Convolutional Neural Networks

Machine Learning Decal

Hosted by Machine Learning at Berkeley

ML@B

Agenda

Motivation

Convolutions

Convolutions in Neural Networks

Demos

Questions

Motivation

Applications of CNNs

ML@B

- 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

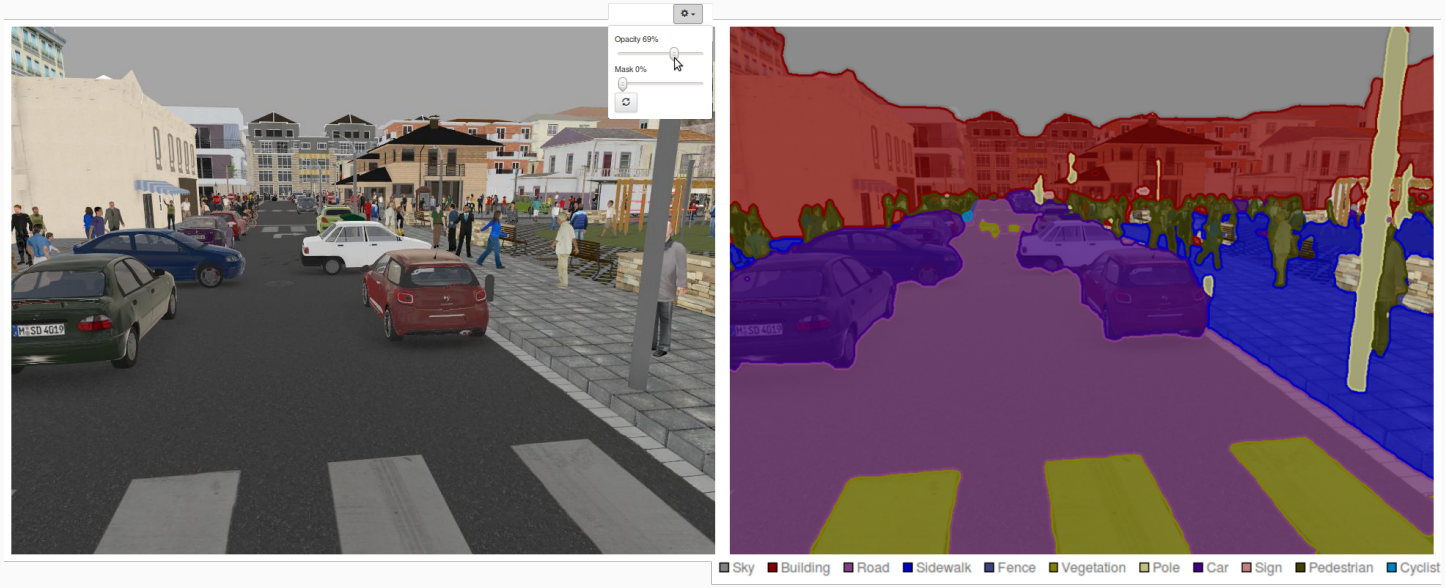


Image Generation

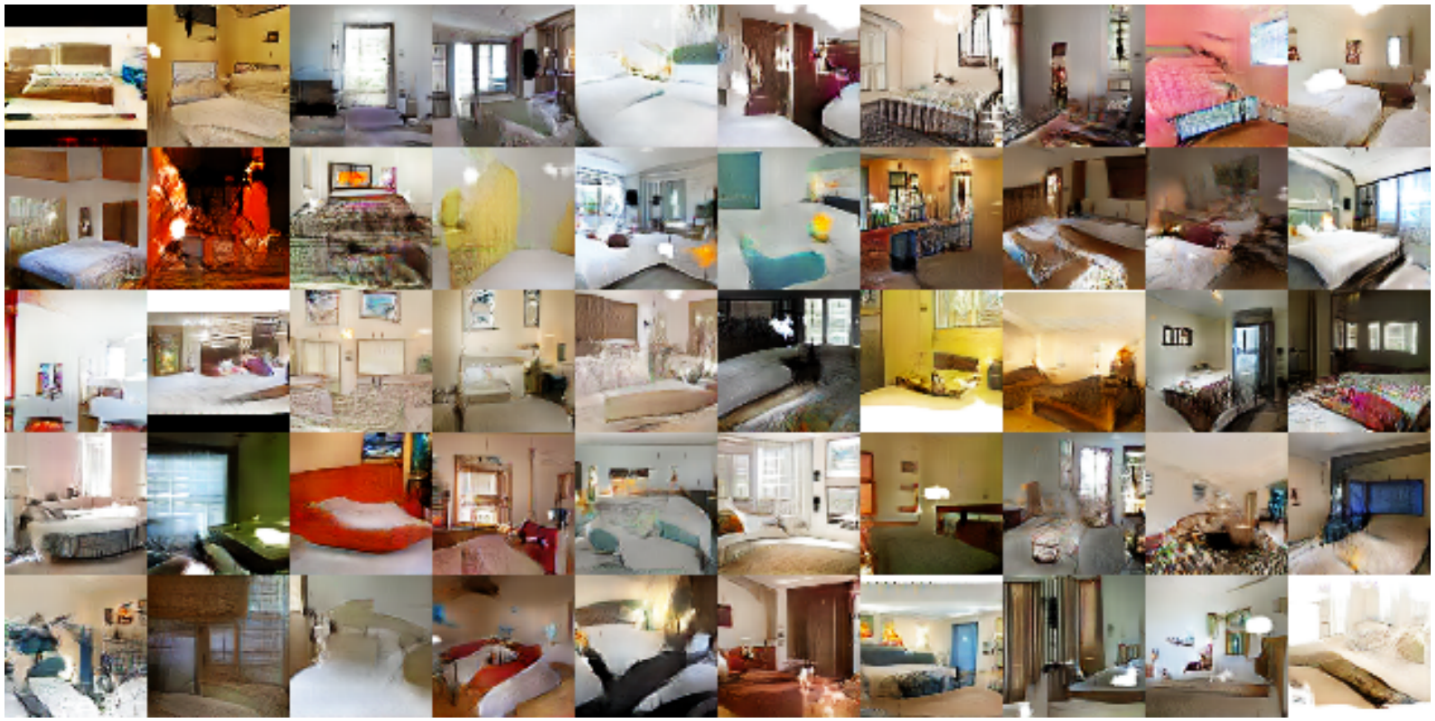


Image Style Transfer

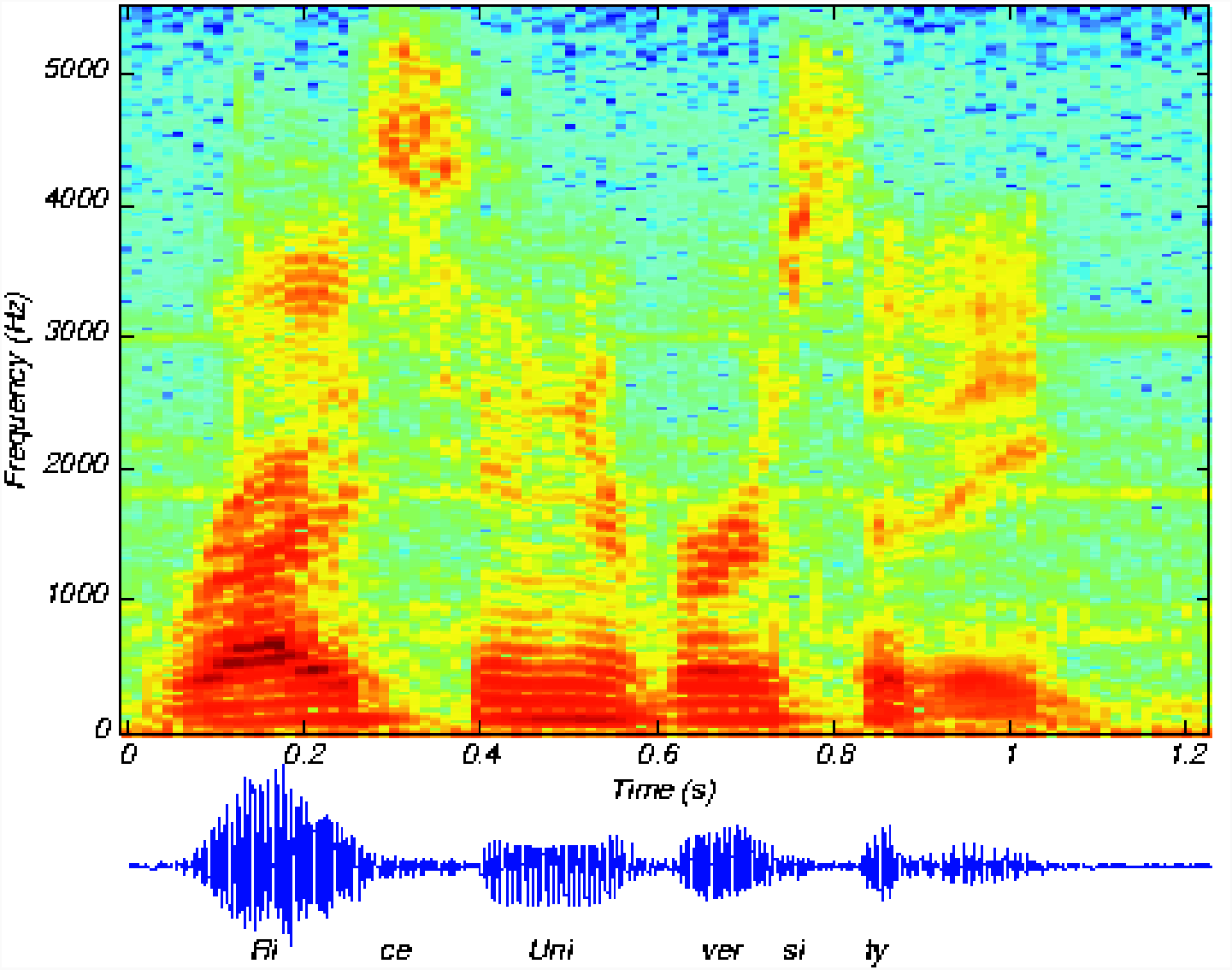
ML@B

Another Style Transfer Example

ML@B

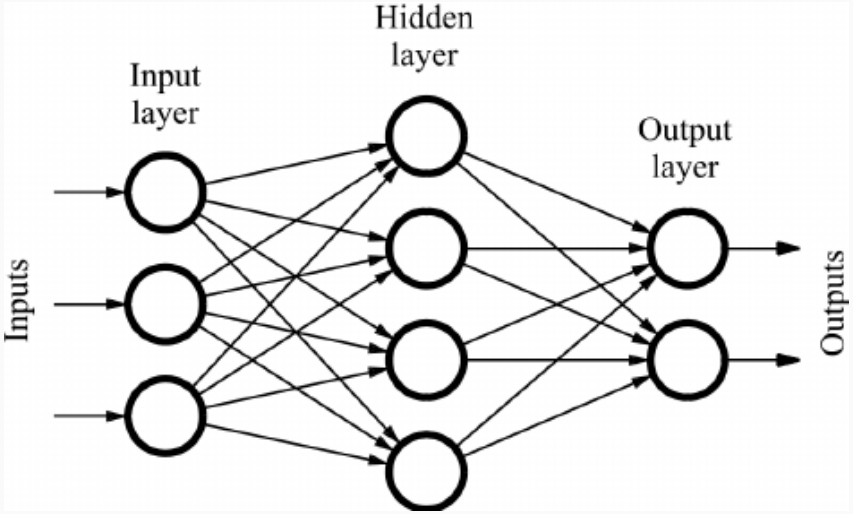
Cycle GAN: https://github.com/junyanz/CycleGAN

Audio Processing



Industry use of CNNs?

What's wrong with feed-forward NNs?



The curse of dimensionality.

Problem: high-dimensional image data

ML@B

- 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

ML@B

Basic Idea: Share parameters for different parts of input.

