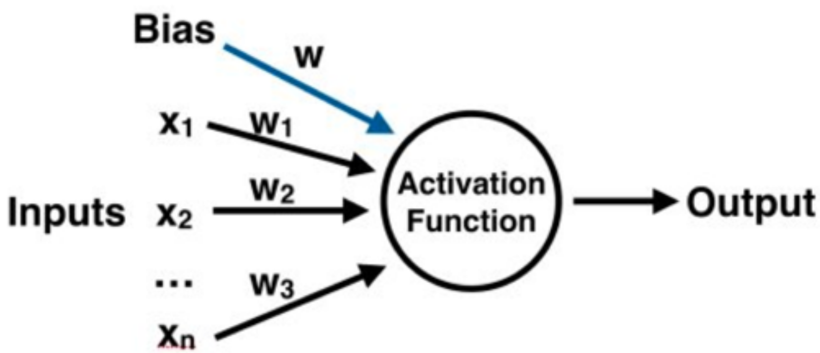
DUKE FUQUA SCHOOL OF BUSINESS

Sudipta Dasmohapatra
sd345@duke.edu

Neural Network and Deep Learning Workshop

Spring 2020

Neural Networks: Weights and Activation Functions



$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j + b \leq 0 \\ 1 & \text{if } \sum_j w_j x_j + b > 0 \end{cases}$$

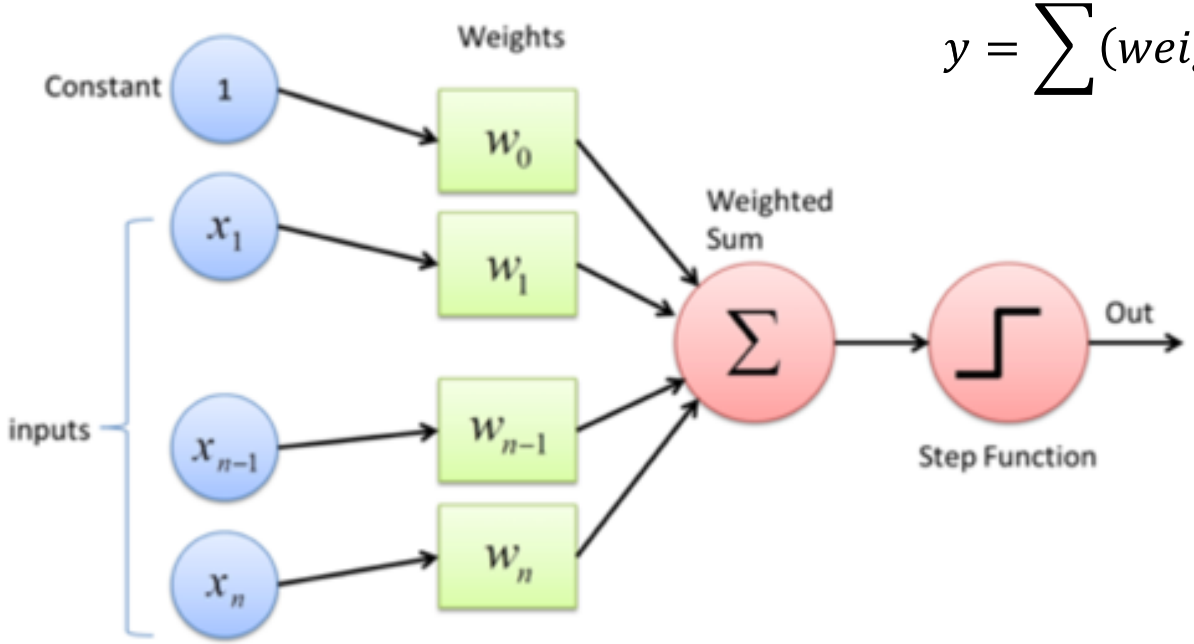
Simply y = $\sum_j w_j x_j + b$

$$y = \sum (weight * input) + bias$$

In this equation b denotes the intercept (also known as bias, and technically a weight itself) and w and x are vectors carrying the weights and values from all inputs

Activation Function

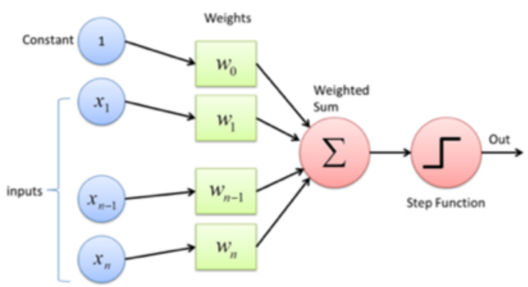
$$y = \sum(weight * input) + bias$$



A perceptron works on simple steps:

1. All the inputs x are multiplied with their weights w, lets call it k
2. Add all the multiplied values = weighted sum
3. Apply that weighted sum to the correct activation function

Activation Function: Step Function

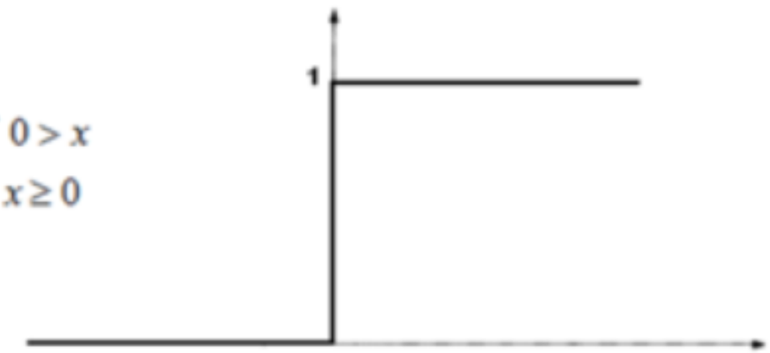


$$y = \sum(weight * input) + bias$$

If value of y is above a threshold > activated
A (function A) =1 if y> threshold, 0 otherwise

