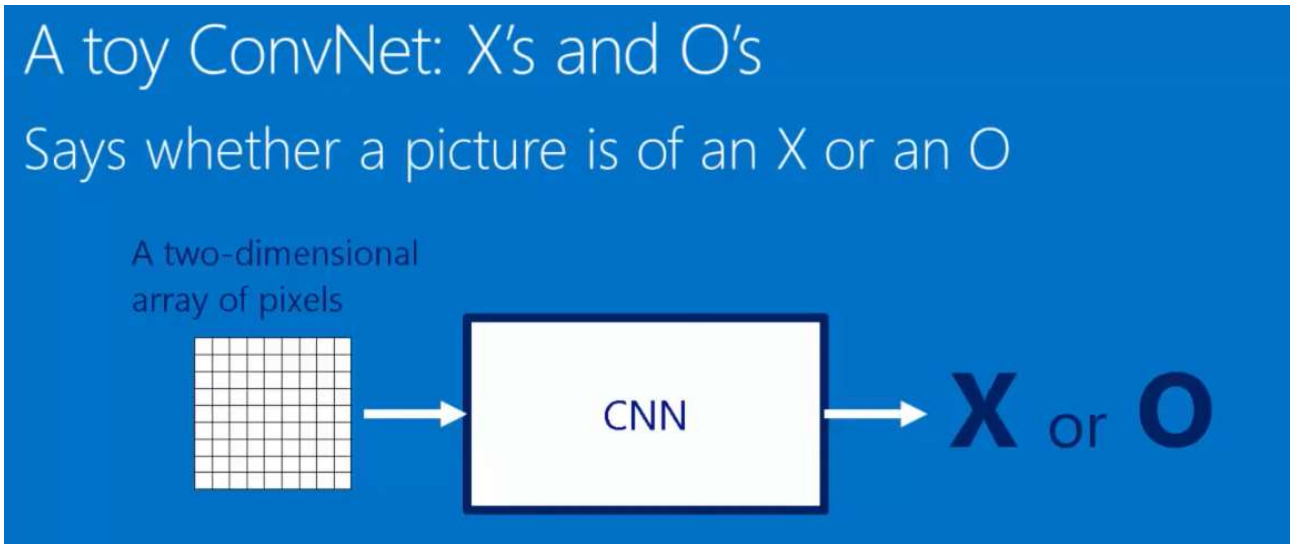
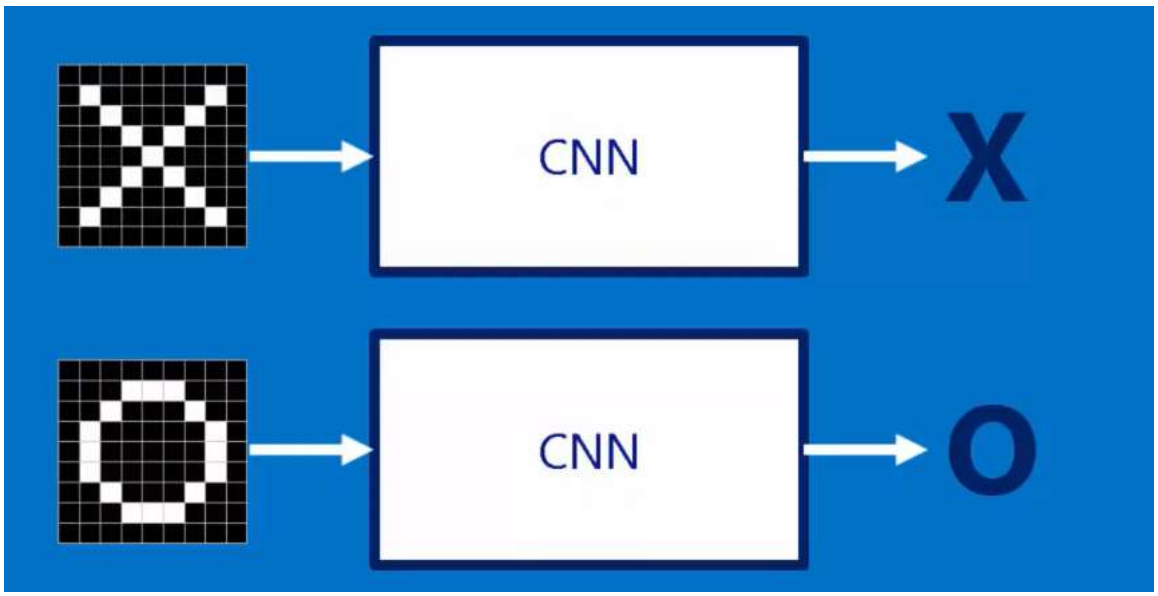
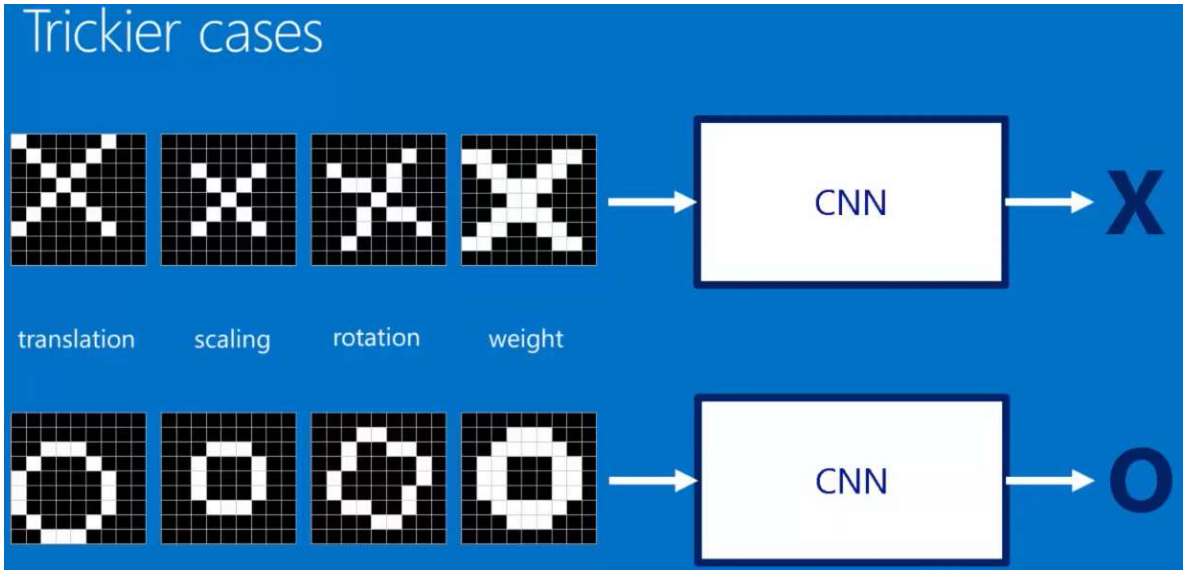
CONVNET:

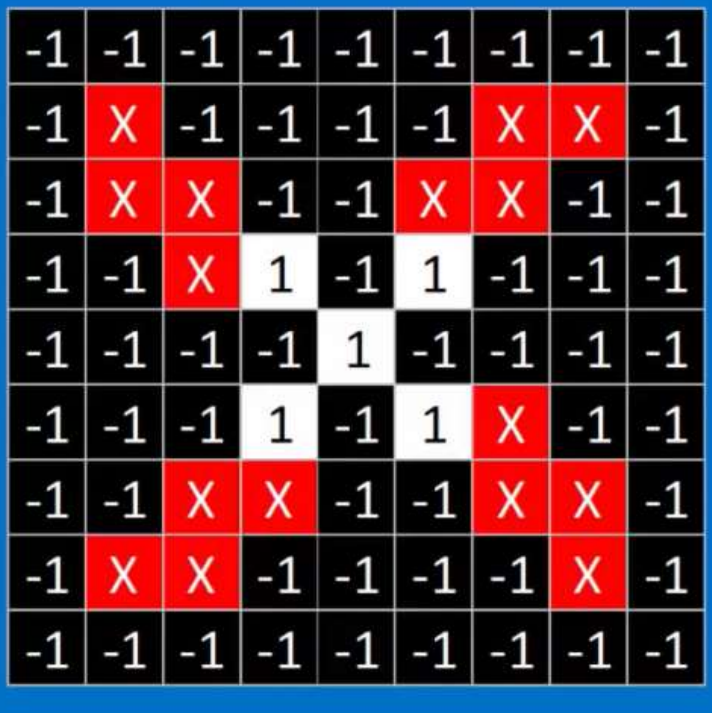

A toy ConvNet: X's and O's
Says whether a picture is of an X or an O
A two-dimensional
array of pixels
CNN
X or O