Convolutions

1D Convolution

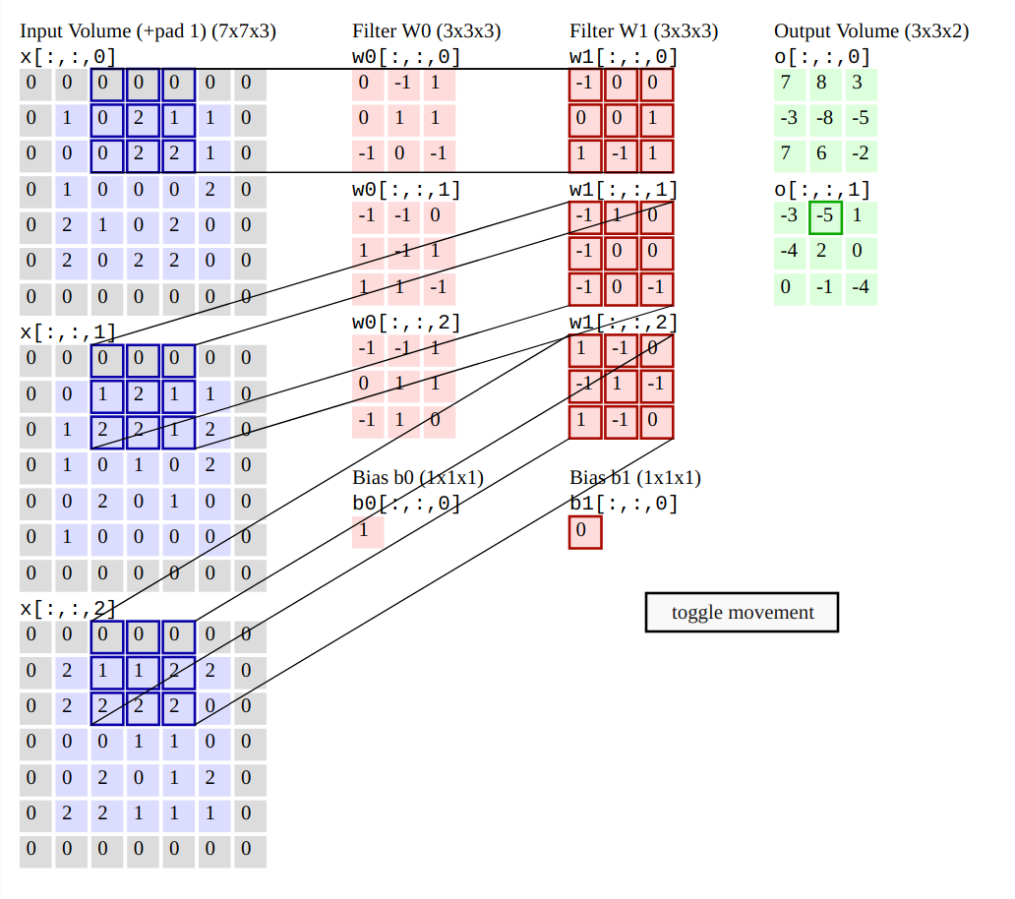
Intuition: sliding dot product.

Suppose f is a length N vector.

$$(g * f)[n] = \sum_{i=0}^{N} g[n+i]f[i]$$

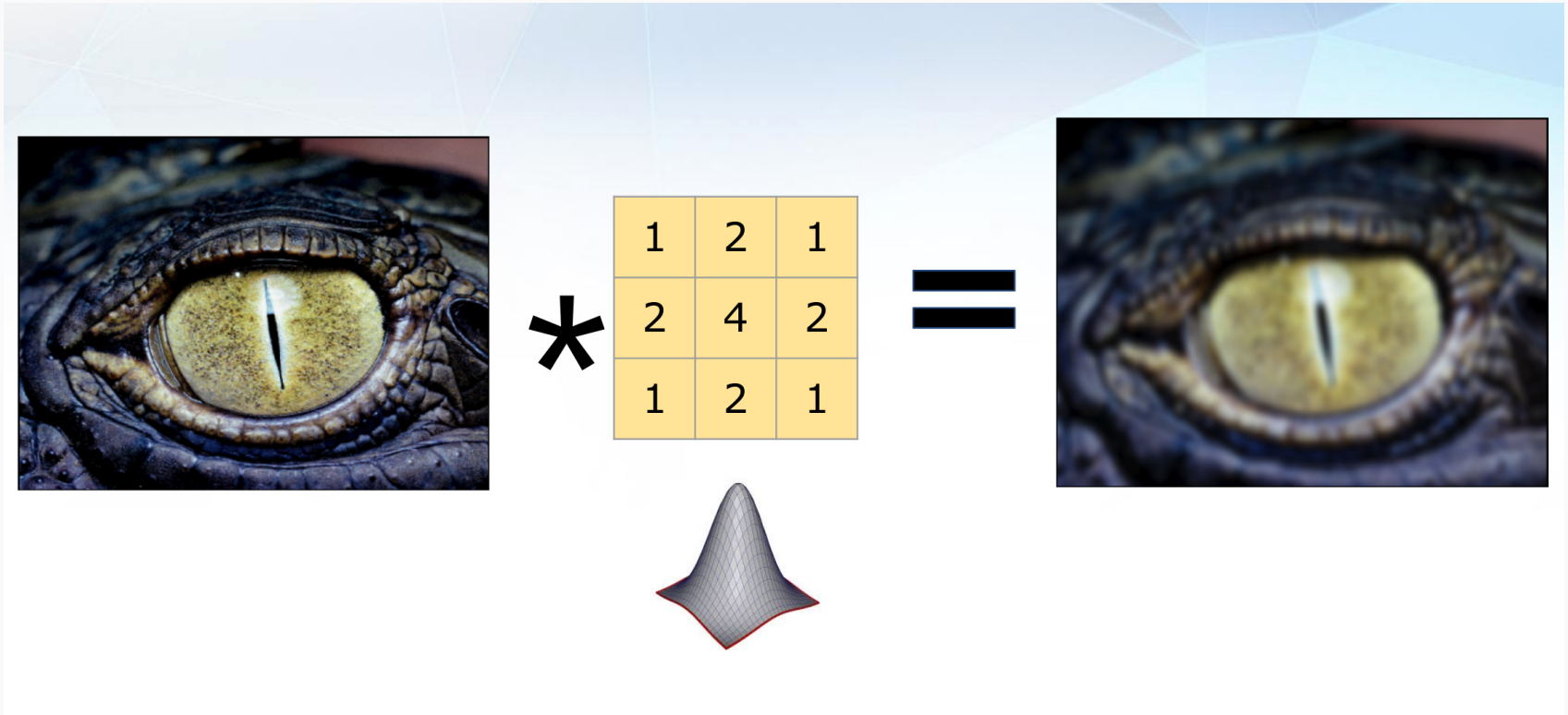
We can think of f as a filter (more on this later).

2D Convolution: Most Common in CNNs



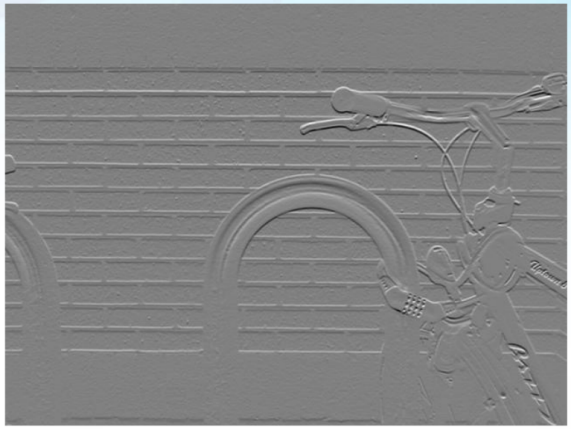
http://cs231n.github.io/convolutional-networks/

Convolutions as filters



Convolutions as filters cont.

$$*\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} =$$



ML@B

Demo

ML@B

This will account for exactly 0% of your grade in this class.

If $f = [5, 1, 2]$ and $g = [3, 1, 4, 5, 6, 0, 0, 1]$, What is the 4 element of the output vector of $f * g$?

Answer

$$f * g[4] = 5 * 5 + 1 * 6 + 0 * 2$$

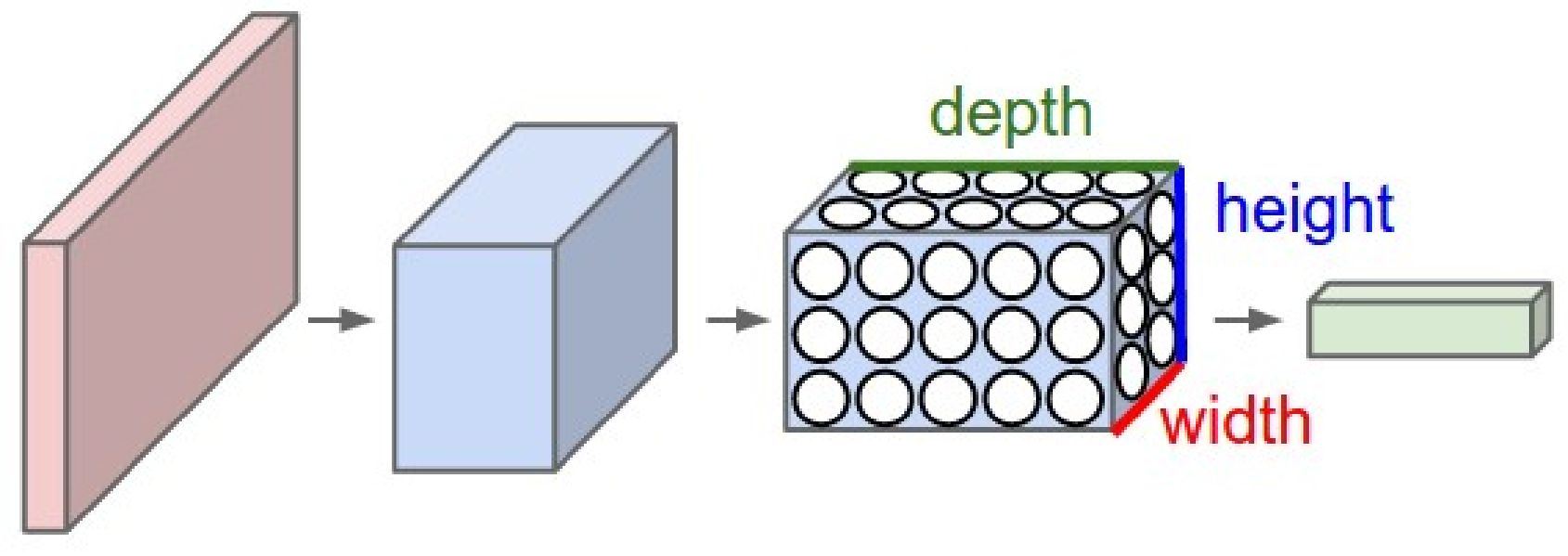
Convolutions in Neural Networks

ML@B

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

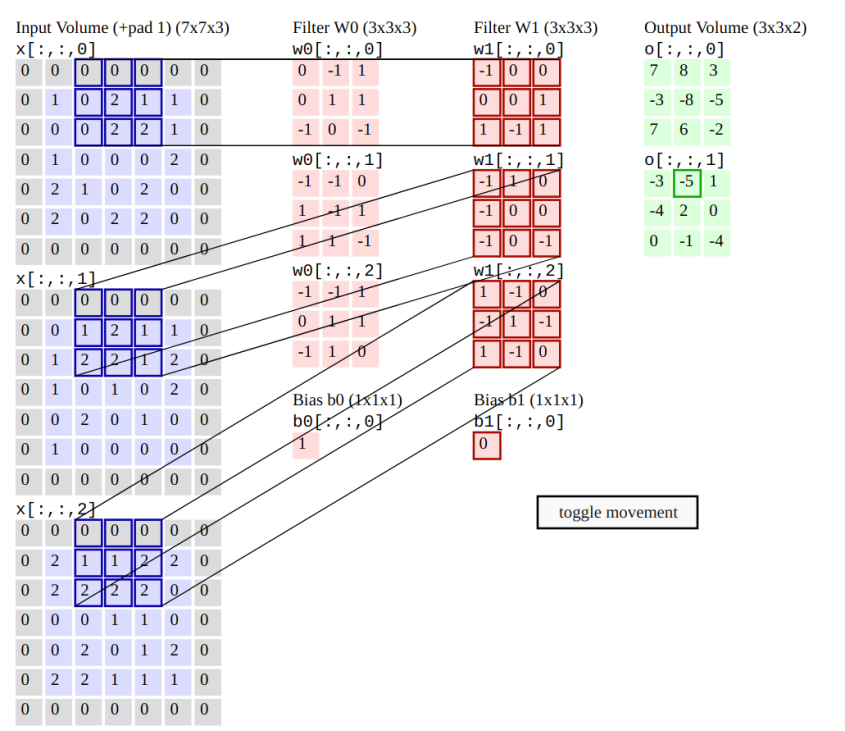


What do our filters look like?

3D filters !!!