Unit step (threshold)

$$f(x) = \begin{cases} 0 \text{ if } 0 > x \\ 1 \text{ if } x \geq 0 \end{cases}$$



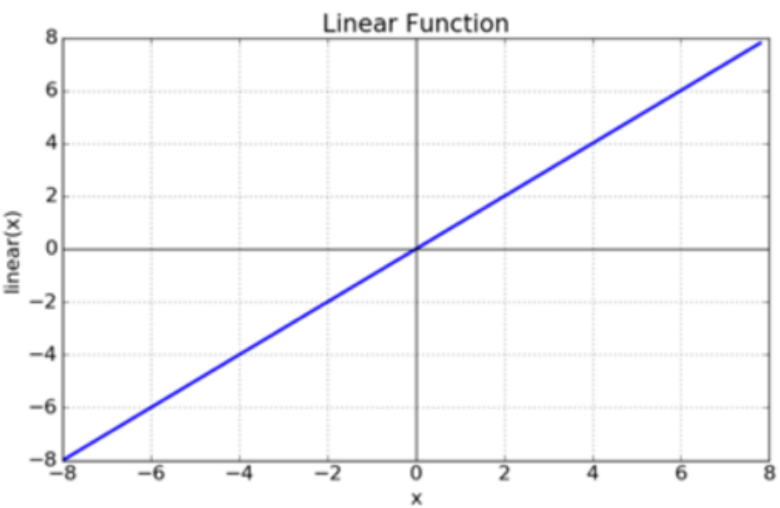
Disadvantage: Multiple classes in outcome (what if multiple neurons are activated)

DUKE FUQUA SCHOOL OF BUSINESS

Activation Functions

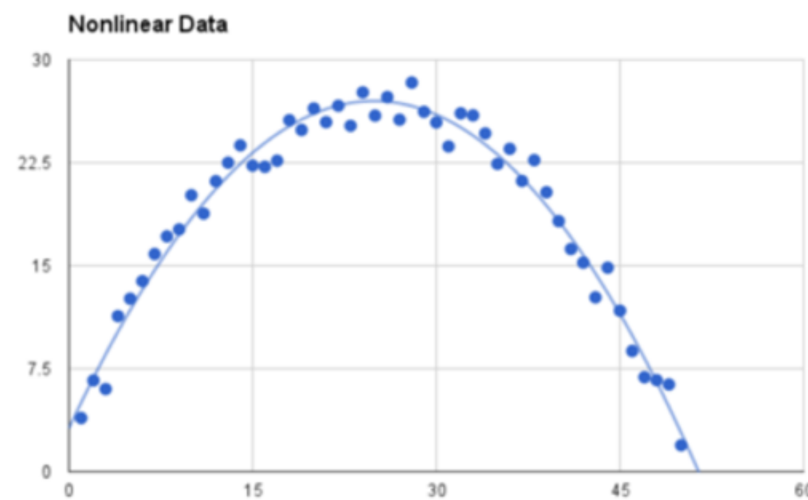
- Maps the resulting values in between output values

Linear activation



A=cx
Equation: f(x) = x
Range: (-infinity to infinity)

Non-linear activation



Makes the model to generalize with variety of data and differentiate between output

Non Linear Activation Functions

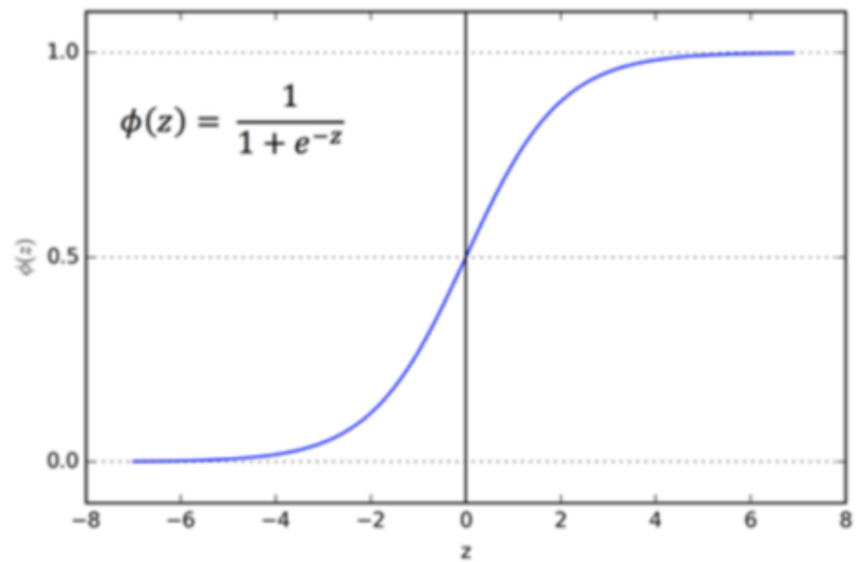
Logistic (A=$\frac{1}{1+e^{-x}}$)

- Smooth gradient
- Output between 0 and 1 (wouldn't blow up activations)
- Between values -2 and +2,

y is steep (any small change in x changes y significantly) but either ends, the y changes slow

