Deep Learning Theory and Applications

CNNs

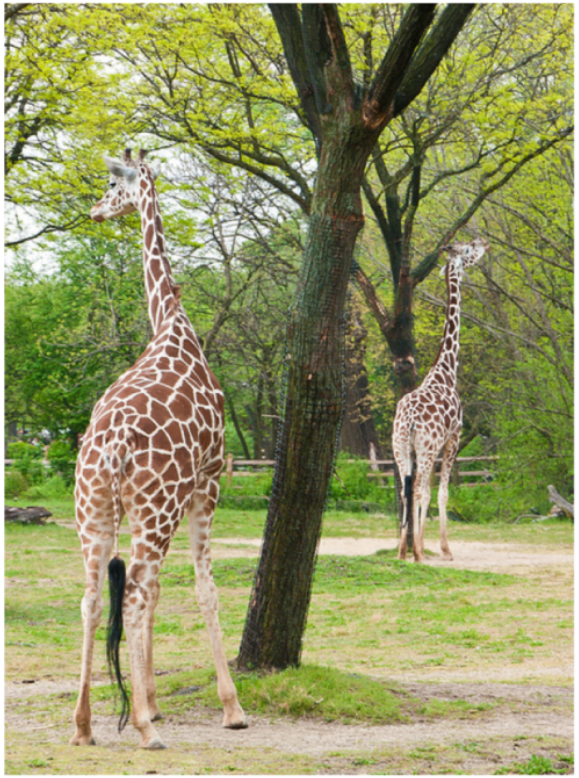
Yale

CPSC/AMTH 663

Image Classification

a soccer player is kicking a soccer ball

a street sign on a pole in front of a building

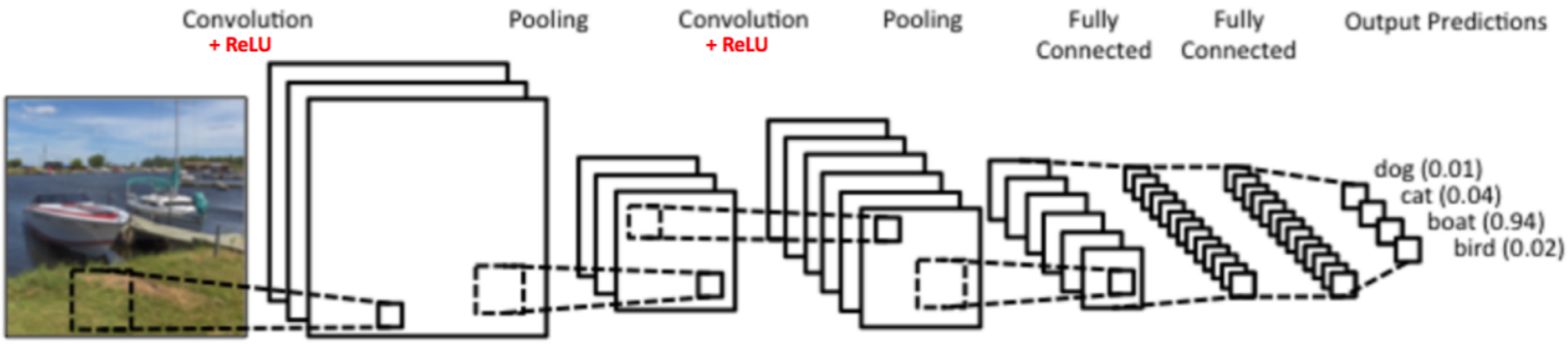


a couple of giraffe standing next to each other

Nnets have been very good at classifying real world images

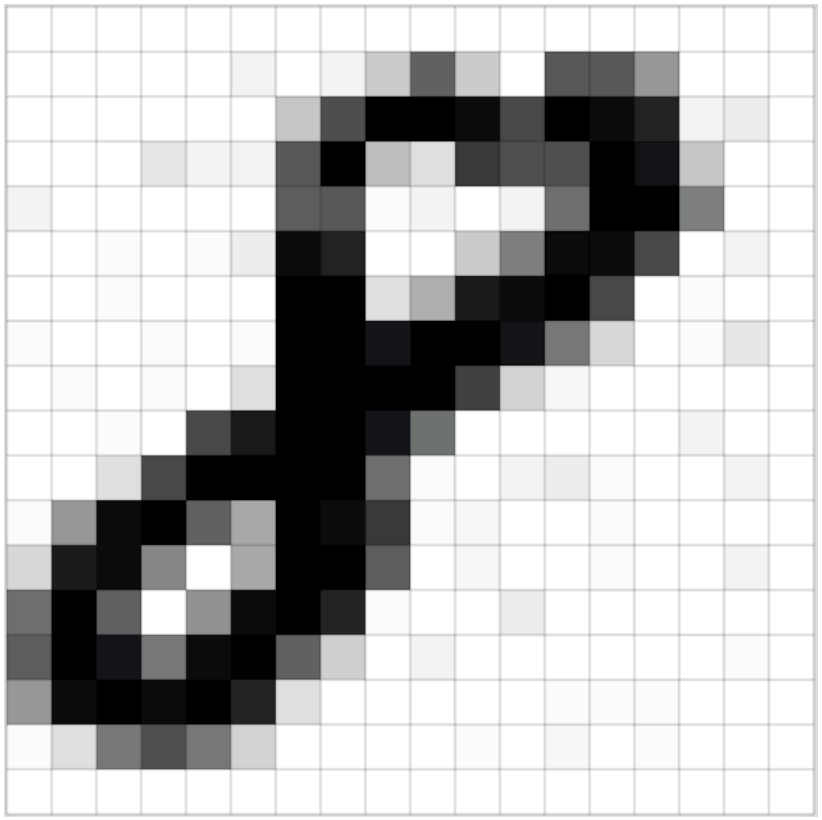
First architecture was LeNet formulated by Yann Lecun in 1988

How to make a CNN



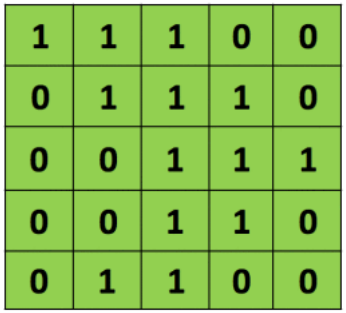
- Convolutions
- Non-linearities
- Pooling
- Classification

Images are a series of Pixel Values

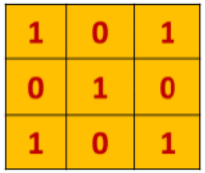


Grayscale images:
0=Black
255 = White

The convolution operation

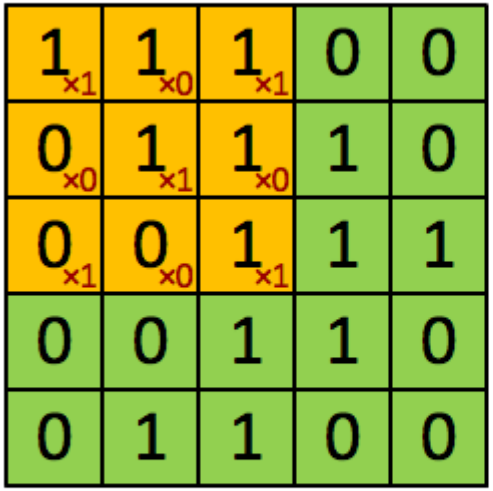


Image

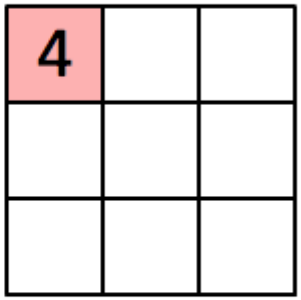


Filter/
Feature detector

1. Pointwise multiply
2. Add results
3. Translate filter



Image



Convolved
Feature

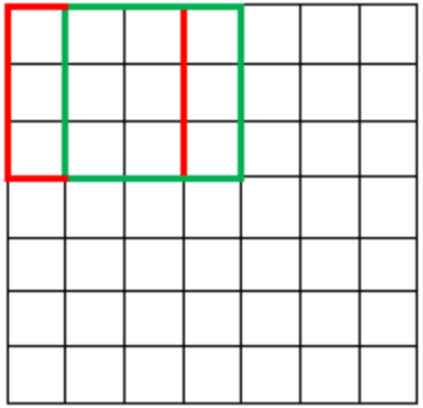
Filters

Original Image

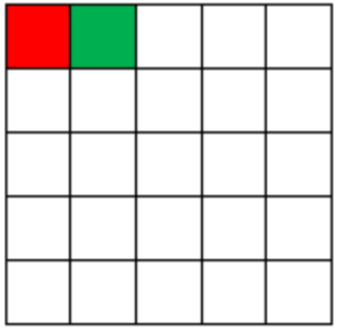
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Stride