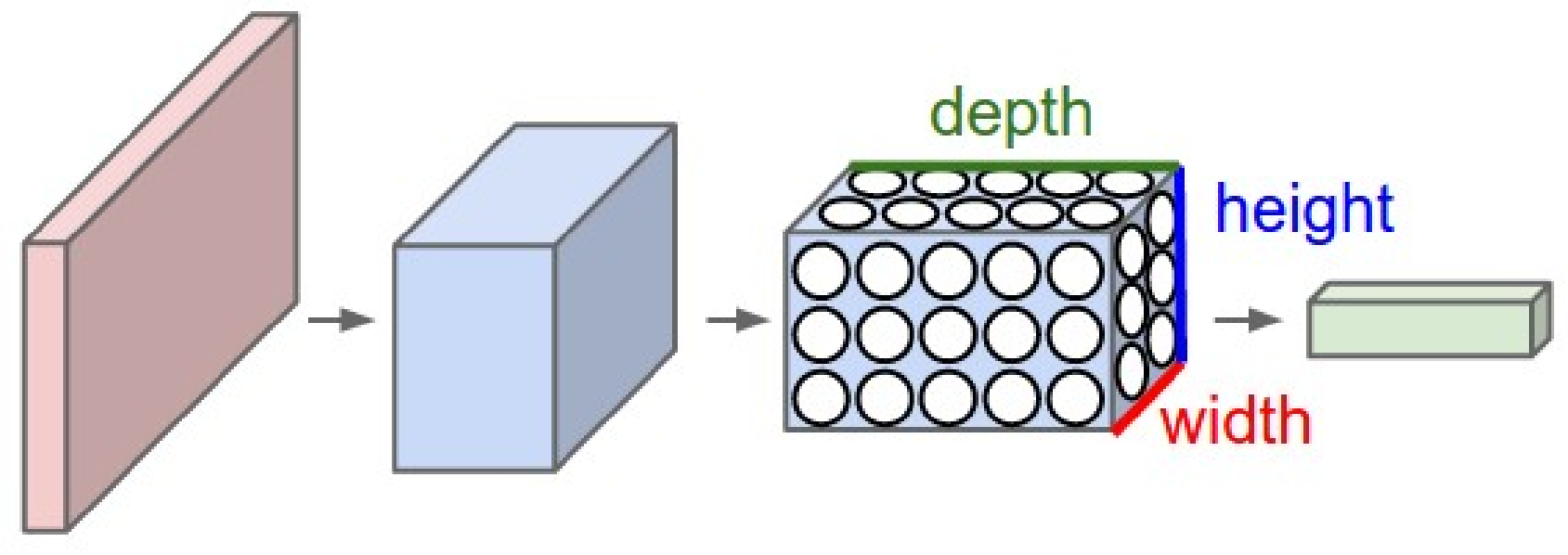
Our filters will then be $n \times n \times 3$, where n is a hyperparameter.

We can also have multiple filters per layer.

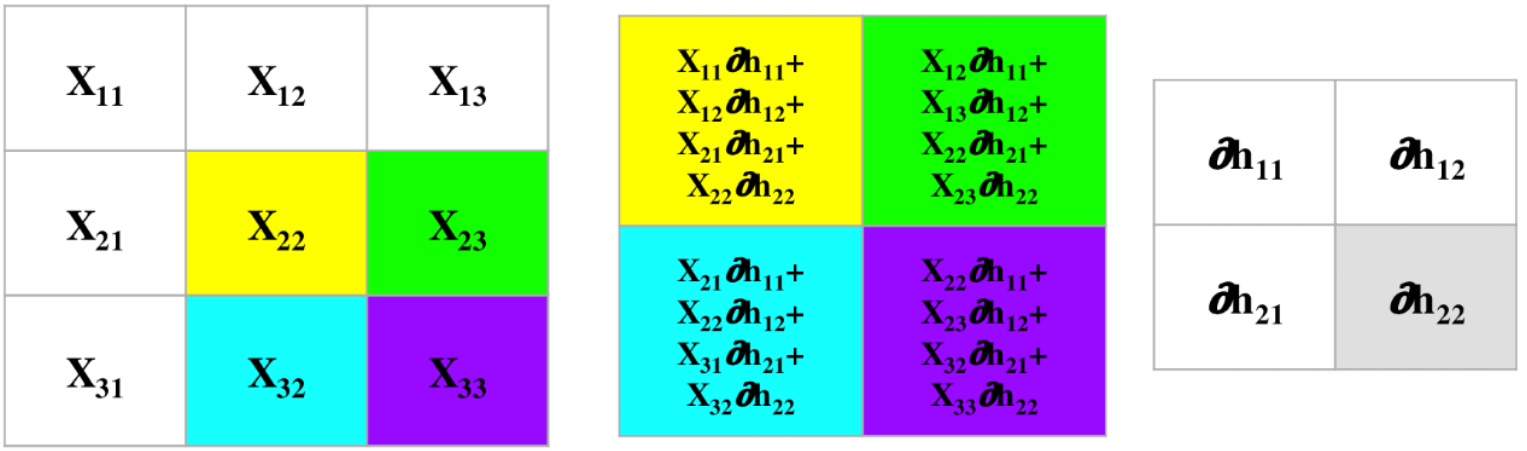


What will the output of a layer look like?

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

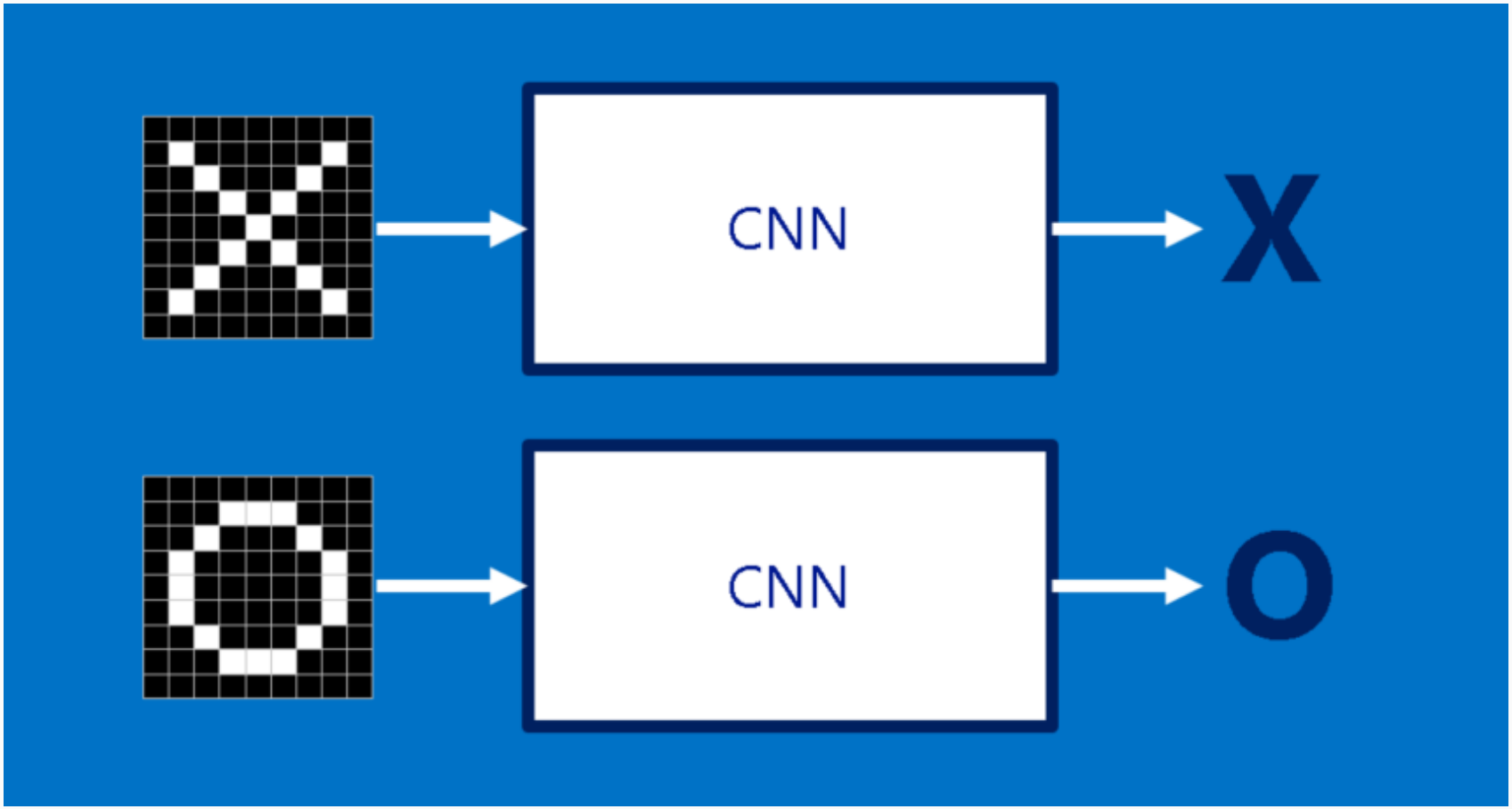


How do we update filters?



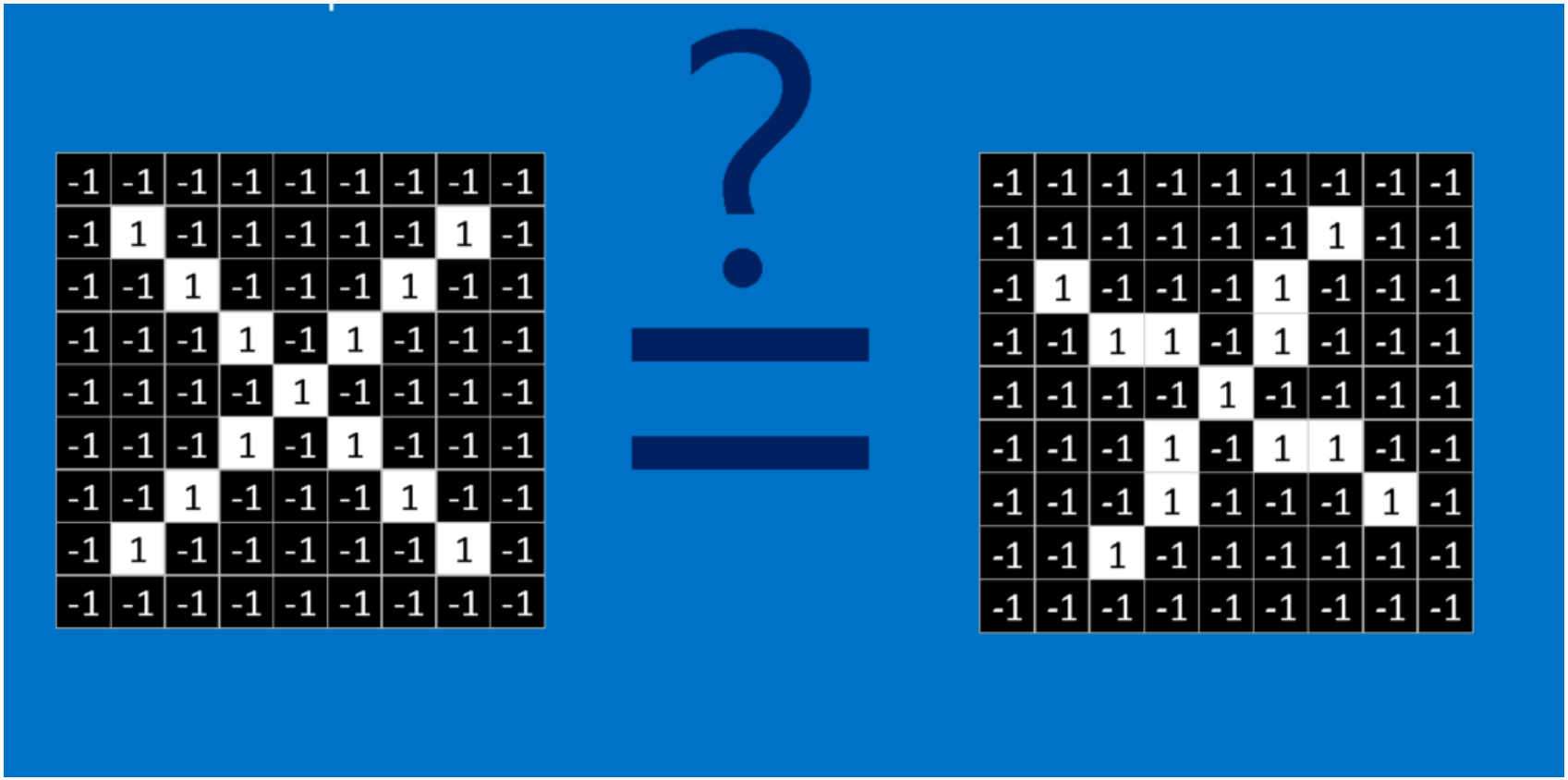
Derivative Computation (Backward pass) since pictures speak more than words

Let's look at a toy example!