= gradient will be small

= network refuses to learn further



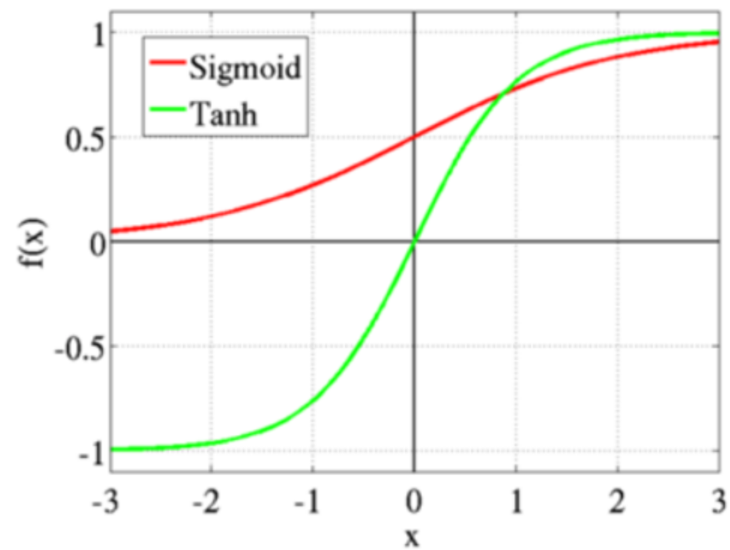
Softmax function: more generalized function for multiclass classification

DUKE FUQUA SCHOOL OF BUSINESS

Non Linear Activation Functions

Tanh or hyperbolic tangent

$$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$$



- Output between -1, +1
- Derivatives are steeper

(gradient is stronger)

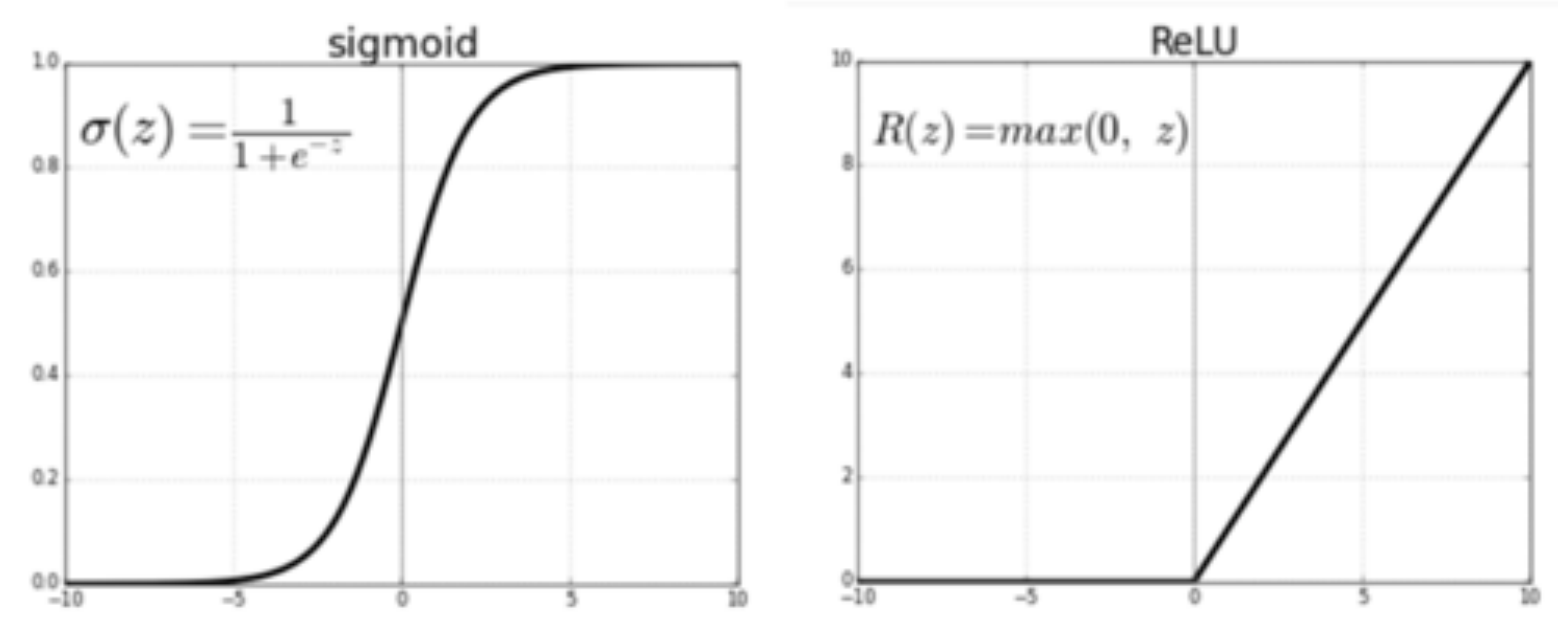
- Also has vanishing gradient problem

Negative inputs will be mapped strongly negative and the zero inputs will be mapped near 0

DUKE FUQUA SCHOOL OF BUSINESS

Non Linear Activation Functions

ReLU (Rectified Linear Unit)



- The ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero
- Range is 0-infinity
- Less computationally expensive compared to sigmoid or tanh = great for deep neural networks