7 x 7 Input Volume

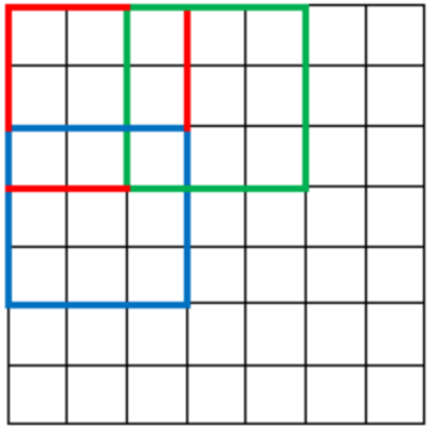


5 x 5 Output Volume

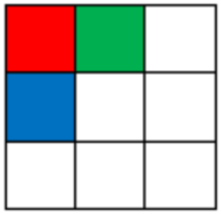


Stride 1

7 x 7 Input Volume



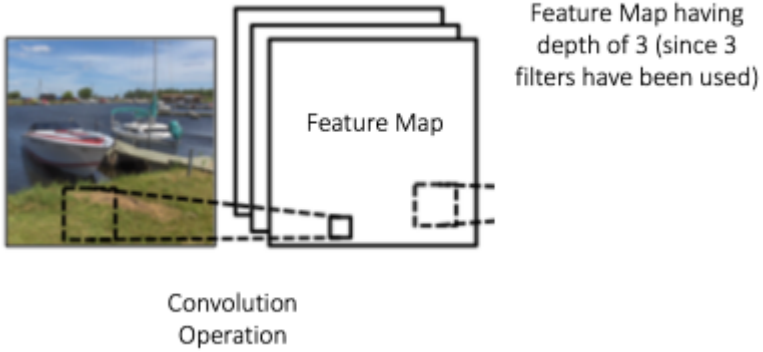
3 x 3 Output Volume



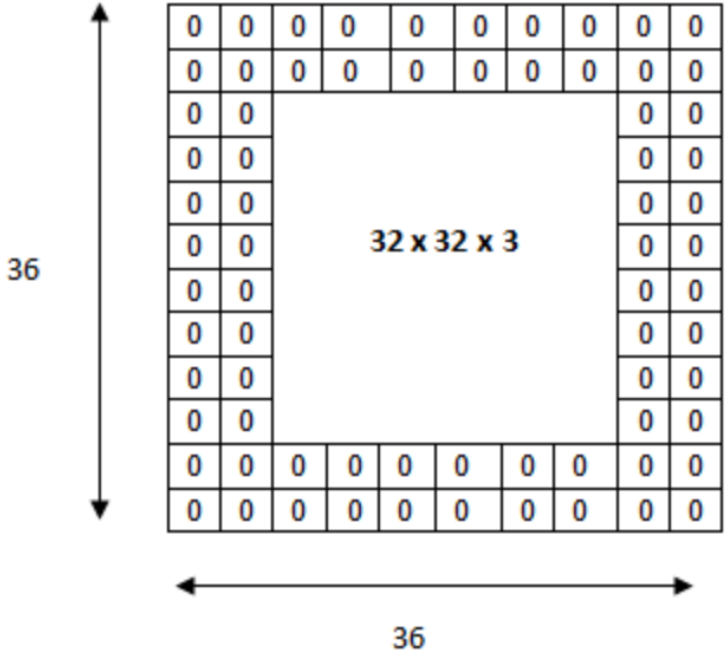
Stride 2

Depth and Zero Padding

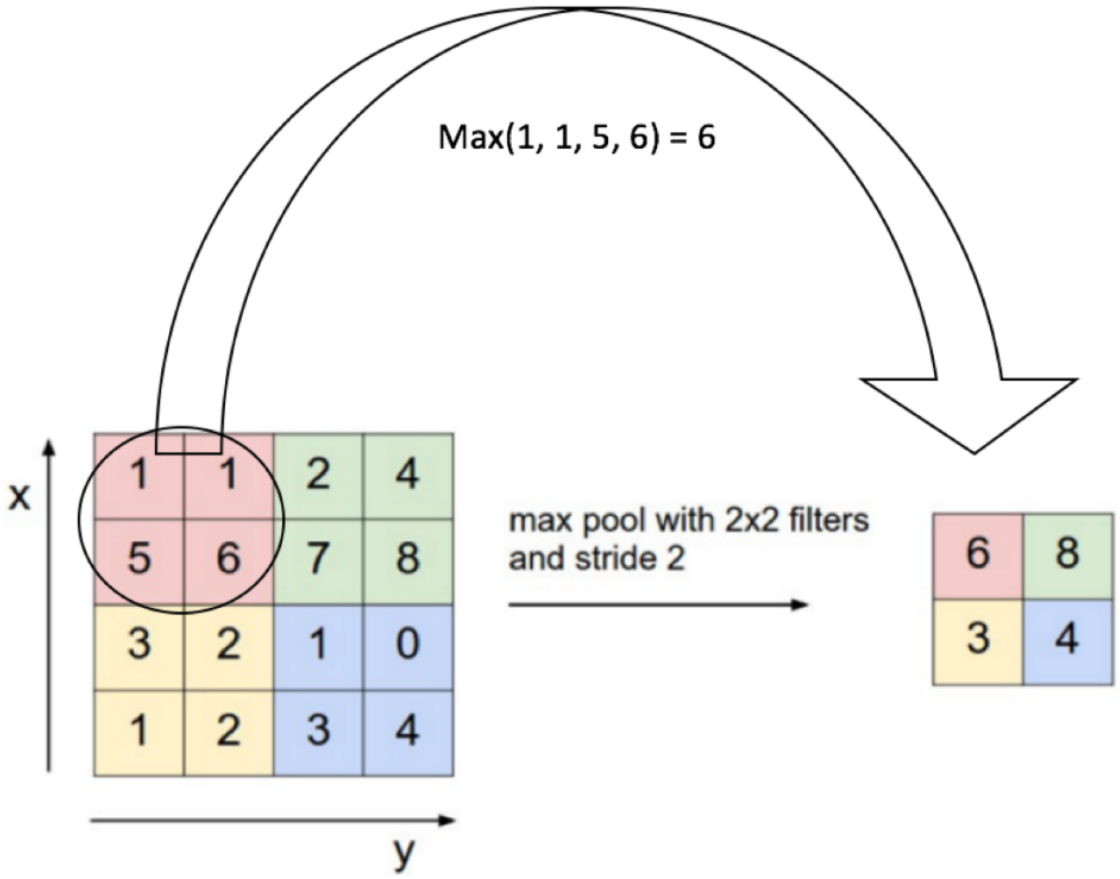
- Depth



- Padding



Pooling

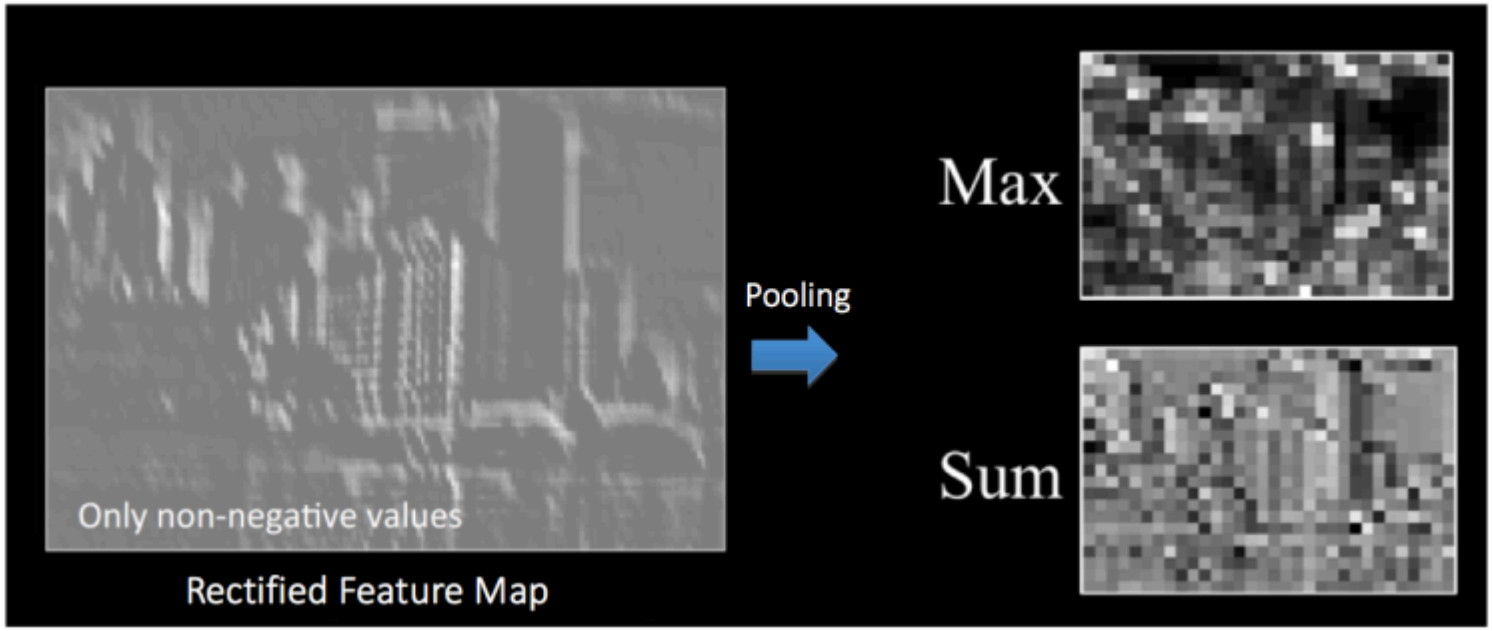


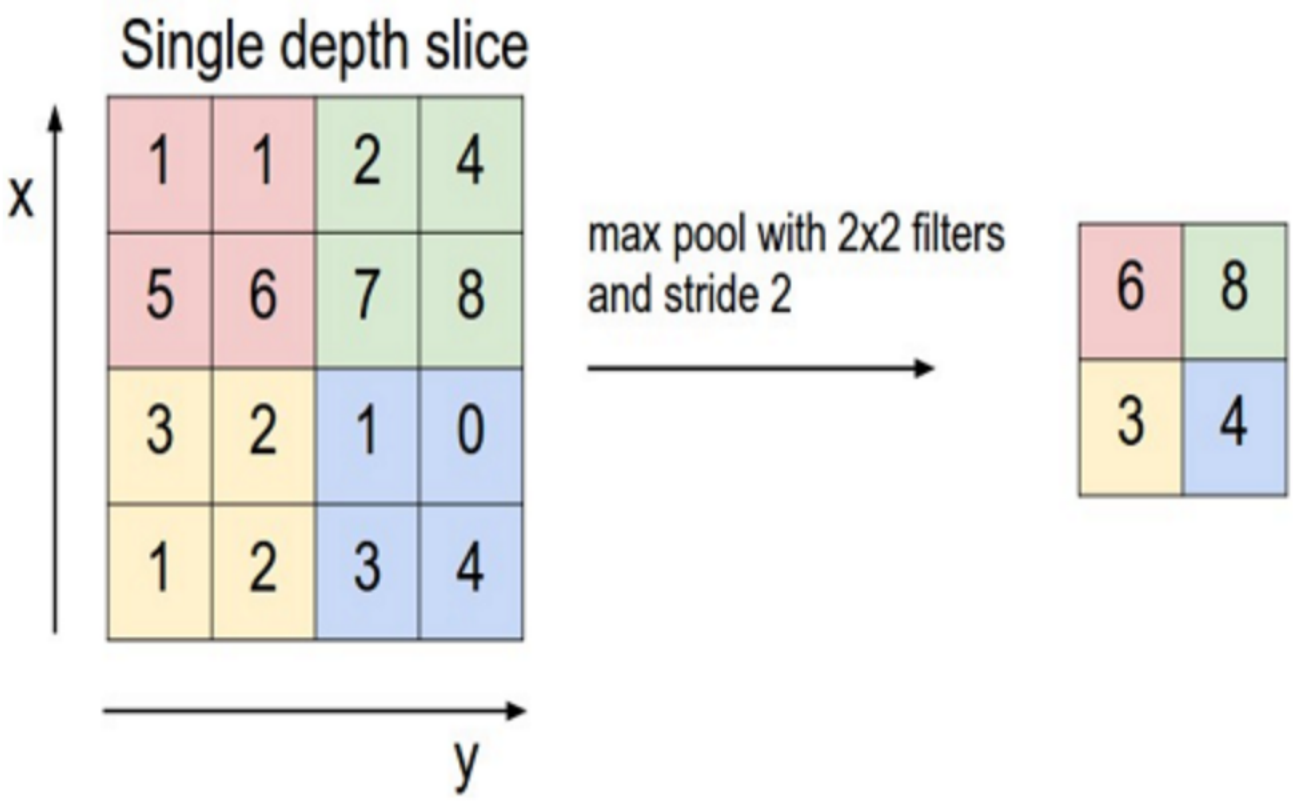
Downsampling the results by pooling values by averaging or max