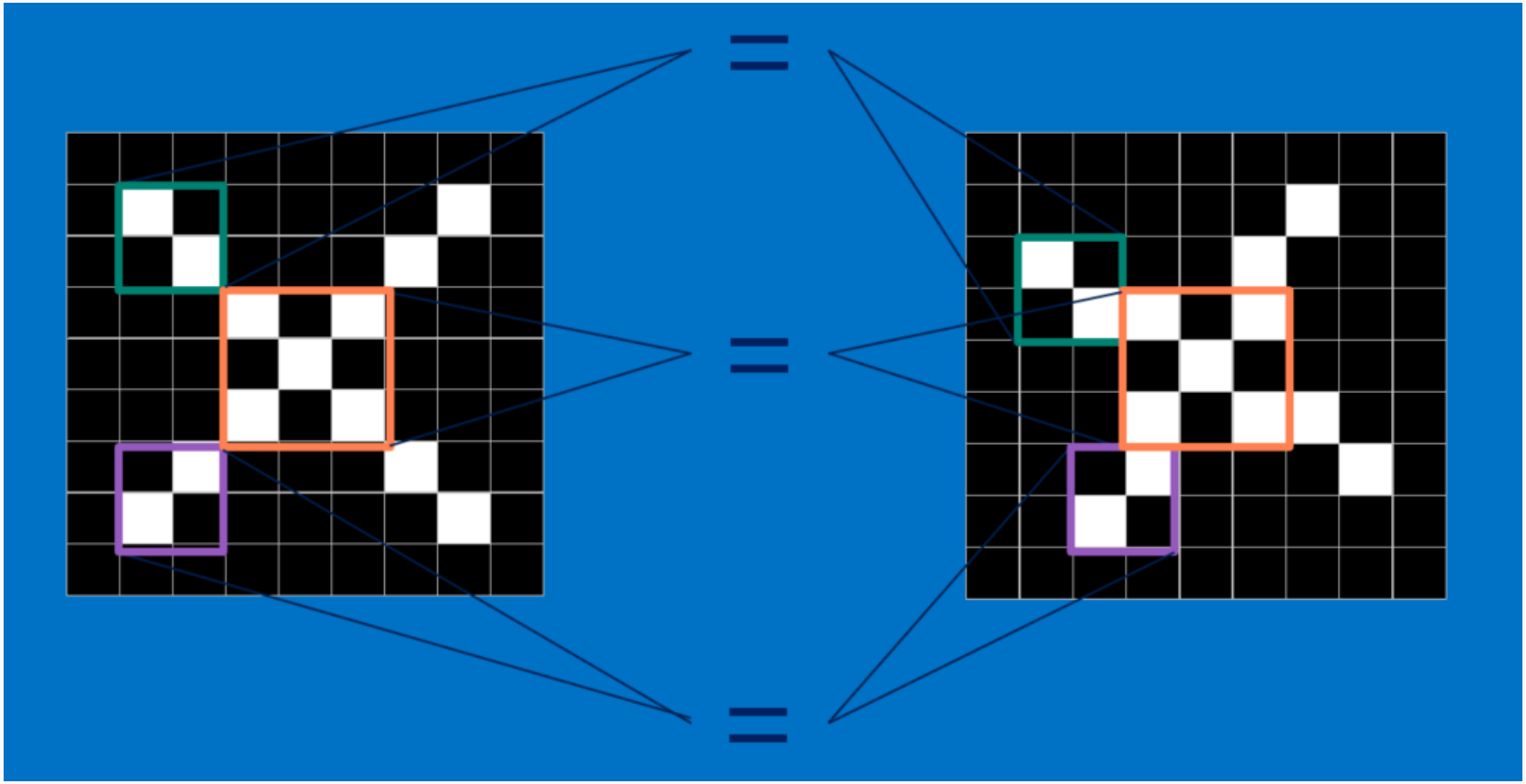
- How would a CNN classify X's vs O's?

Would you classify this as an X?



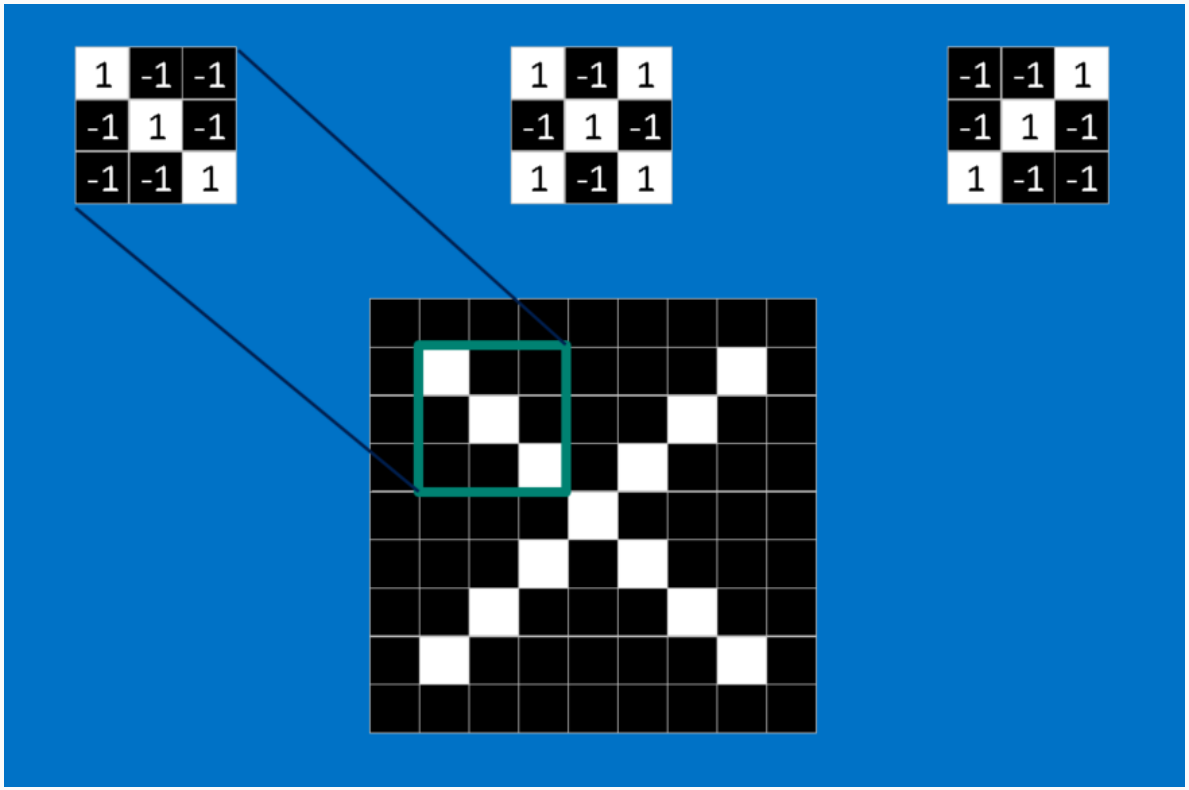
- A computer can't easily distinguish this example as an X.. so what can we do?

What is a feature?



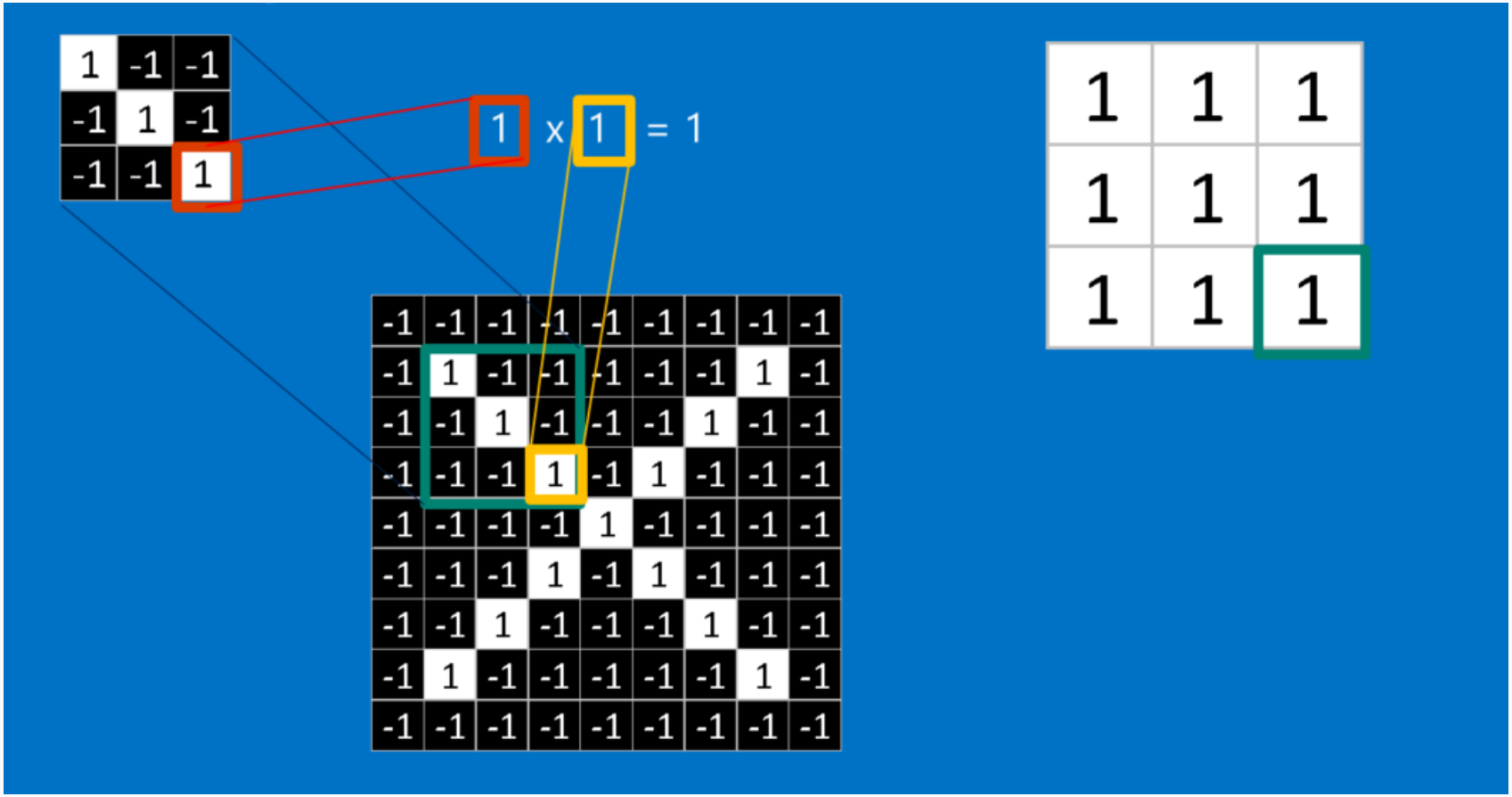
- We can break down the image into smaller images and look for patterns!
- Can we see here the similiar patterns?
- These diagonals and crosses can be thought of as features!

Convolutional Filters



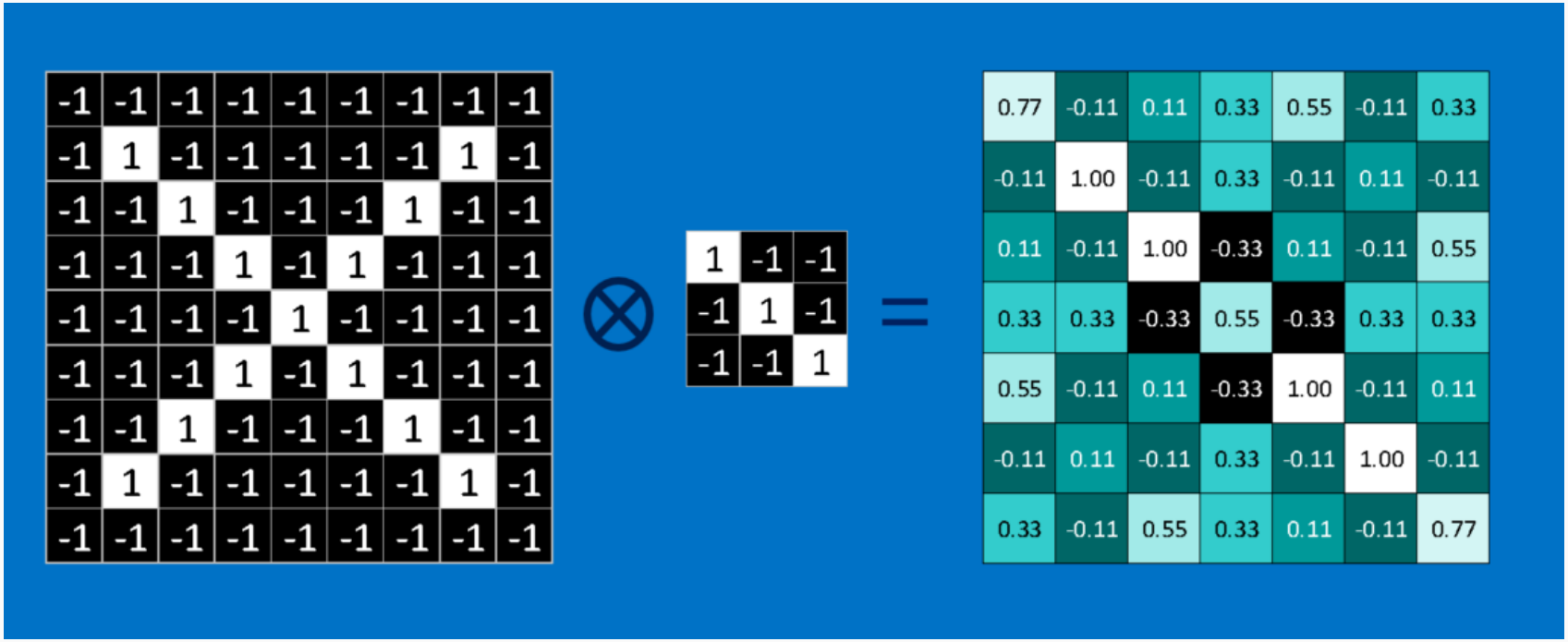
- Let's break this up into smaller problems... what about smaller patterns!