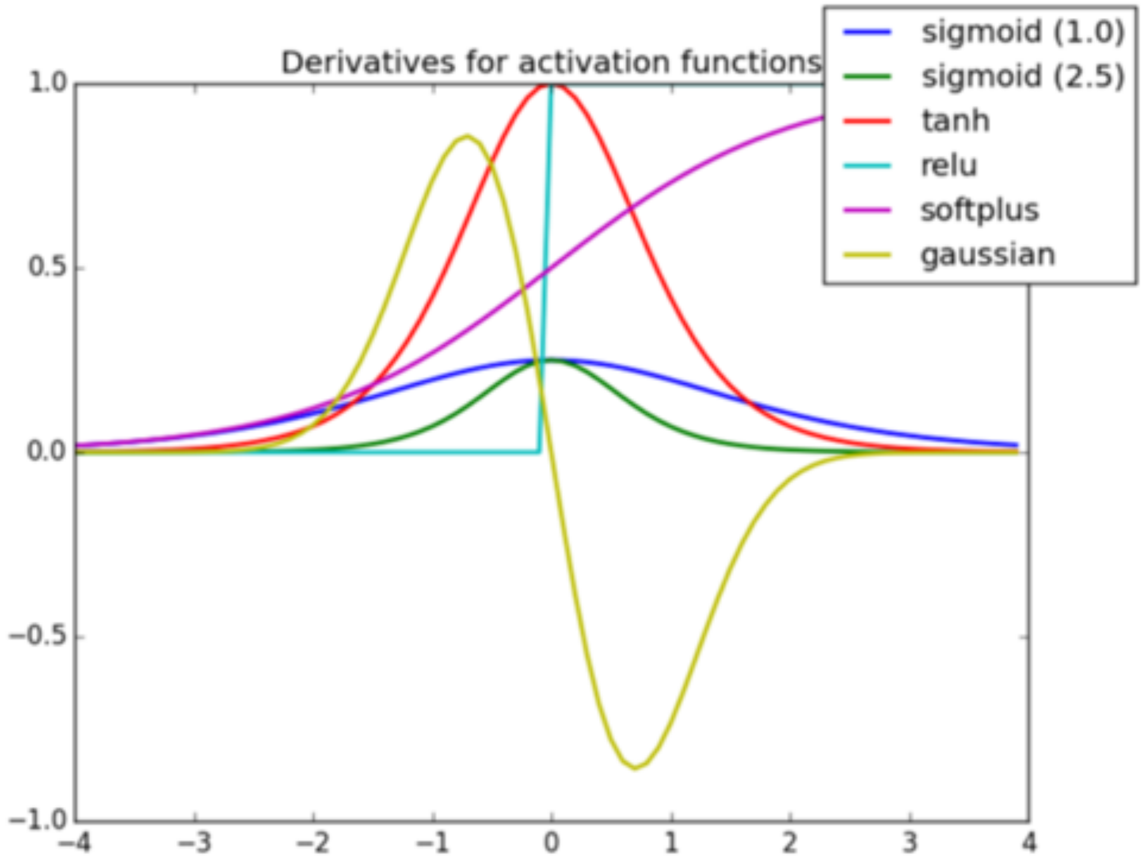
DUKE FUQUA SCHOOL OF BUSINESS

List of Activation Functions

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

DUKE FUQUA SCHOOL OF BUSINESS

Derivatives of Activation Functions

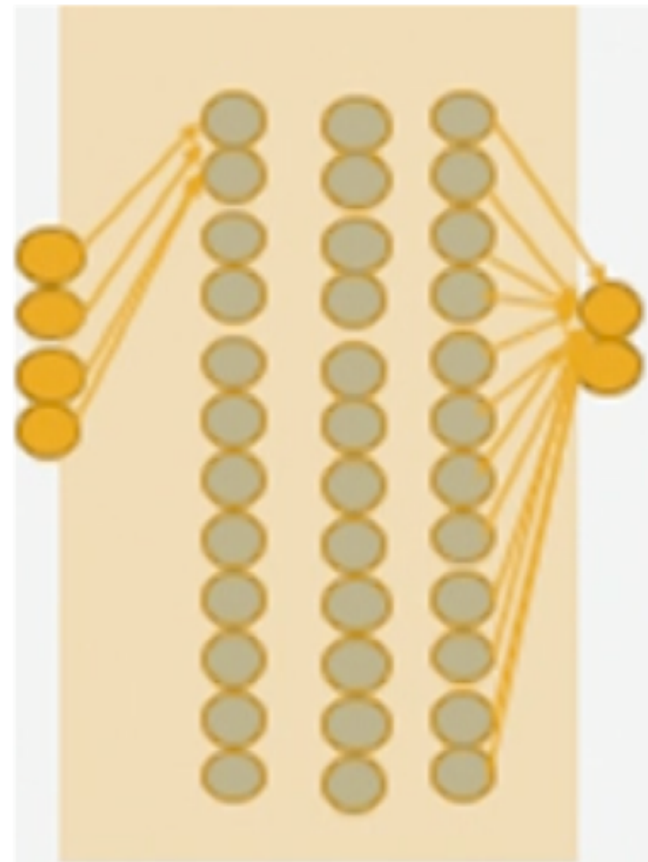


Back to it: What is a Deep Learning Neural Network?

Collection of inputs, wired to some central layers of perceptrons, and then to a desired number of inputs

