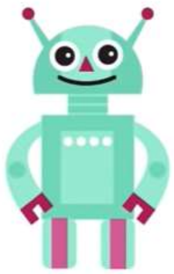
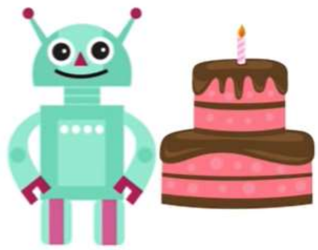
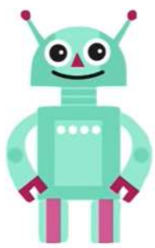
<u>INTRODUCTION TO DEEP LEARNING:</u>

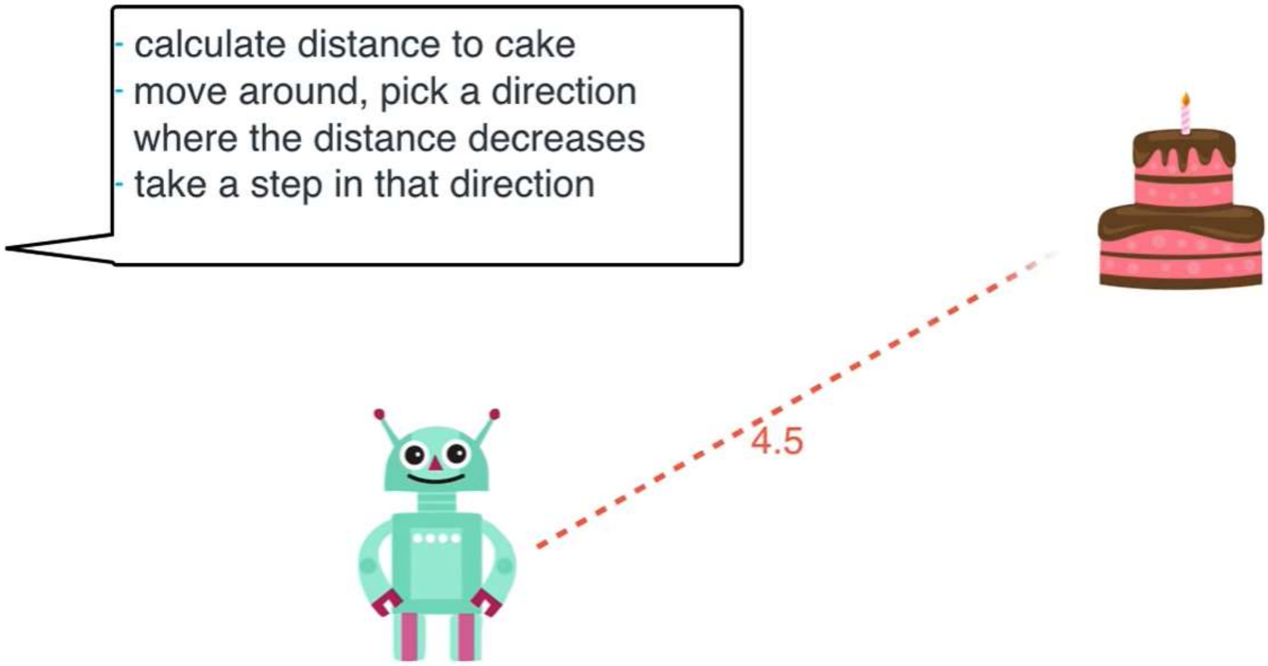


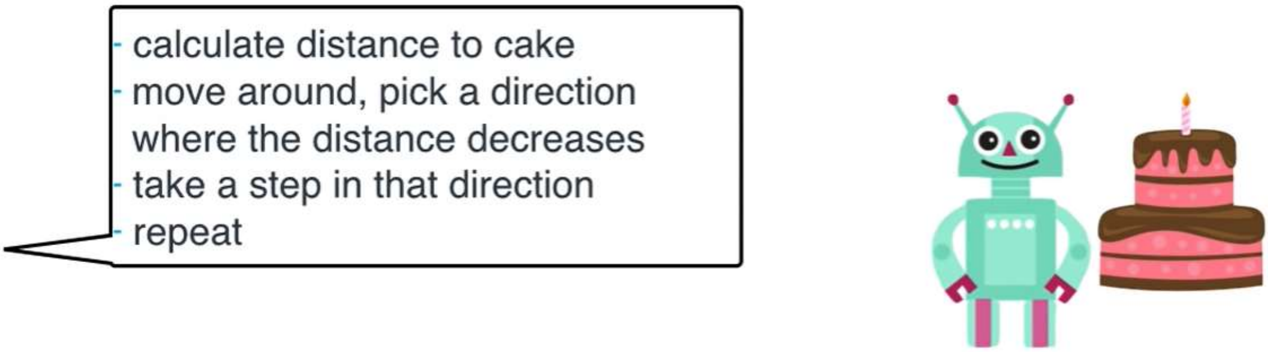
- turn right
- go 10 steps
- turn left
- go 4 steps
- grab cake

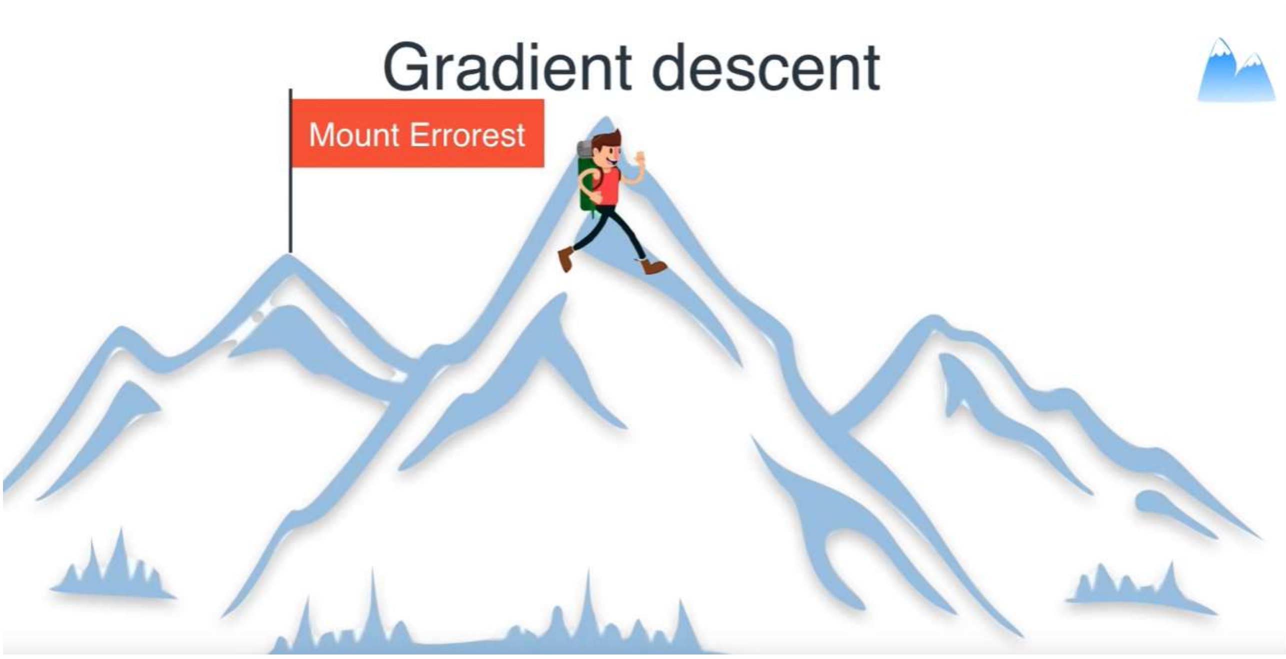


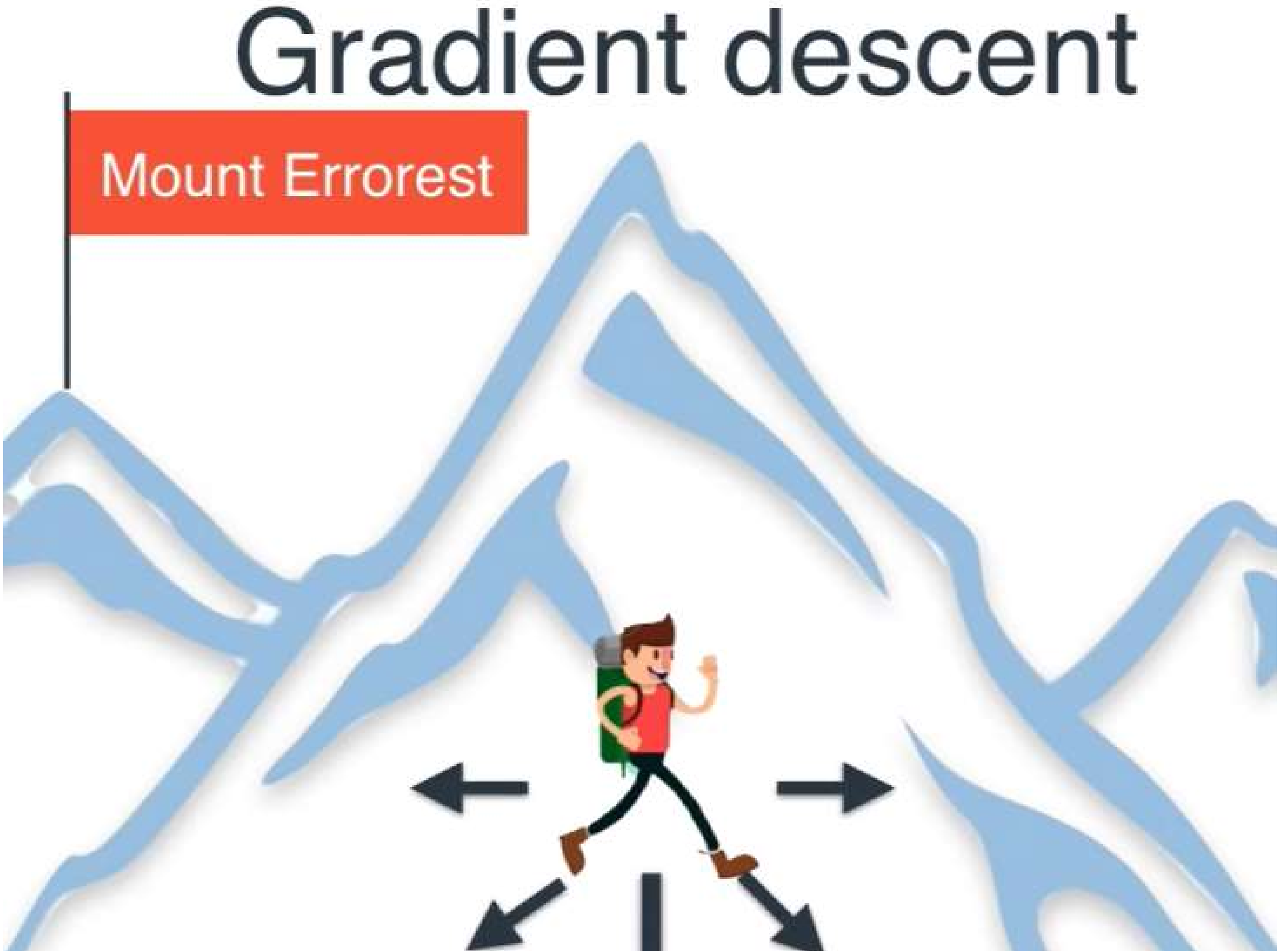
- calculate distance to cake
- move around, pick a direction where the distance decreases




- calculate distance to cake
- move around, pick a direction where the distance decreases
- take a step in that direction
4.5


- calculate distance to cake
- move around, pick a direction where the distance decreases
- take a step in that direction
- repeat


Gradient descent
Mount Errorest



WE TOOK THE GRADIENT – THE DERIVATIVE

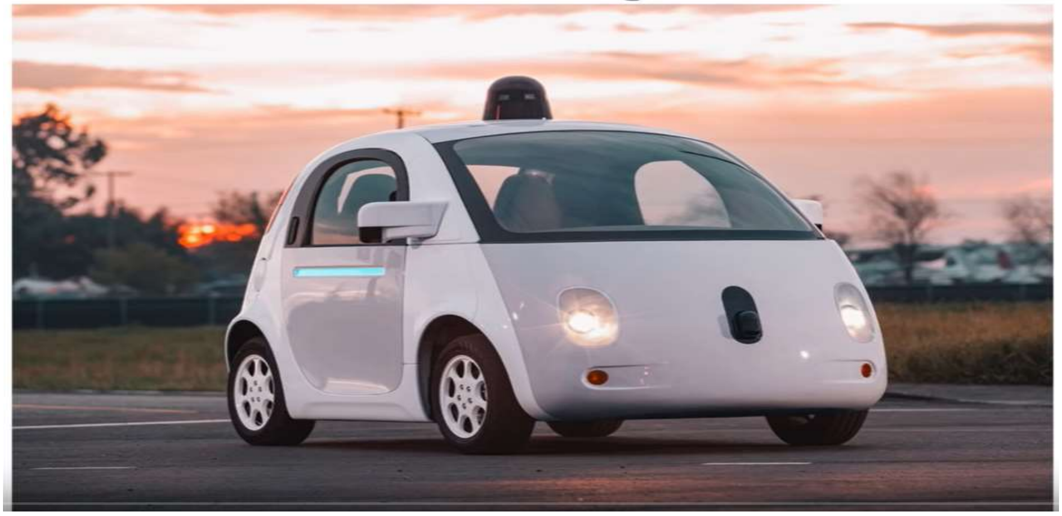
Gradient descent

Get cake
Minimize distance to cake

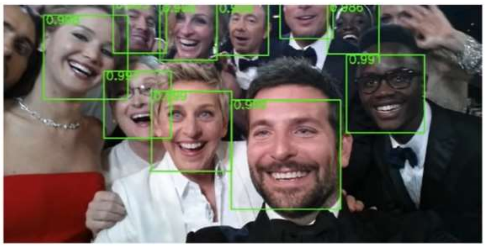
Descend from mountain
Minimize height

Solve any problem
Minimize error

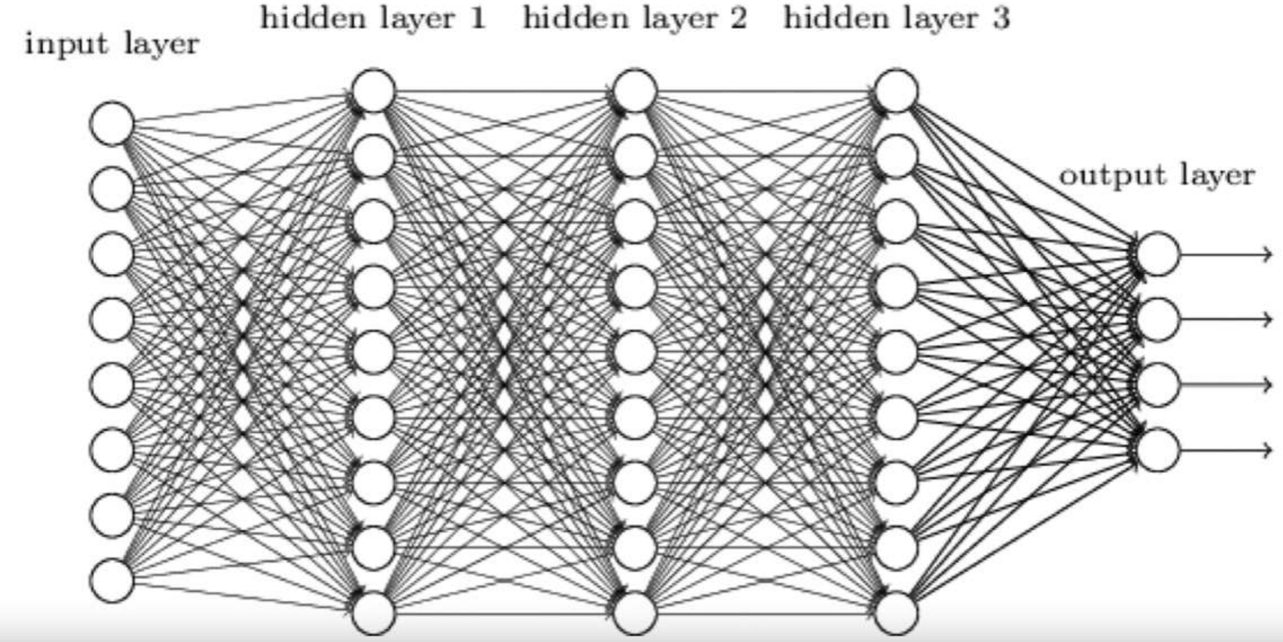
Self Driving Car



Many more things

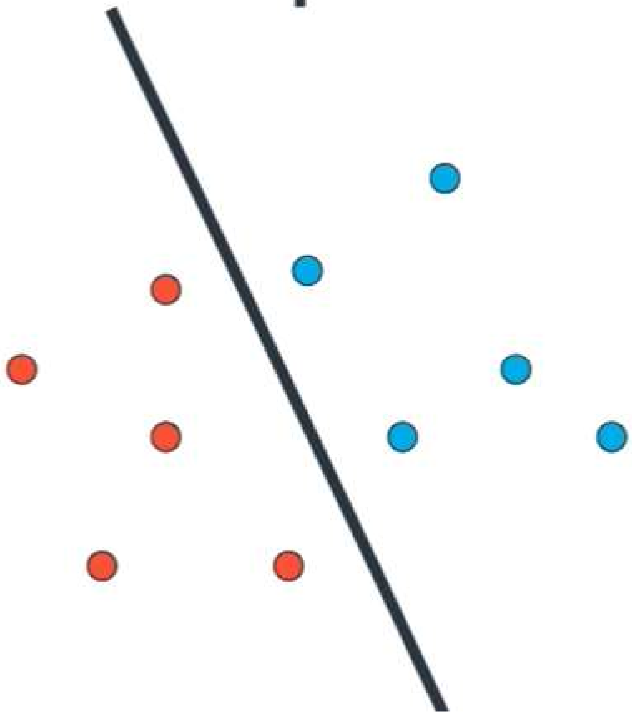


Neural Networks

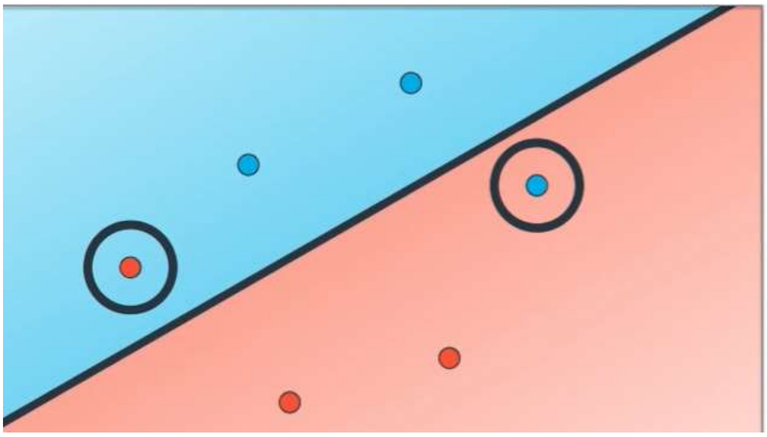


Neural Networks

Goal: Split Data

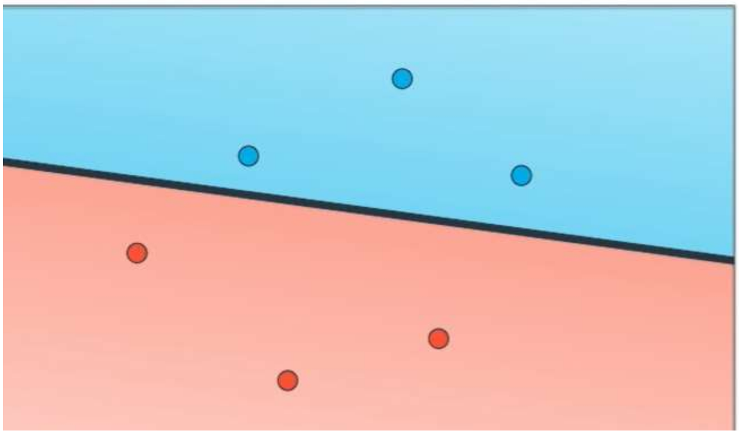


Goal: Split Data



2 errors

Goal: Split Data



0 errors

Hot!