CNN
X
CNN
O

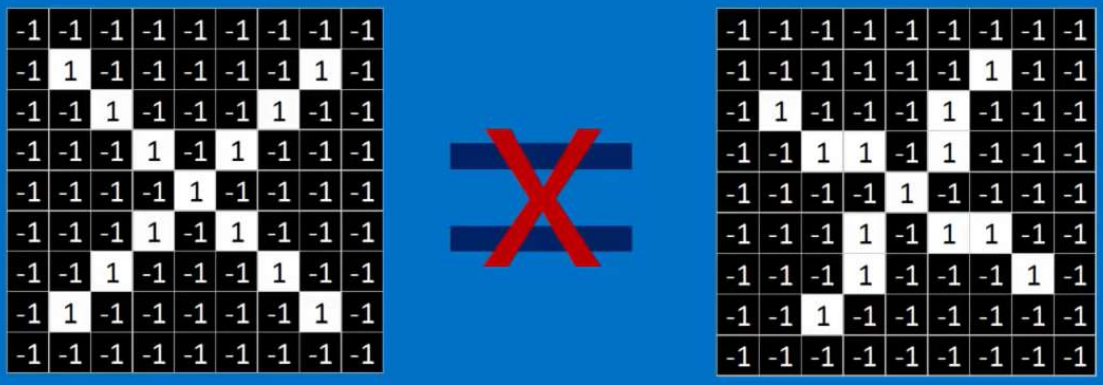

Trickier cases
CNN
X
translation
scaling
rotation
weight
CNN
O

What computers see

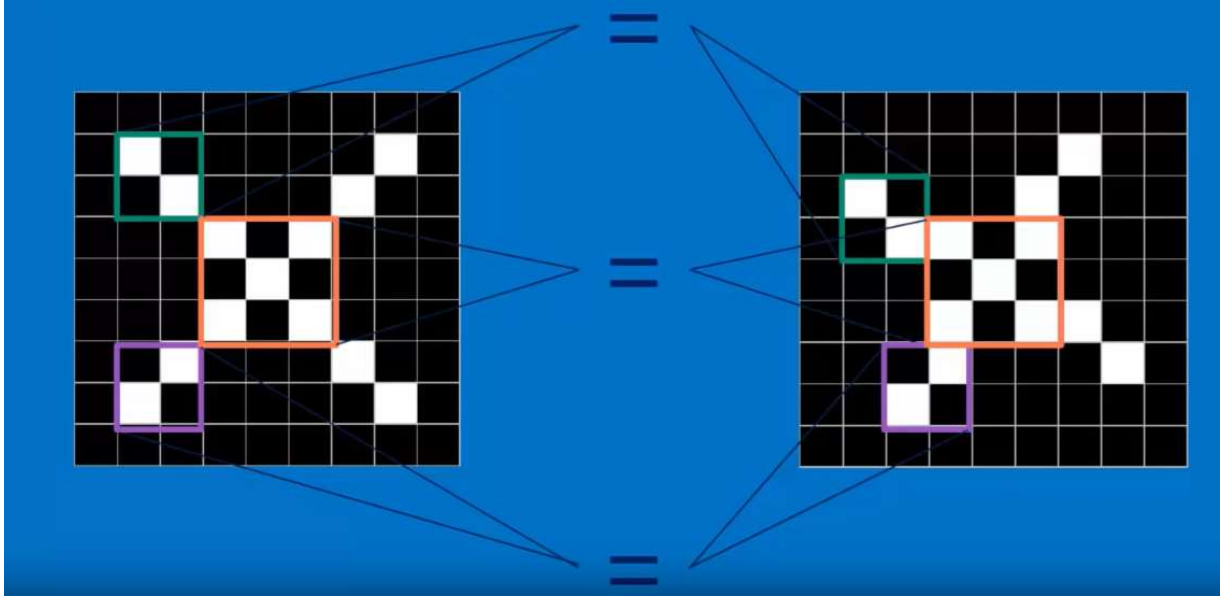
What computers see