There is some element of brute force high computing power to these approaches

Intense data requirements because there are vast numbers of parameters



DUKE FUQUA SCHOOL OF BUSINESS

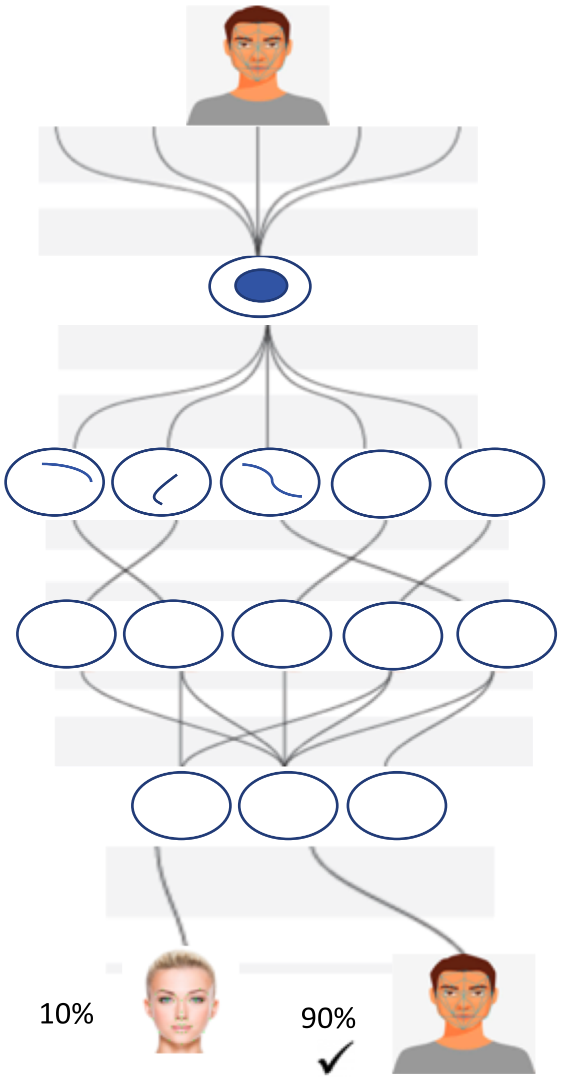
Tools and Deep Learning Progress

- Perceptron 1960
- Torch
- CUDA
- Theano
- TensorFlow 0.1 2015
- PyTorch 0.1 2017
- TensorFlow 1.0 2018
- PyTorch 1.0 2018
- TensorFlow 2.0 2019

- Perceptron 1957
- Backpropagation, RNN
- CNN, RNN
- Deep Learning 2006
- ImageNet 2009
- DeepFace 2014
- AlphaGo 2016
- BERT (Google) 2018

DUKE FUQUA SCHOOL OF BUSINESS

Deep Learning: How Neural Networks Recognize an Object



Training: During this phase, a neural network is fed thousands of labeled images of various faces, learning to classify them

Input: An unlabeled image is shown to a pretrained network

First Layer: the neurons respond to different simple shapes like edges

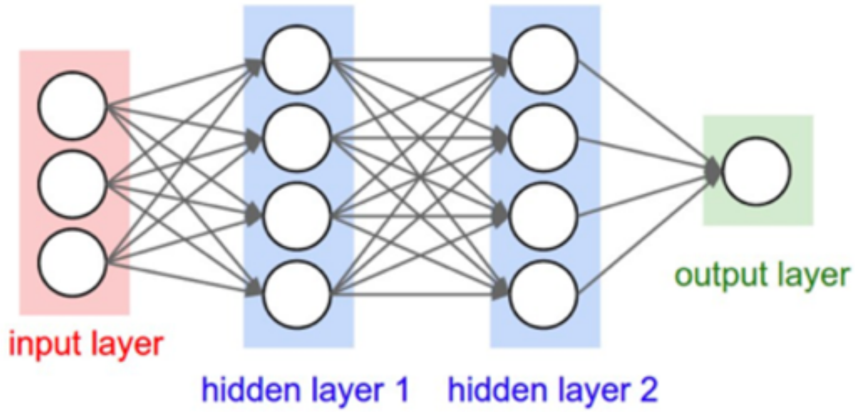
Higher Layer: Neurons respond to more complex structures such as nose, lips, forehead

Top Layer: Neurons respond to highly complex abstract concepts that we would identify as different faces

Output: The network predicts what the object most likely is, based on its training

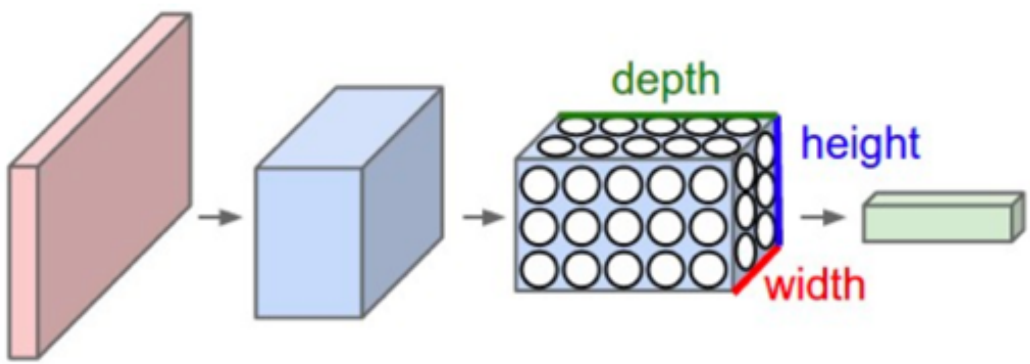
DUKE FUQUA SCHOOL OF BUSINESS

Convolutional Neural Network



Regular 3-layer NN

The convolutional neural networks are formed by neurons that have parameters in the form of weights and biases that can be learned