Gradient descent

Get cake

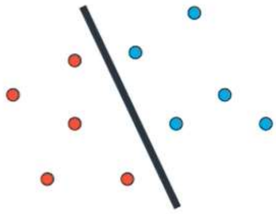
Distance to cake

continuous function

Descend from mountain

Height

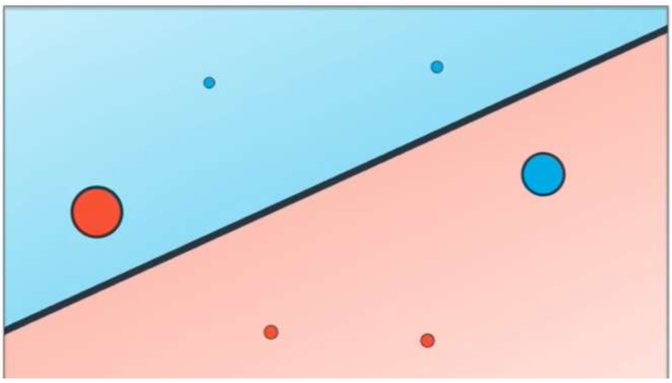
continuous function

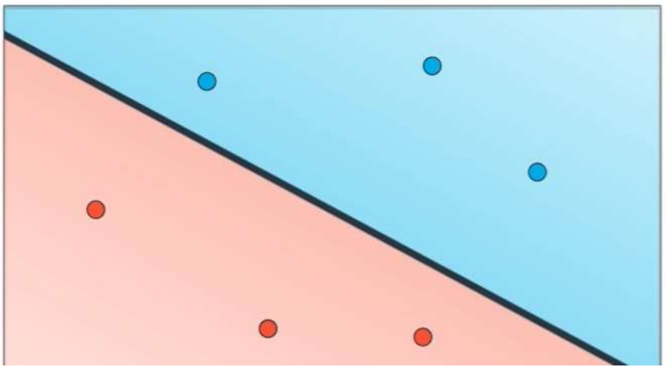


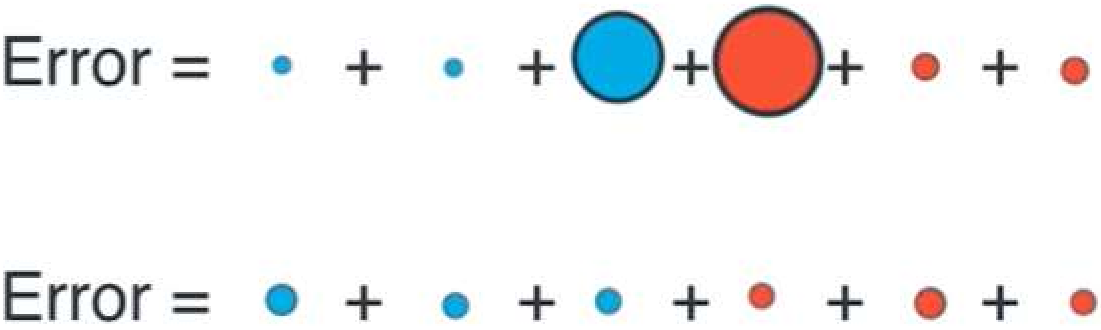
Split data

Number of errors

discrete function



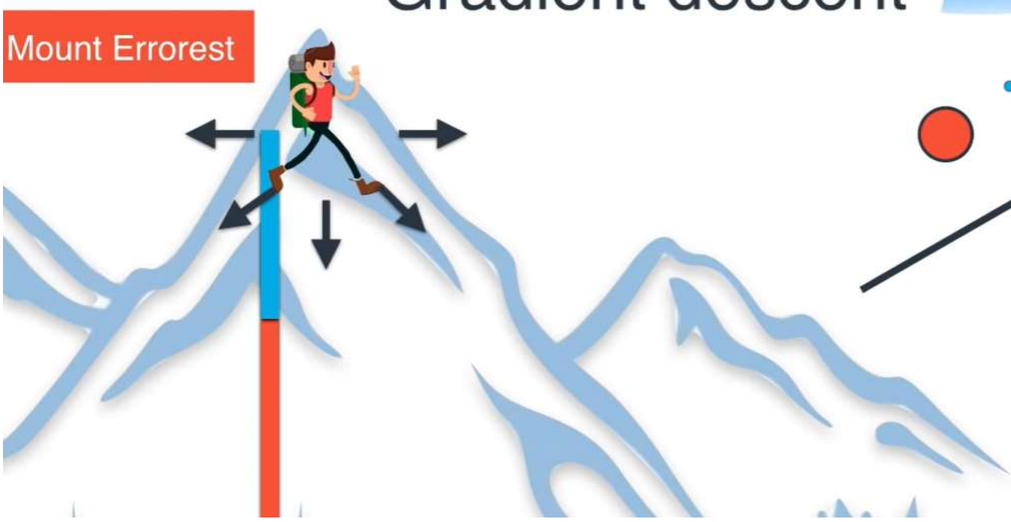


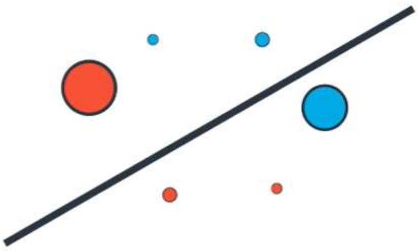


Minimize error

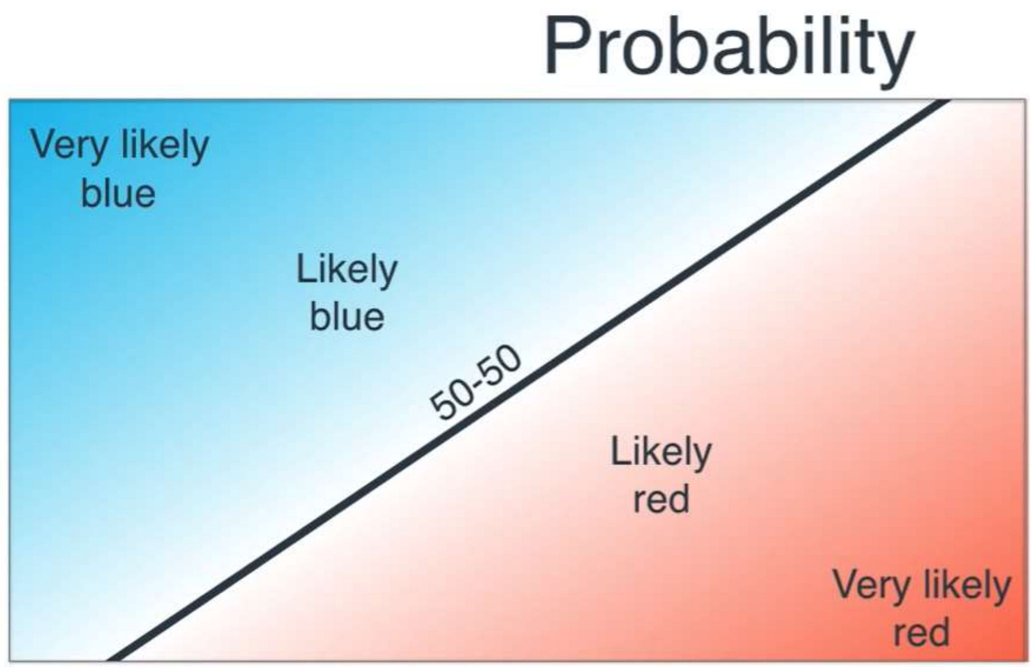
Gradient descent

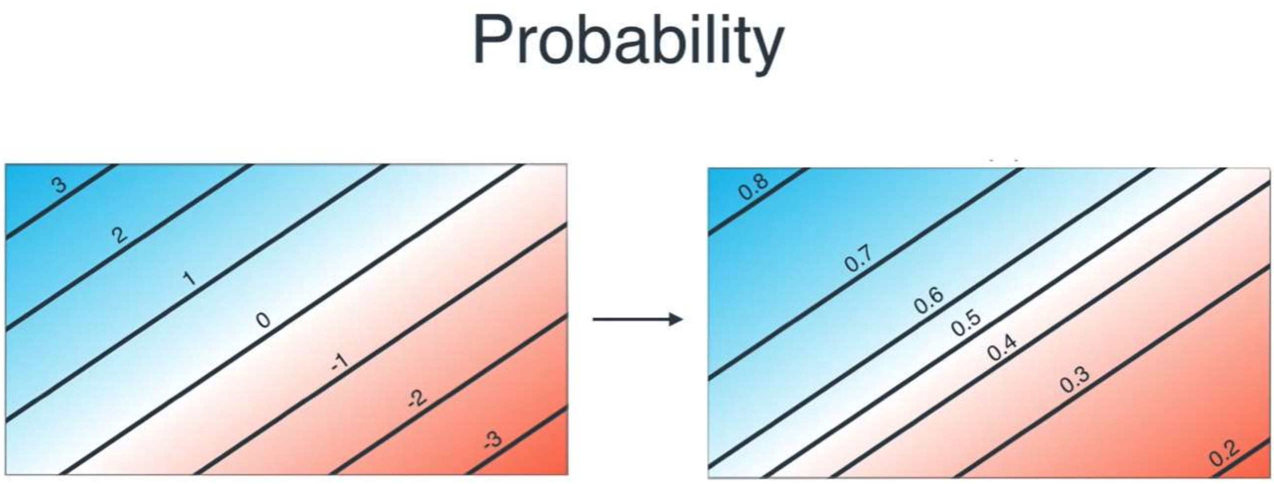
Gradient descent 

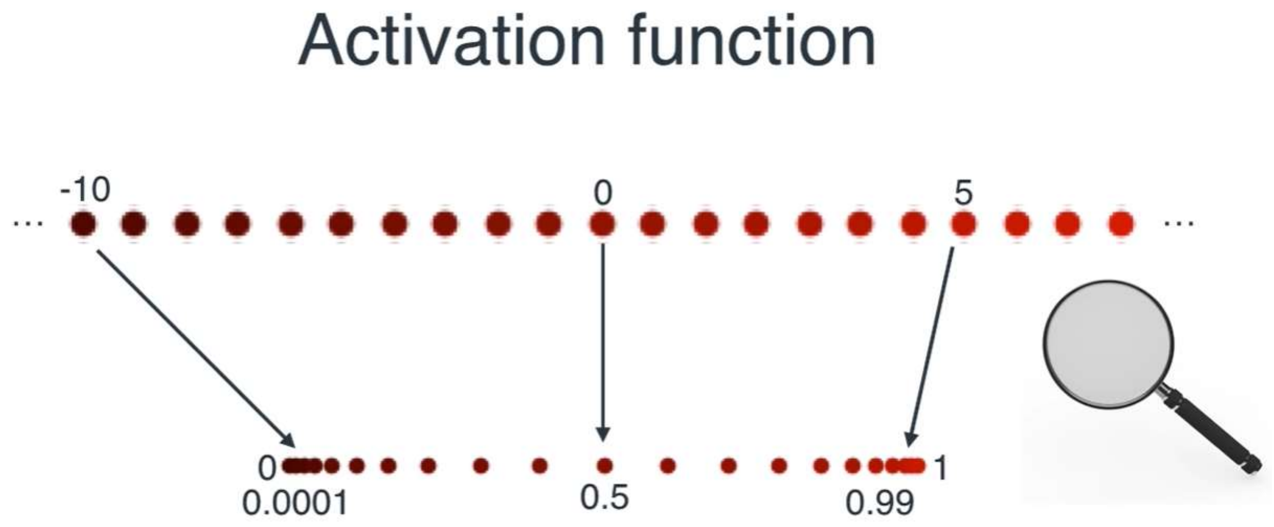




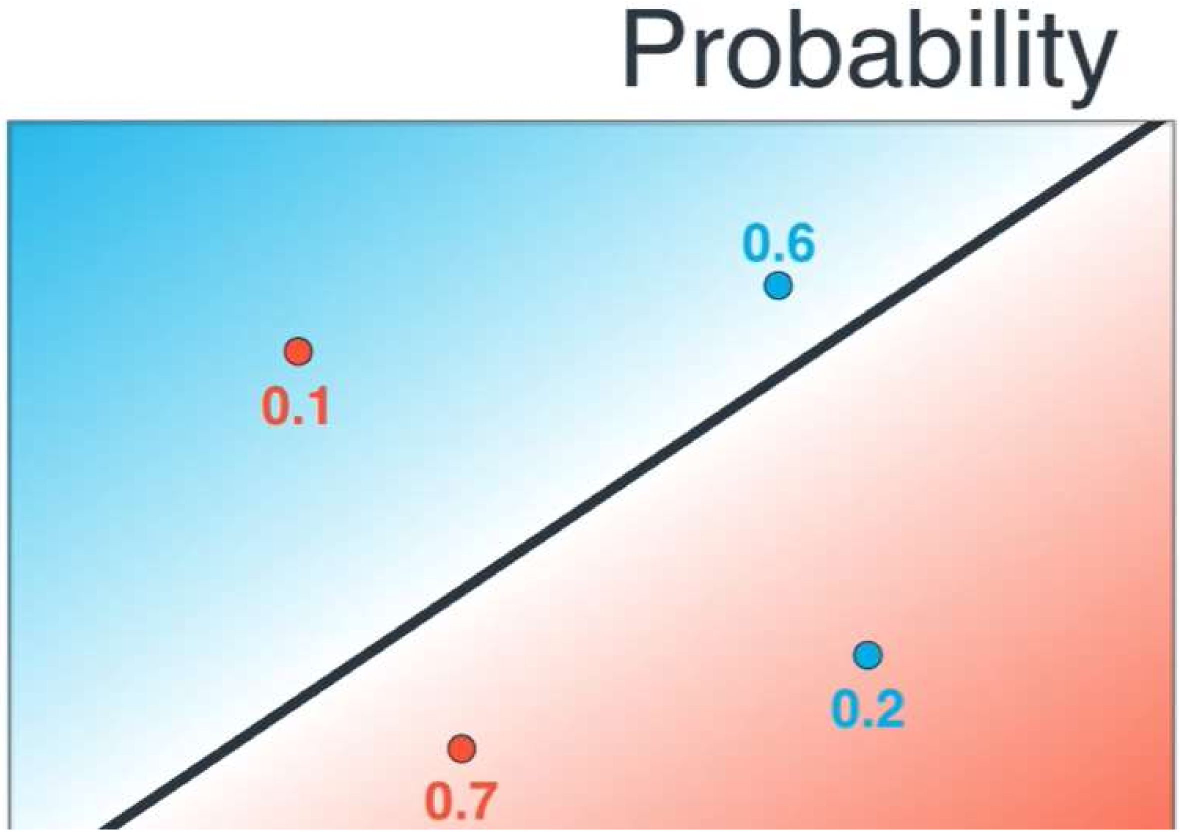

Gradient descent
Mount Errorest


Probability
Very likely
blue
Likely
blue
50-50
Likely
red
Very likely
red

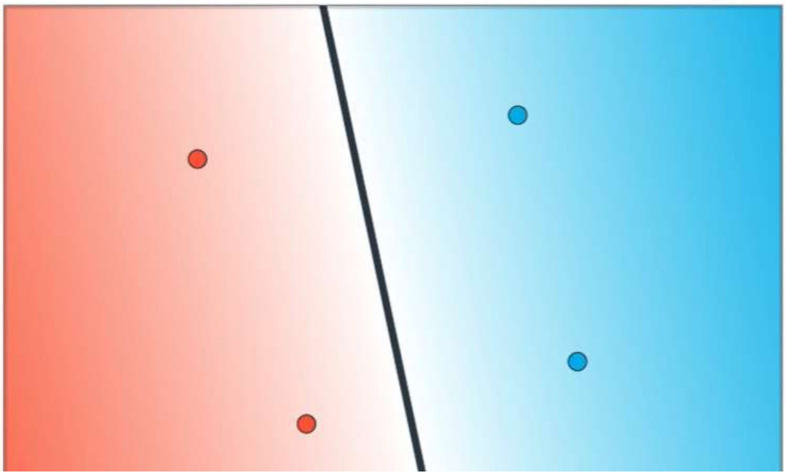

Probability
3
2
1
0
-1
-2
-3
0.8
0.7
0.6
0.5
0.4
0.3
0.2



IF WE THINK THESE ARE INDIPENDENT EVENTS, THE PROBABILITY ALL FOUR HAPPENING, IS THE PRODUCT



Probability

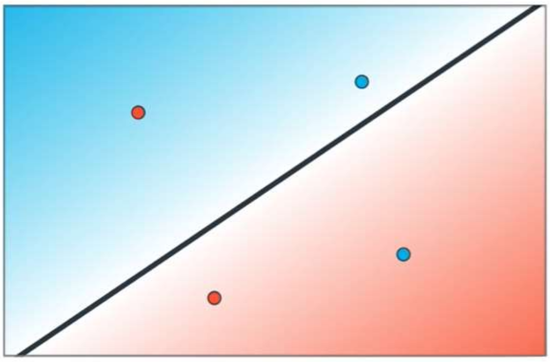


$0.6*0.2*0.1*0.7 = 0.0084$

$0.7*0.9*0.8*0.6 = 0.3024$

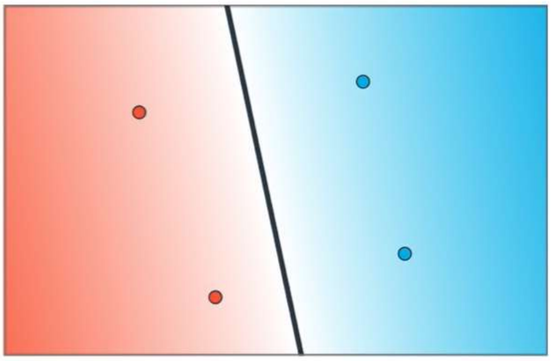
Maximum Likelihood

Error function



$0.6*0.2*0.1*0.7 = 0.0084$

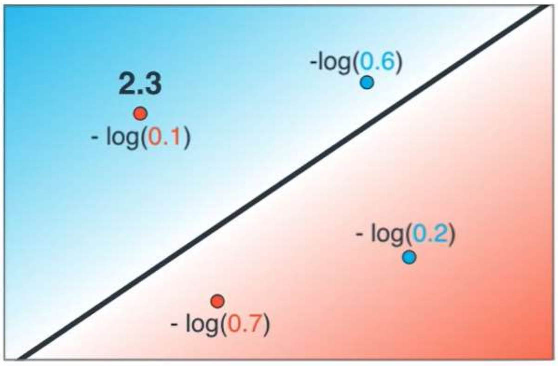
$-\log(0.6) - \log(0.2) - \log(0.1) - \log(0.7) = 4.8$