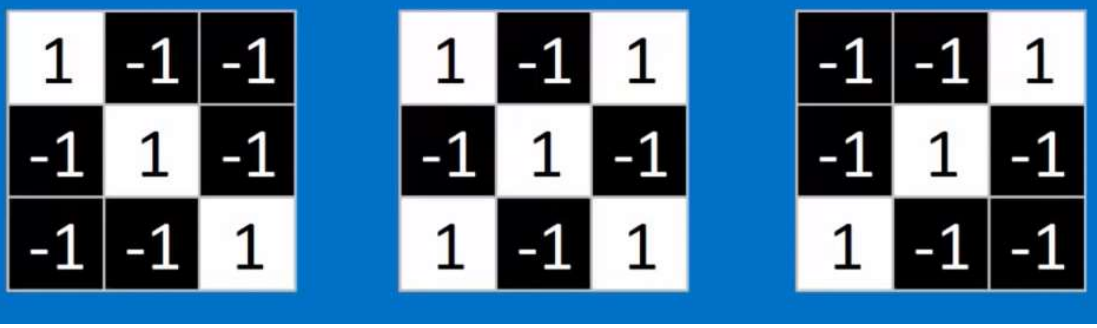
Computers are literal



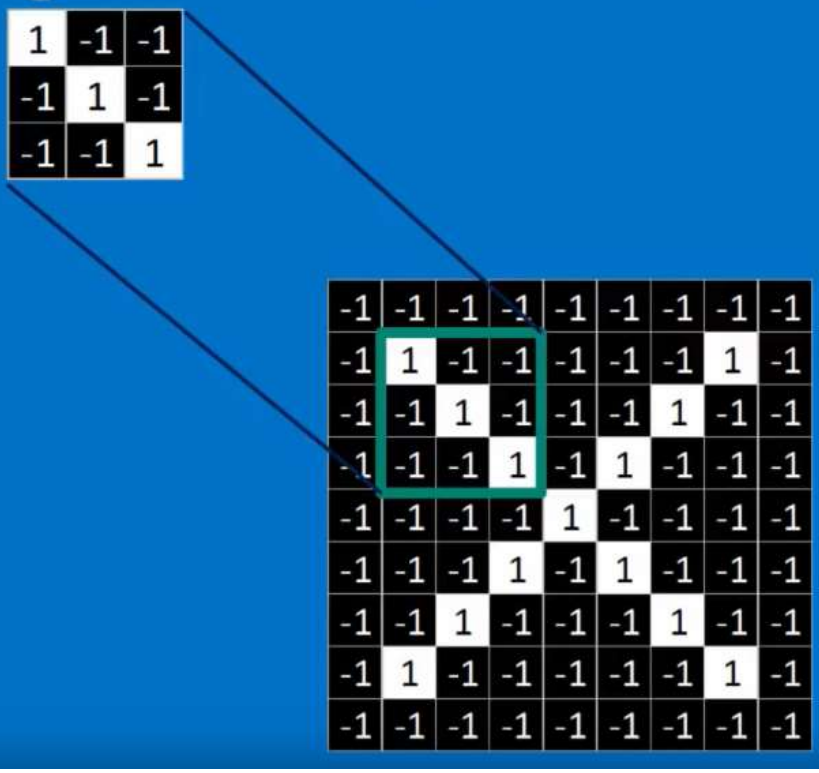
ConvNets match pieces of the image



Features match pieces of the image

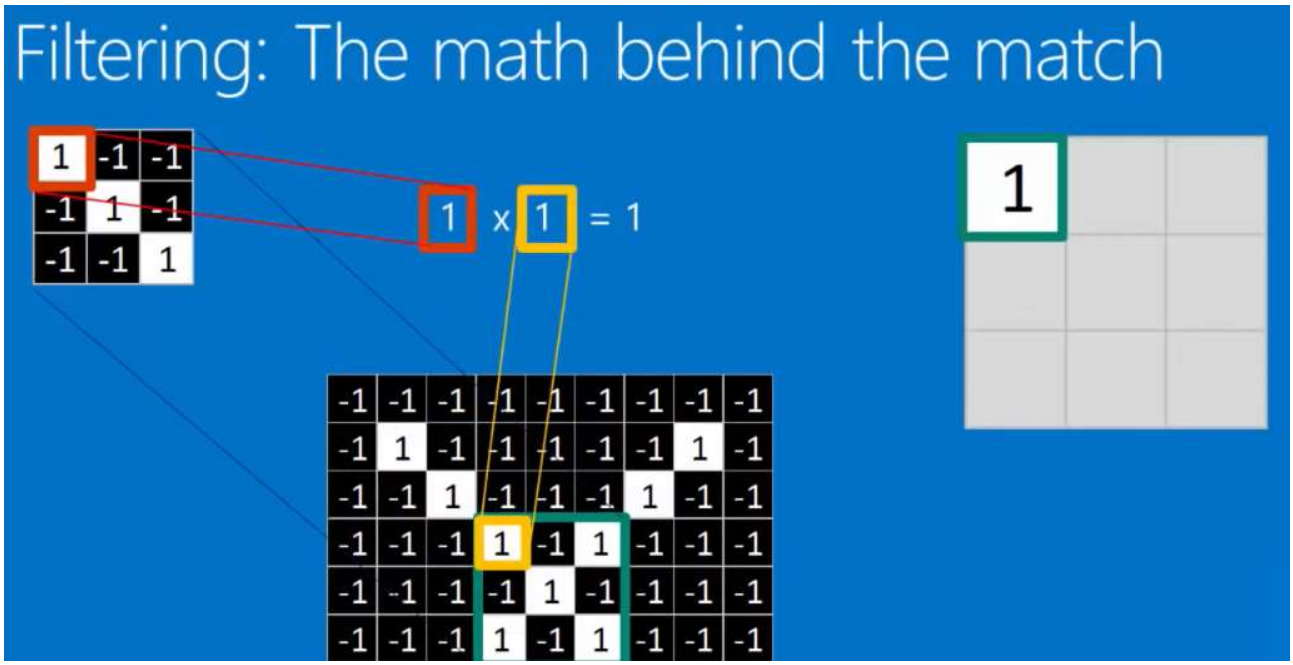