CNN Layers: Convolutions



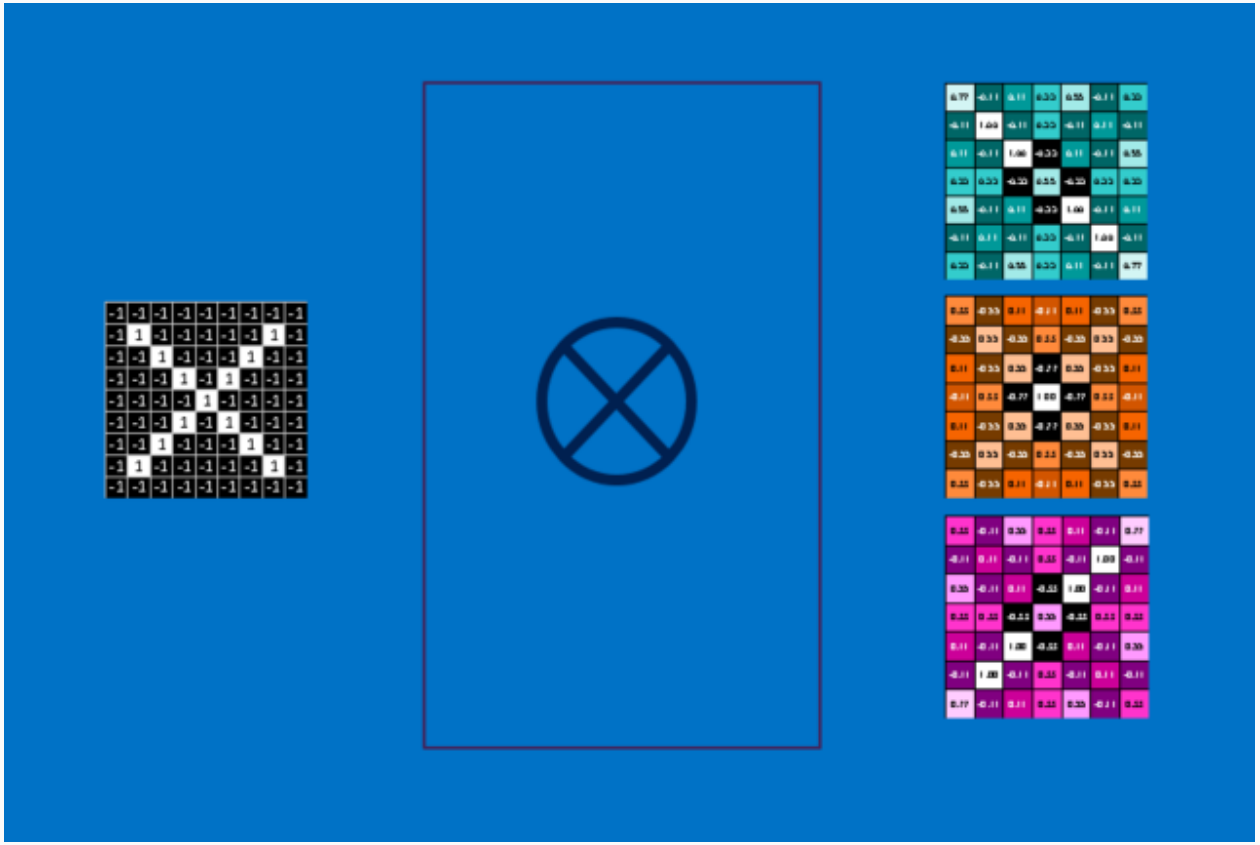
- Now let's compare our smaller pattern with the image.
- This is the basic idea behind convolutions!

CNN Layers: Convolutions



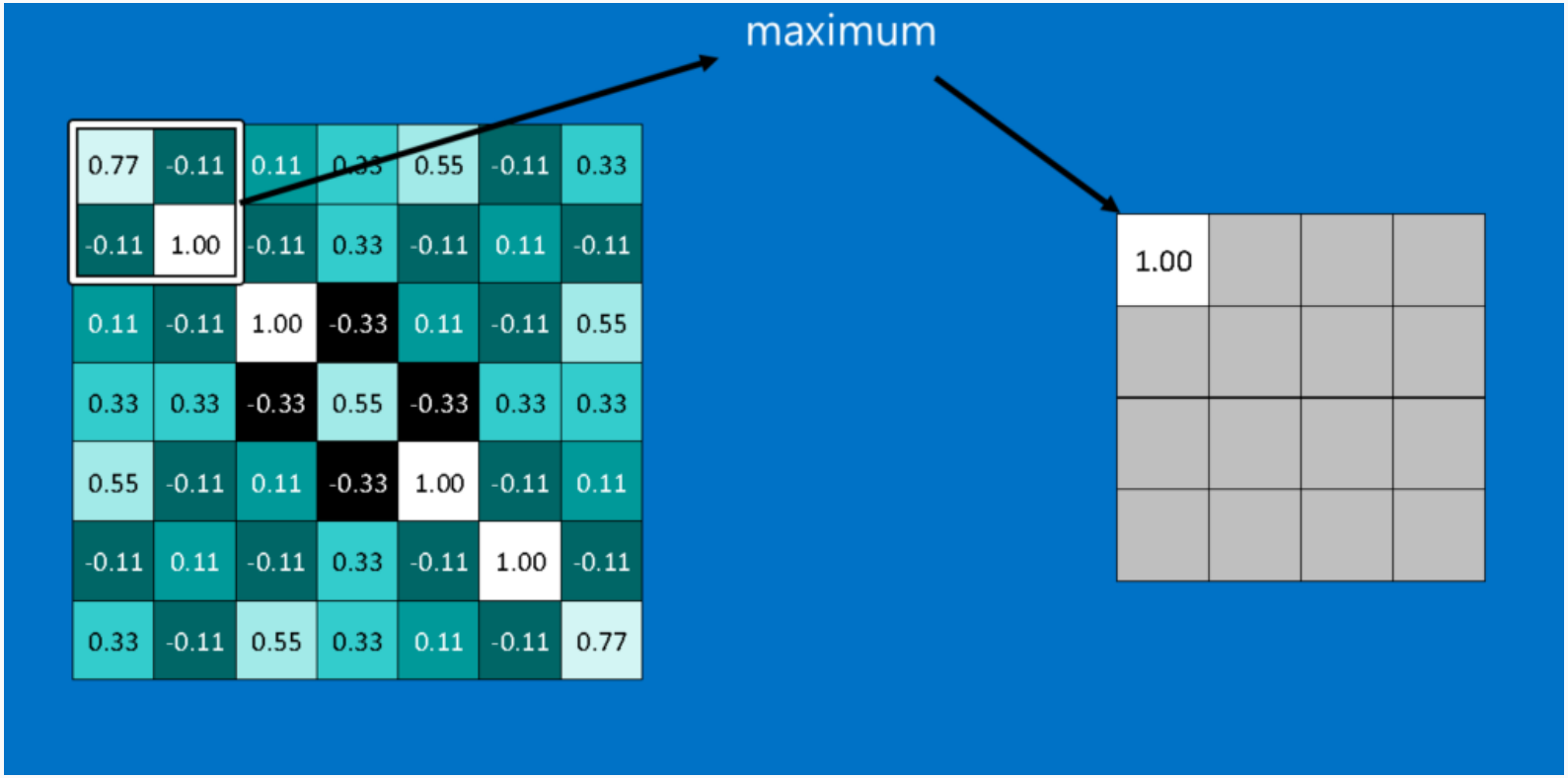
- Now lets apply this process to every part of the board.
- We can look at the new board and call it a filtered version of this specific feature
- Shows us where on the image the feature matches the most

CNN Layers: Convolutions



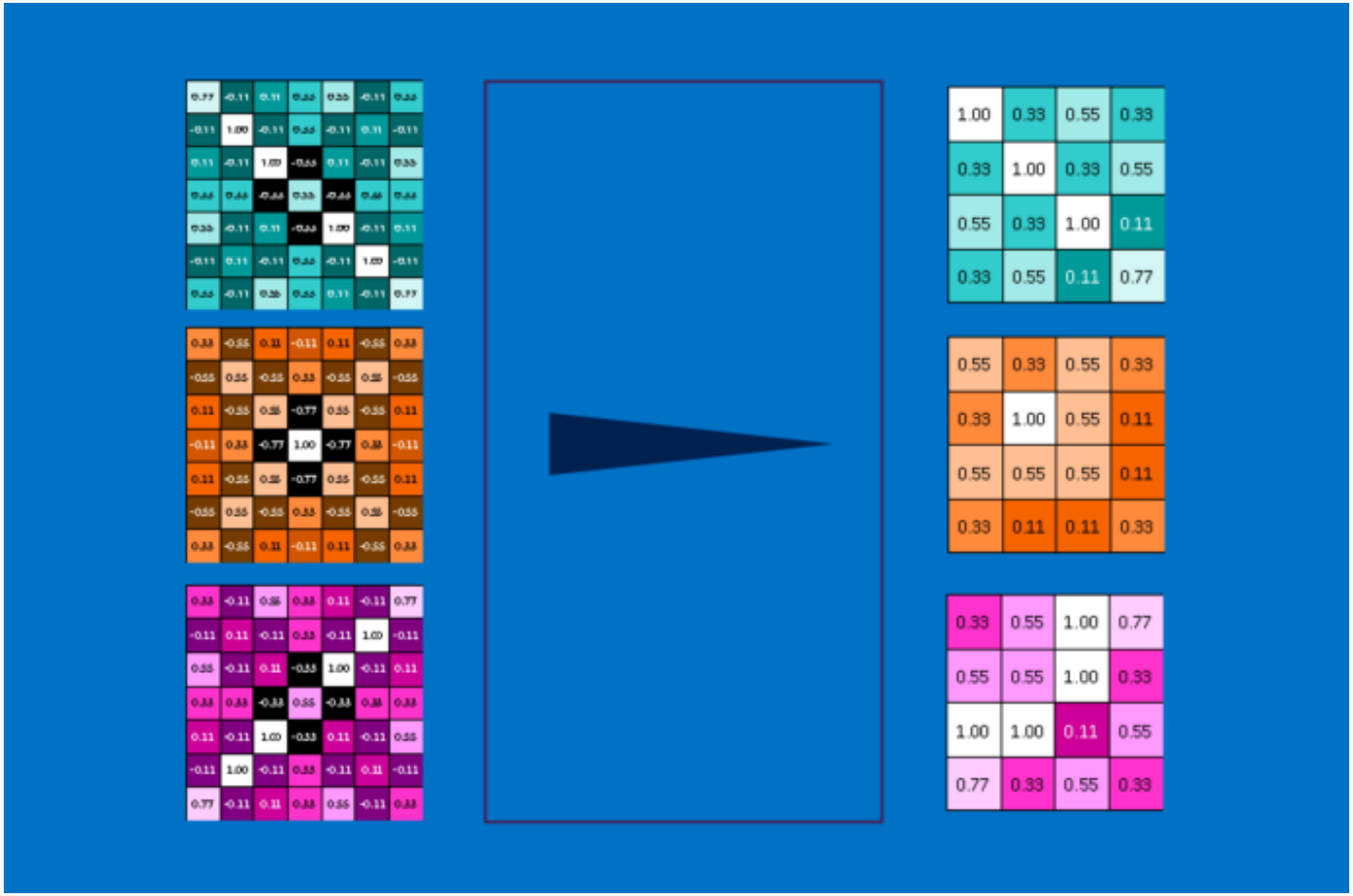
- Now lets apply this process to every part of the image for different features.