$0.7*0.9*0.8*0.6 = 0.3024$

$-\log(0.7) - \log(0.9) - \log(0.8) - \log(0.6) = 1.2$

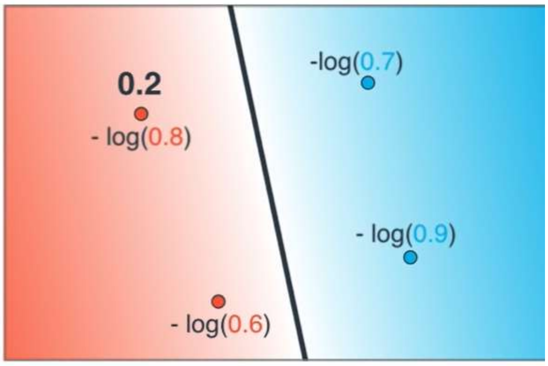
Error function



$0.6*0.2*0.1*0.7 = 0.0084$

$-\log(0.6) - \log(0.2) - \log(0.1) - \log(0.7) = 4.8$

2.3



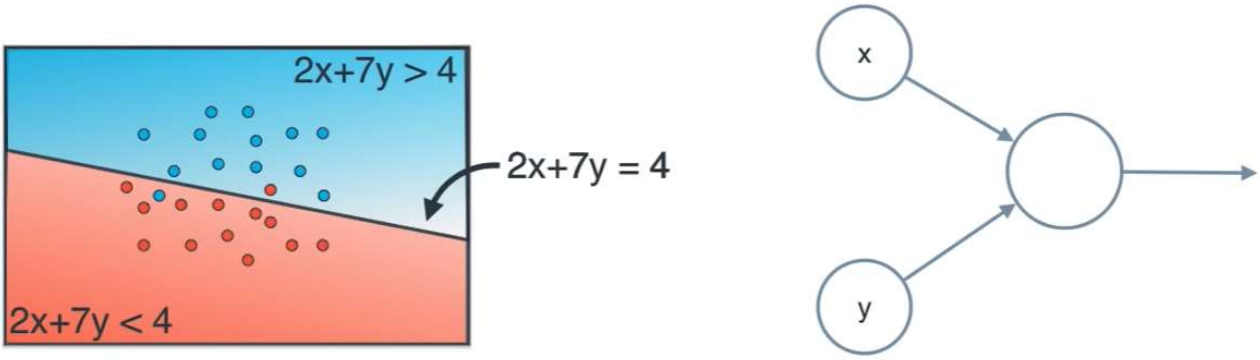
$0.7*0.9*0.8*0.6 = 0.3024$

$-\log(0.7) - \log(0.9) - \log(0.8) - \log(0.6) = 1.2$

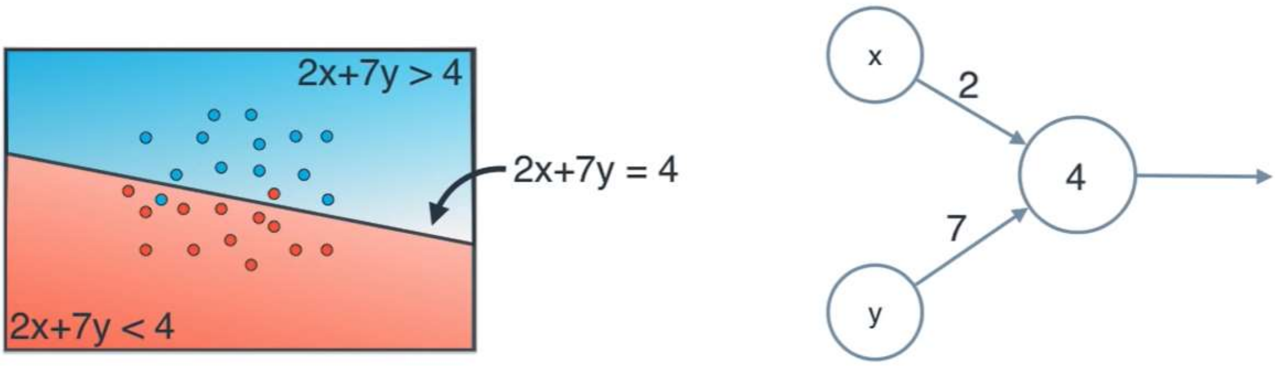
0.2

ERROR FUNCTION AS A PENALTY FOR EVERY POINT

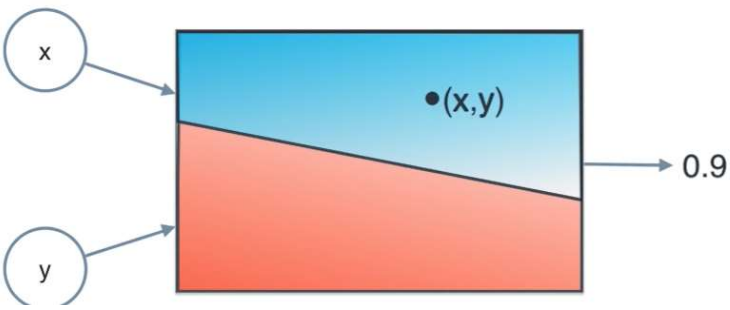
Neuron

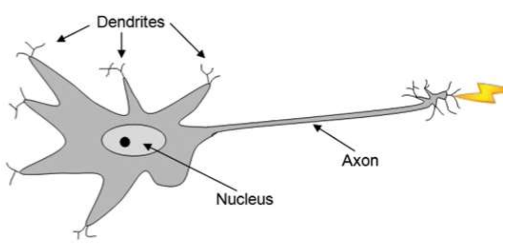


Neuron

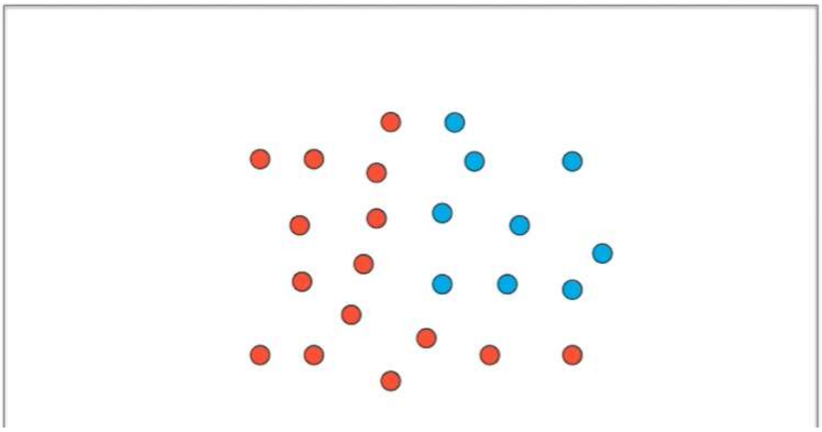


Neuron

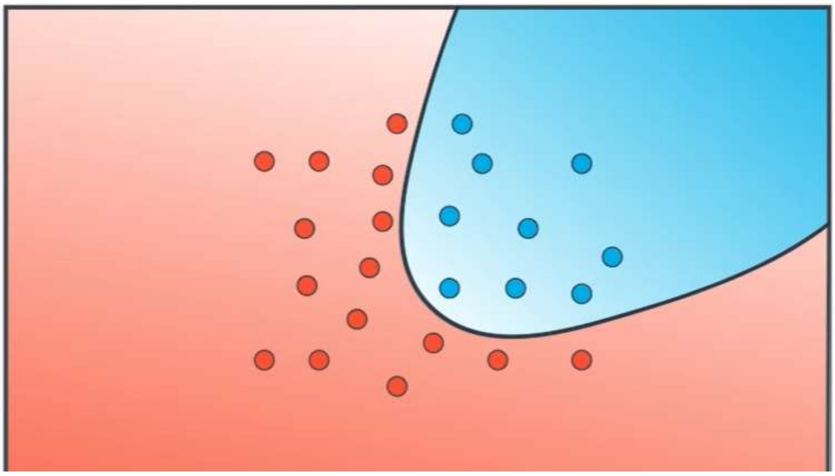




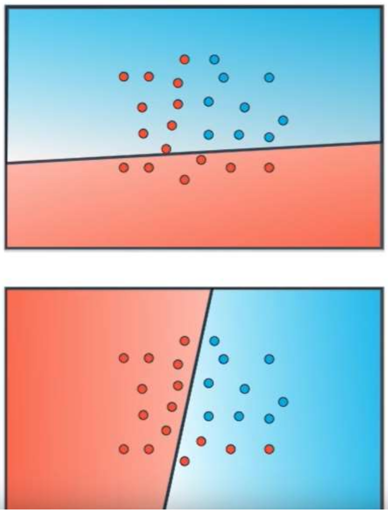
Non-linear regions

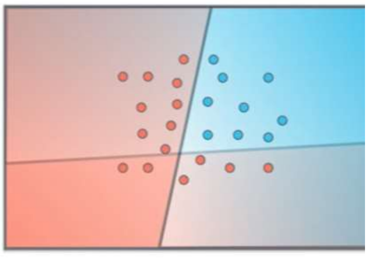


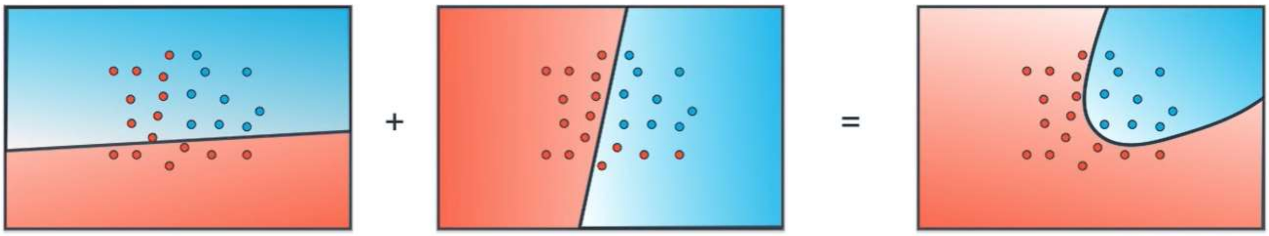
Non-linear regions



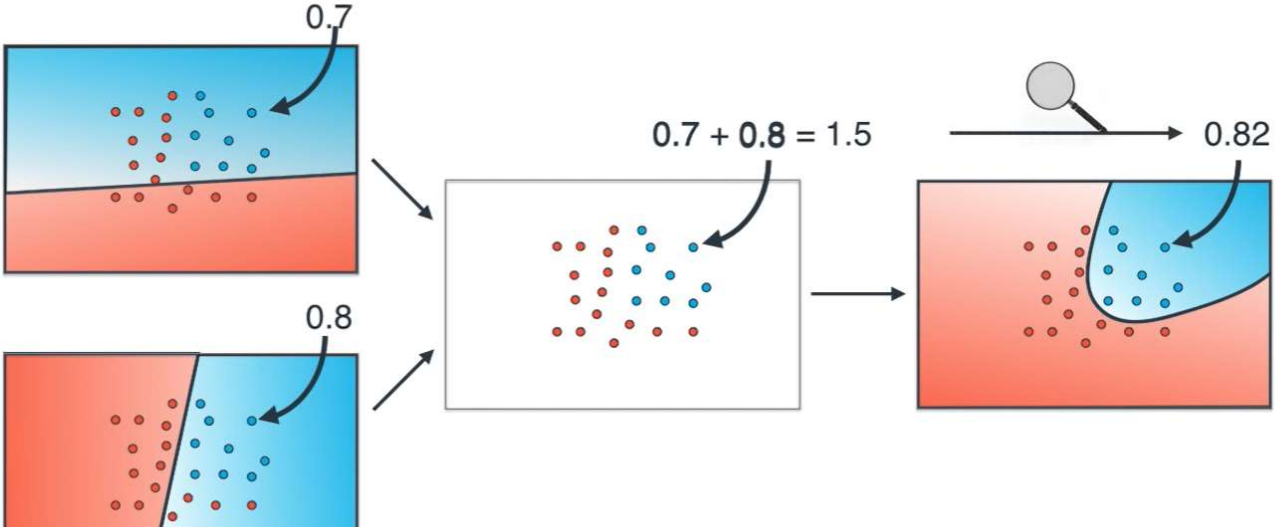
Combining Regions



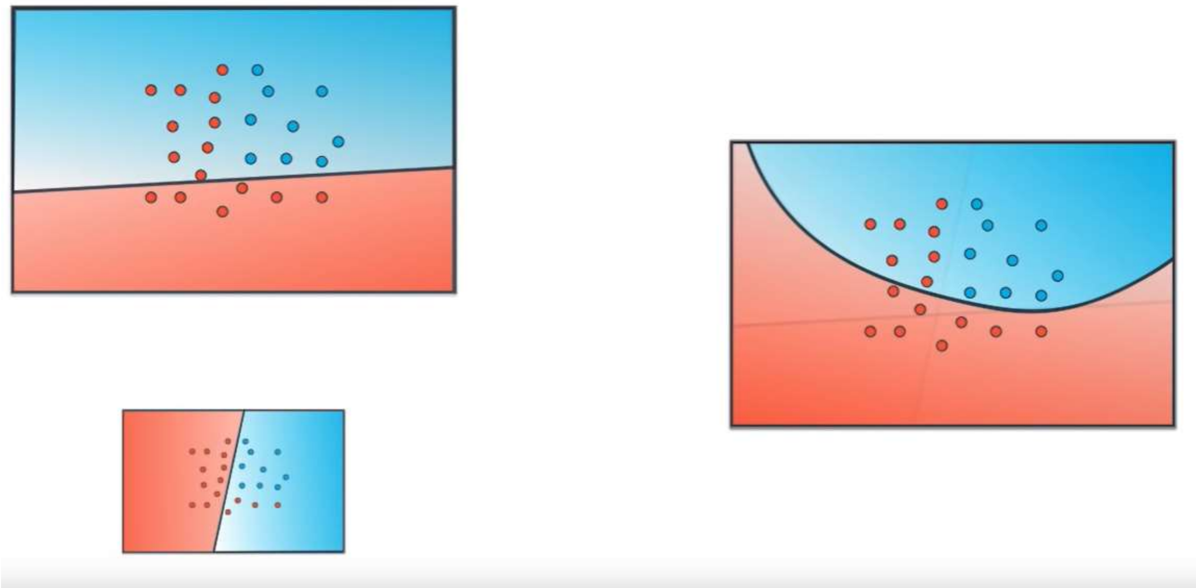