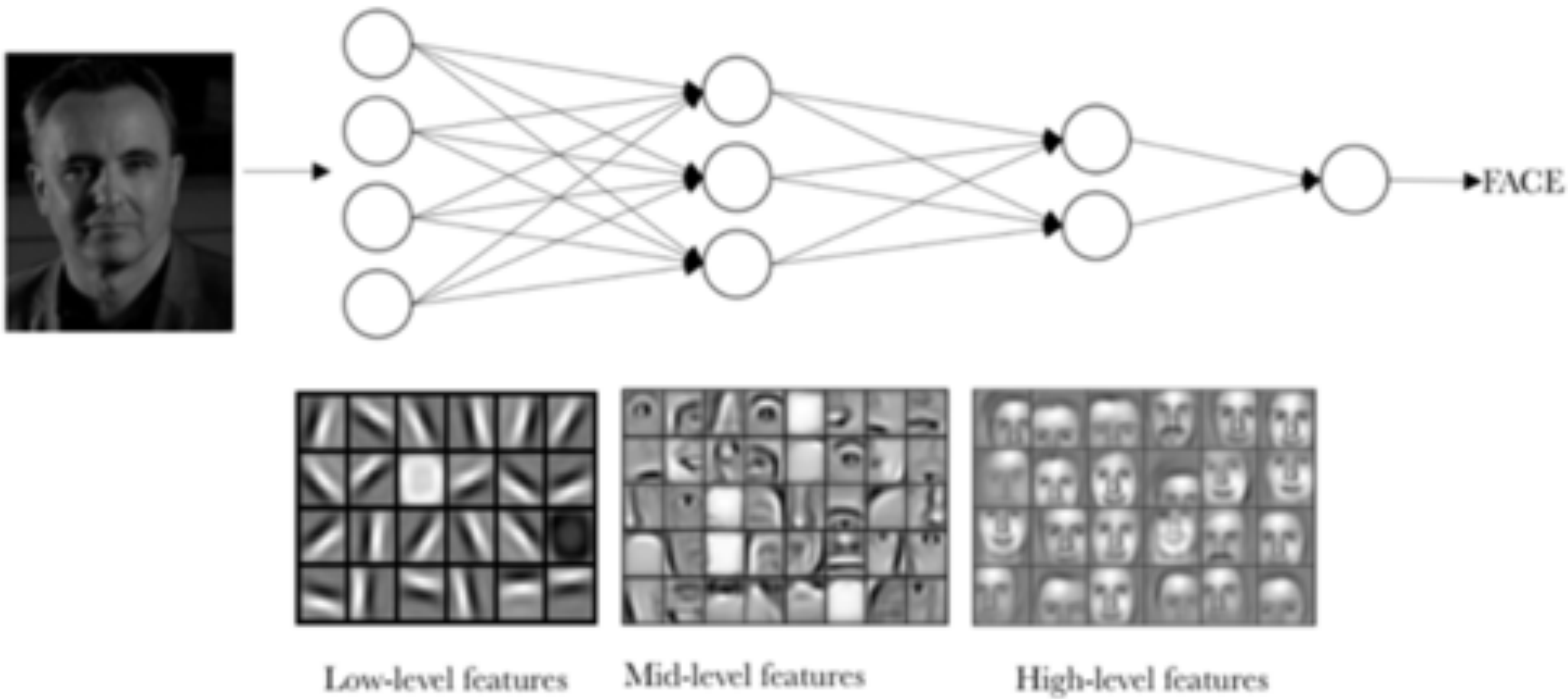


CNN

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

DUKE FUQUA SCHOOL OF BUSINESS

Convolutional Neural Network



Source: https://torres.ai/en/deeplearning/

CNN: Image Classification

What We See

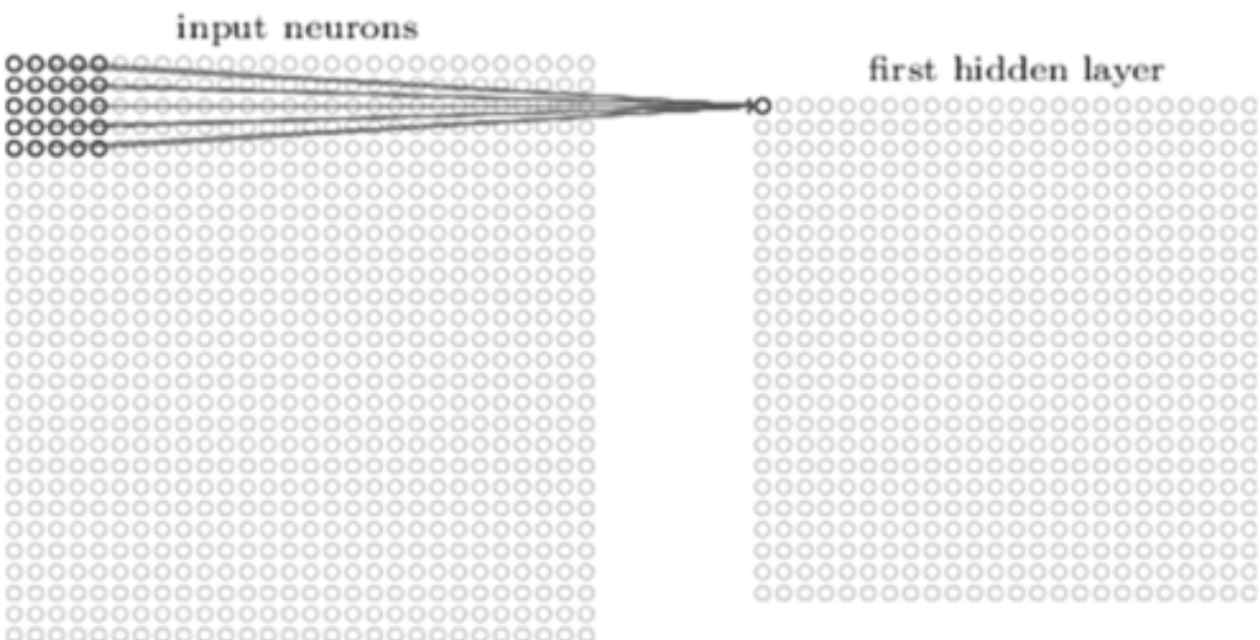
What Computers See

DUKE FUQUA
SCHOOL OF BUSINESS

CNN: Pixel Representation

What do you Want the Computer to do?

Take the image, pass it through a series of convolutional, nonlinear, pooling (or what is defined as downsampling), and fully connected layers, and get an output layer



Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

Image Processing: First Layer

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Visualization of a curve detector filter

Here you see a pixel representation of a filter which is a curve detection filter. Filters are feature identifiers.