Max pool is used more commonly

Effects of Pooling

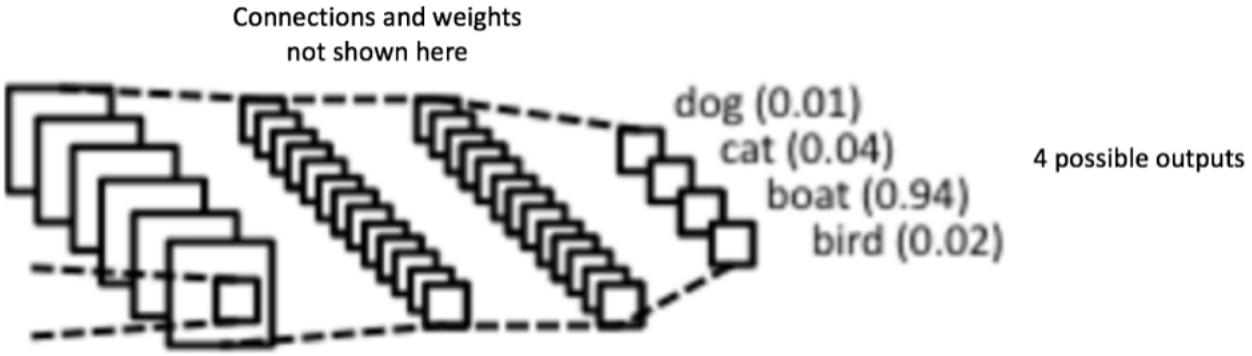
- Reducing spatial size of the image
- Making CNNs invariant to distortions and noise
- Reduces parameters
- Scale invariance



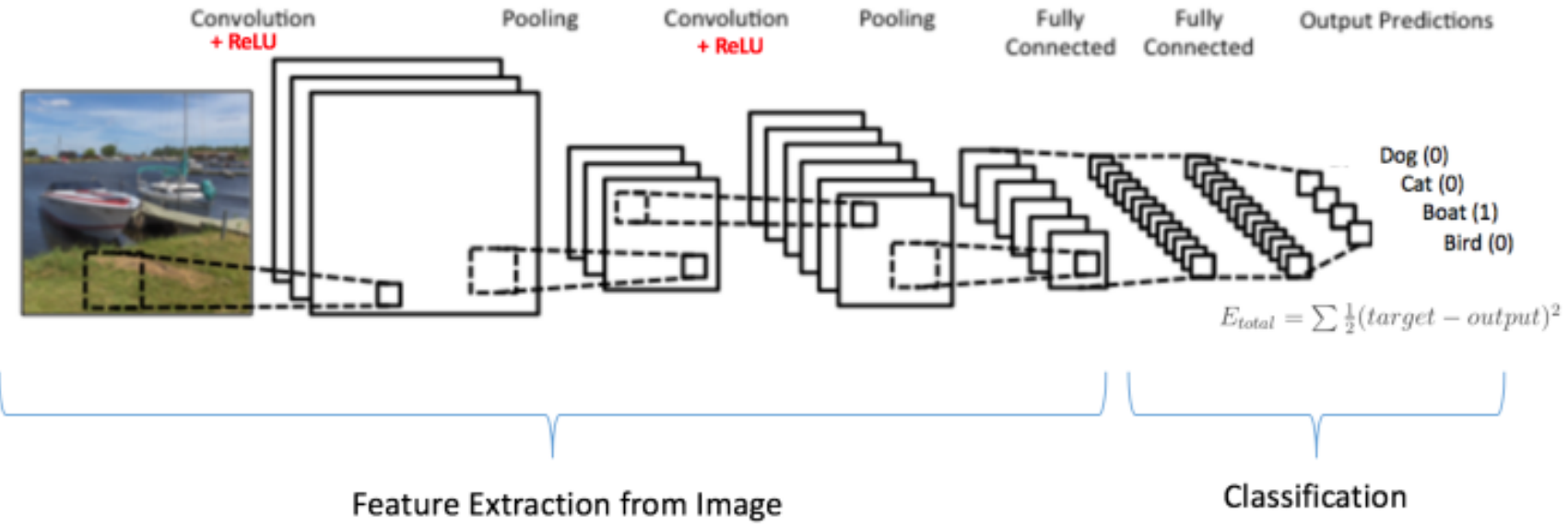

Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Fully connected layer to combine

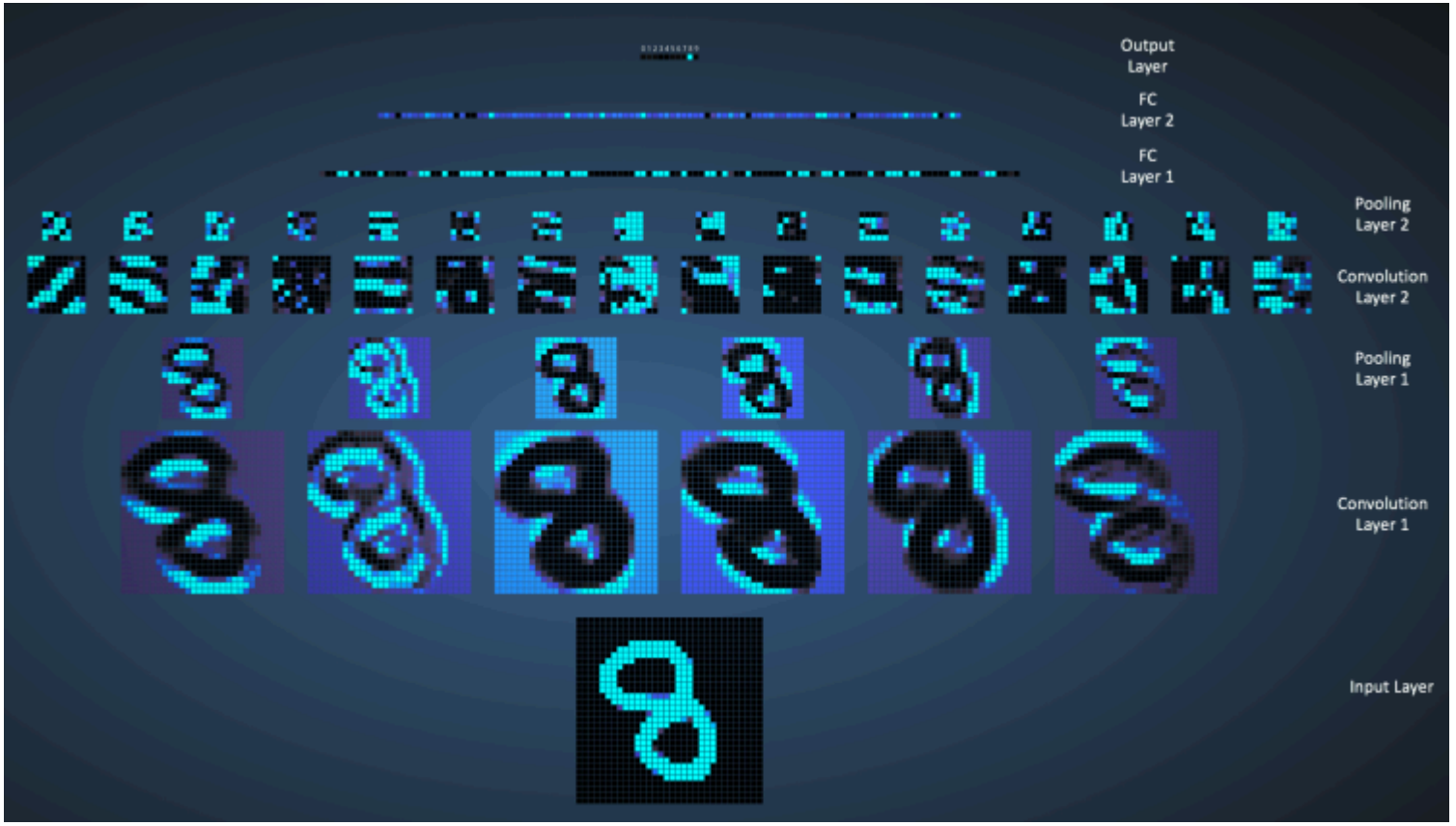
- Convolutional layers detected features
- Pooling layers reduced complexity
- Now we have a set of feature maps
- Can combine them nonlinearly to classify