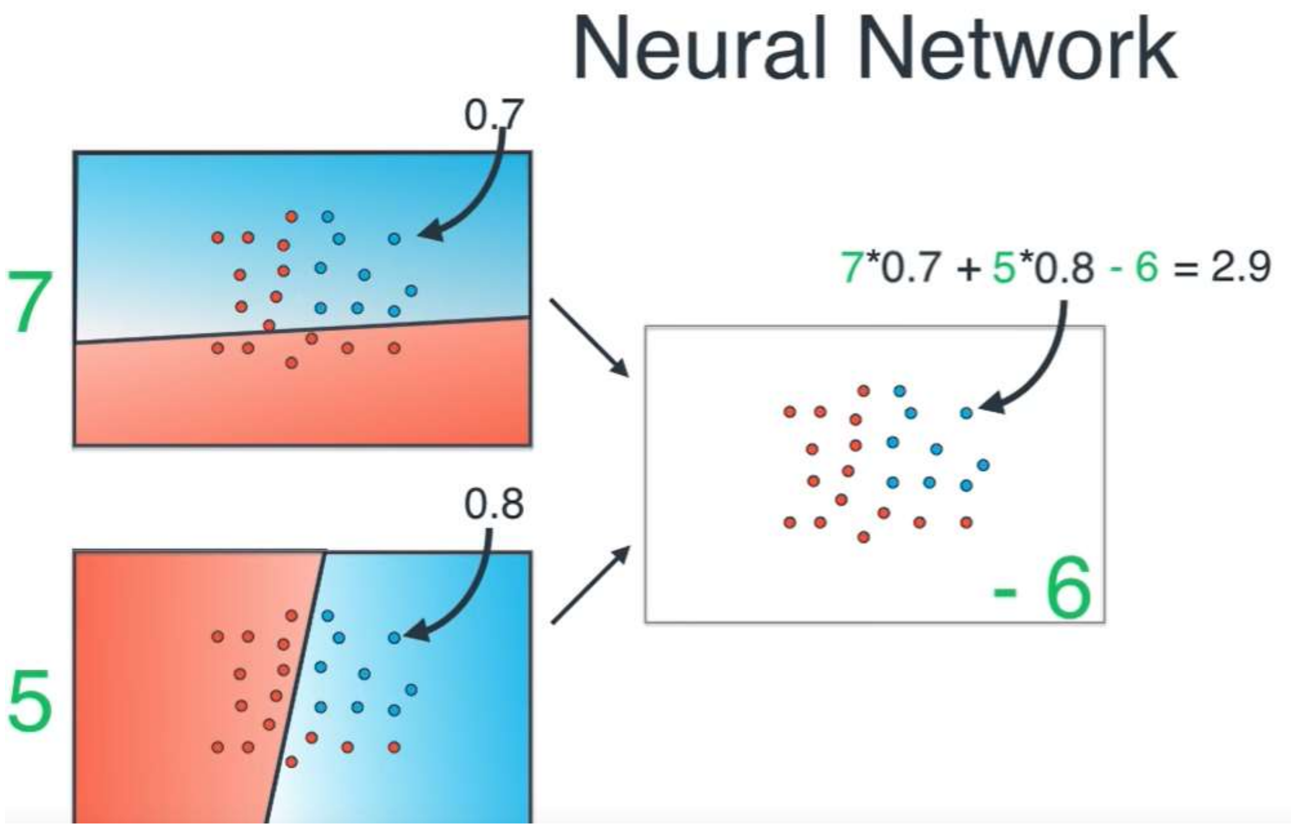


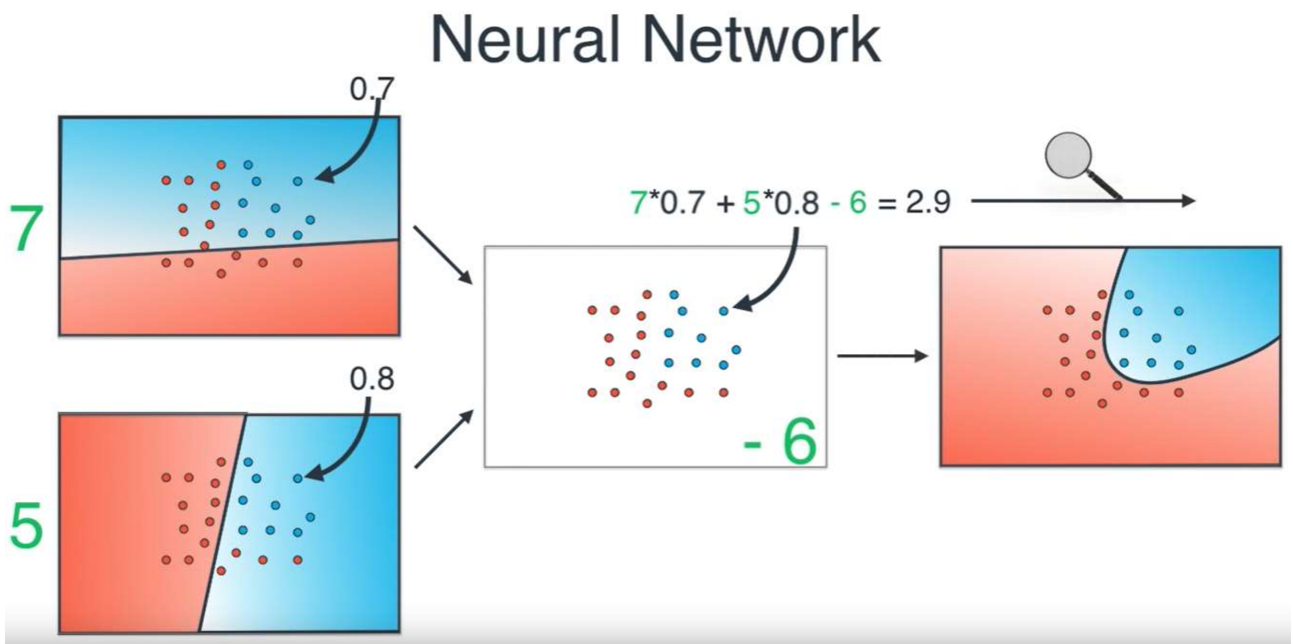
Neural Network

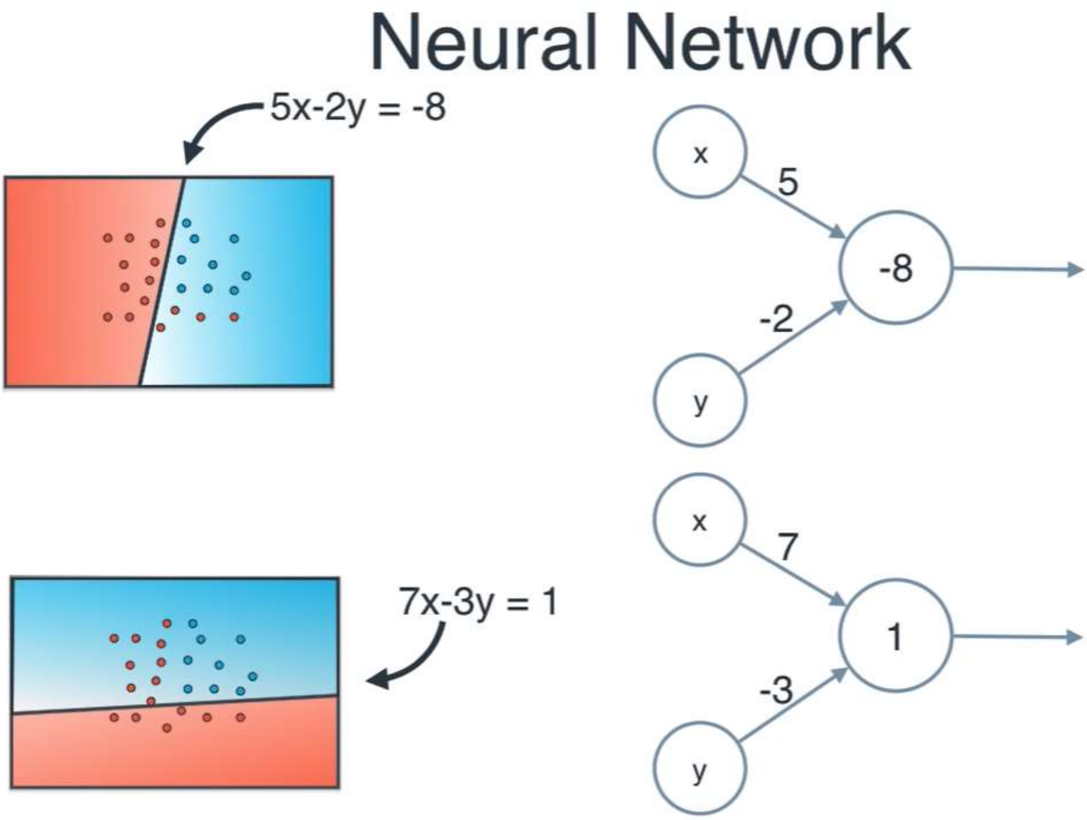


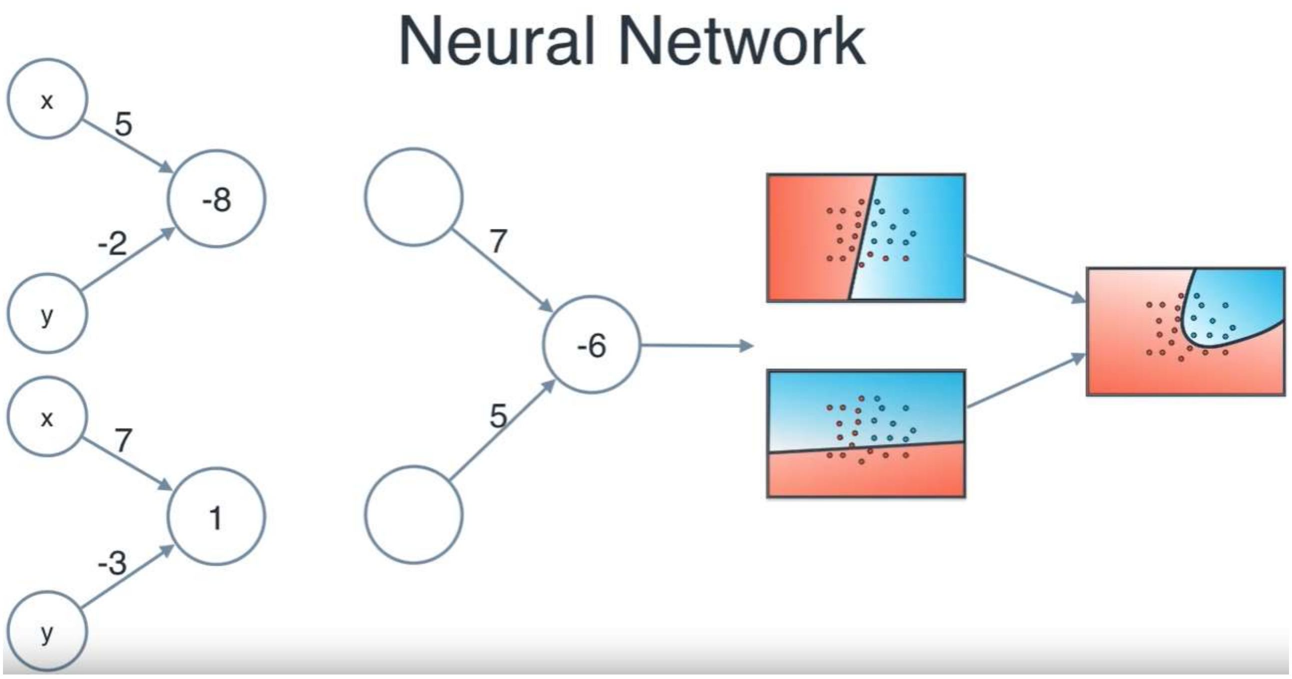
Combining Regions

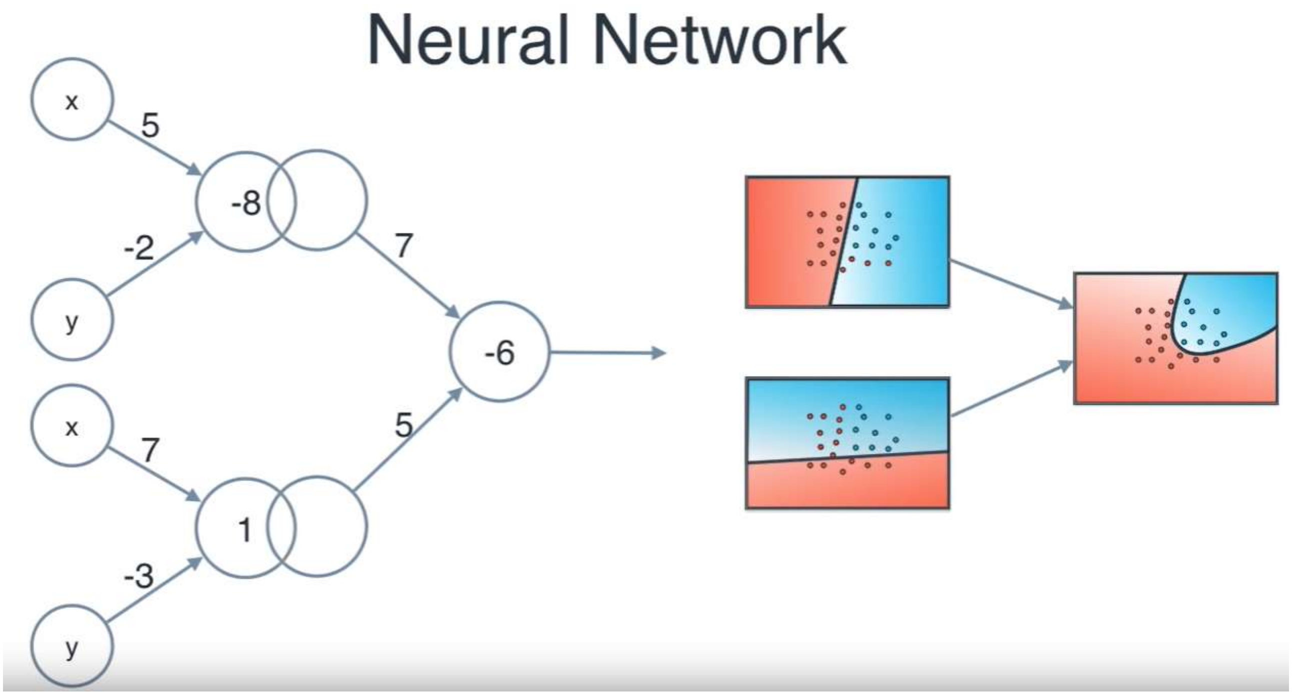


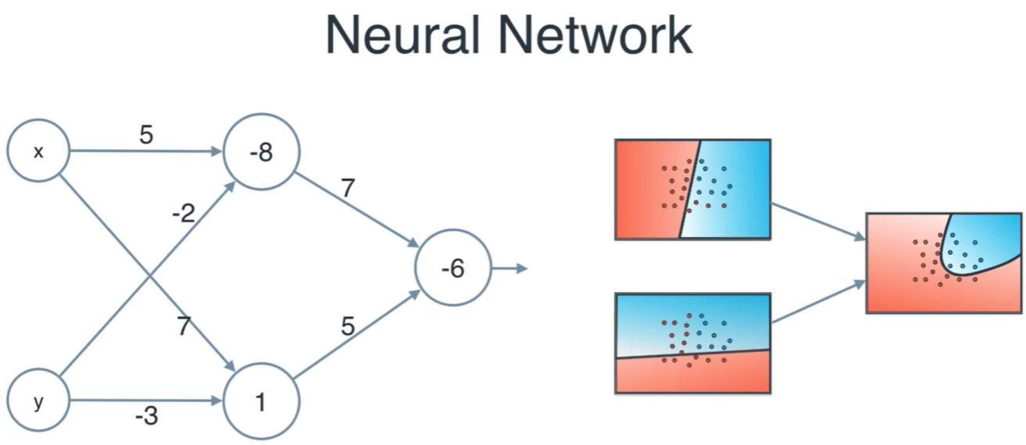

Neural Network
0.7
7
0.8
5
7*0.7 + 5*0.8 - 6 = 2.9
- 6

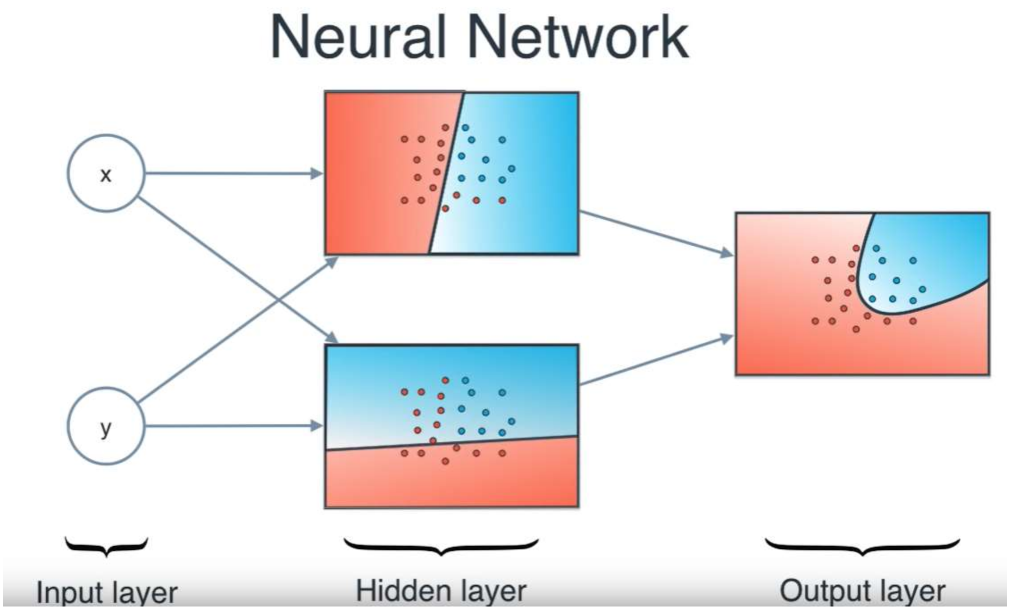

Neural Network
0.7
7
0.8
5
7*0.7 + 5*0.8 - 6 = 2.9
- 6

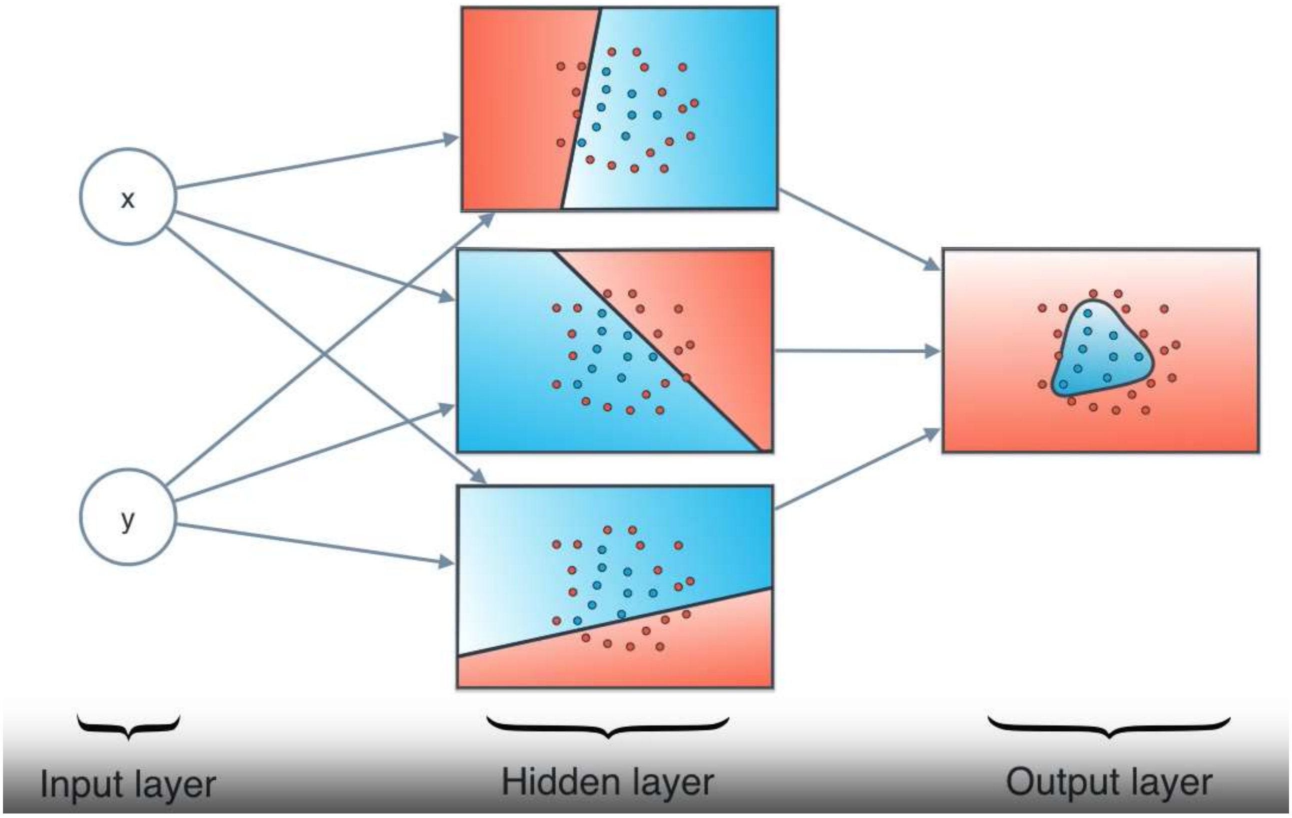

Neural Network
5x-2y = -8
x
5
-8
-2
y
7x-3y = 1
x
7
1
-3
y

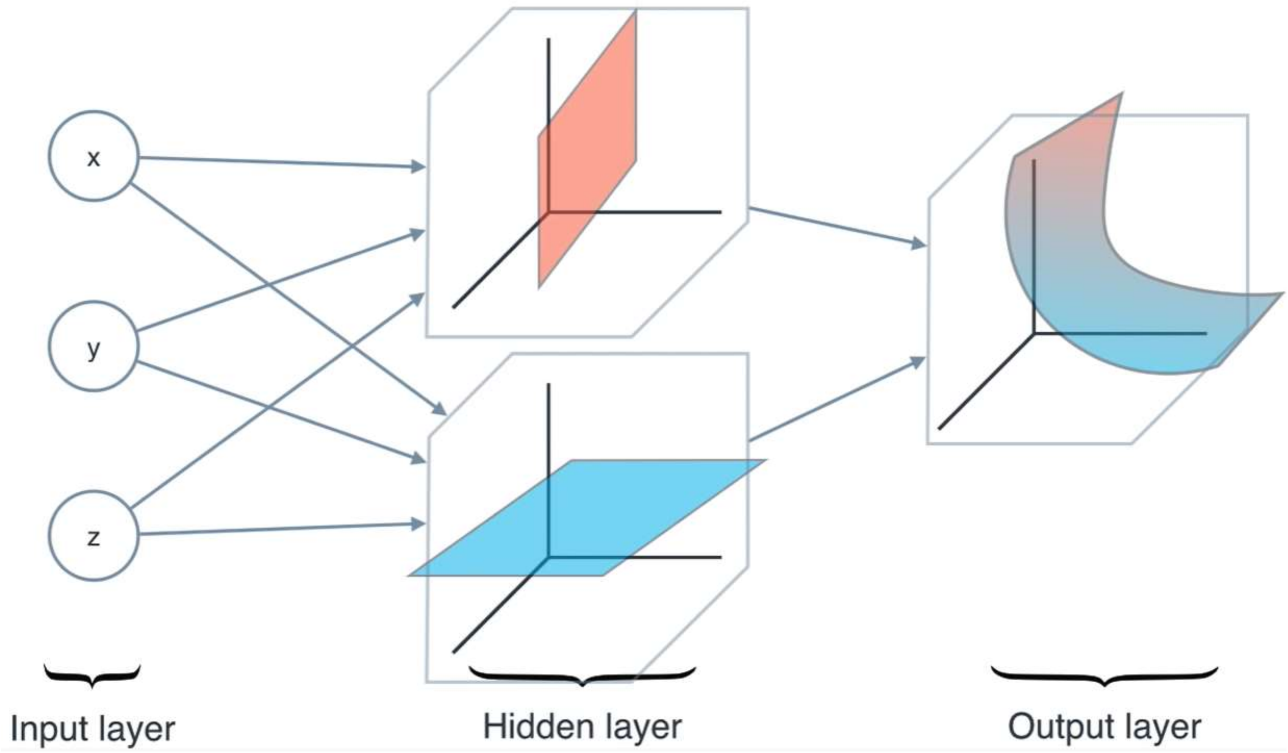

Neural Network
x
5
-8
-2
y
x
7
1
-3
y
7
-6
5


Neural Network
x
5
-8
-2
y
7
-6
x
7
5
1
-3
y

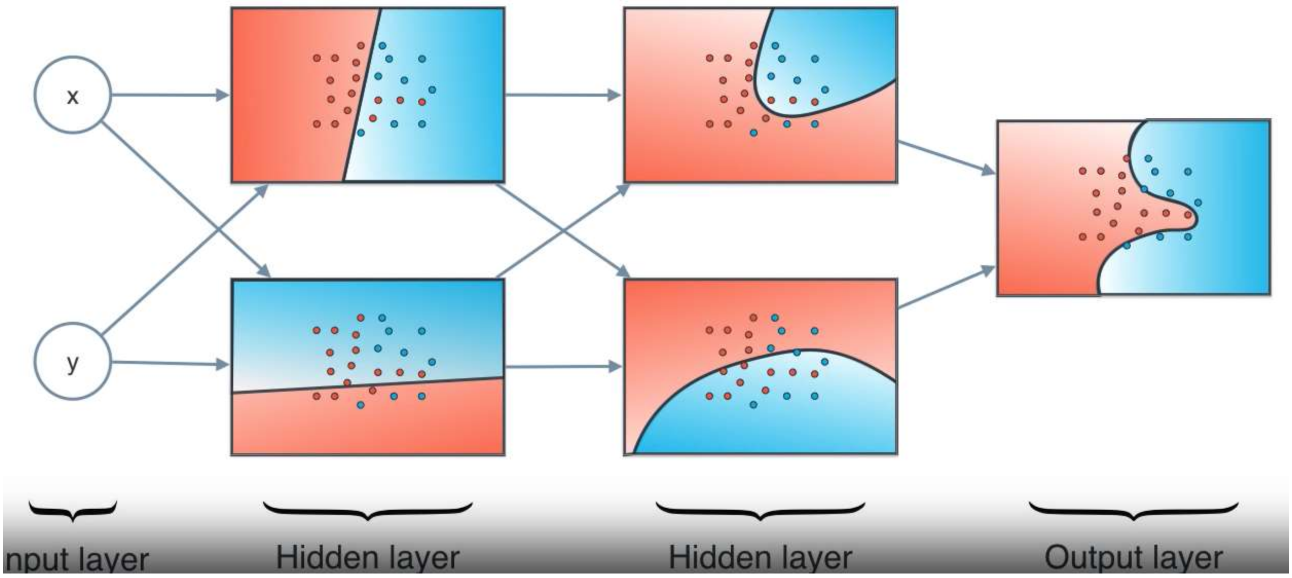

Neural Network
x
5
-8
-2
7
-6
7
5
y
-3
1


Neural Network
x
y
Input layer
Hidden layer
Output layer

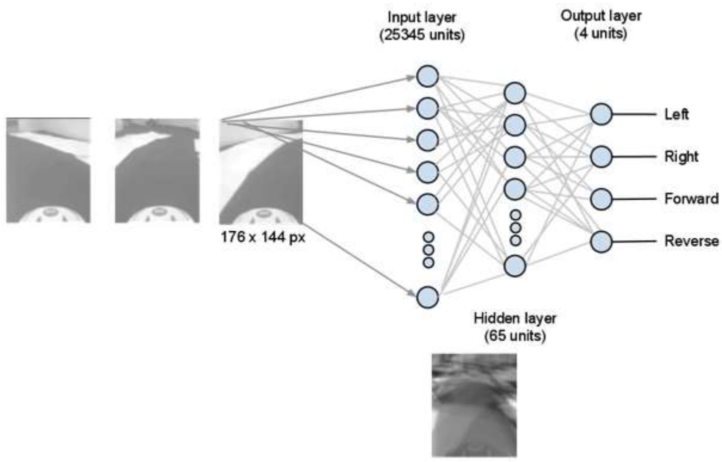

x
y
Input layer
Hidden layer
Output layer


x
y
z