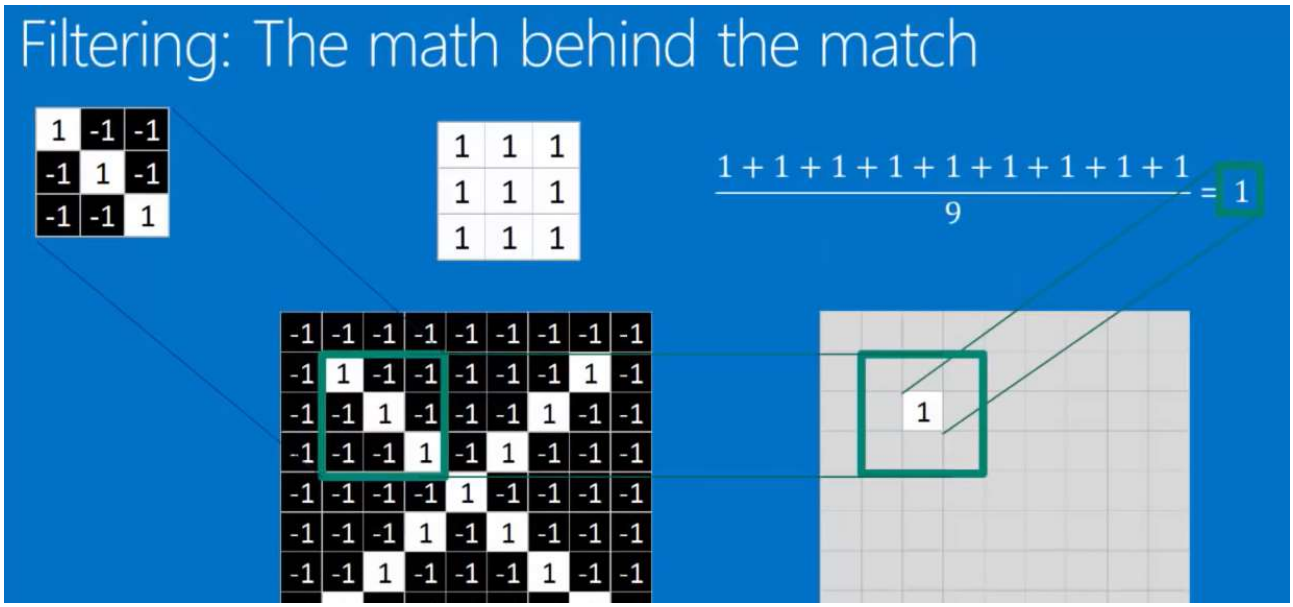
Filtering: The math behind the match



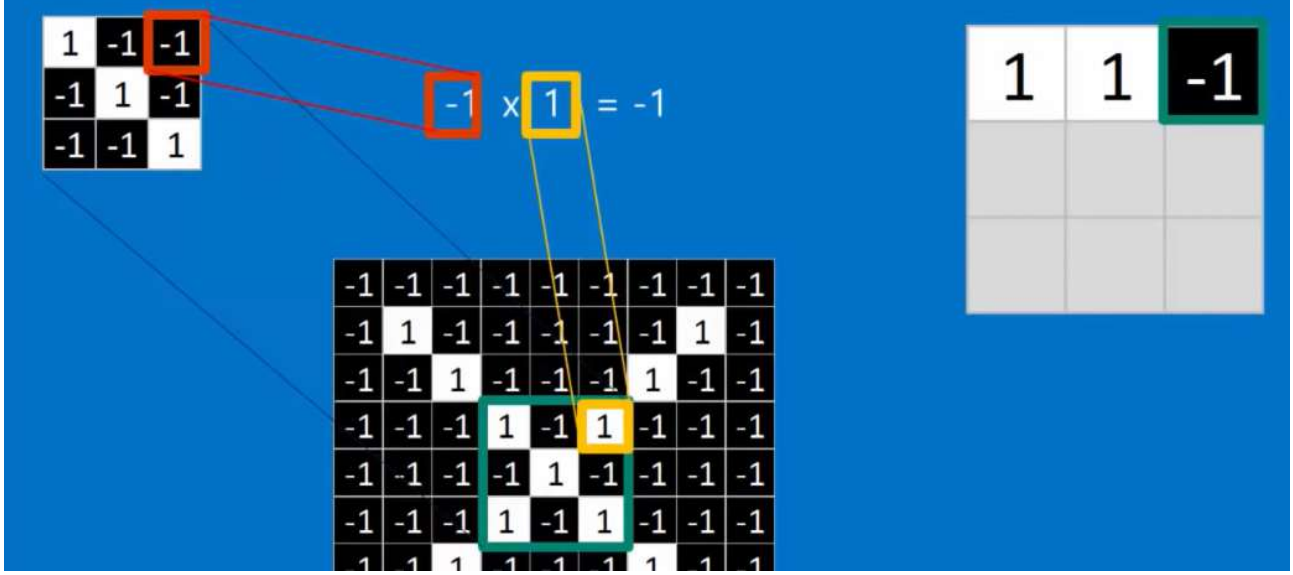
Filtering: The math behind the match

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.