Full Network



Visualizing a CNN



CNNs use three basic ideas

1. Local receptive fields
2. Shared weights
3. Pooling

Local receptive fields

Standard Neural Network (all nodes are connected, input is a vector)

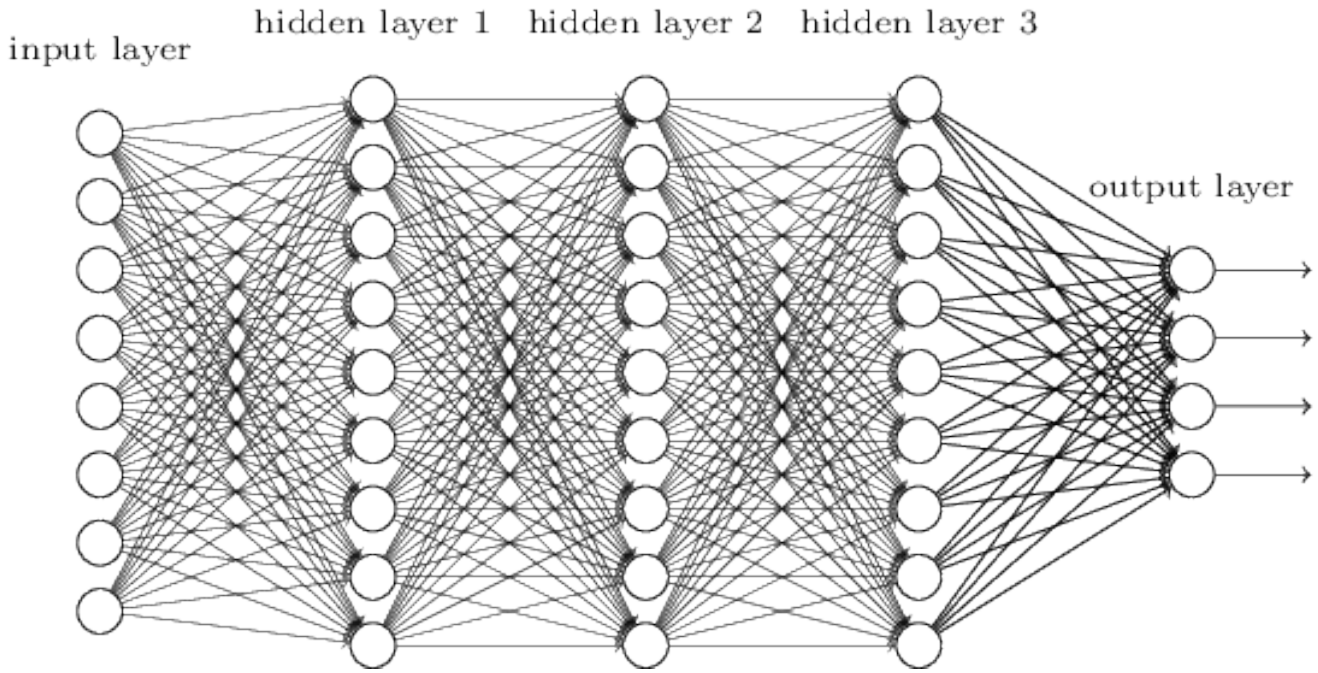


Image taken from Michael Neilsen's book "Neural Networks and Deep Learning"

Local receptive fields

View input as an image

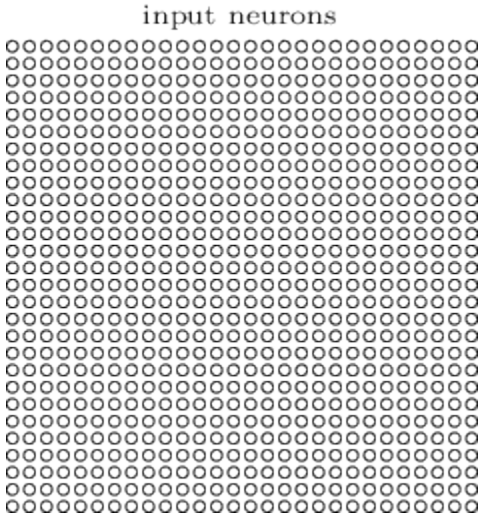


Image taken from Michael Neilsen's book "Neural Networks and Deep Learning"

Local receptive fields

Make connections in small, localized regions of the input image

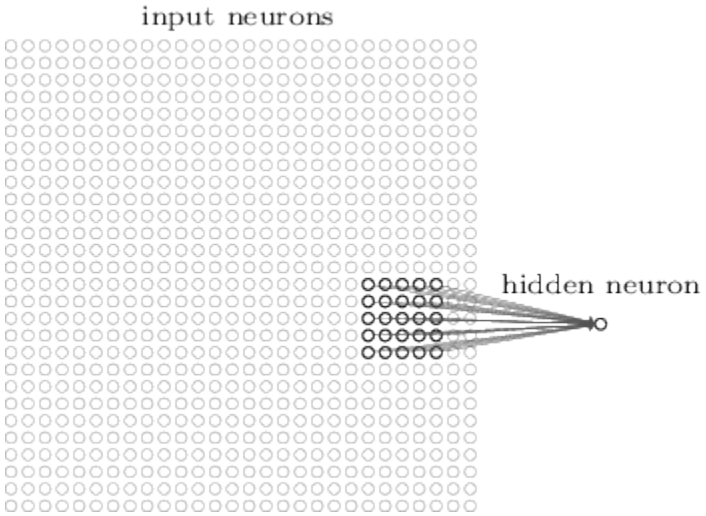


Image taken from Michael Neilsen's book "Neural Networks and Deep Learning"

Local receptive fields

Make connections in small, localized regions of the input image

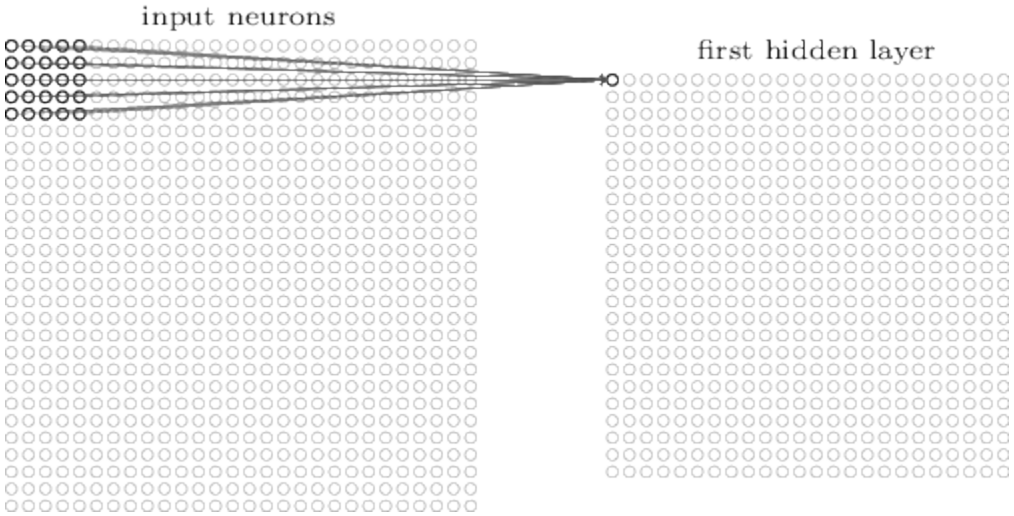


Image taken from Michael Neilsen's book "Neural Networks and Deep Learning"

Local receptive fields

Slide the local receptive field over by one (or more) pixel and repeat

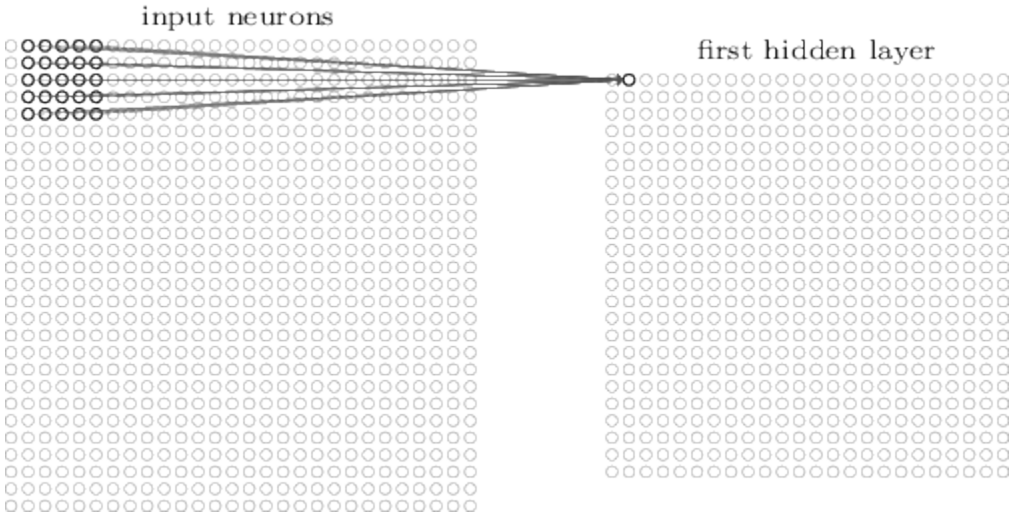


Image taken from Michael Neilsen's book "Neural Networks and Deep Learning"