Input layer
Hidden layer
Output layer

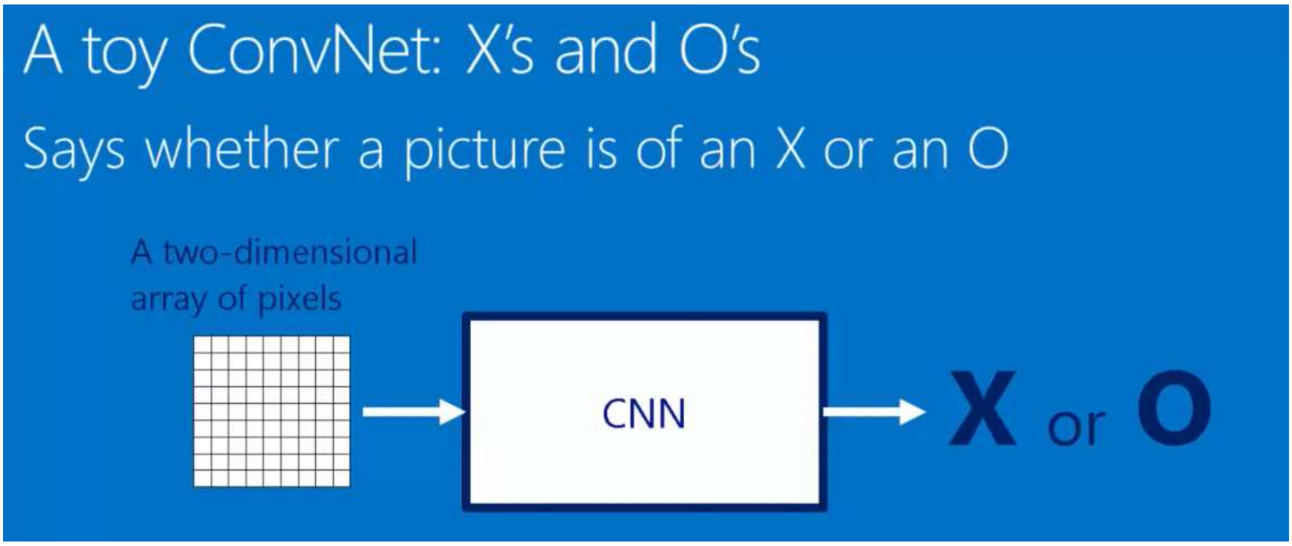
Deep Neural Network

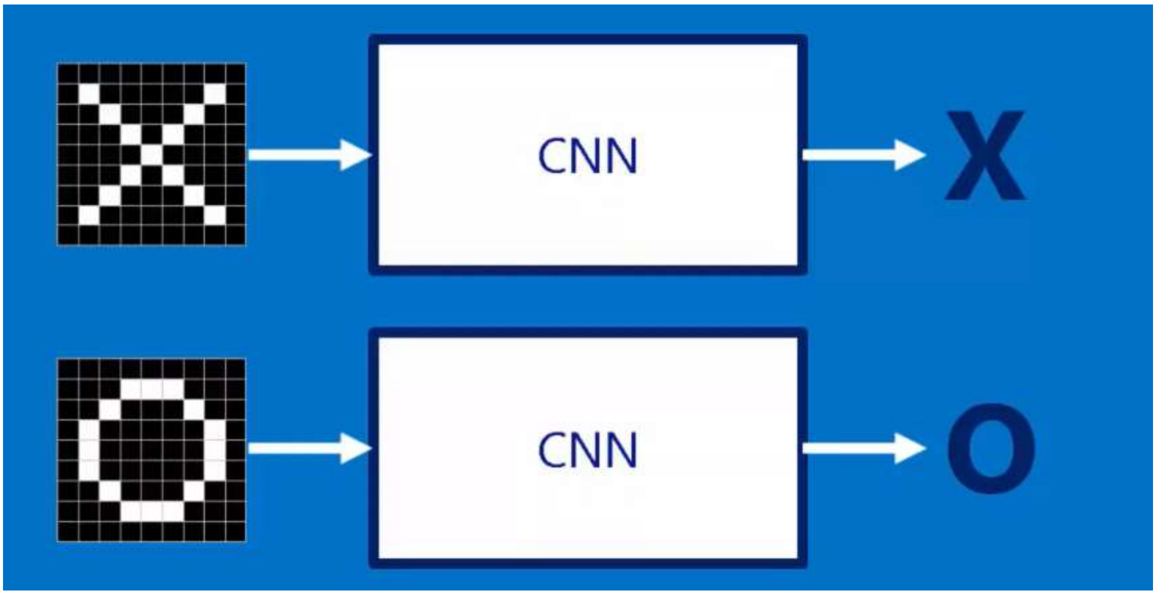


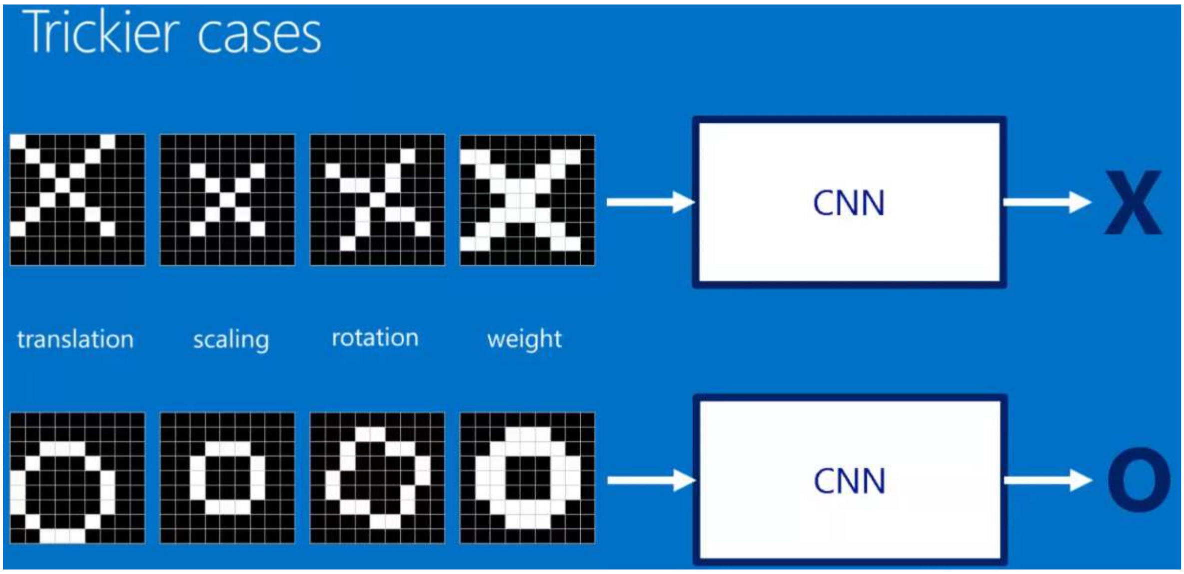
Self Driving Car

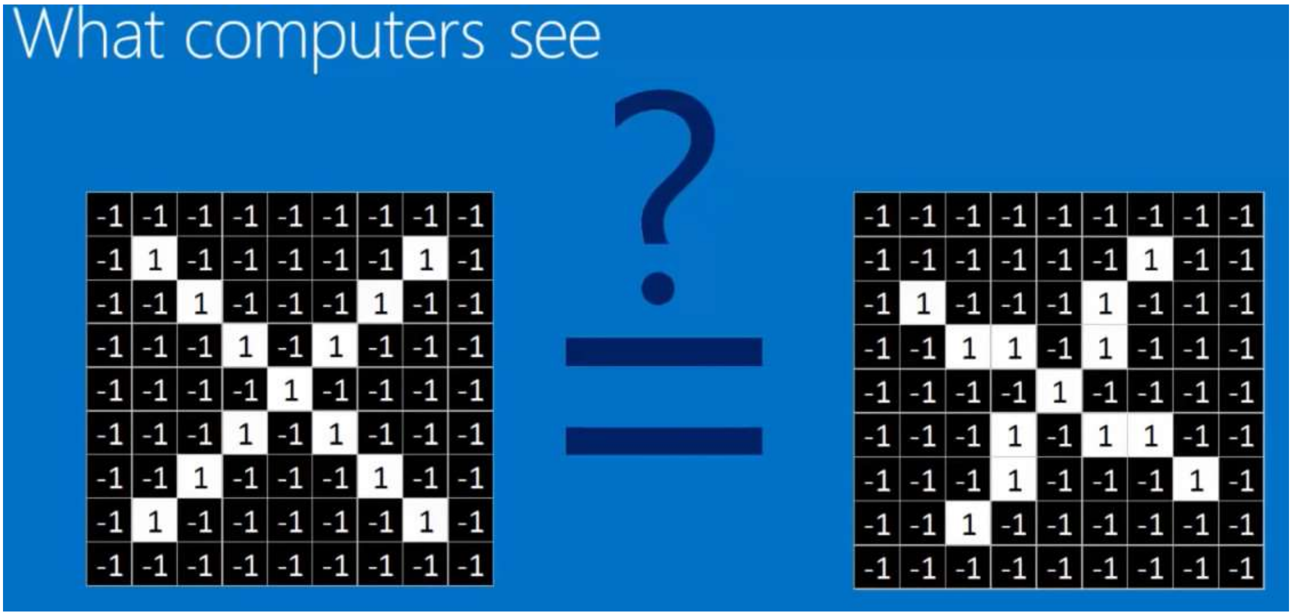


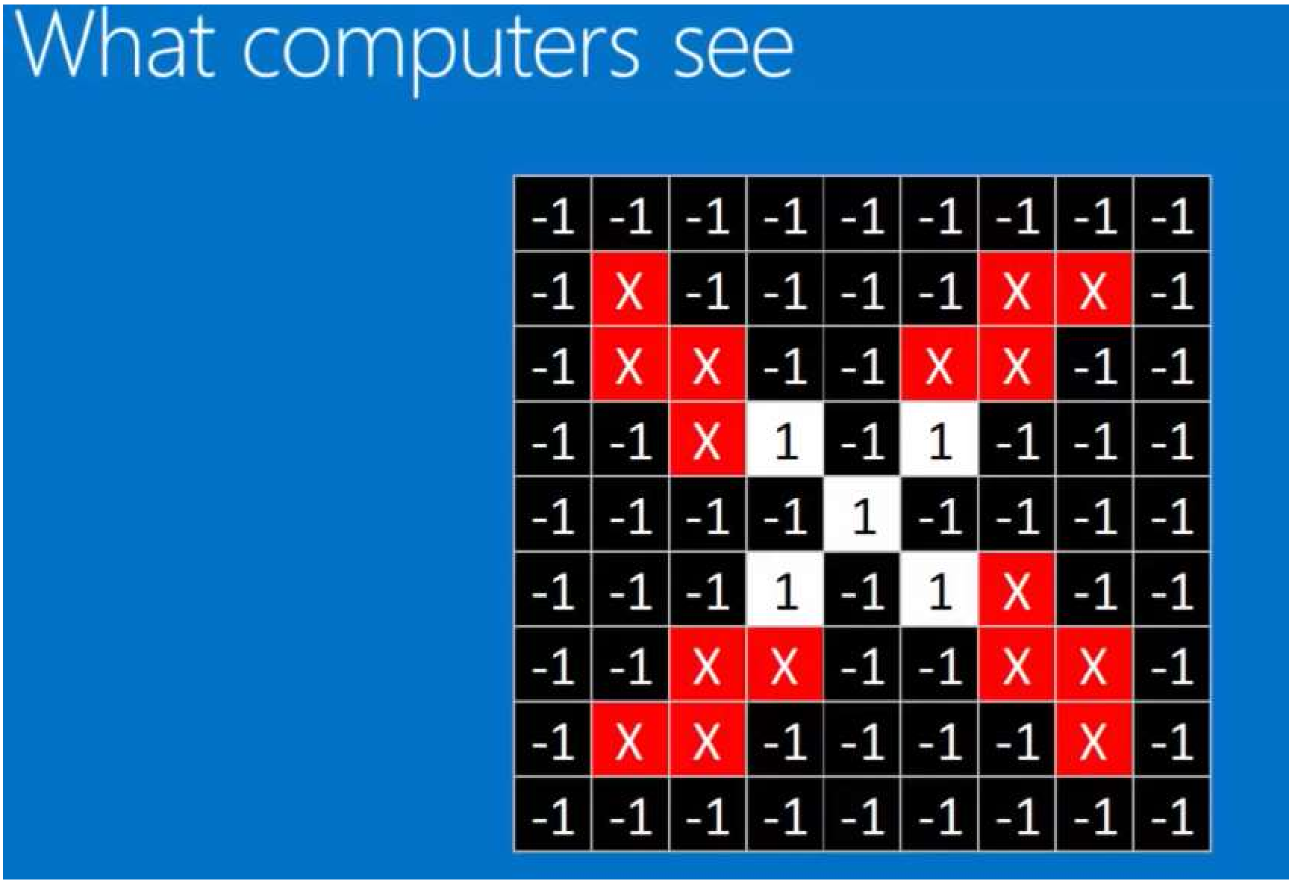
CONVNET:


A toy ConvNet: X's and O's
Says whether a picture is of an X or an O
A two-dimensional array of pixels
CNN
X or O


CNN
X
CNN
O


Trickier cases
translation
scaling
rotation
weight
CNN
X
CNN
O


What computers see
?
=
-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 -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


What computers see
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 X -1 -1 -1 -1 X X -1
-1 X X -1 -1 X X -1 -1
-1 -1 X 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 X -1 -1
-1 -1 X X -1 -1 X X -1
-1 X X -1 -1 -1 -1 X -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

