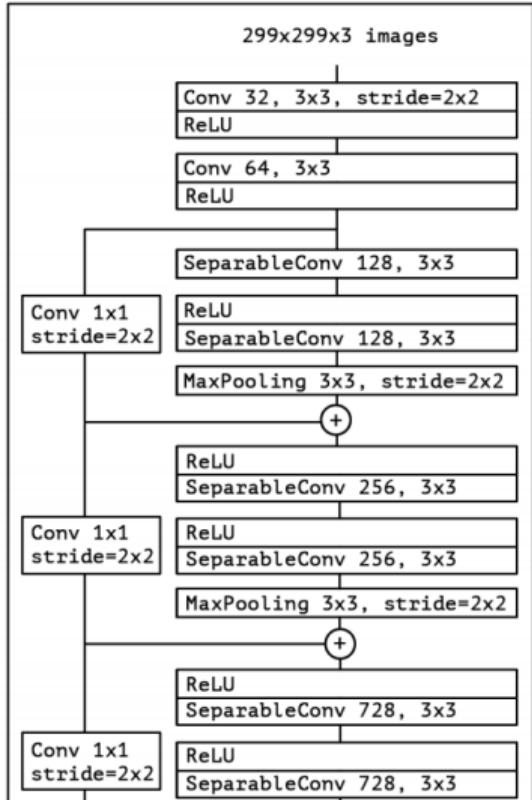
Network in a network in a network...

Convolution
Pooling
Softmax
Other

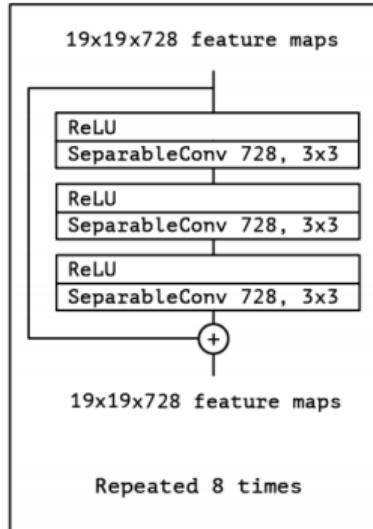
Xception (2016)