CNN Layers: Pooling



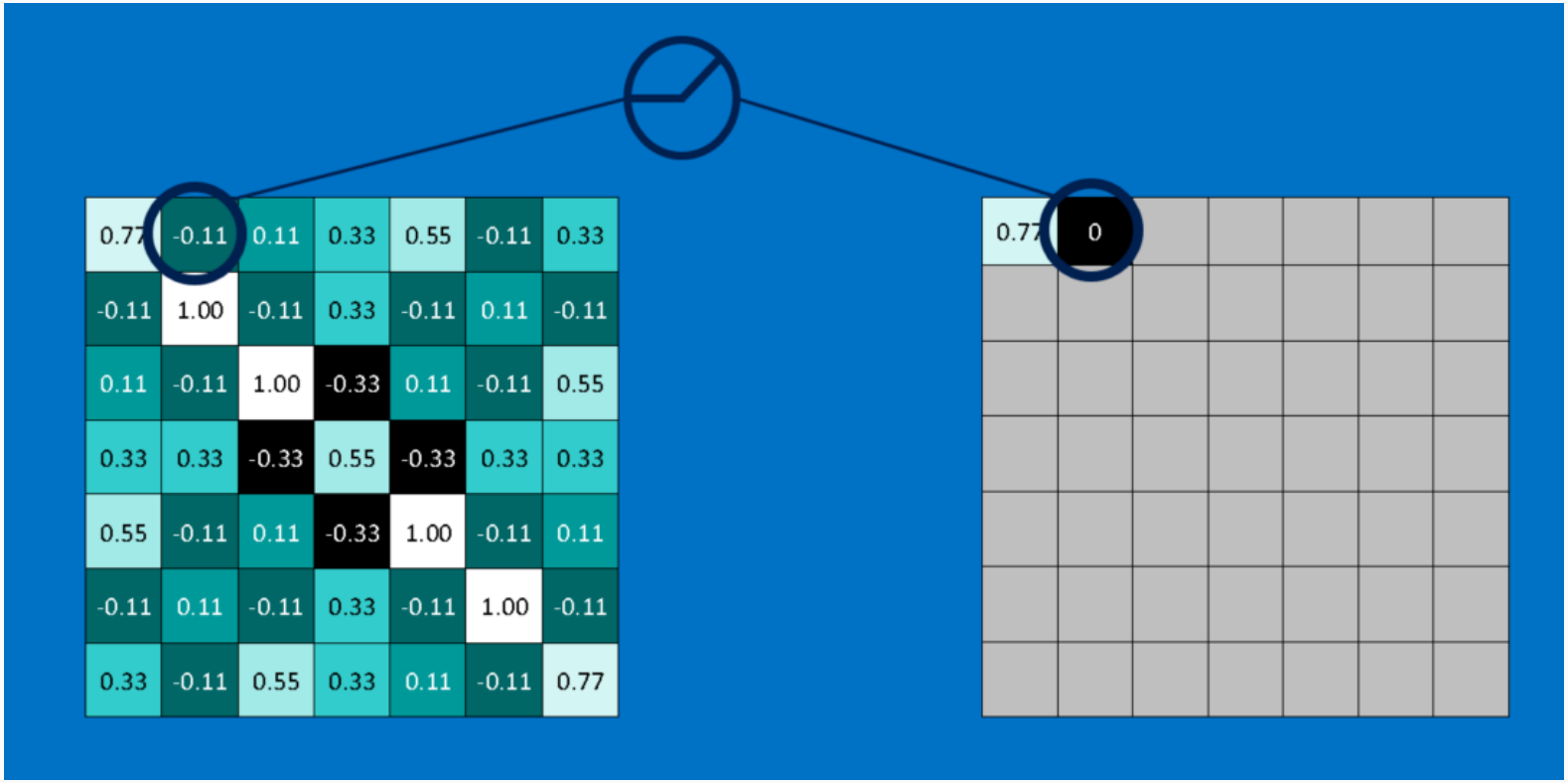
- If CNN's are really deep, doing that math for every possible pixel gets to be quite computationally expensive
- That's why we have pooling layers, that down sample our image!

CNN Architecture



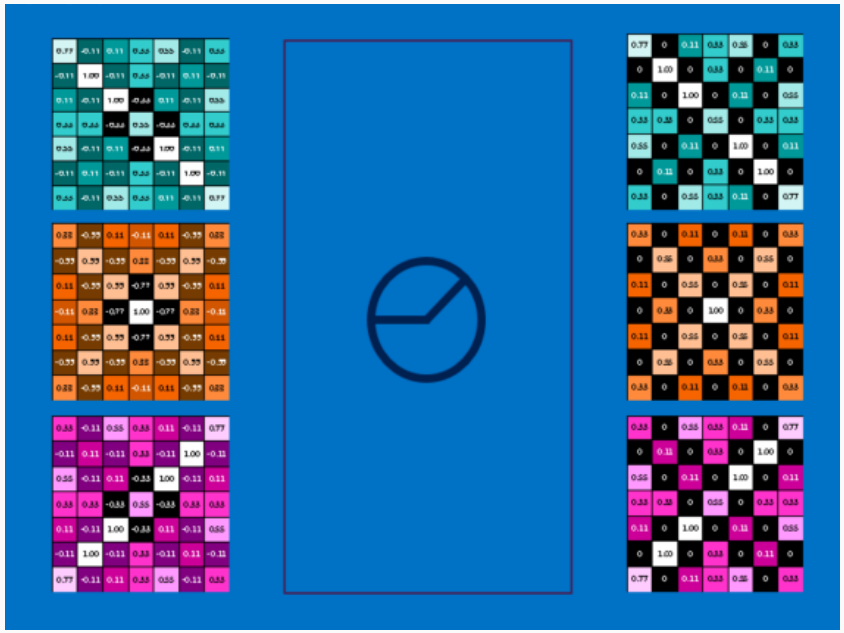
- By applying these pooling layers, we try to capture the patterns and get rid of extra unneeded information.

CNN Layers: Activation



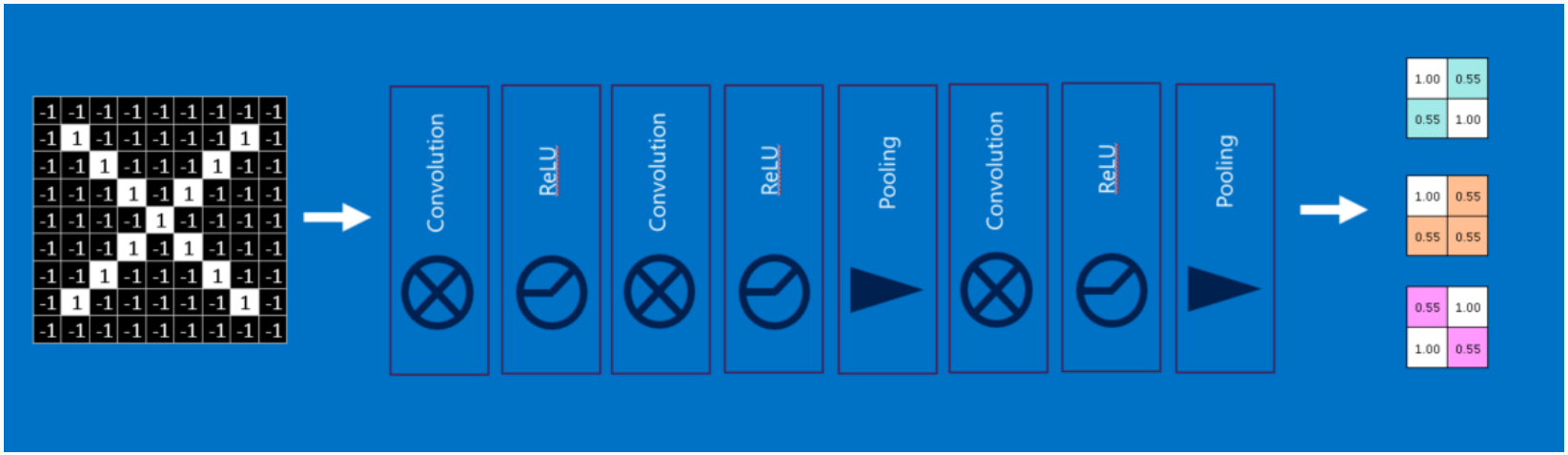
- Remember those activation functions you learned from the last lecture? We use them here too!
- Applying RELU we simply get rid of all the negatives!

CNN Pooling Layers

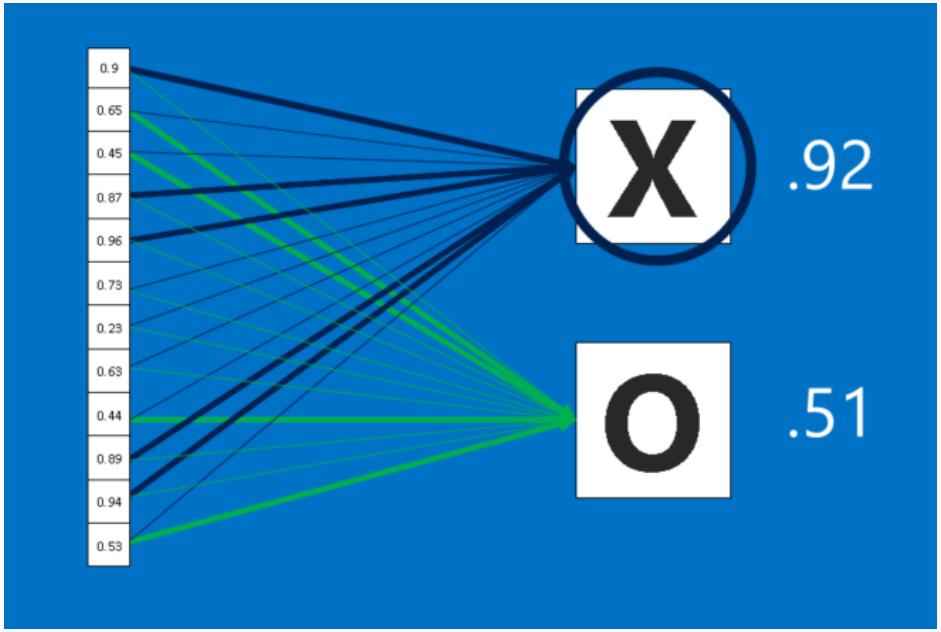


- The idea behind this, is to help keep any one number stuck at 0, or block up towards infinity
- This may not seem like a big issue in our toy example, but in much larger more complicated CNN's having stable values is important!

Let's put it all together!



CNN: Fully Connected Layer



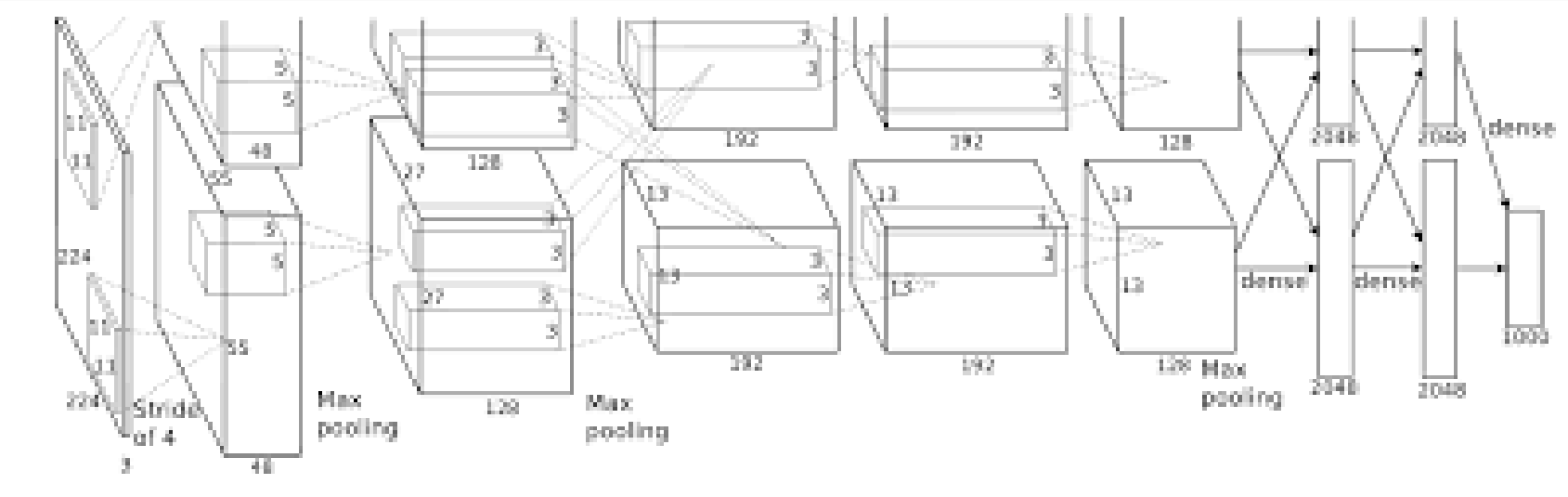
- At the end we use a fully connected layer to hold a vote if the image is either an X or O
- Each value has a weight though, so not all values contribute equally
- This goes back to the ideas you learned last week!