Shared weights and biases

- Same weights and biases are used within the hidden layer
- E.g., output of the j, kth hidden neuron is

$$\sigma\left(b + \sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m} a_{j+l,k+m}\right)$$

  - A convolutional operator
- All neurons in the first hidden layer detect the same feature (helps with translations)

Shared weights and biases

A single layer in a CNN includes multiple feature maps

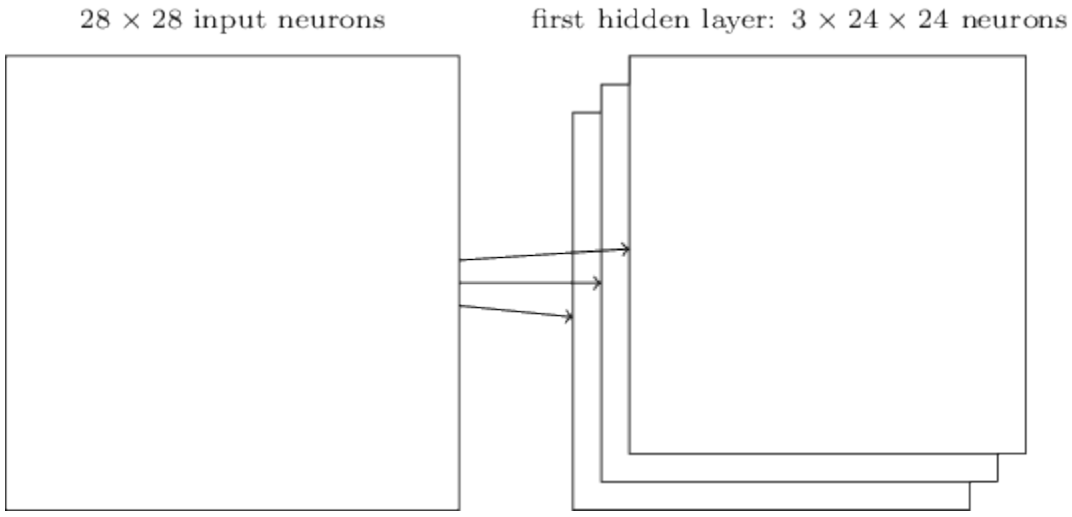


Image taken from Michael Neilsen's book "Neural Networks and Deep Learning"

Shared weights and biases

Some example features when trained on MNIST

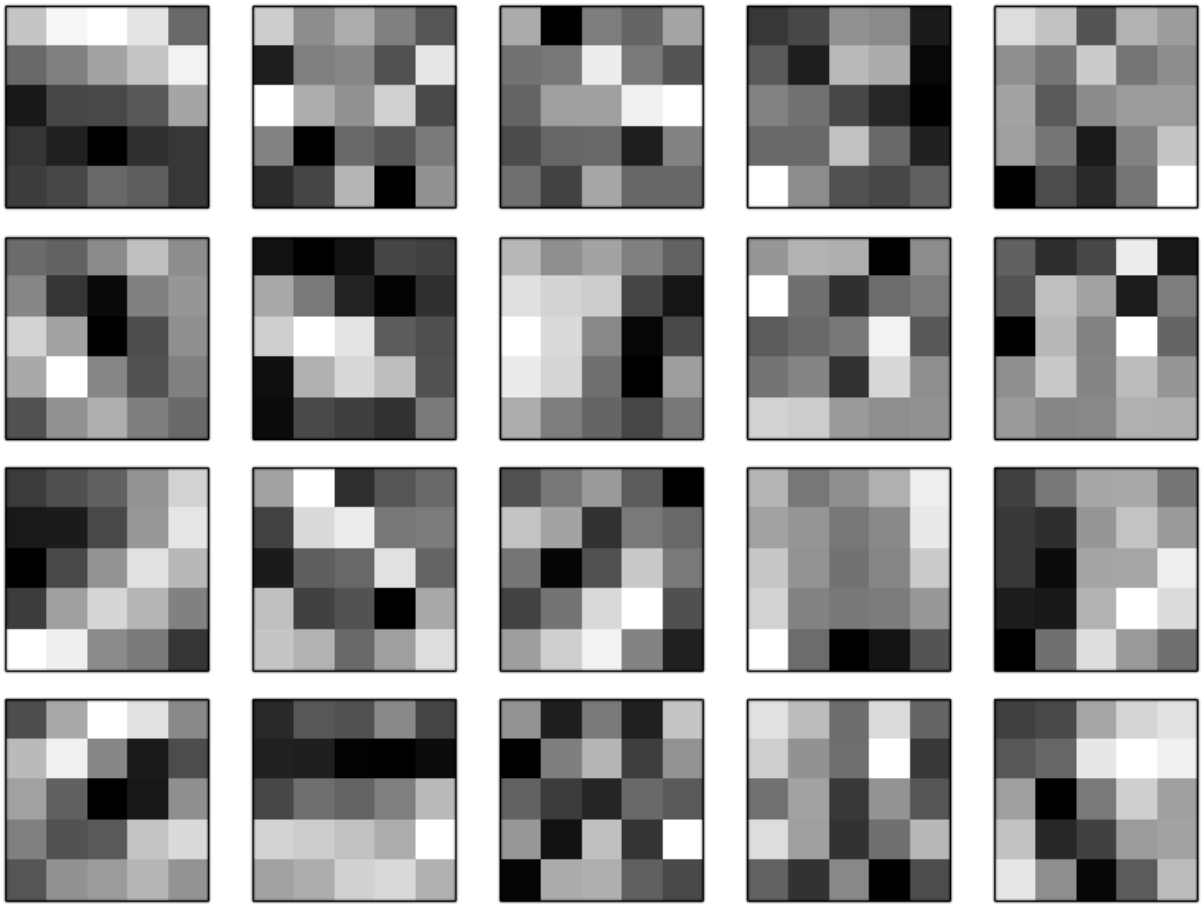


Image taken from Michael Neilsen's book "Neural Networks and Deep Learning"

Shared weights and biases

Why use shared weights?

- Greatly reduces the number of parameters (faster training)
- MNIST Example
  - CNN layer with 20 feature maps: 520 parameters
  - Fully connected neural net with 30 hidden neurons: 23,550 parameters

Shared weights and biases

Why use shared weights?

- Greatly reduces the number of parameters (faster training)
- MNIST Example
  - CNN layer with 20 feature maps: 520 parameters
  - Fully connected neural net with 30 hidden neurons: 23,550 parameters

Pooling layers

- Usually used immediately after convolutional layers
- Used to simplify the output from the convolutional layer
  - E.g., max-pooling outputs the maximum activation in a region

Pooling layers

Pooling applied to a feature map

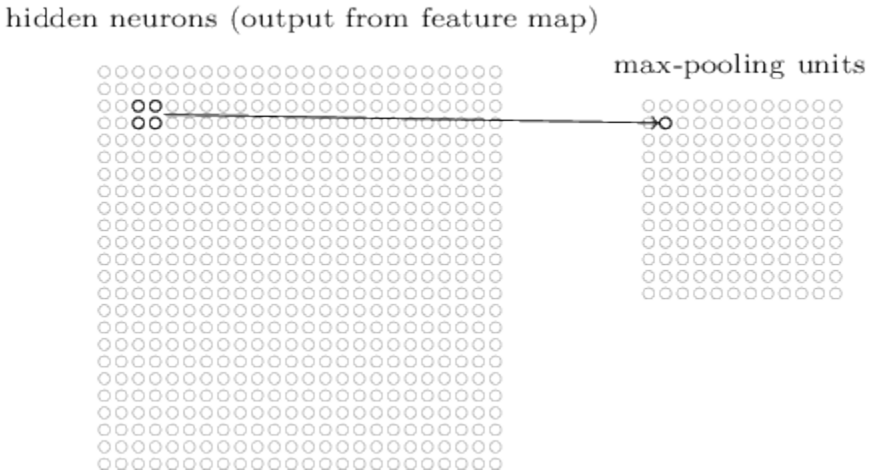


Image taken from Michael Neilsen's book "Neural Networks and Deep Learning"

Pooling layers

Pooling applied to multiple feature maps

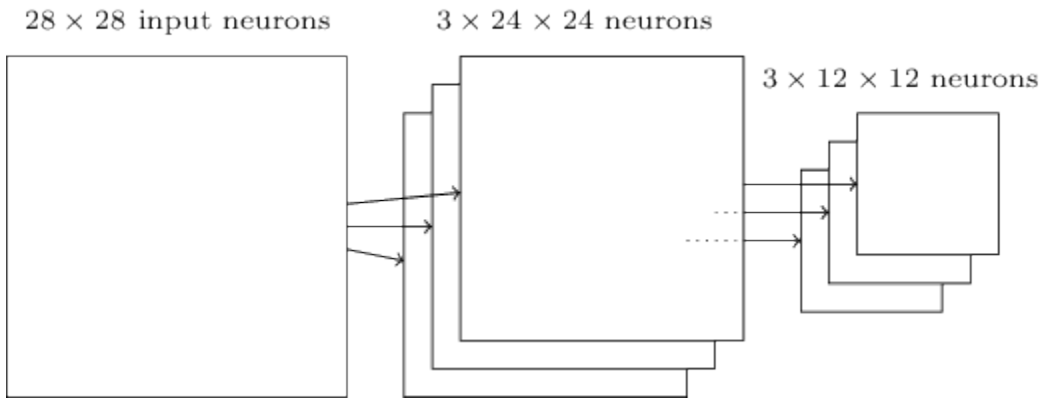


Image taken from Michael Neilsen's book "Neural Networks and Deep Learning"