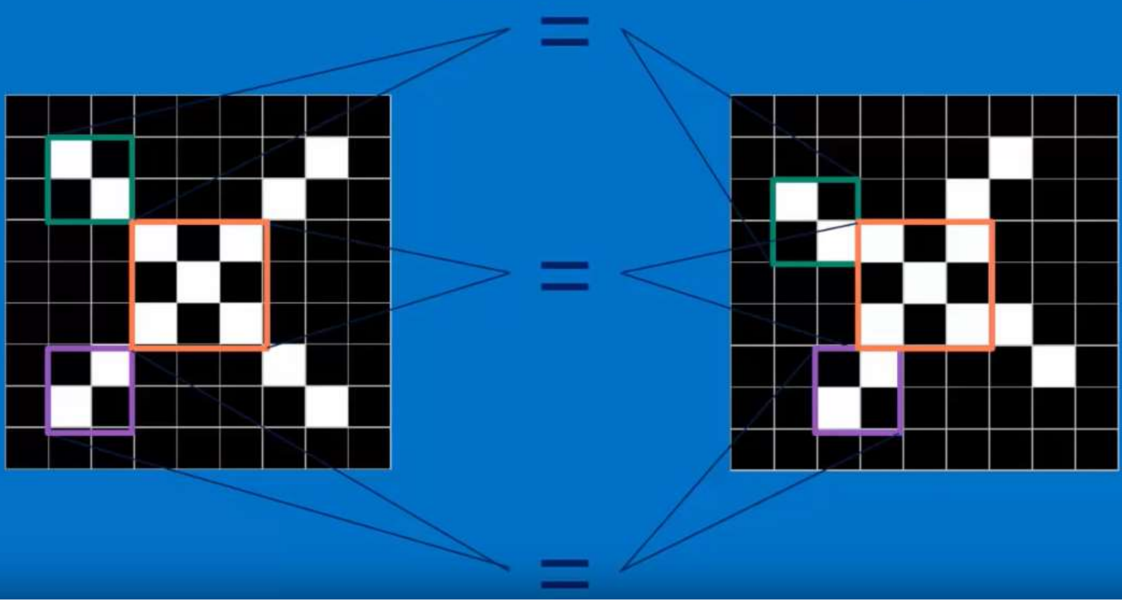
Computers are literal

-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

$\neq$

-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

ConvNets match pieces of the image



Features match pieces of the image

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

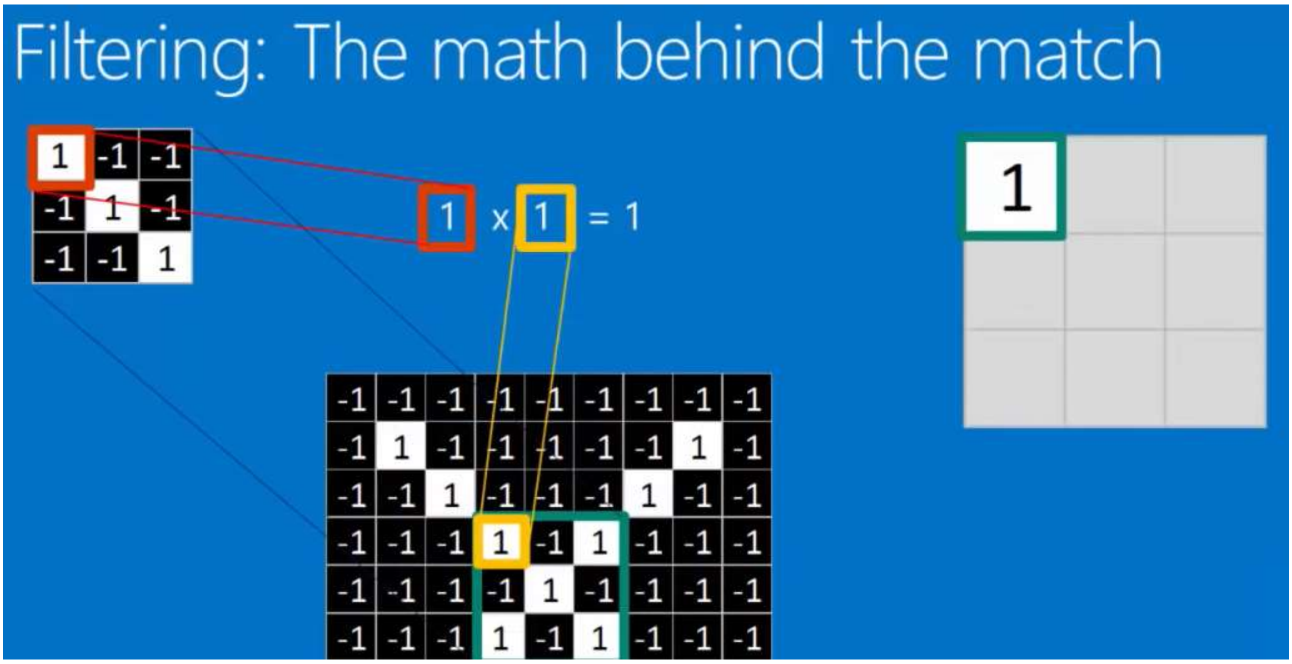
Filtering: The math behind the match

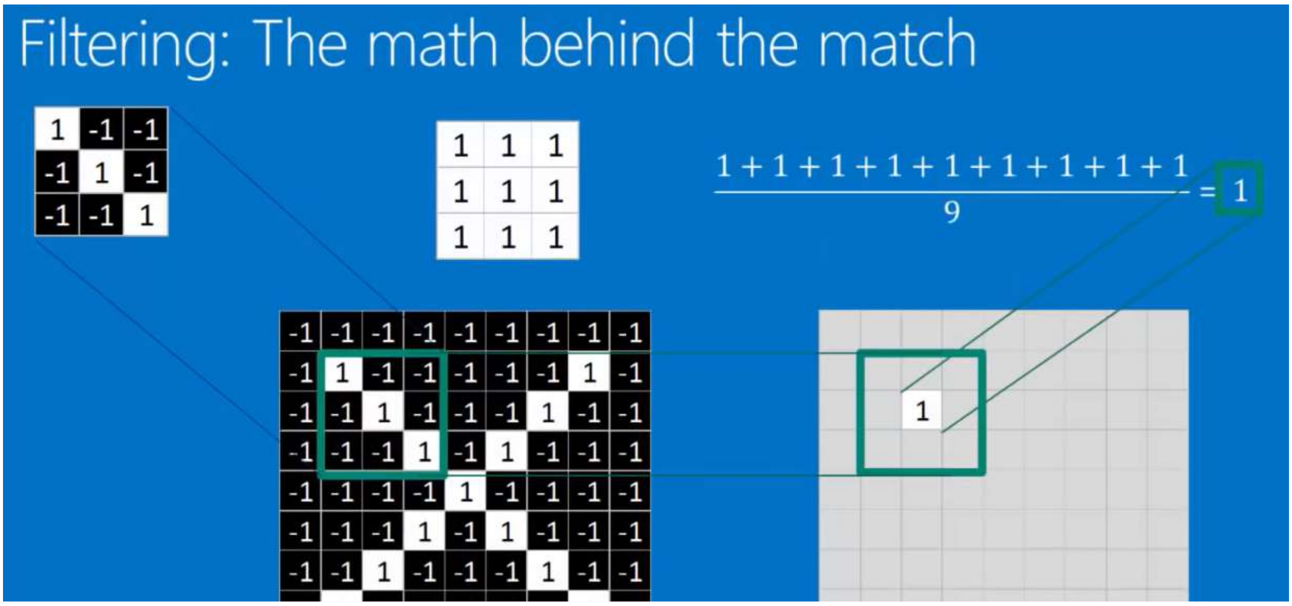
1	-1	-1
-1	1	-1
-1	-1	1

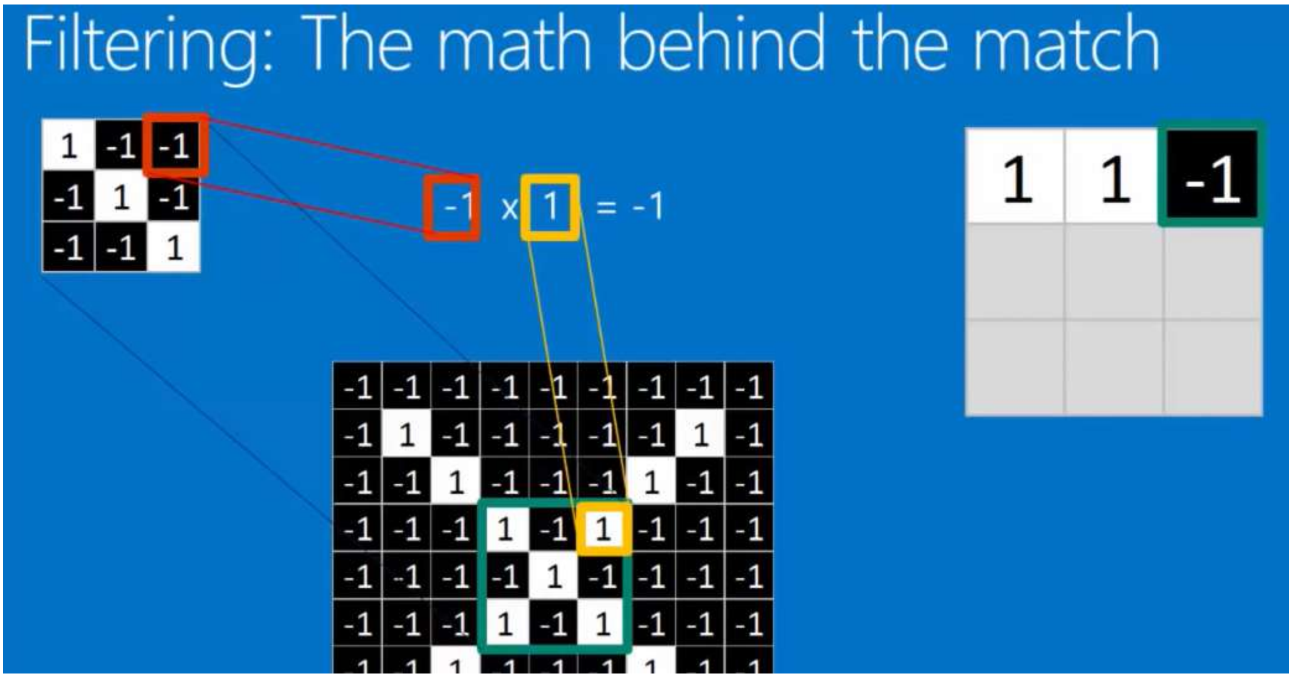
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

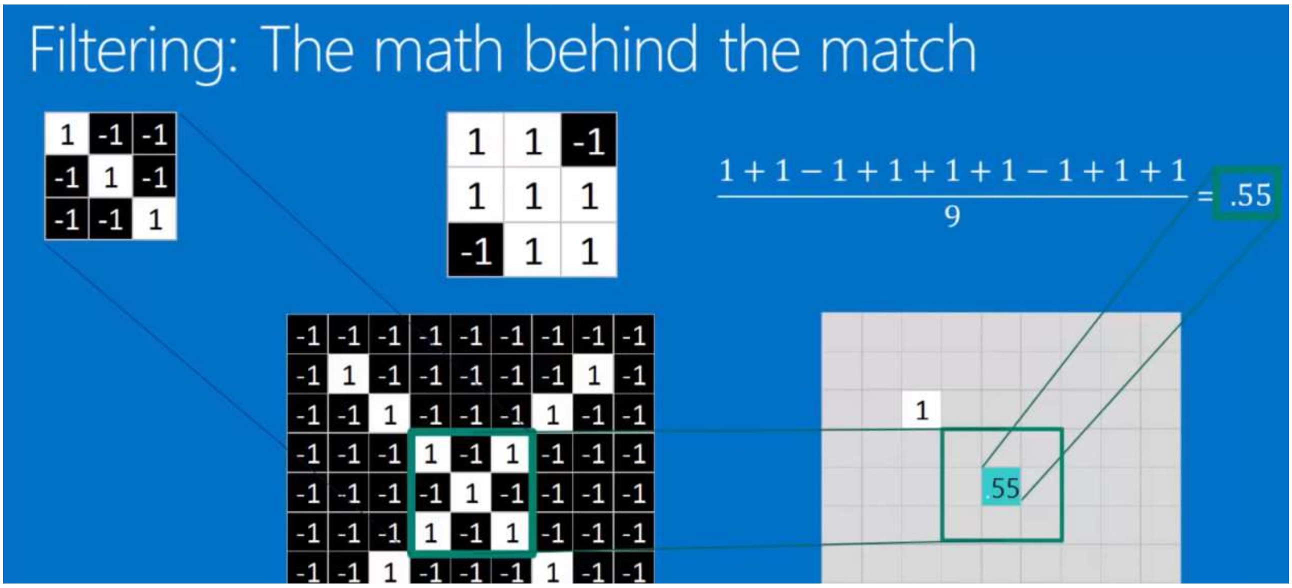
Filtering: The math behind the match

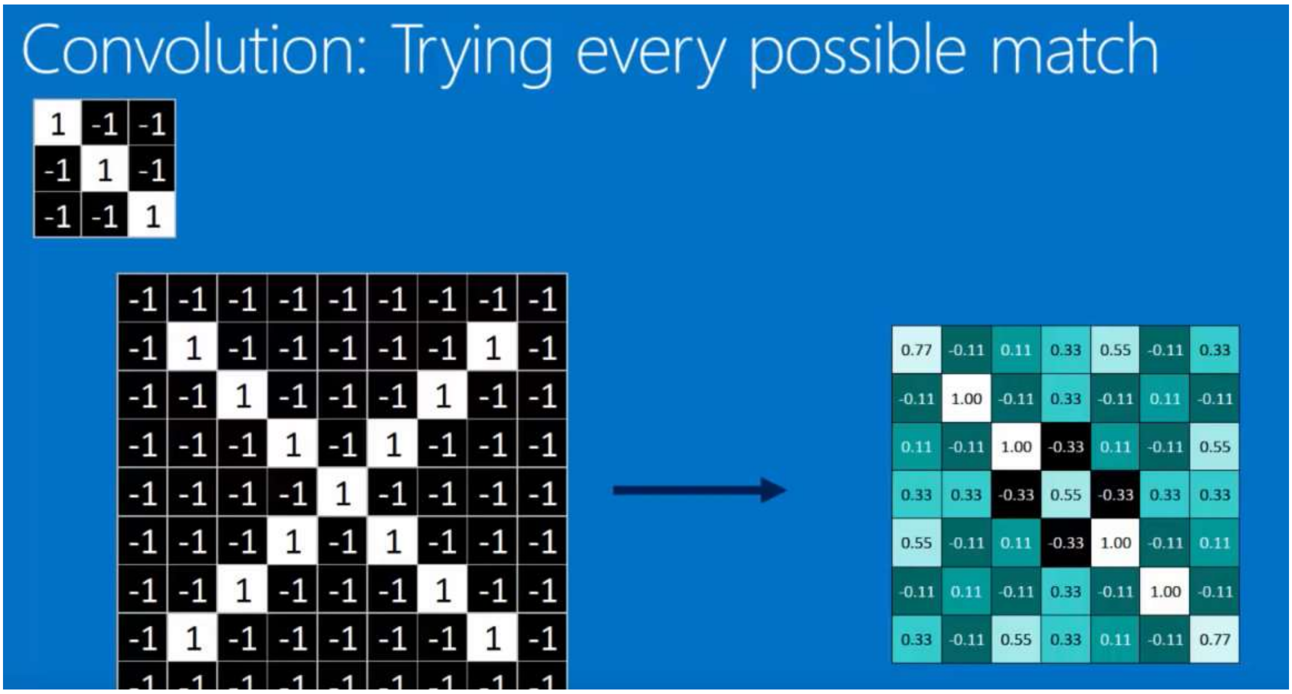
1. Line up the feature and the image patch.
2. Multiply each image pixel by the corresponding feature pixel.
3. Add them up.
4. Divide by the total number of pixels in the feature.

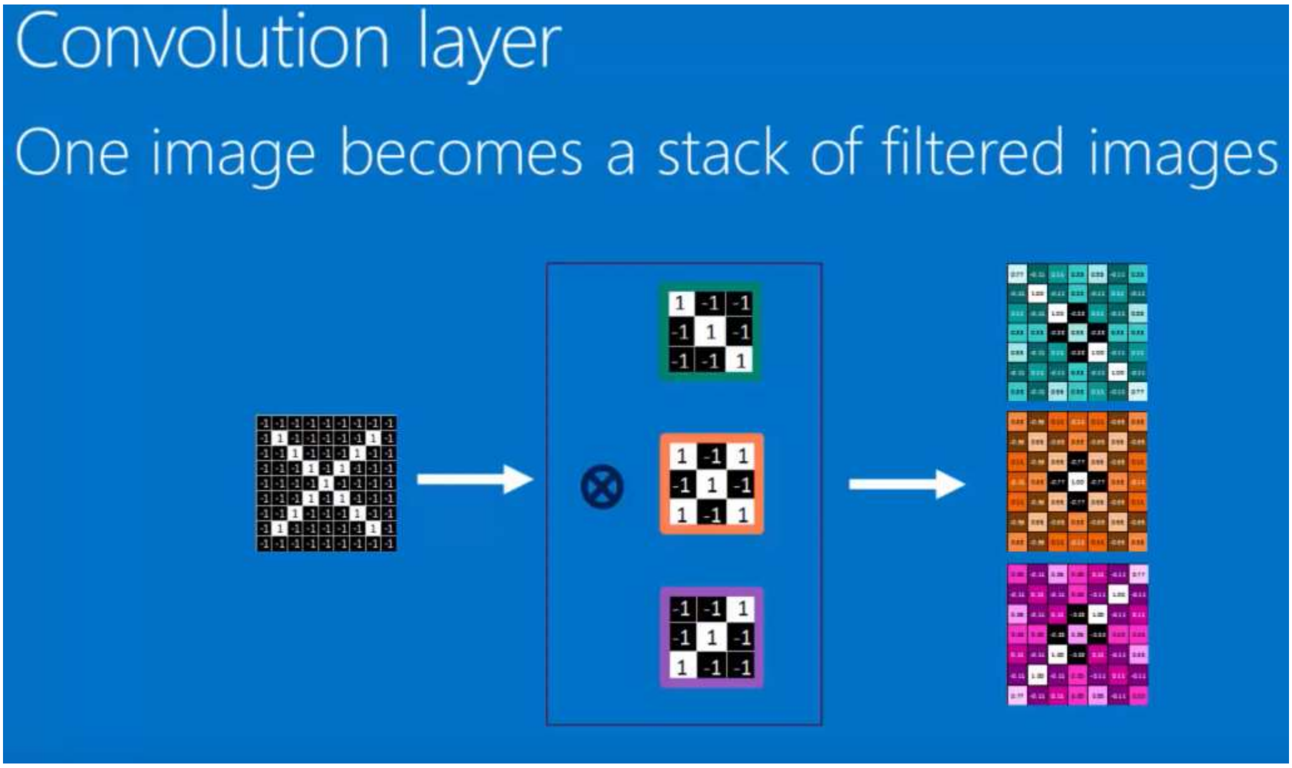

Filtering: The math behind the match
1 x 1 = 1
1

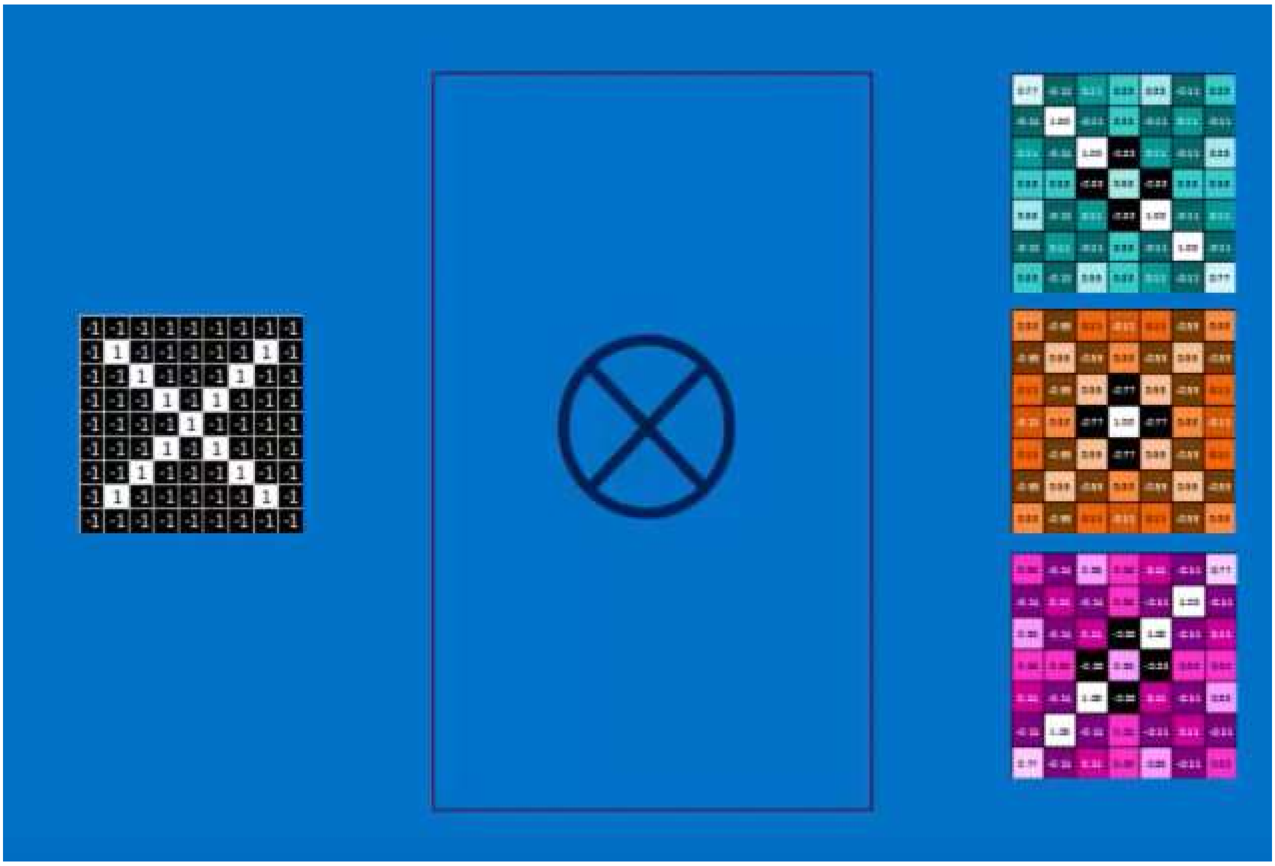

Filtering: The math behind the match
1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1
9
= 1


Filtering: The math behind the match
-1 x 1 = -1


Filtering: The math behind the match
$\frac{1+1-1+1+1+1-1+1+1}{9} = .55$
.55


Convolution: Trying every possible match

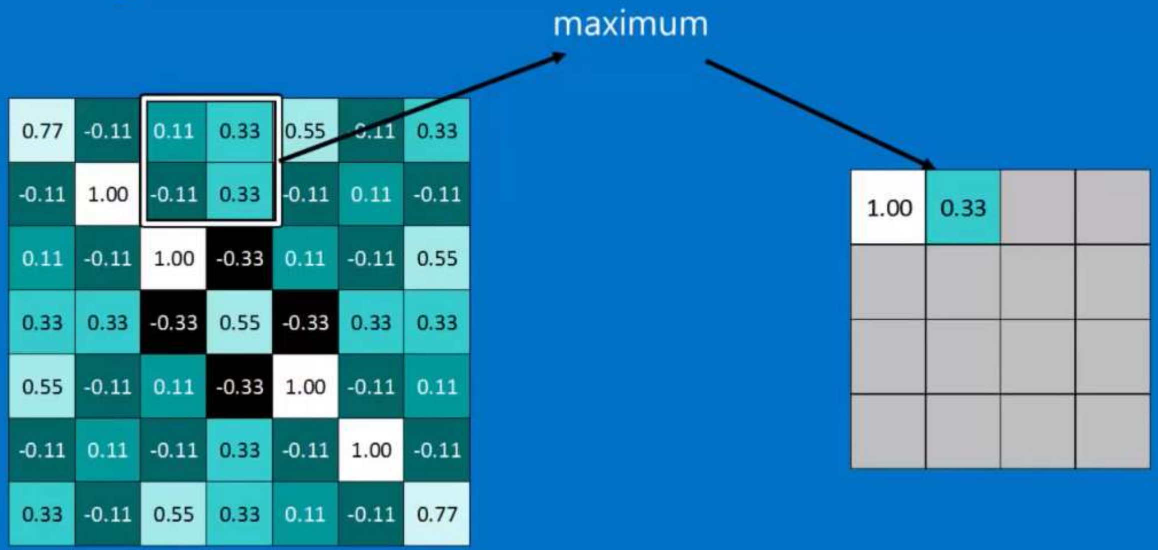

Convolution layer
One image becomes a stack of filtered images



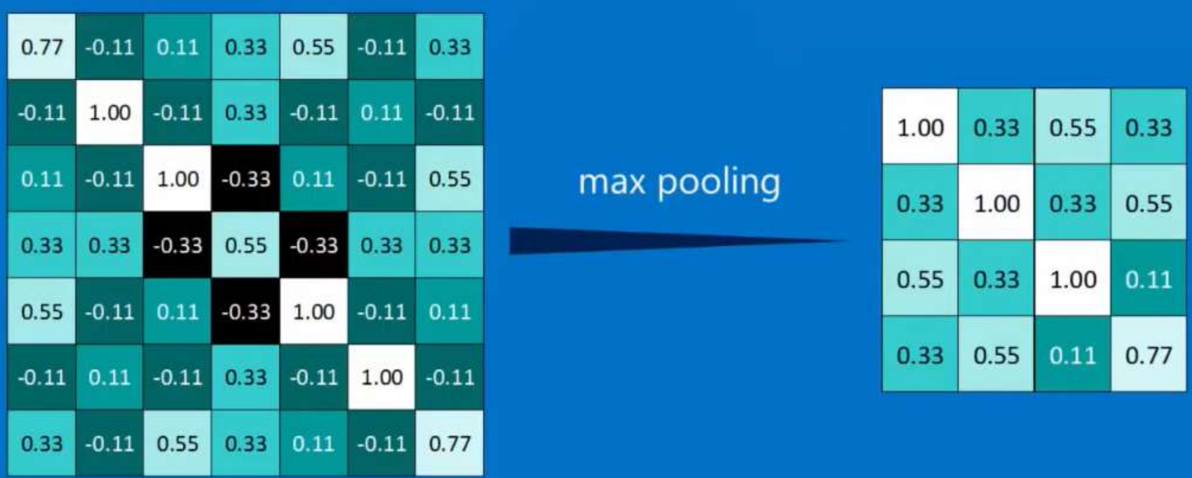
Pooling: Shrinking the image stack

1. Pick a window size (usually 2 or 3).
2. Pick a stride (usually 2).
3. Walk your window across your filtered images.
4. From each window, take the maximum value.