Idolized CNN architectures

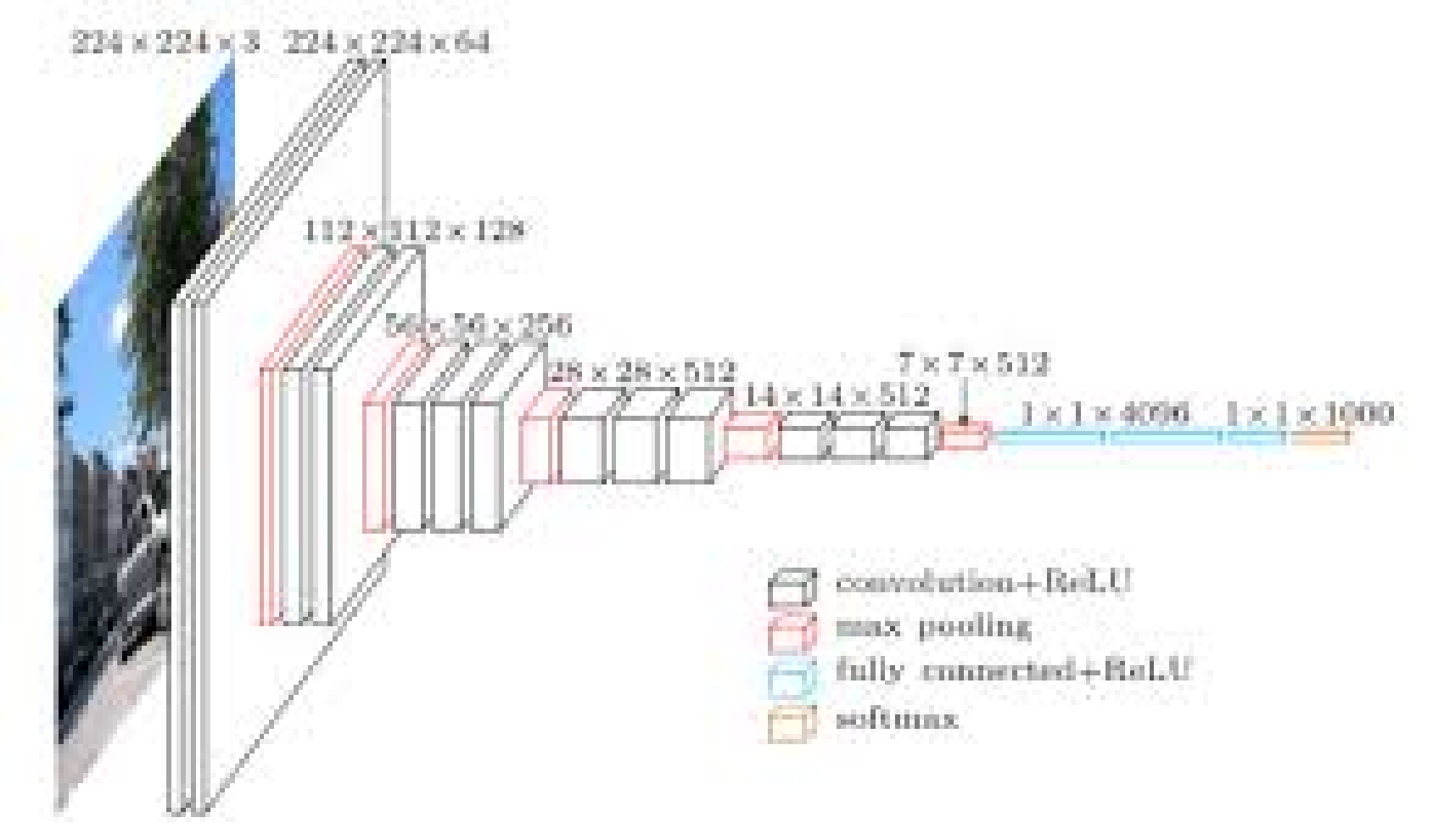
ML@B

- "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...

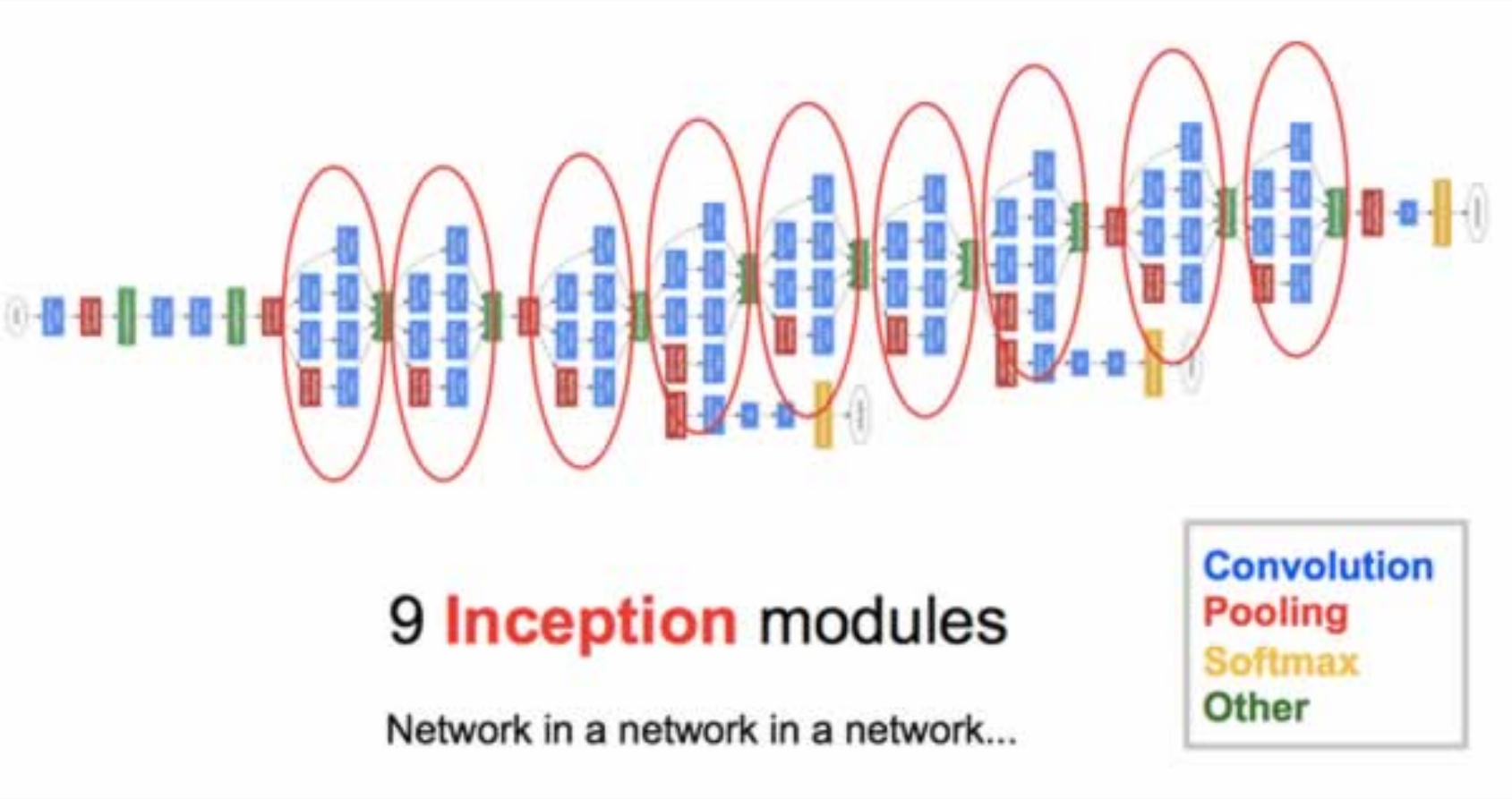
AlexNet (2012)



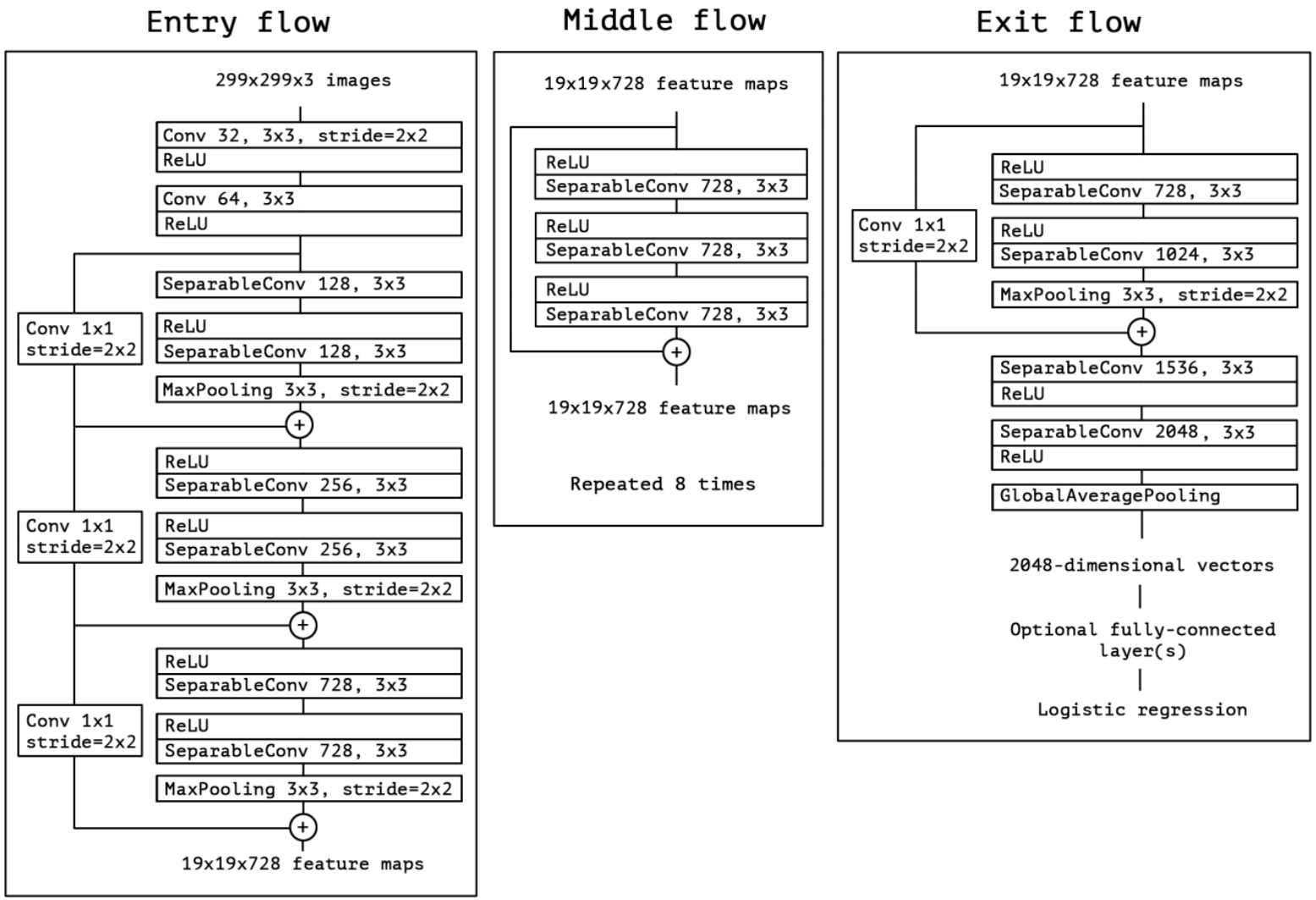
VGG-16 (2014)


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

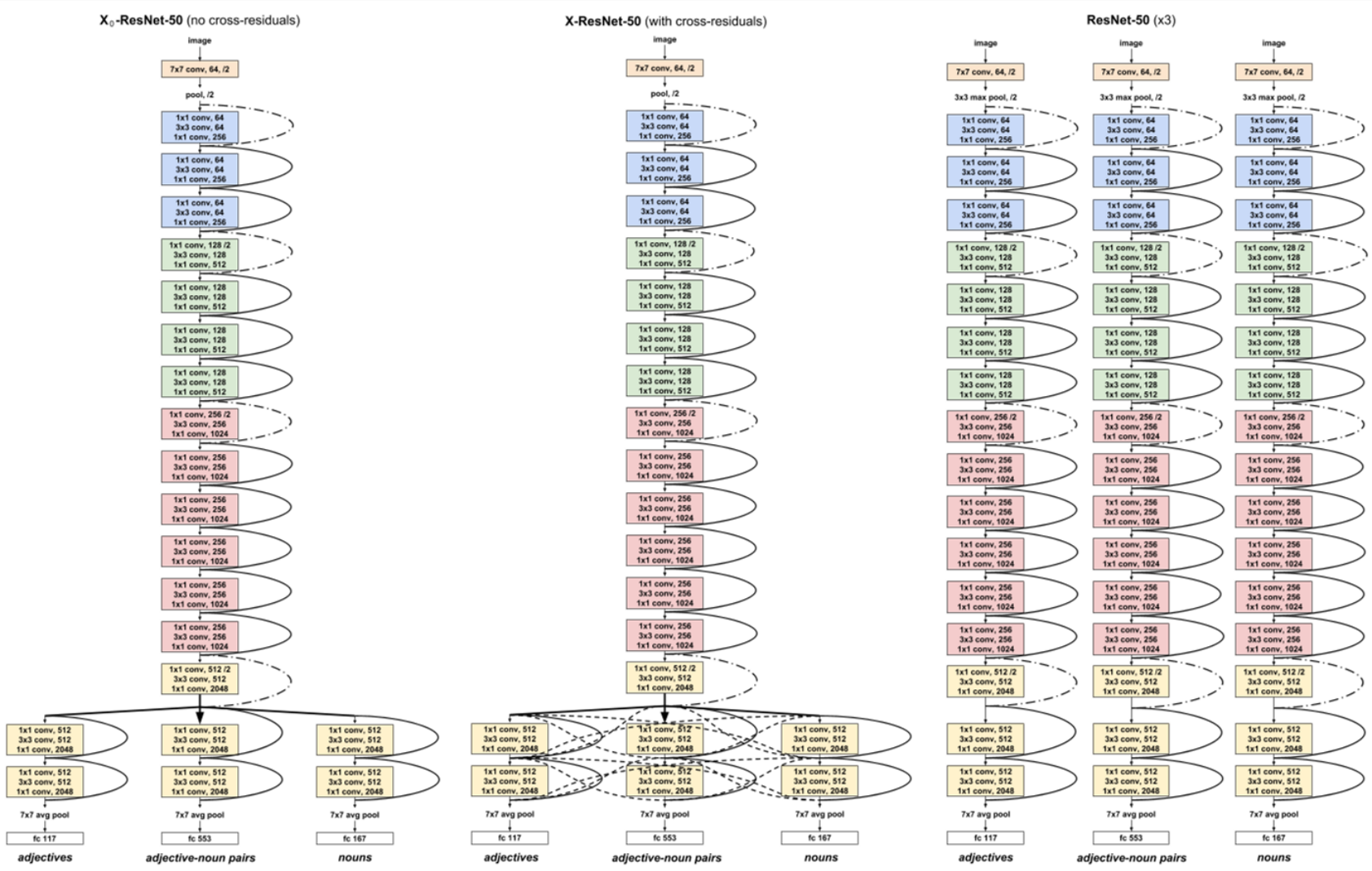
GoogLeNet (2015)