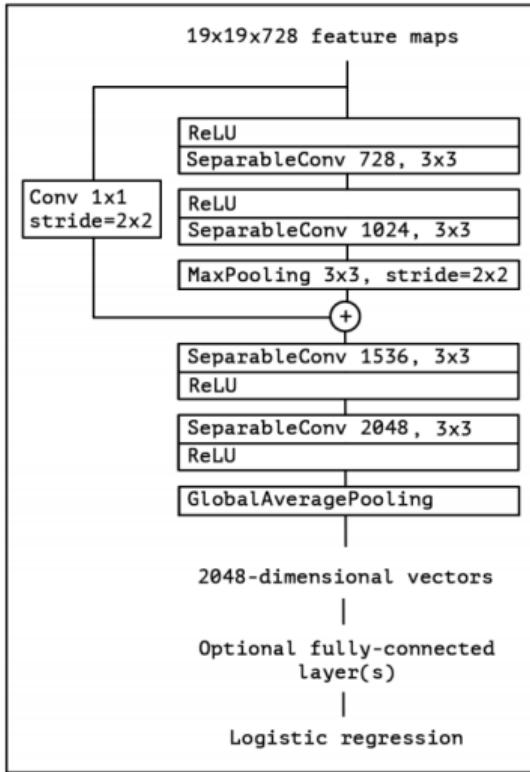
Entry flow



Middle flow



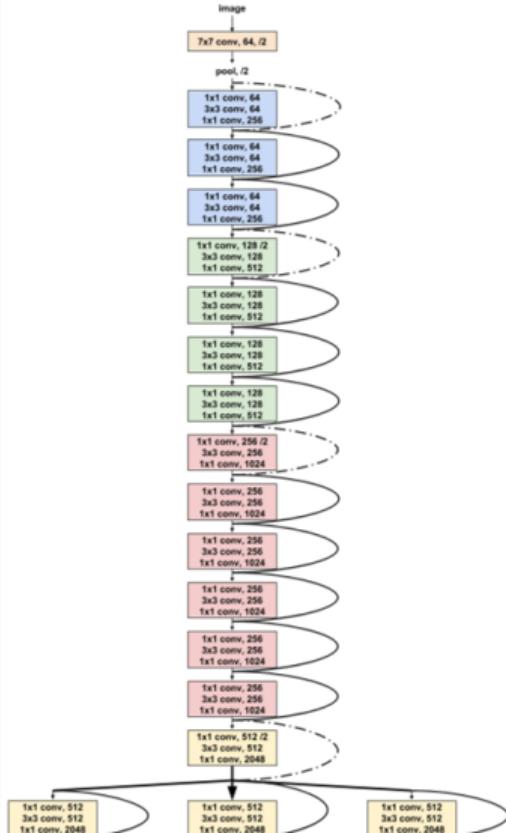
Exit flow



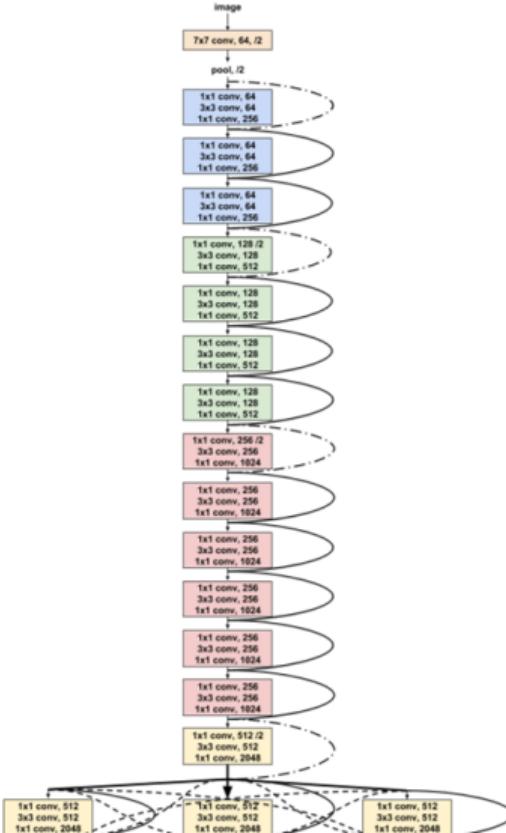
ResNet-50 (2015)



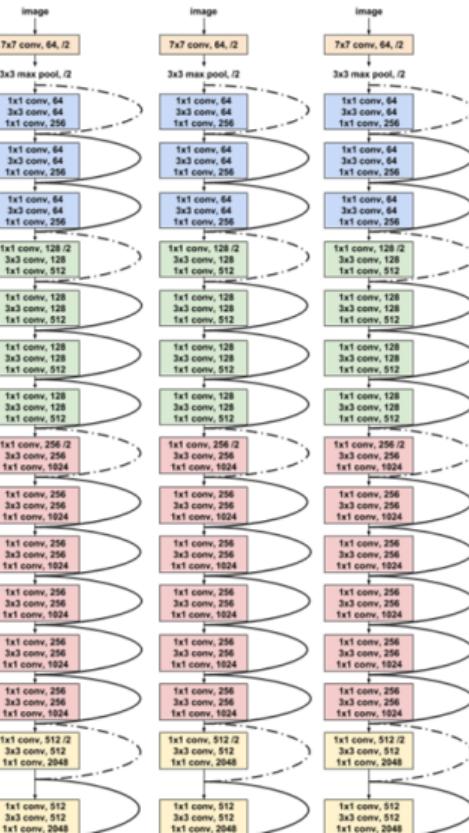
X₀-ResNet-50 (no cross-residuals)



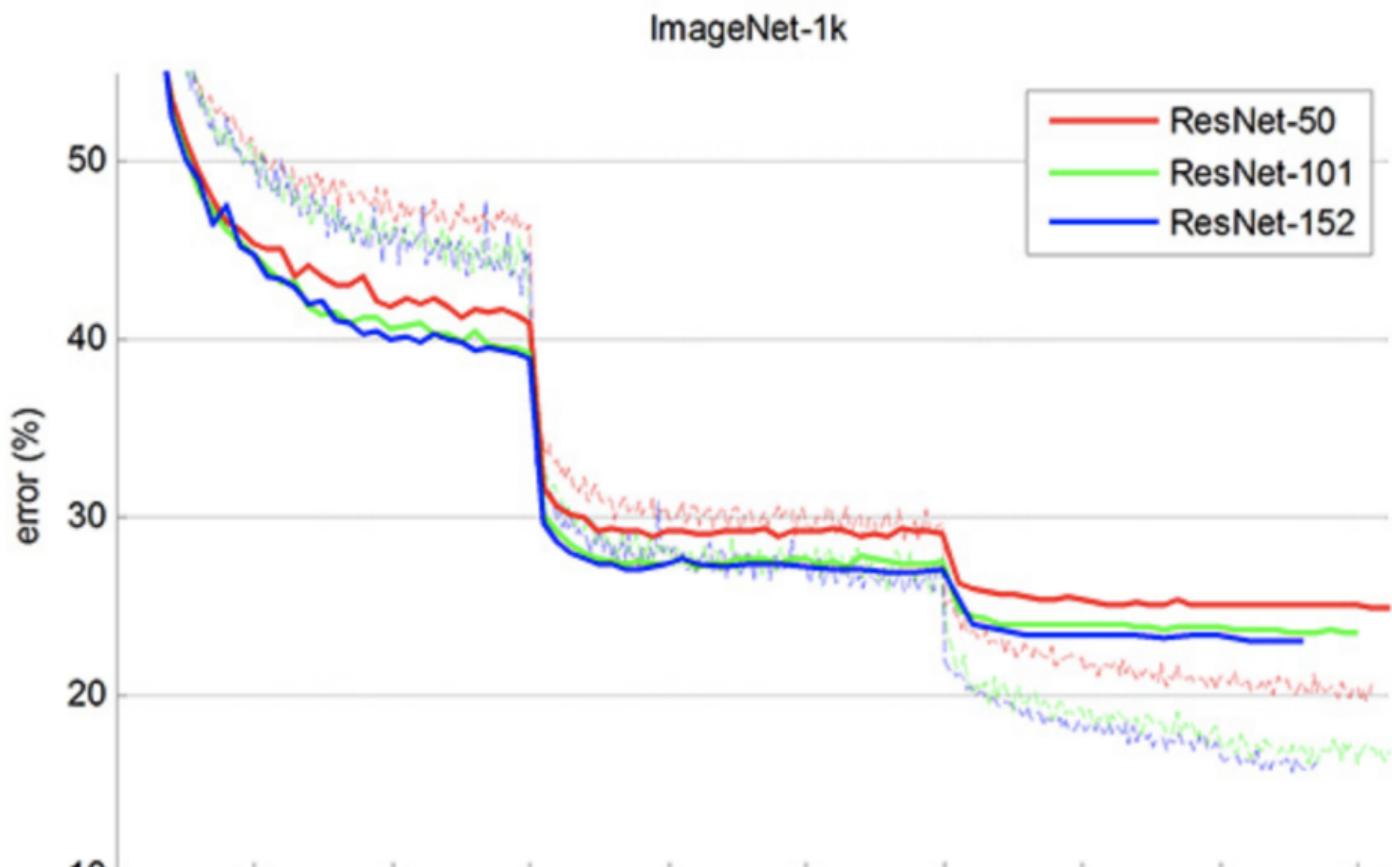
X-ResNet-50 (with cross-residuals)