Pooling layers

- Intuition: the exact location of a feature isn't as important as its rough location
  - Helps prevent overfitting
- Reduces the number of parameters needed in later layers
- L_2 pooling is also common (L_2 norm)

A full CNN

Slight modifications required for back propagation

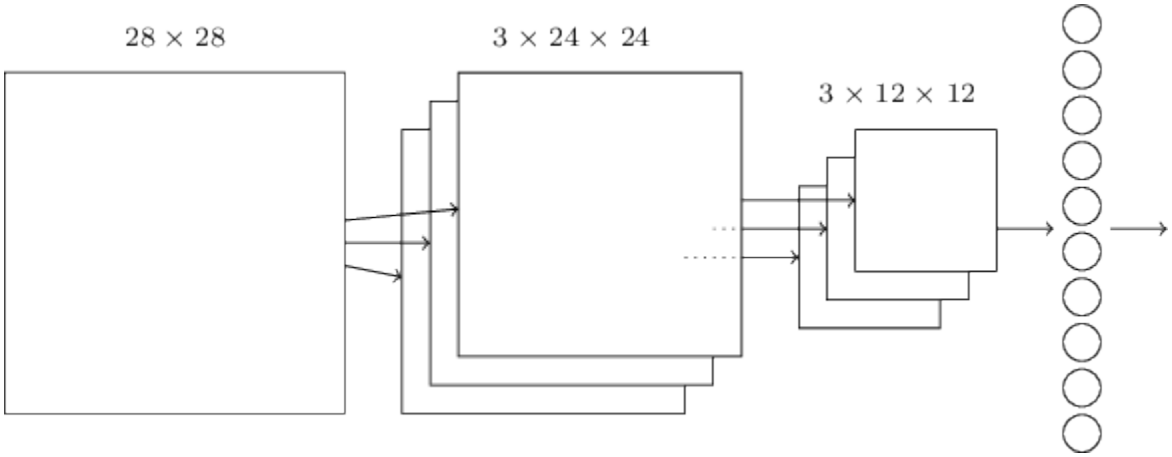
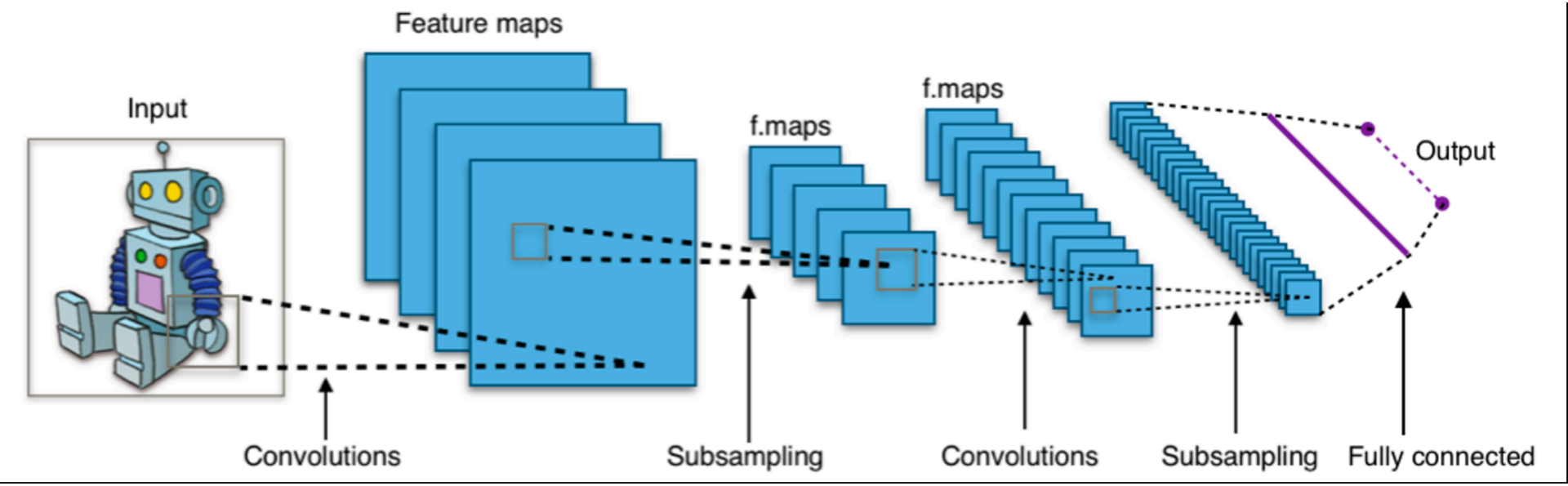


Image taken from Michael Neilsen's book "Neural Networks and Deep Learning"

CNN Architecture



Each convolutional unit has the same weight.

Other Modifictions: Inception

- GoogleNet 2014: Introduced the Inceptuion module that reduced parameters from 60 M to 4M (order of magnitude)
- ResNet 2015: Residual network

Inception Network

- CNNs drastically increased popularity of neural networks, due to their empirical success
- One of the most popular: Inception
  - Deep classification network
  - Set state-of-the-art benchmarks on the Imagenet dataset

(a) Siberian husky

(b) Eskimo dog

Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Inception Network

- Built on previous CNN architectures, but engineered it up to scale
  - Large networks, large data = “smart” model
- Training a large network can be harder than training a small network
  - Sometimes they actually perform worse, as a result
- They built an “Inception Block” to address this

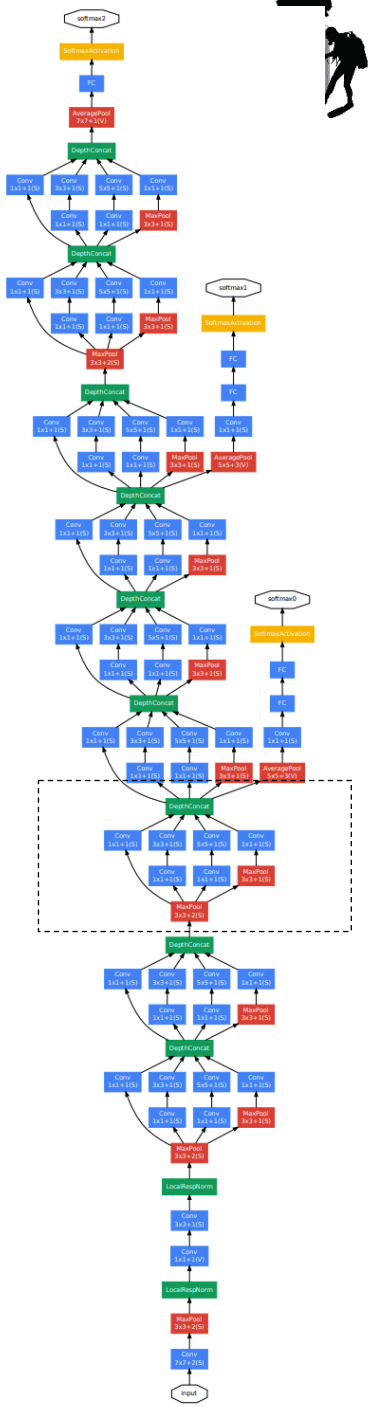
Inception Architecture

Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Inception Network

- Built on previous CNN architectures, but engineered it up to scale
  - Large networks, large data = “smart” model
- Training a large network can be harder than training a small network
  - Sometimes they actually perform worse, as a result
- They built an “Inception Block” to address this