Pooling

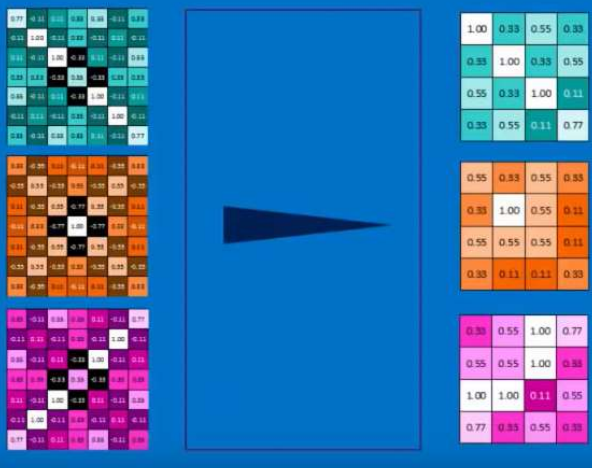


Pooling



Pooling layer

A stack of images becomes a stack of smaller images.



Normalization

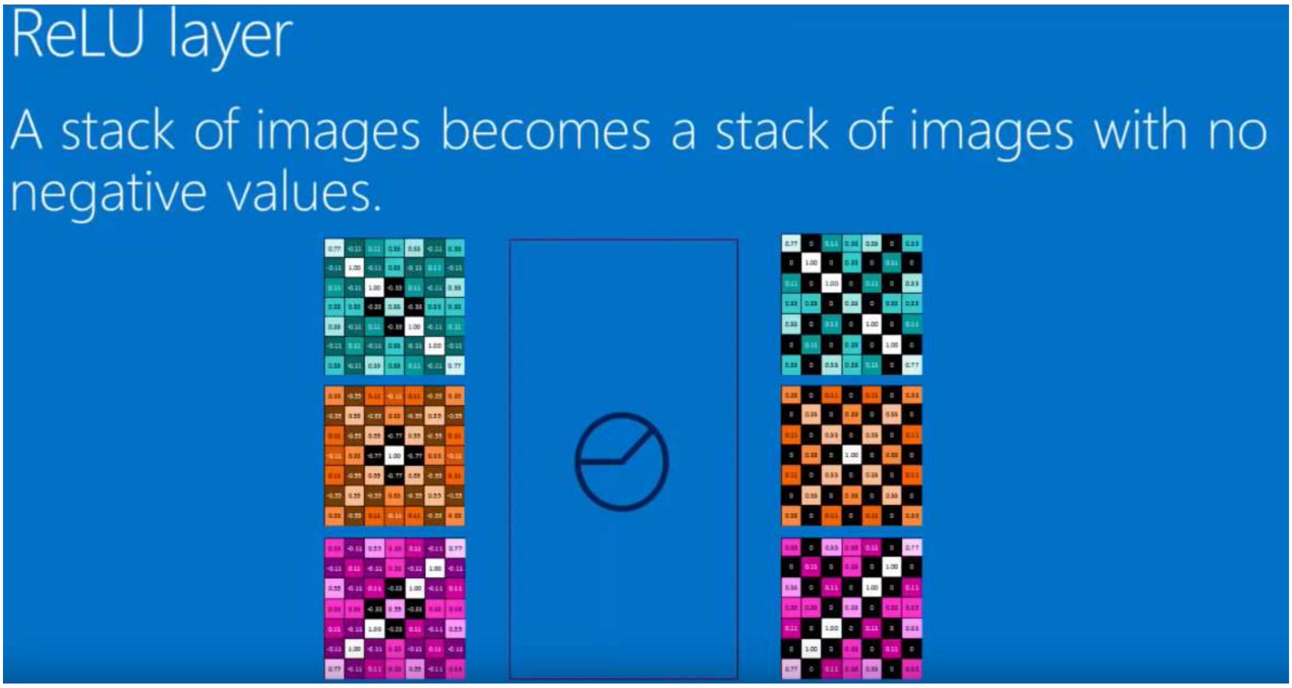
Keep the math from breaking by tweaking each of the values just a bit.

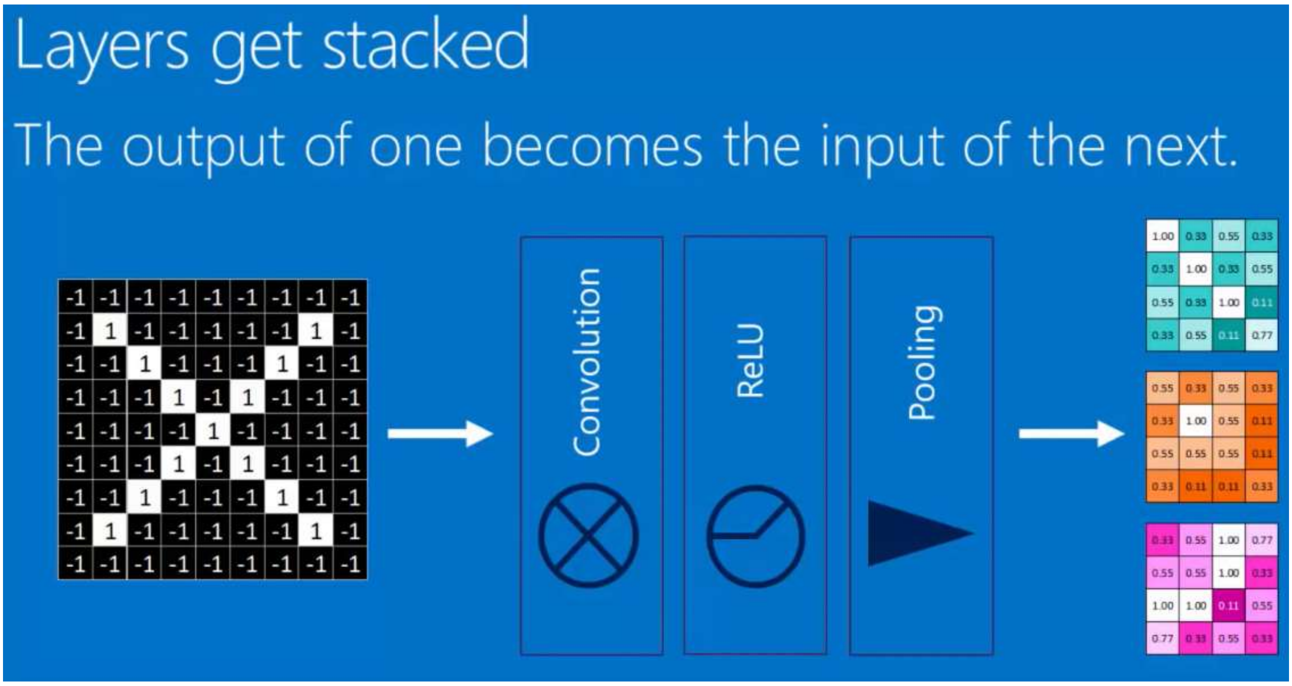
Change everything negative to zero.

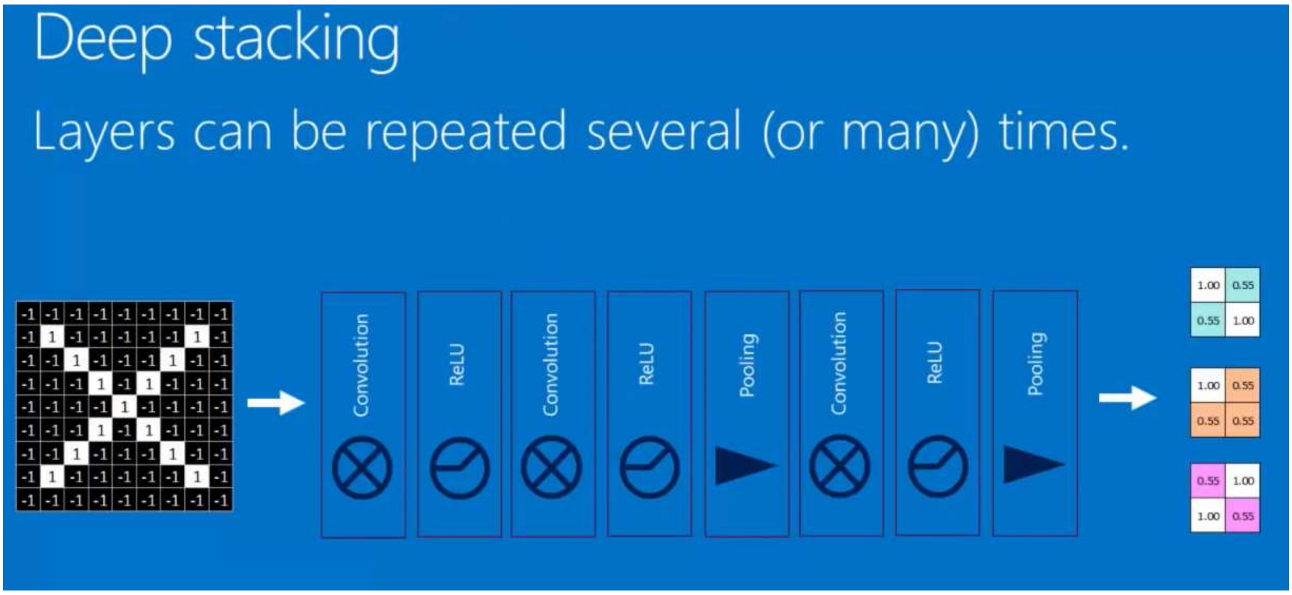
Rectified Linear Units (ReLUs)

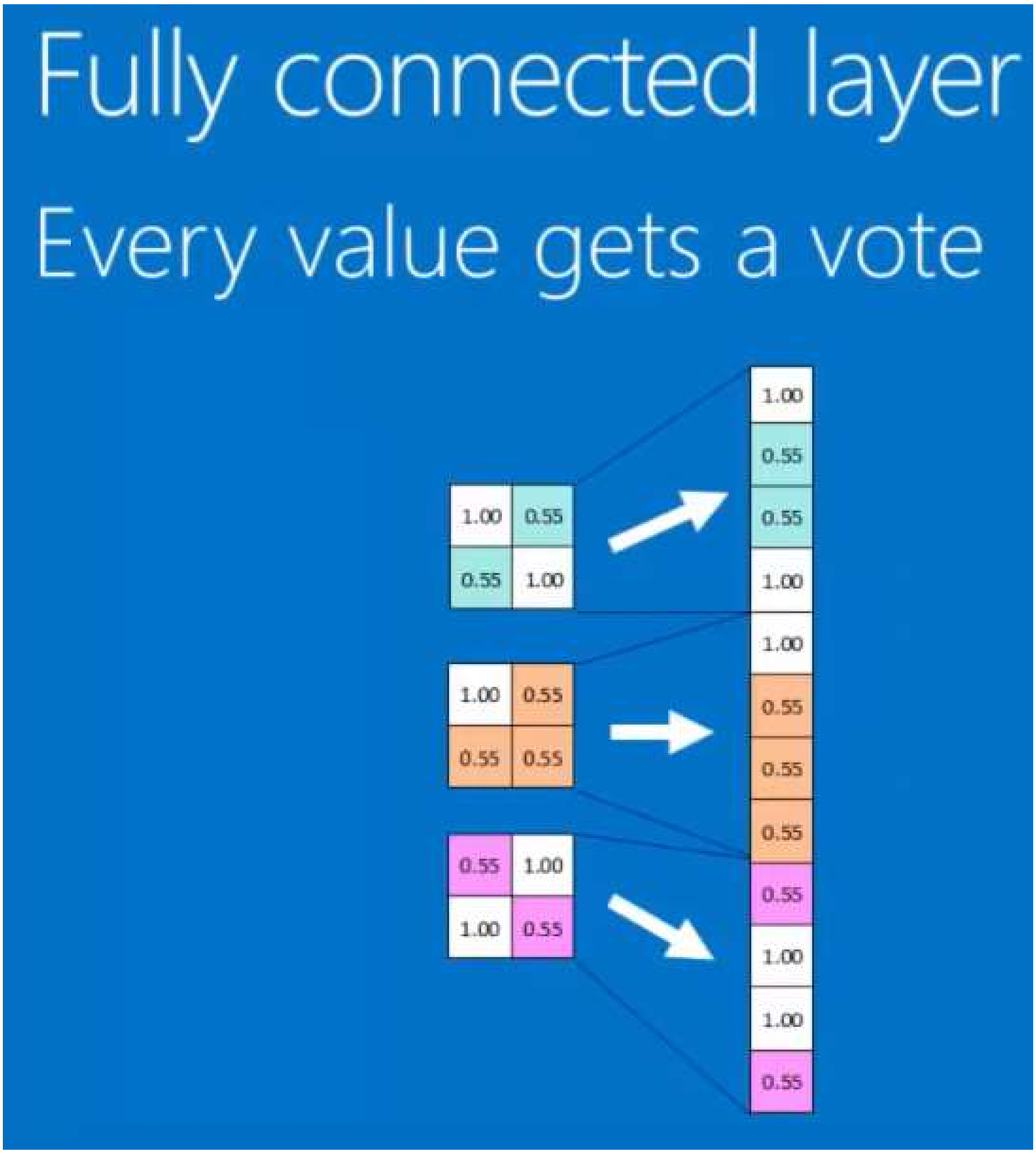
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

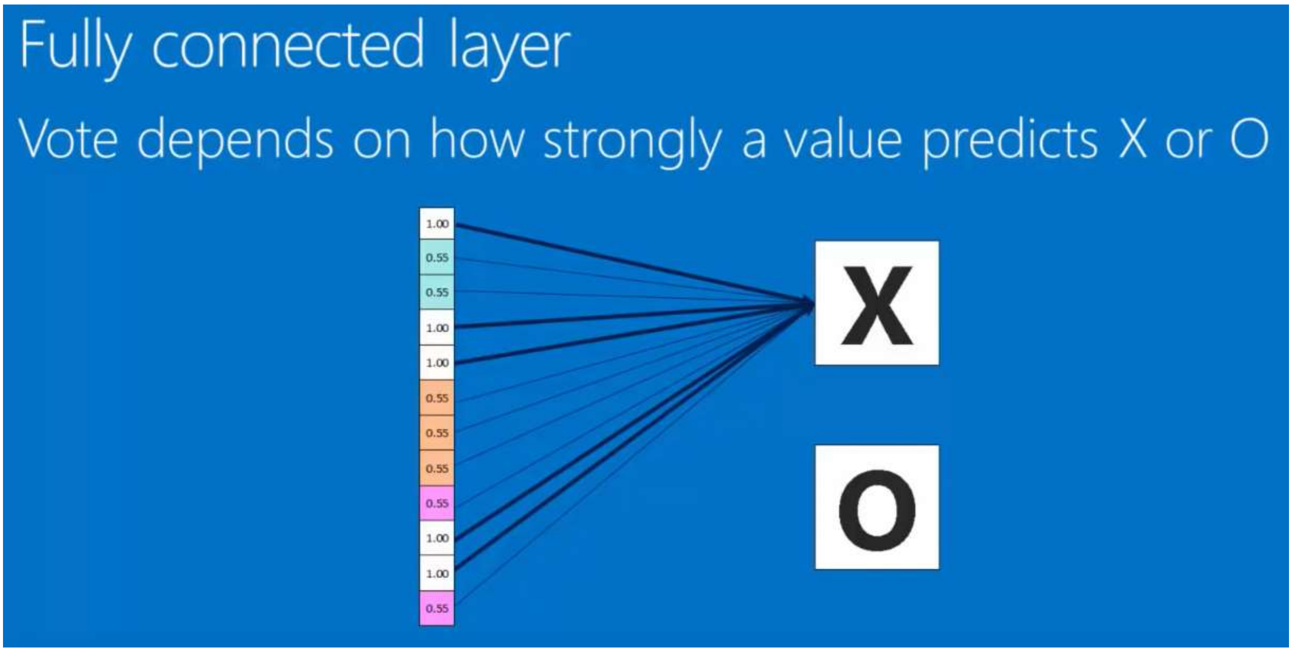
0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

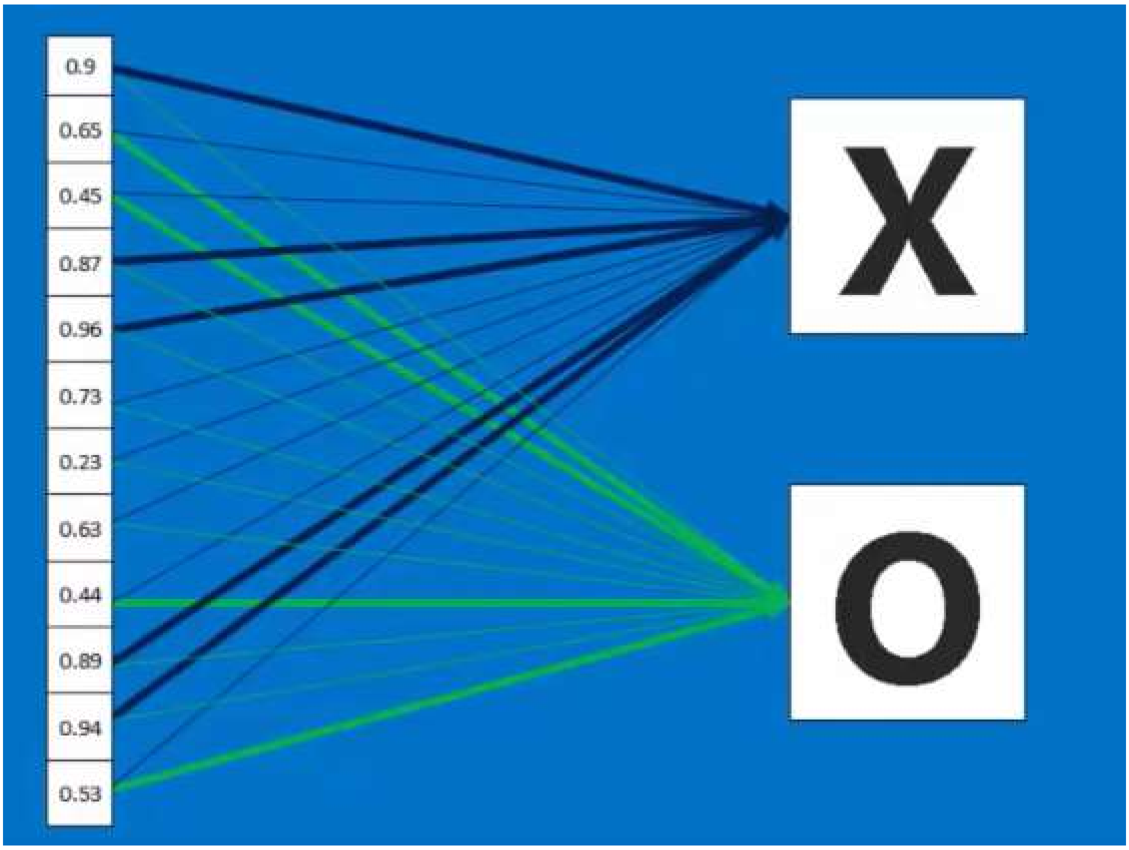

ReLU layer
A stack of images becomes a stack of images with no negative values.