ResNet-50 (x3)



ResNet-152 (2015)



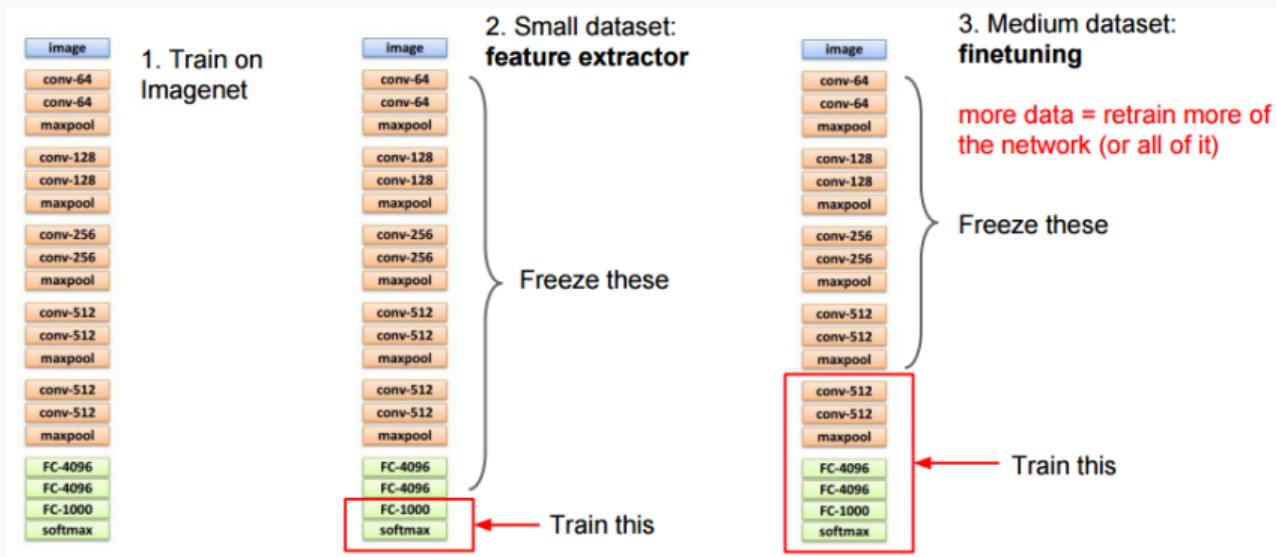
Transfer Learning



- In practice, we don't train an entire ConvNet from scratch
- One can pretrain the CNN on a large dataset like ImageNet and then finetune the ConvNet so that it does well on our dataset.
- There are several scenarios when we might want to use transfer learning.
- Let's look at a few...



How Can We Transfer Learn?



Why does it work?



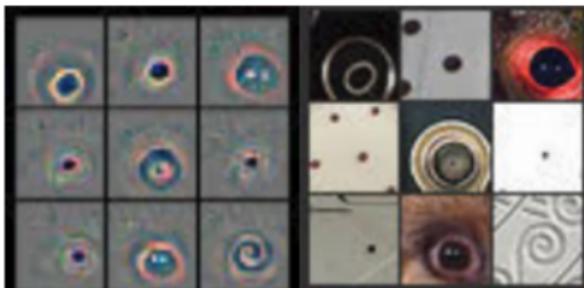
Early layers learn simple features



Why does it work?



Later layers learn more abstract features



Layer 2 of AlexNet



Layer 4 of AlexNet

Questions

Questions?

Demo

CNN demo: CIFAR-10