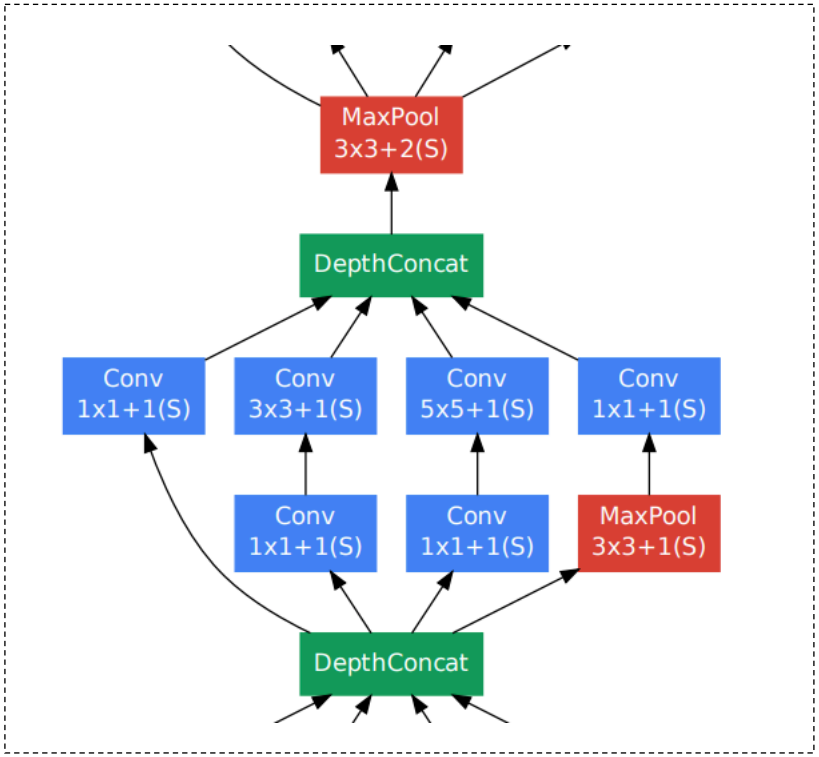
Inception Architecture



Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

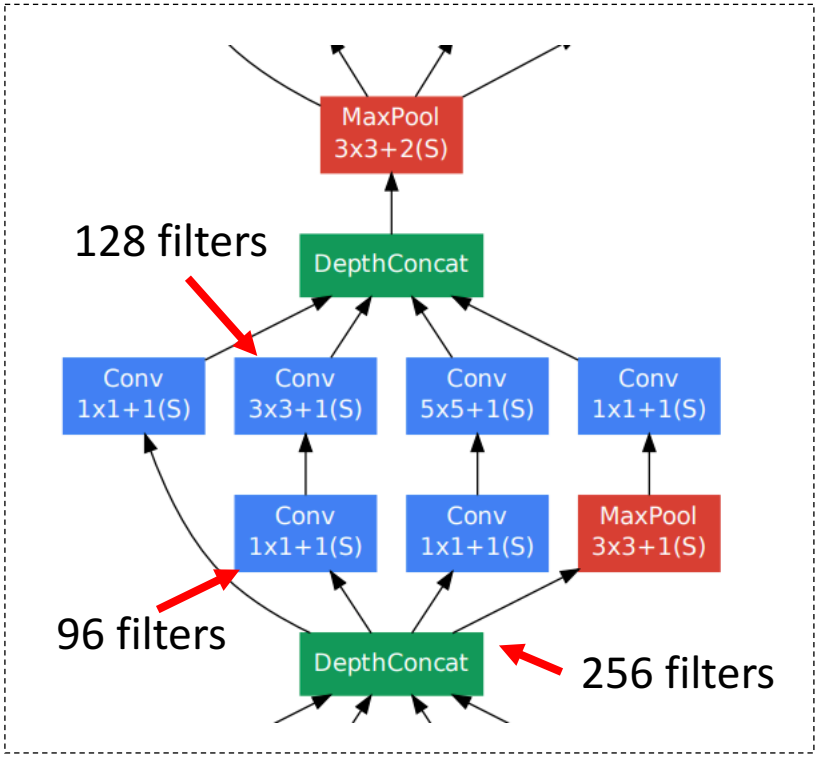
Inception Network

- Combine convolutions of different spatial size (centered at each pixel)
- If naively done, this will greatly expand the dimensionality at each step and thus the number of parameters
  - Each convolutional filter is connected to all filters in the input
- So instead, reduce the dimensionality of the input before being put into the 3x3 and 5x5 convolutions



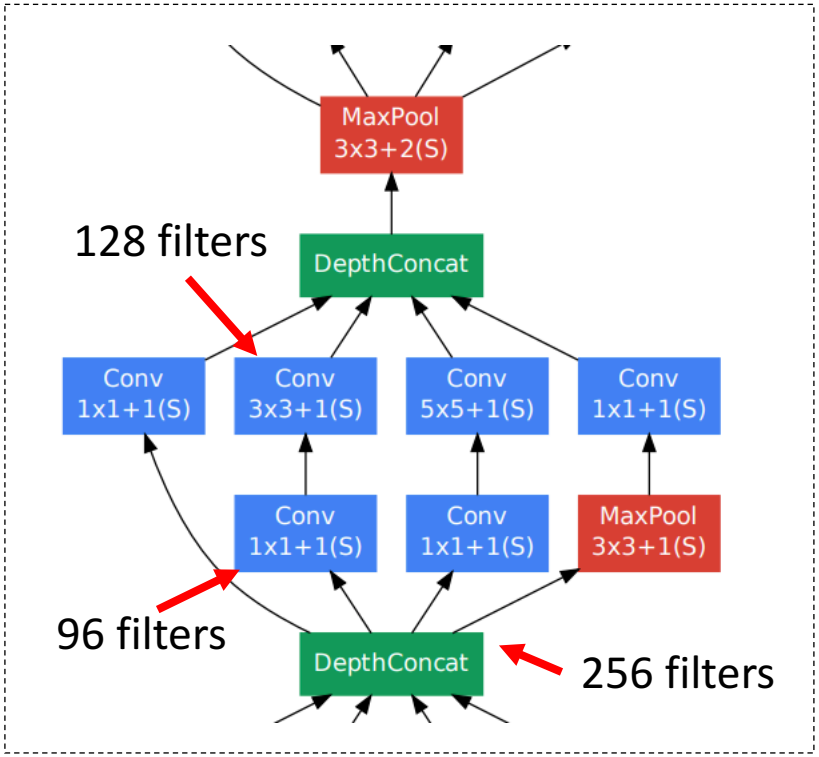
Inception Network

- Combine convolutions of different spatial size (centered at each pixel)
- If naively done, this will greatly expand the dimensionality at each step and thus the number of parameters
  - Each convolutional filter is connected to all filters in the input
- So instead, reduce the dimensionality of the input before being put into the 3x3 and 5x5 convolutions



Inception Network

- What is a 1x1 convolution?
  - I'm pixel at the position [322, 10]
  - Look at all the input dimensions for pixel [322, 10] in the previous layer
  - Do a standard non-linear neuron activation to produce an output number
  - Move to pixel [322, 11] and perform the *same* operation, but with its input values instead
- Important: if there are *n* filters coming in and you are learning *m* 1x1 convolutions, there are only *mxn* weights
  - Whether there are 784 pixels or 1,000,000+!



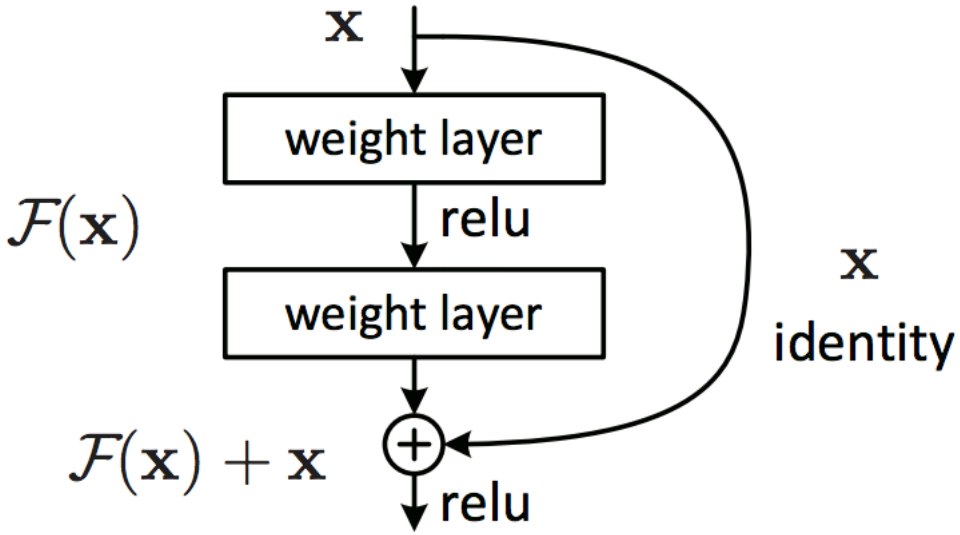
ResNets

- Inception network performed very well
- However, making it even deeper did not improve performance
- This motivated a new type of block
  - One that you can stack more and more of and get better performance with each additional block
- Gradient flow through the network needs to be improved
- Idea: use “skip connections”

He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

ResNets

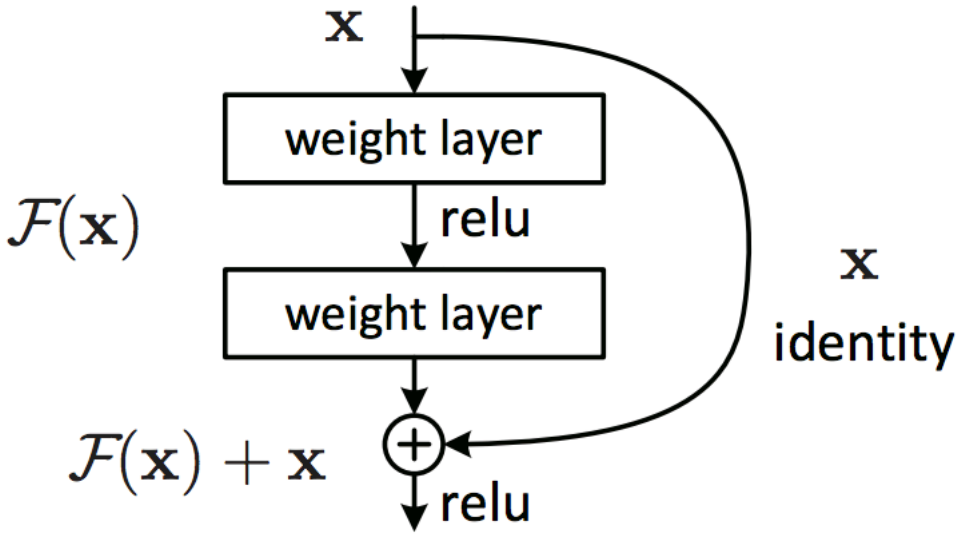
- The normal nonlinearity is learned and then *added* to the input
- Instead of the activation being ***a = F(x)*** it is ***a = F(x) + x***



He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

ResNets

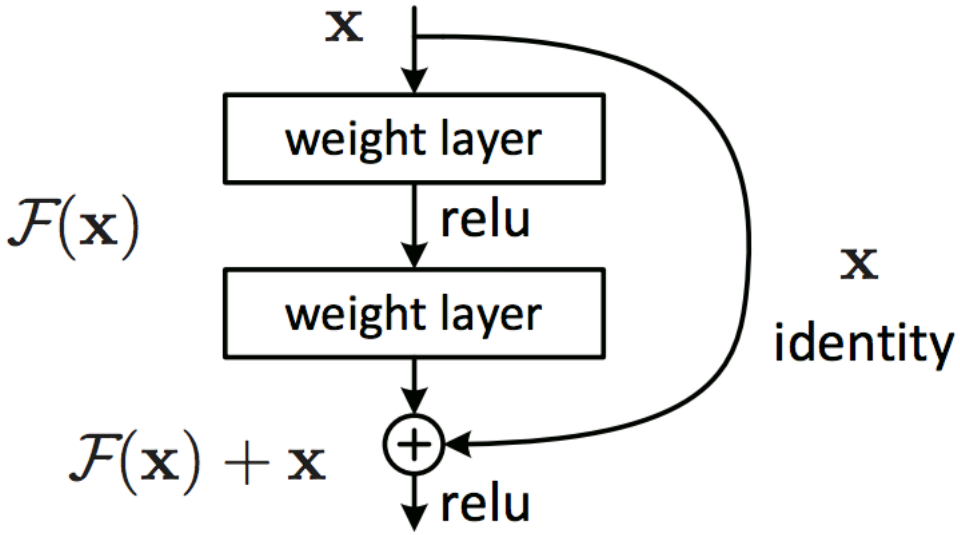
- How does this address vanishing gradients?