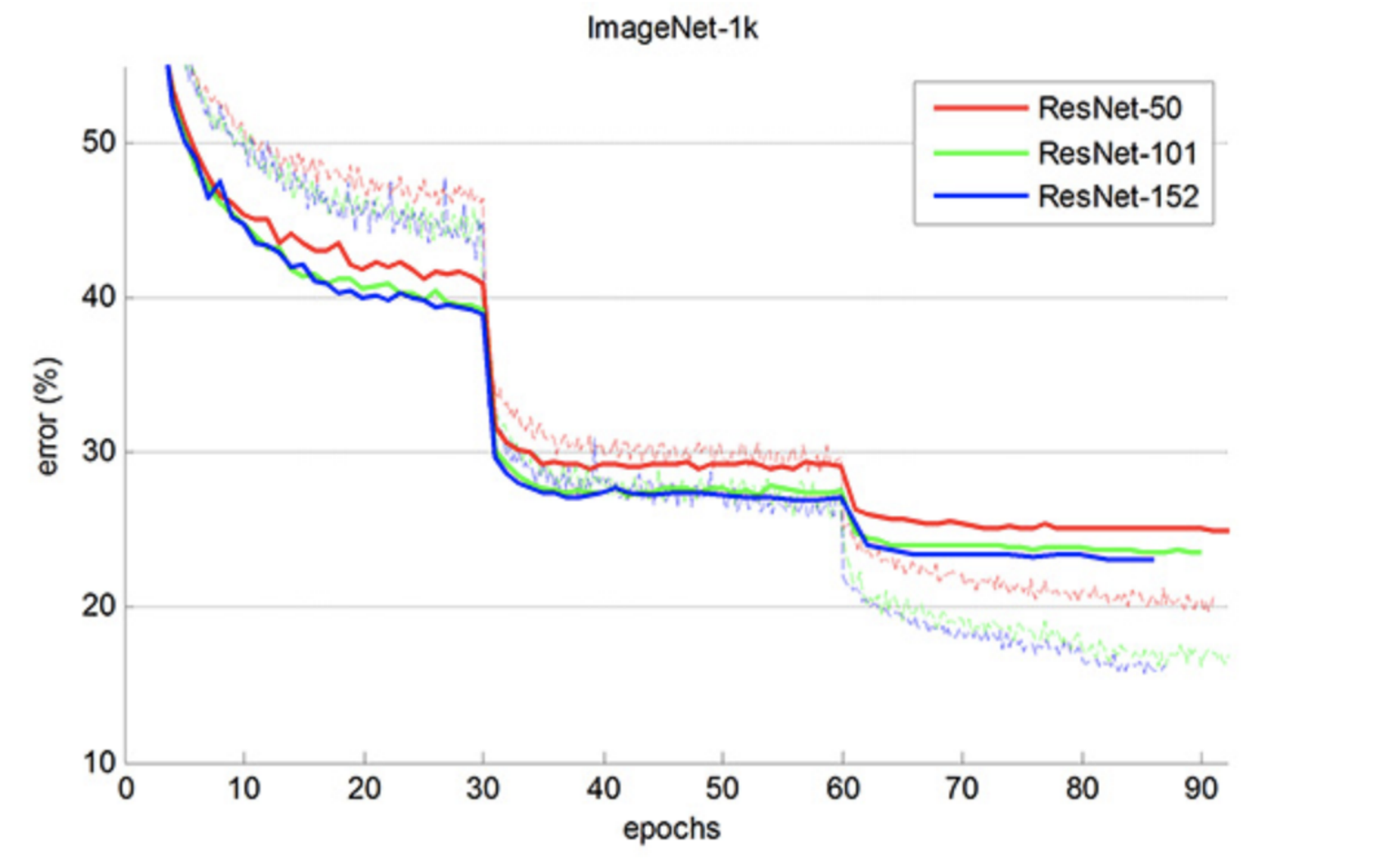
Xception (2016)



ResNet-50 (2015)



ResNet-152 (2015)

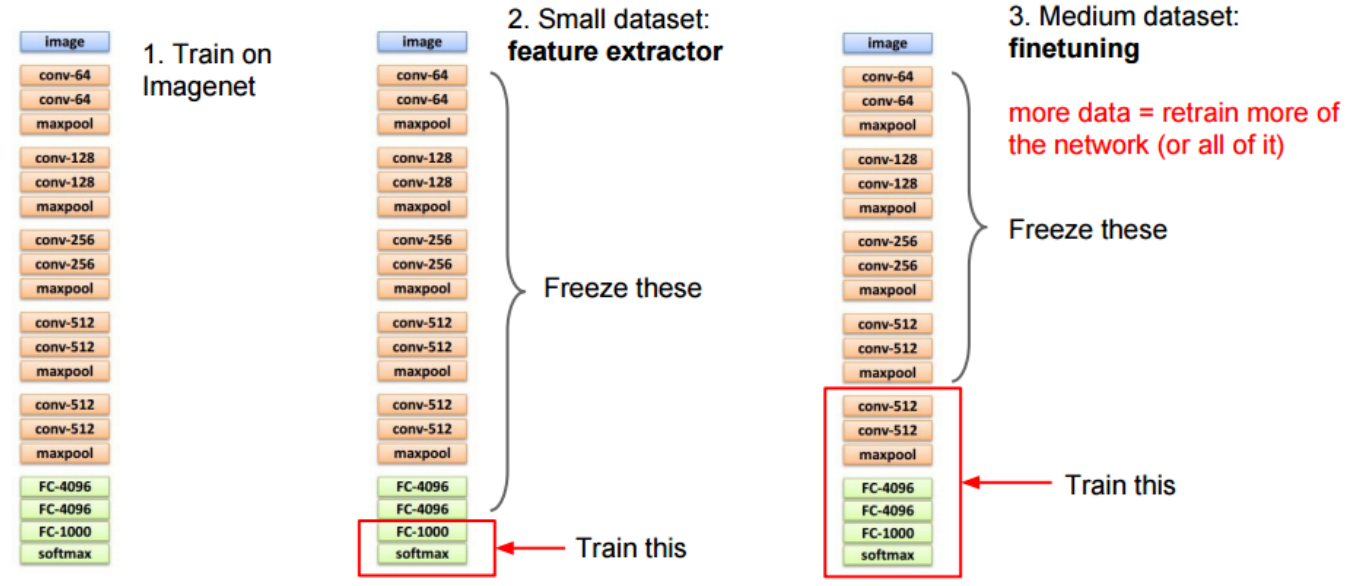


Transfer Learning

ML@B

- 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...

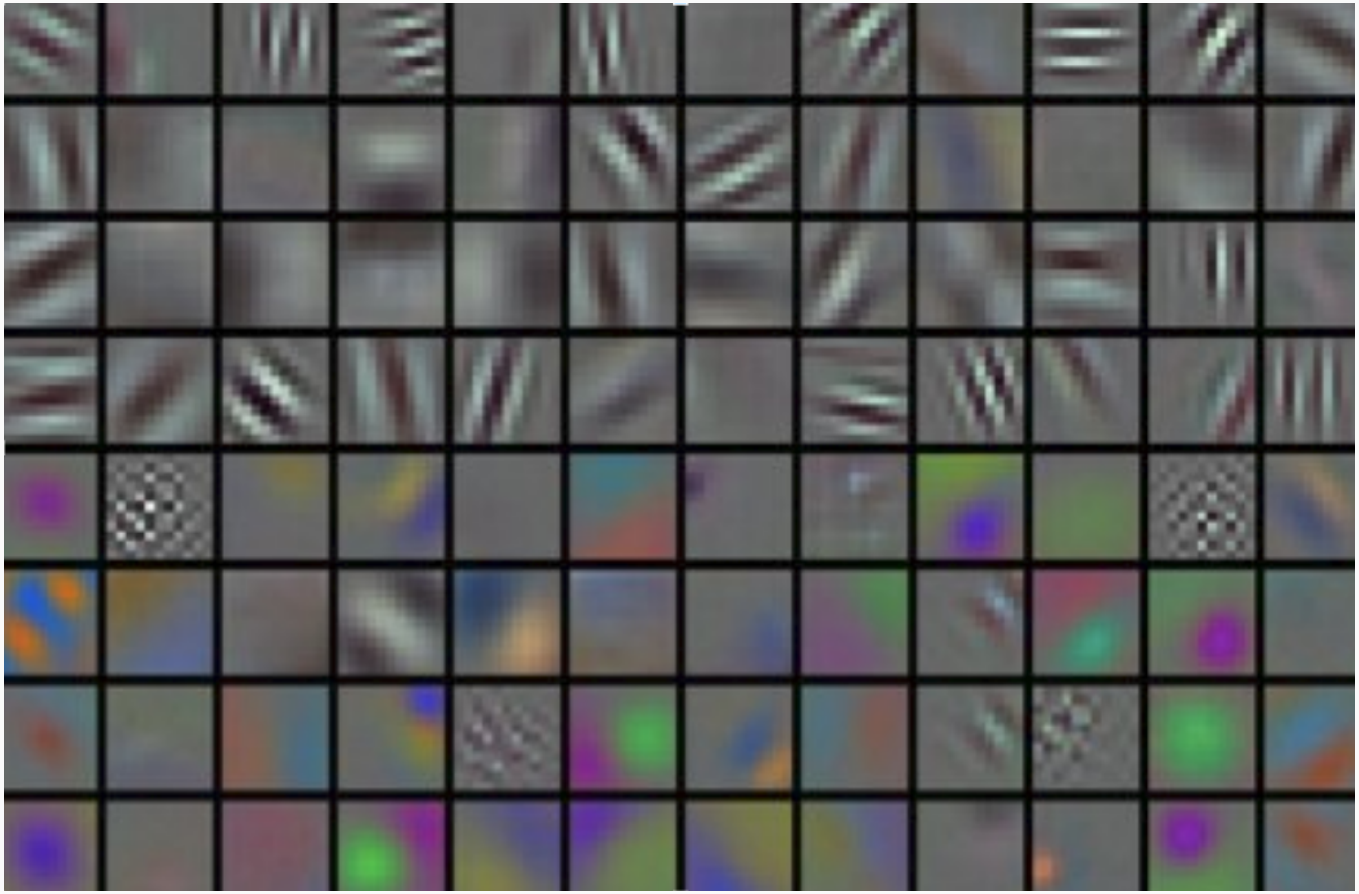
How Can We Transfer Learn?



Why does it work?

ML@B

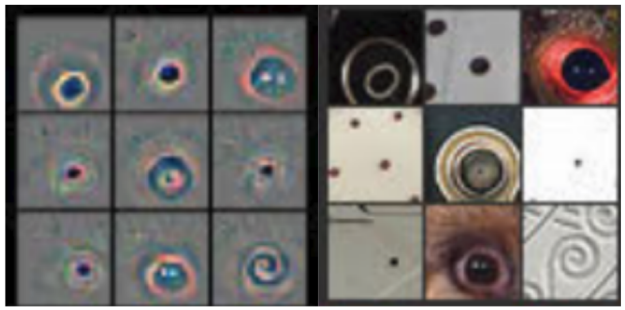
Early layers learn simple features



Why does it work?

ML@B

Later layers learn more abstract features



Layer 2 of AlexNet

Layer 4 of AlexNet

Demos

CNN demo: FashionMNIST

Cool things we did not have time to talk about today

- Training networks using GPUs
- Network visualization and interpretation
- Other types of neural networks: auto-encoders, Neural Turing machines, Generative adversarial networks, ...

Questions

Questions?