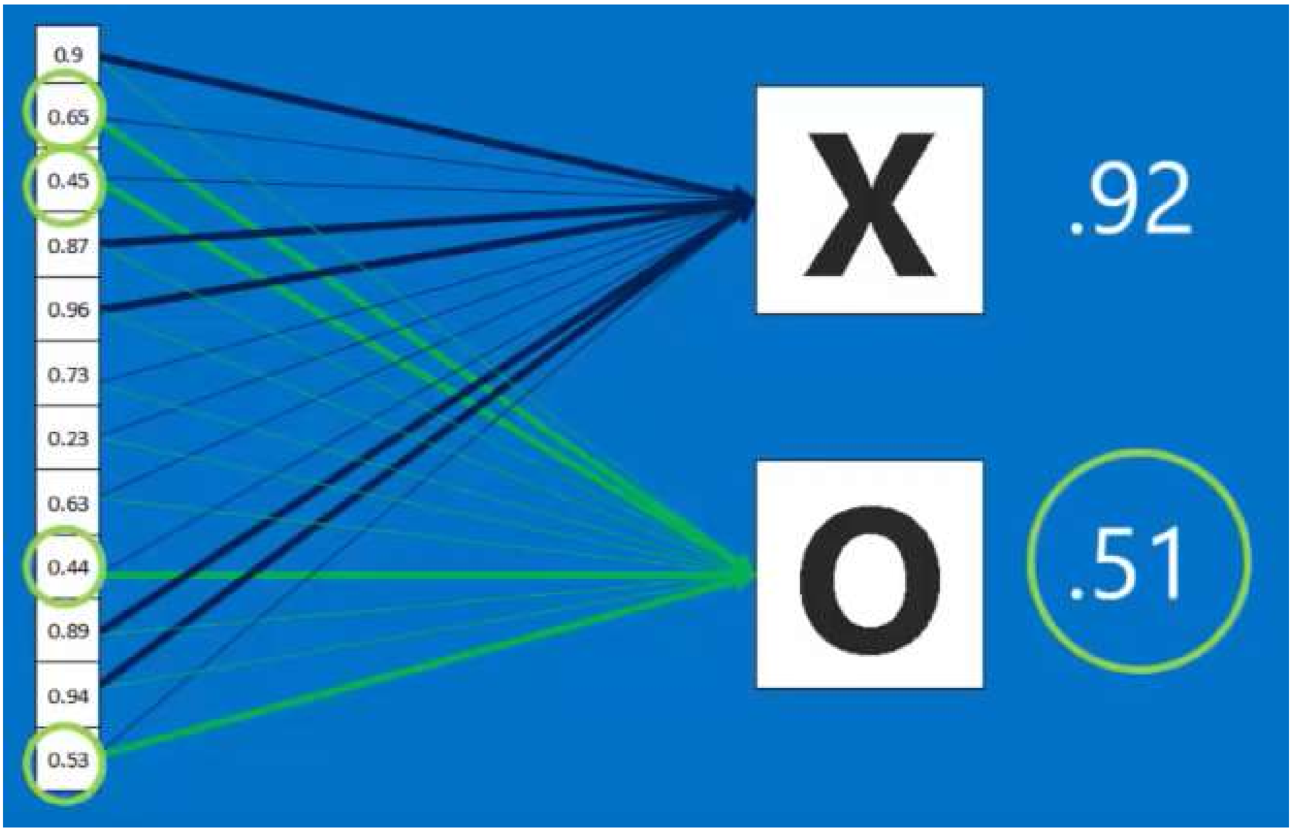

Layers get stacked
The output of one becomes the input of the next.
Convolution
ReLU
Pooling

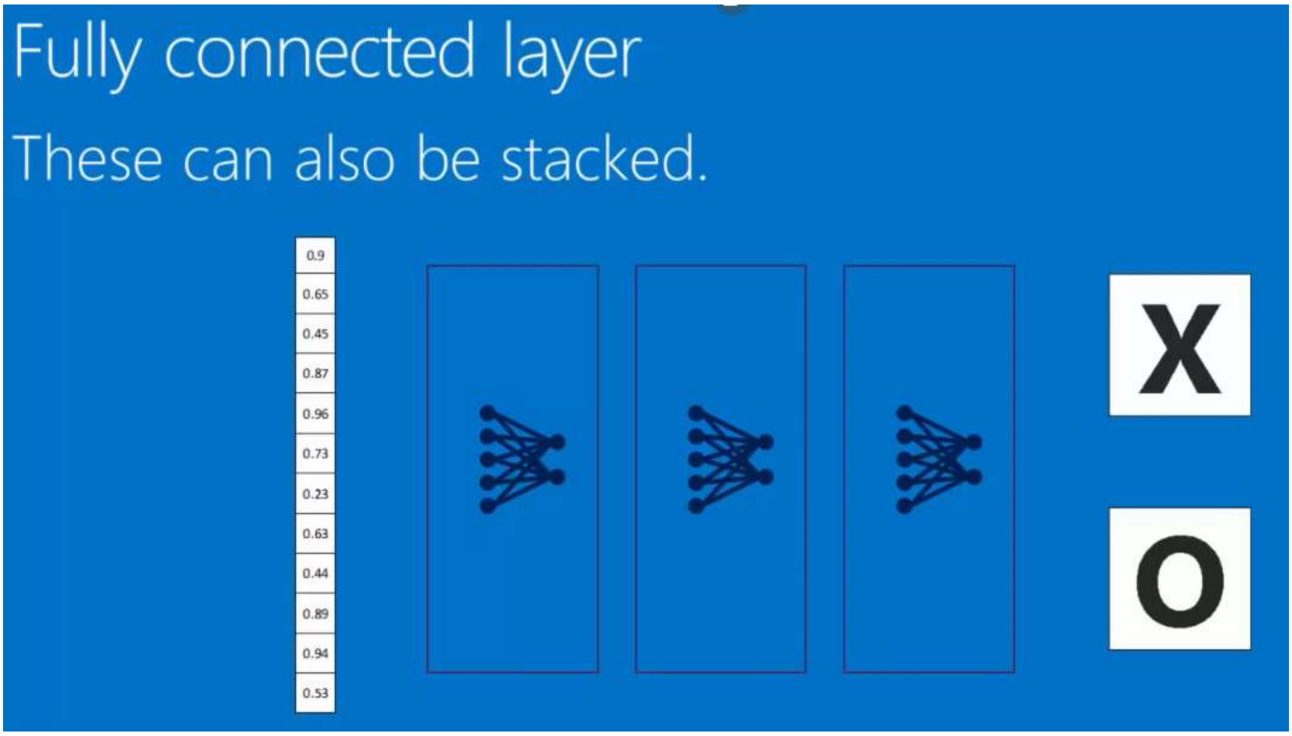

Deep stacking
Layers can be repeated several (or many) times.
Convolution
ReLU
Convolution
ReLU
Pooling
Convolution
ReLU
Pooling
1.00 0.55
0.55 1.00
1.00 0.55
0.55 0.55
0.55 1.00
1.00 0.55


Fully connected layer
Every value gets a vote
1.00 0.55
0.55 1.00
1.00 0.55
0.55 0.55
0.55 1.00
1.00 0.55
1.00
0.55
0.55
1.00
1.00
0.55
0.55
0.55
0.55
1.00
1.00
0.55


Fully connected layer
Vote depends on how strongly a value predicts X or O
1.00
0.55
0.55
1.00
1.00
0.55
0.55
0.55
0.55
1.00
1.00
0.55
X
O


0.9
0.65
0.45
0.87
0.96
0.73
0.23
0.63
0.44
0.89
0.94
0.53
X
O


0.9
0.65
0.45
0.87
0.96
0.73
0.23
0.63
0.44
0.89
0.94
0.53
X
.92
O
.51


Fully connected layer
These can also be stacked.
0.9
0.65
0.45
0.87
0.96
0.73
0.23
0.63
0.44
0.89
0.94
0.53
X
O

Putting it all together

A set of pixels becomes a set of votes.

Learning

Q: Where do all the magic numbers come from?

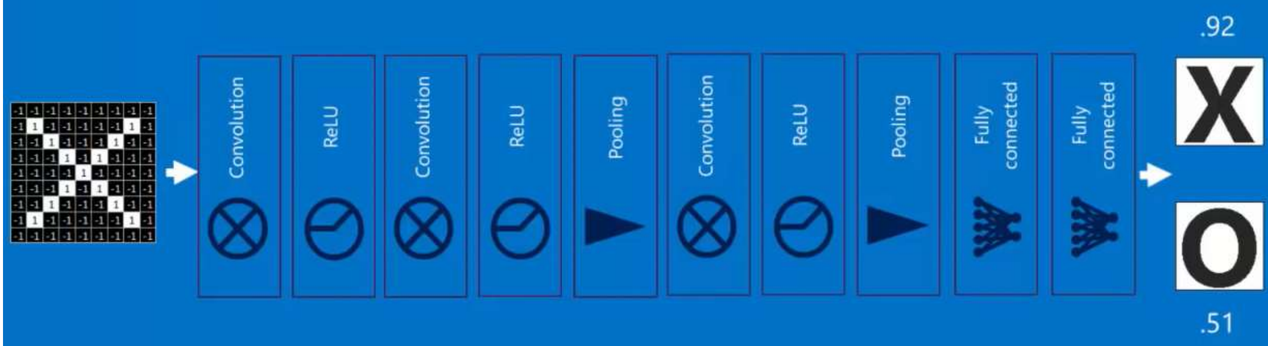
Features in convolutional layers

Voting weights in fully connected layers

A: Backpropagation

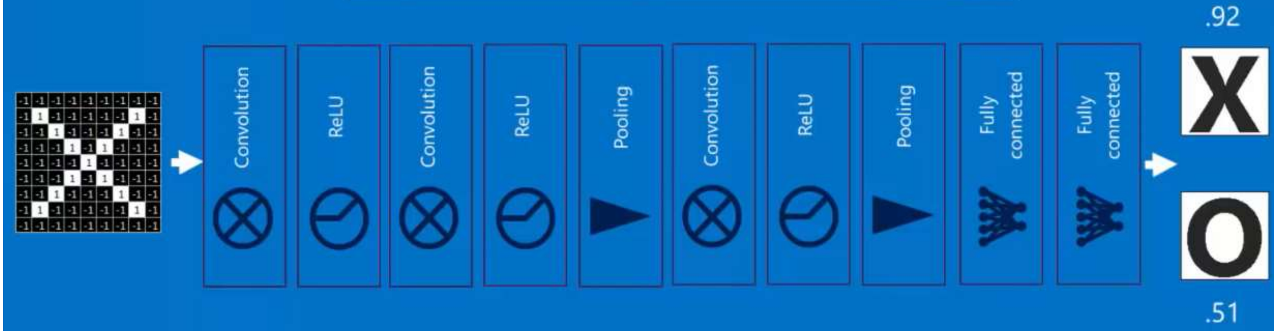
Backprop

Error = right answer – actual answer



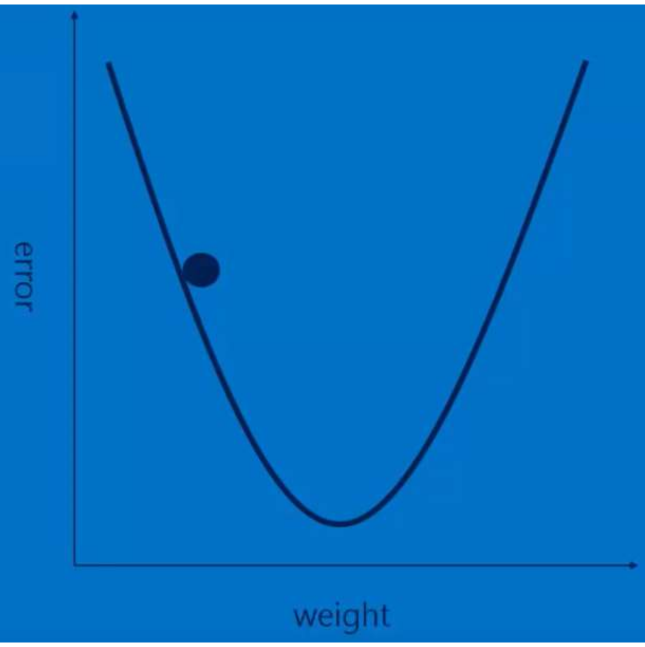
Backprop

	Right answer	Actual answer	Error
X	1	0.92	0.08
O	0	0.51	0.49
		Total	0.57



Gradient descent

For each feature pixel and voting weight, adjust it up and down a bit and see how the error changes.



Hyperparameters (knobs)

Convolution
- Number of features
- Size of features

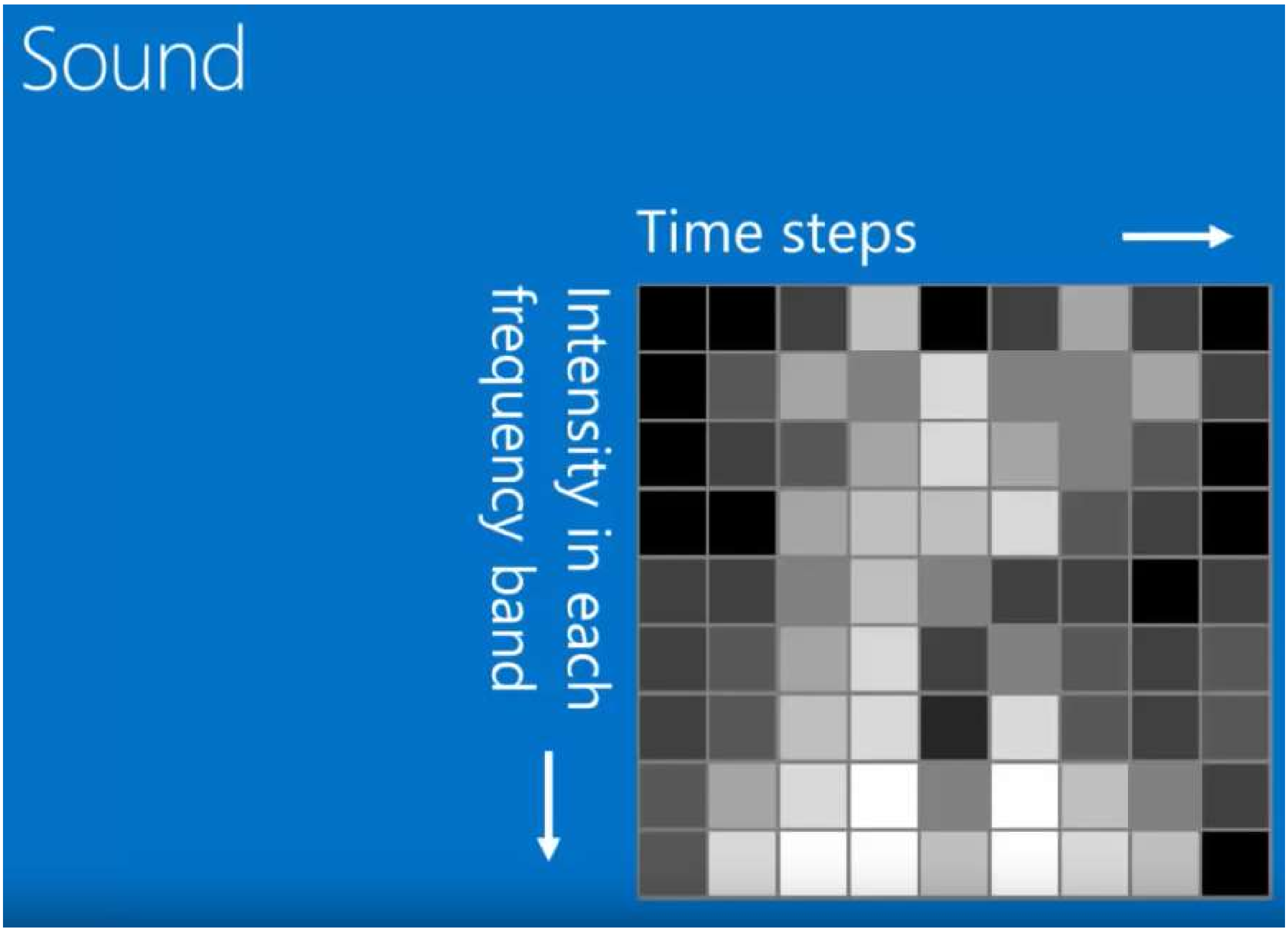
Pooling
- Window size
- Window stride

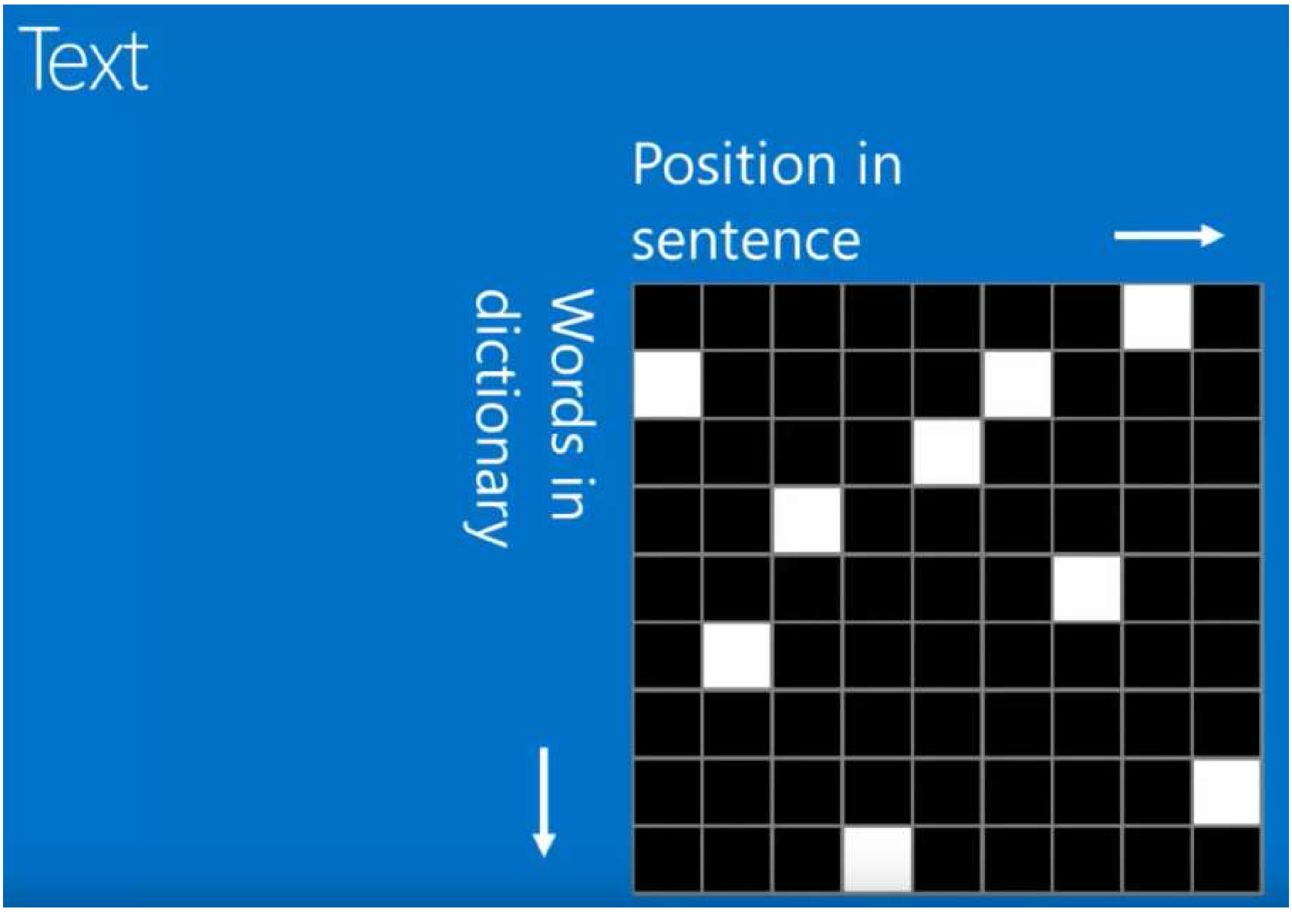
Fully Connected
- Number of neurons

Architecture

How many of each type of layer?

In what order?


Sound
Time steps
Intensity in each frequency band


Text
Position in sentence
Words in dictionary

Some ConvNet/DNN toolkits

Caffe (Berkeley Vision and Learning Center)
CNTK (Microsoft)
Deeplearning4j (Skymind)
TensorFlow (Google)
Theano (University of Montreal + broad community)
Torch (Ronan Collobert)
Many others