He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

ResNets

- How does this address vanishing gradients?
- No matter what the derivative of ***F*** is, the derivative going through the identity branch is constant



He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

ResNets

- This allows for training **much** deeper networks
- But is each layer still as expressive as a regular layer? How does the network perform?

- Various architectures of ResNets at just 34 layers already outperform Inception (GoogLeNet)
- But now we can train even up to **152** layers!
- More layers give better performance

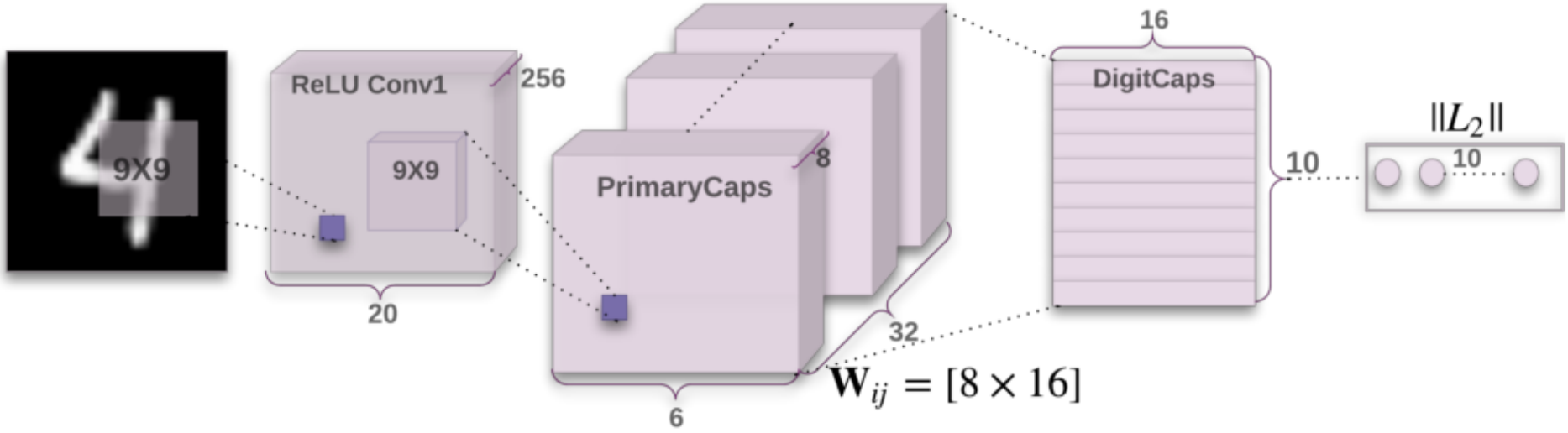
model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Table 3. Error rates (%, **10-crop** testing) on ImageNet validation. VGG-16 is based on our test. ResNet-50/101/152 are of option B that only uses projections for increasing dimensions.

He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Capsule Nets

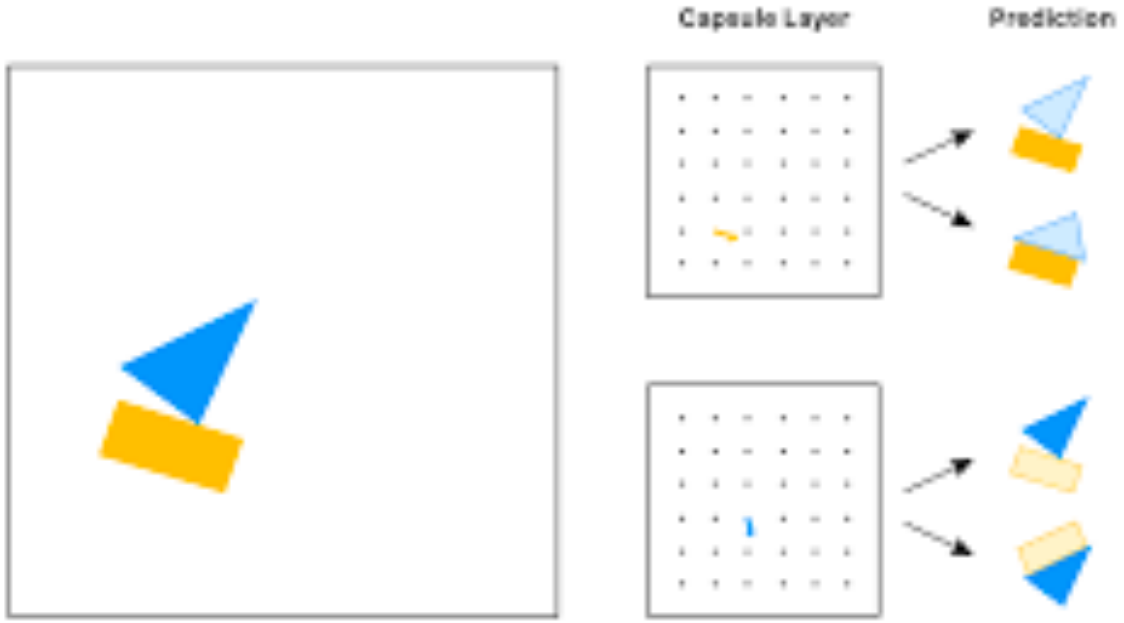
- Each output neuron is a vector (capsule) instead of a scalar
- Not optimized by gradient descent, but by “dynamic routing”



https://medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Capsule Nets

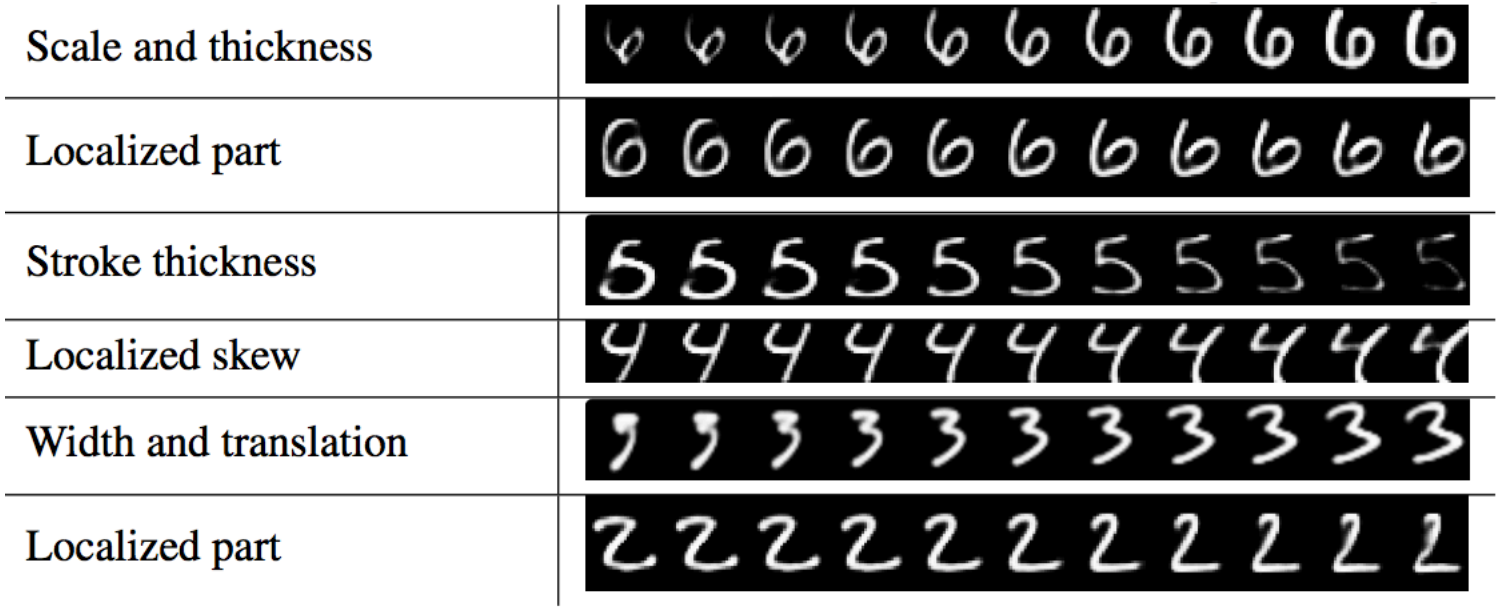
- Dynamic routing:
  - One capsule identifies blue triangle
  - Another capsule identifies yellow rectangle
  - Is it a house or a sailboat?
  - They need to agree, and thus need to know what the other ones thinks
  - Dynamic routing passes information between them



Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic routing between capsules." *Advances in neural information processing systems*. 2017.

Capsule Nets

- Different capsules represent different aspects of variation within the data



Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic routing between capsules." *Advances in neural information processing systems*. 2017.

Suggested Readings

- Chapter 6 Nielsen
- http://cs231n.github.io/convolutional-networks/
- https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
- Inception https://arxiv.org/abs/1409.4842
- ResNet https://arxiv.org/abs/1512.03385