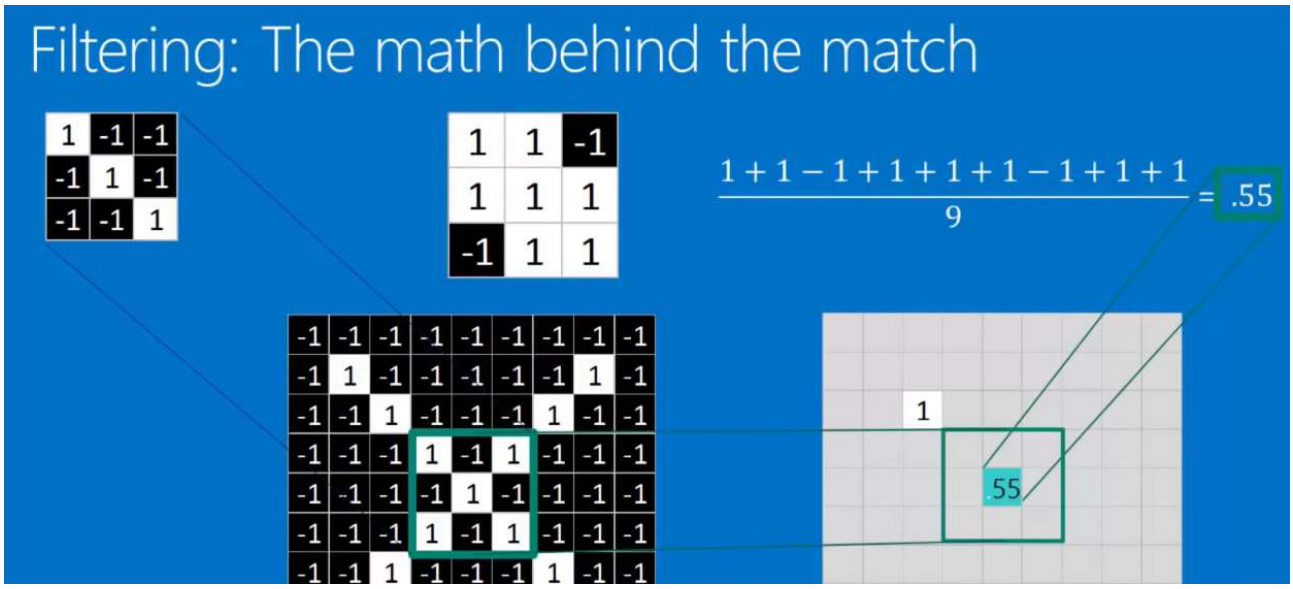

Filtering: The math behind the match
1 -1 -1
-1 1 -1
-1 -1 1
1 x 1 = 1
1

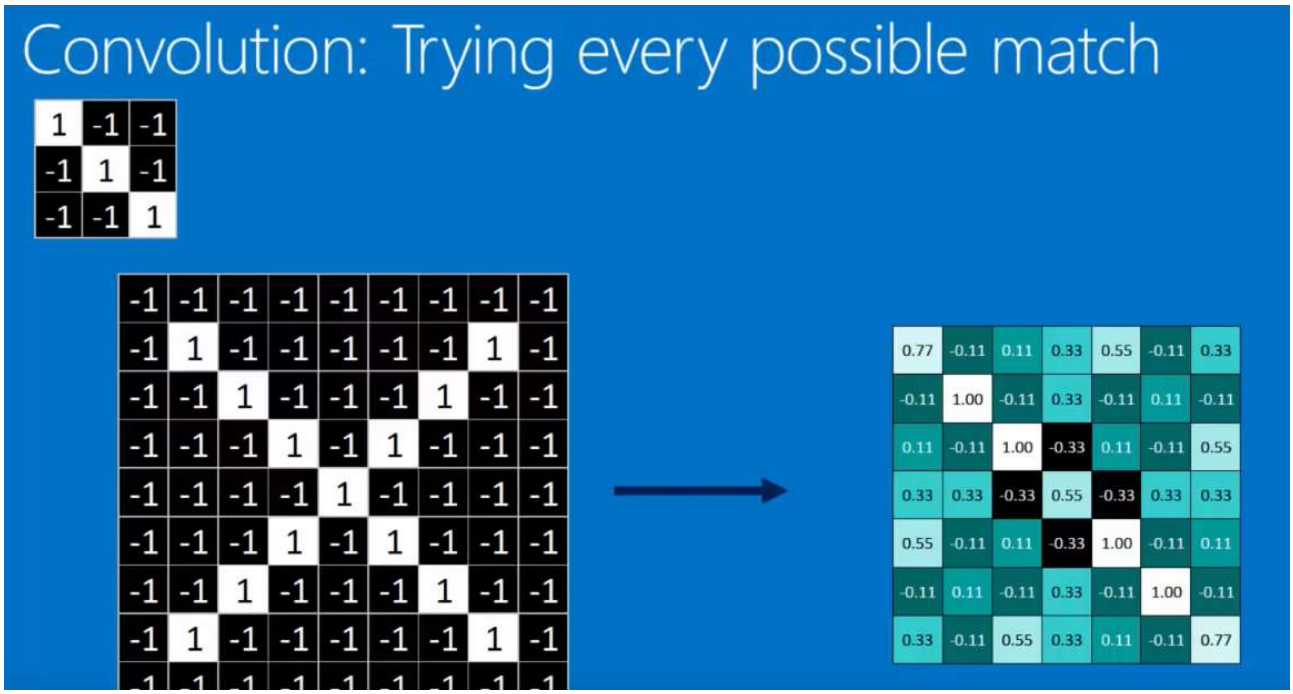

Filtering: The math behind the match
1 -1 -1
-1 1 -1
-1 -1 1
1 1 1
1 1 1
1 1 1
1+1+1+1+1+1+1+1+1 / 9 = 1
1