DUKE FUQUA SCHOOL OF BUSINESS

Visualization of the First Layer

Original image

Visualization of the filter on the image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

In the input image, if there is a shape that generally resembles the curve that this filter is representing, then all of the multiplications summed together will result in a large value

DUKE FUQUA SCHOOL OF BUSINESS

Visualization of the First Layer

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

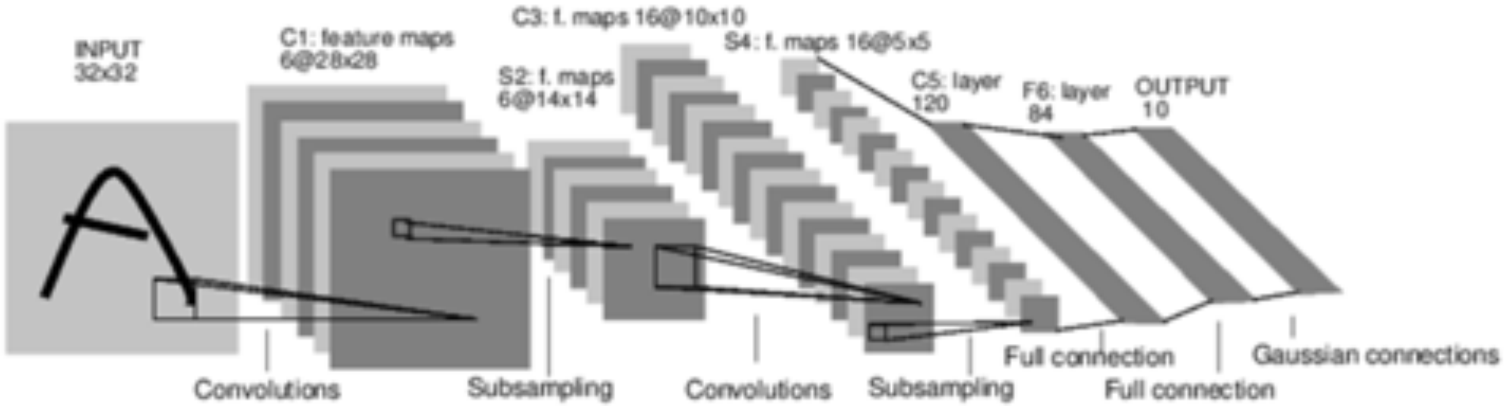
Multiplication and Summation = 0

There isn't anything in the image section that responded to the curve detector filter shown in earlier slide and thus the value in the activation map is zero

DUKE FUQUA SCHOOL OF BUSINESS

Going Deeper into the Layer

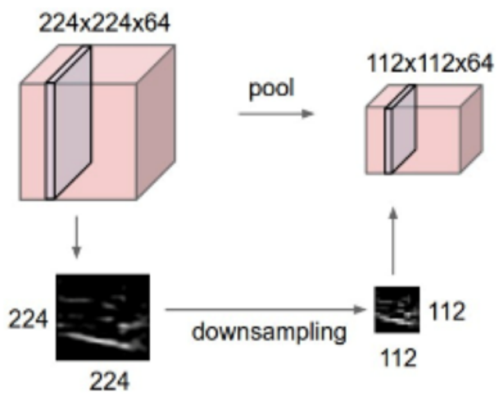
Input -> Conv -> ReLU -> Conv -> ReLU -> Pool -> ReLU -> Conv -> ReLU -> Pool ->Fully Connected



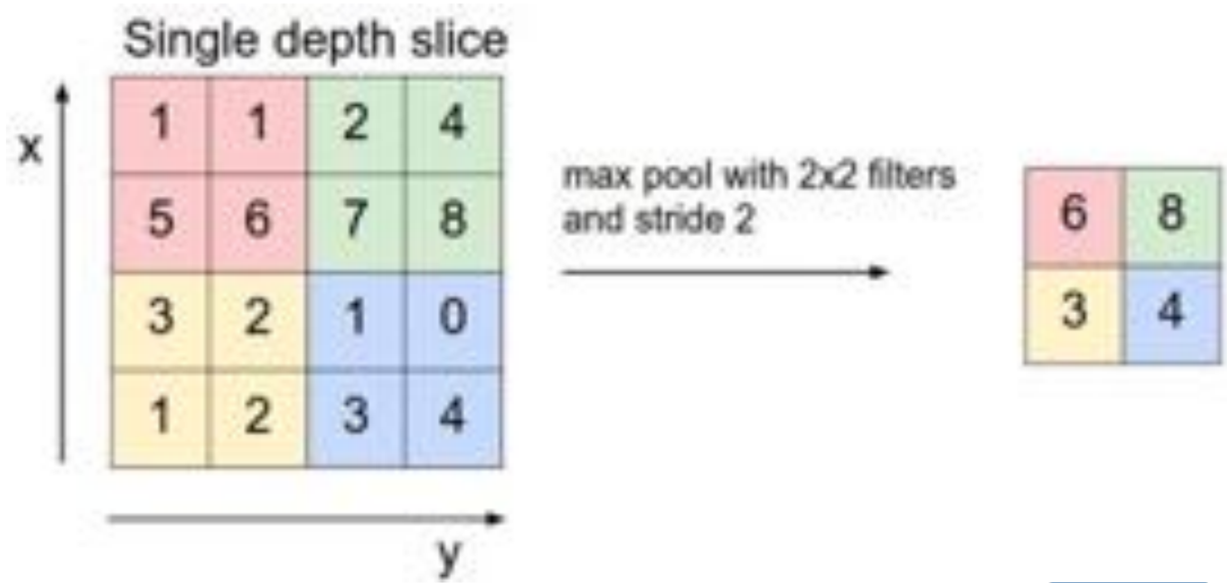
A Full Convolutional Neural Network (LeNet)

Source: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

CNN: Pooling Layer



Pooling layer creates a strategic down-sampling from a convolutional layer, rendering representations of predominant features in lower dimensions

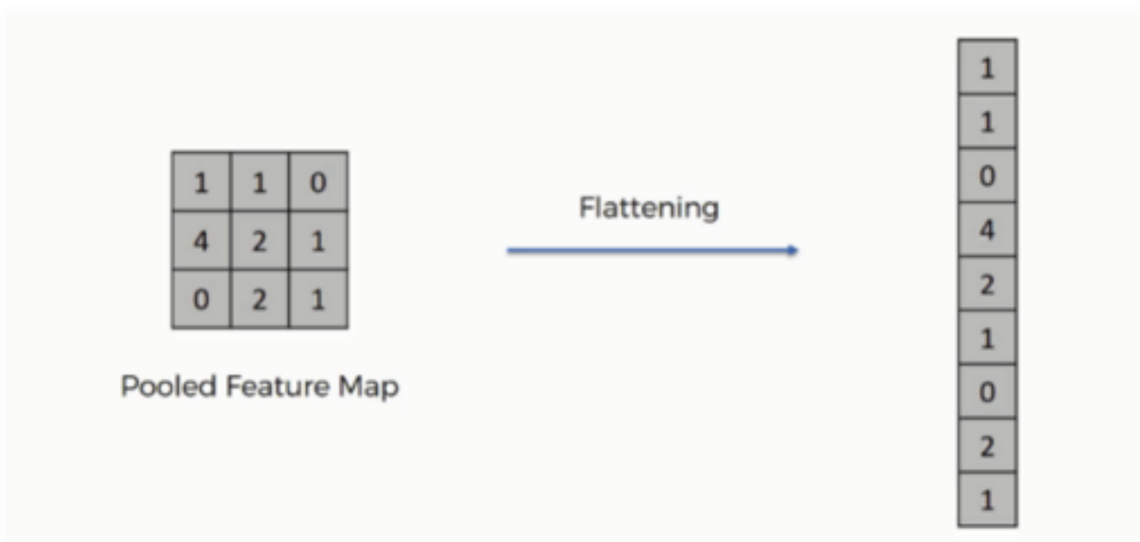


Source: https://cs231n.github.io/convolutional-networks/

CNN: Pooling and Flattening

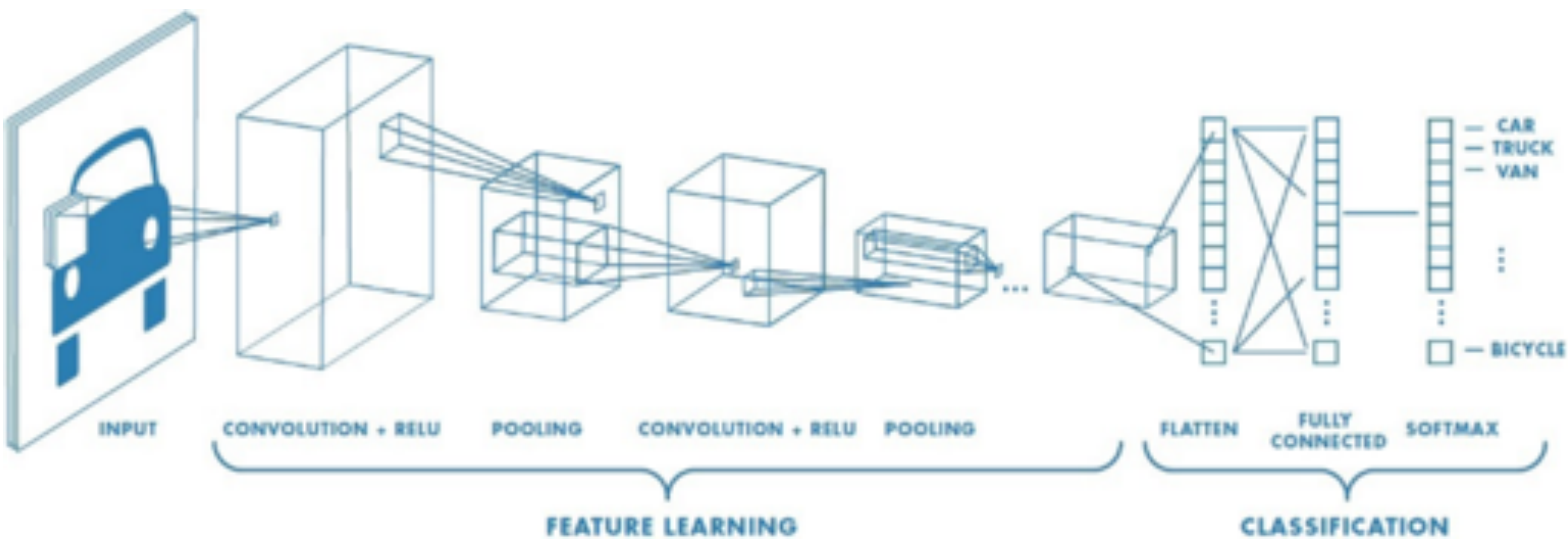
- Advantages of Pooling:
  - The first is that the amount of parameters or weights is reduced by 75%, thus lessening the computation cost
  - Prevents overfitting

- Flattening: Converts the last convolutional layer into a one dimensional neural network layer



CNN: Fully Connected Layer

This layer basically takes an input volume and outputs an N dimensional vector where N is the number of classes that the model has to choose from



If number of classes is four (car, truck, van and bicycle; the final output is a 4-dimensional vector (.55 .10 .30 .05):

A 55% probability that the image is a car

A 10% probability that image is a truck

A 30% probability that image is a van and 5% probability that it is a bicycle

DUKE FUQUA SCHOOL OF BUSINESS

CNN Challenges and Opportunities

- Data, data, data (missing data; open source data)

The more training data that you can give to a network, the more training iterations you can make, the more weight updates you can make, and the better tuned to the network is when it goes to production

- Transfer Learning
- RNN: Sequential Data (e.g., time series, audio, video)
- Assist AI in HealthCare : Taking over standard interactions to relieve burden on healthcare (memorize and track flow charts)

Thanks
sd345@duke.edu