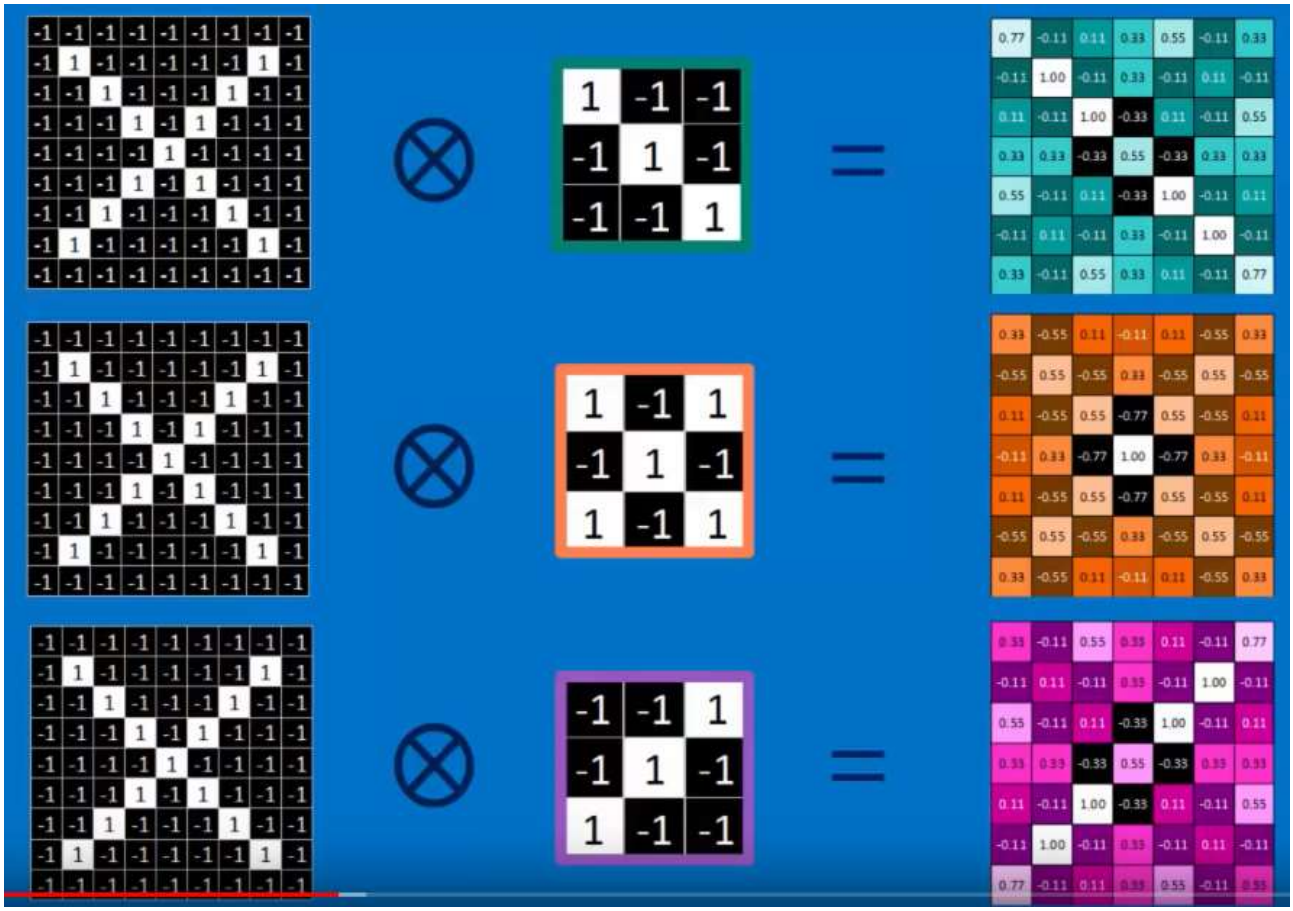
Filtering: The math behind the match



Filtering: The math behind the match

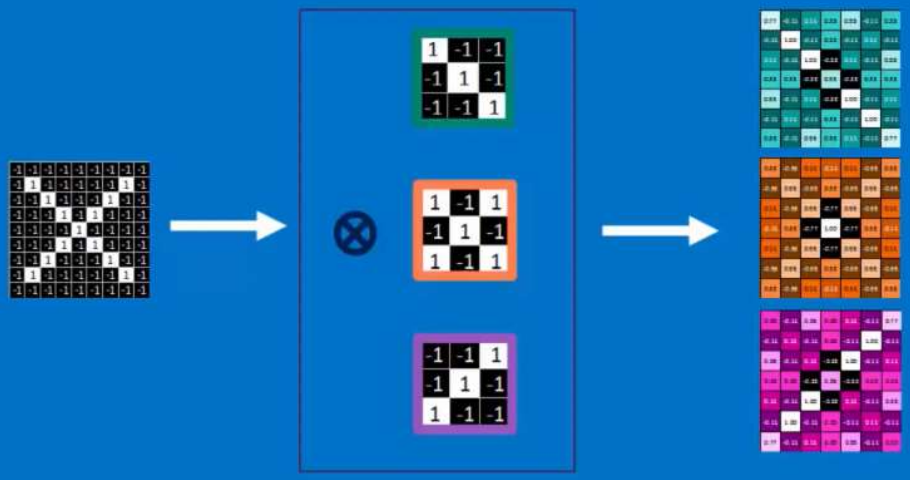


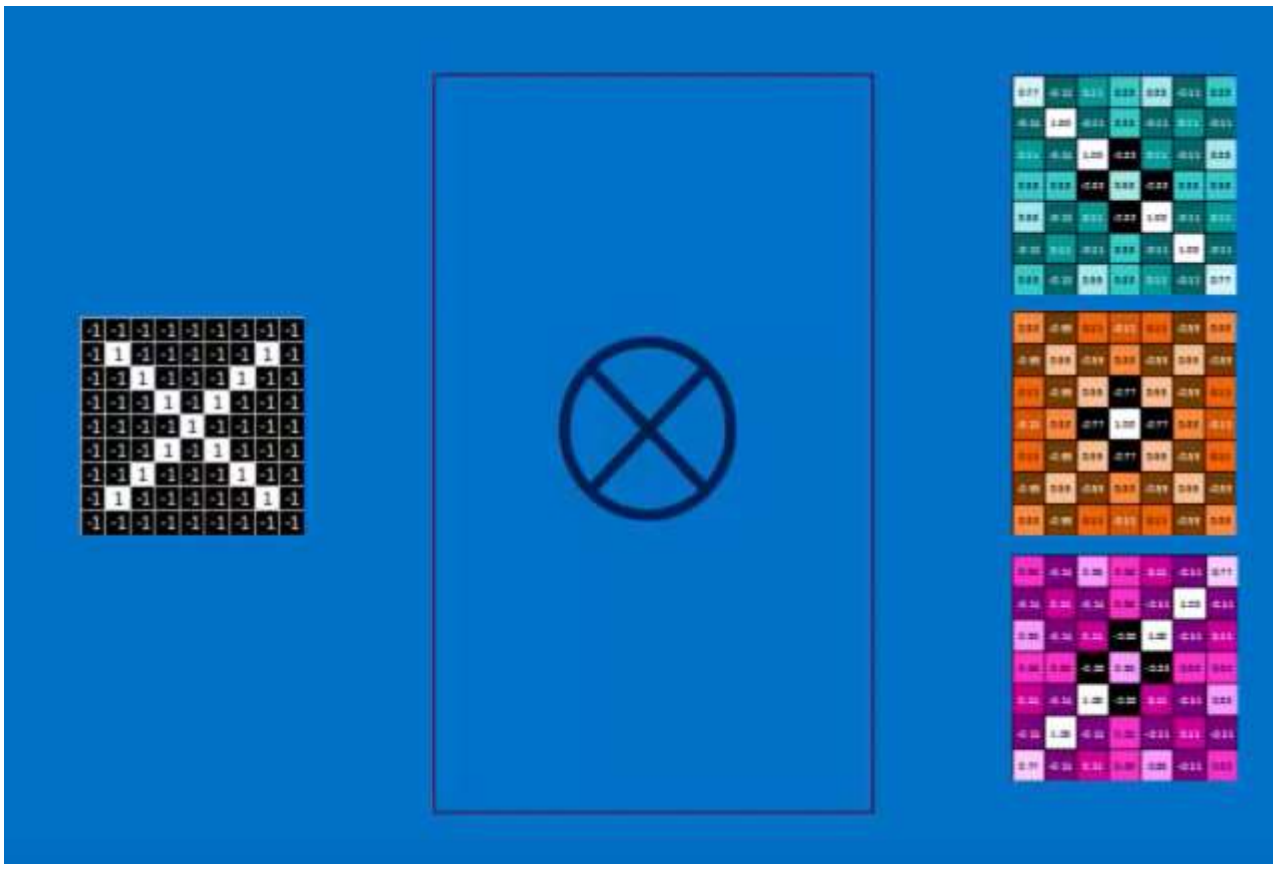

Convolution: Trying every possible match
1 -1 -1
-1 1 -1
-1 -1 1
0.77 -0.11 0.11 0.33 0.55 -0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77



Convolution layer

One image becomes a stack of filtered images

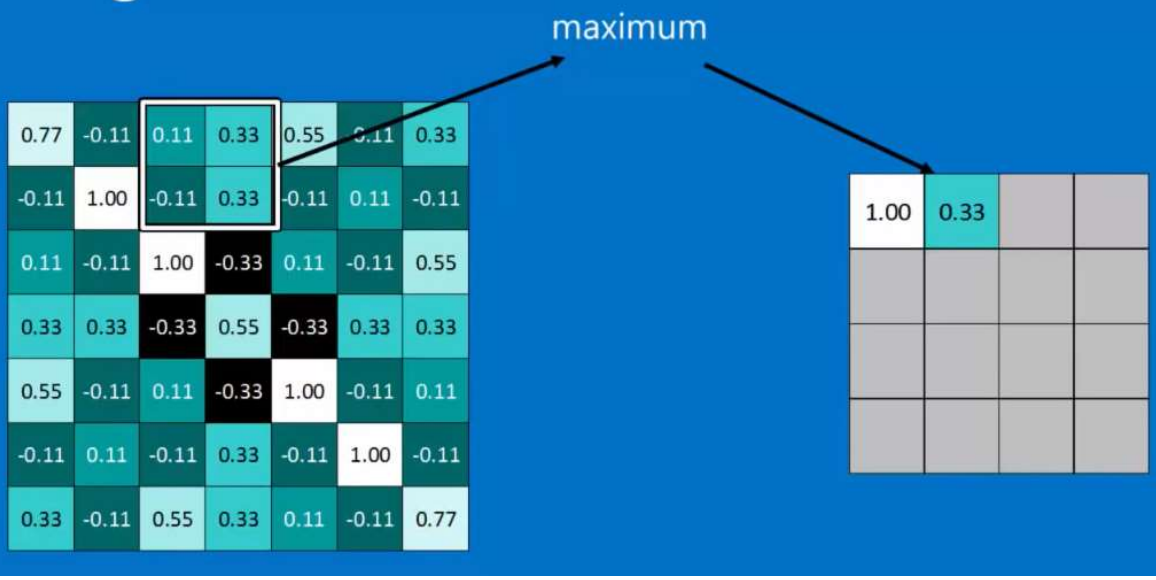




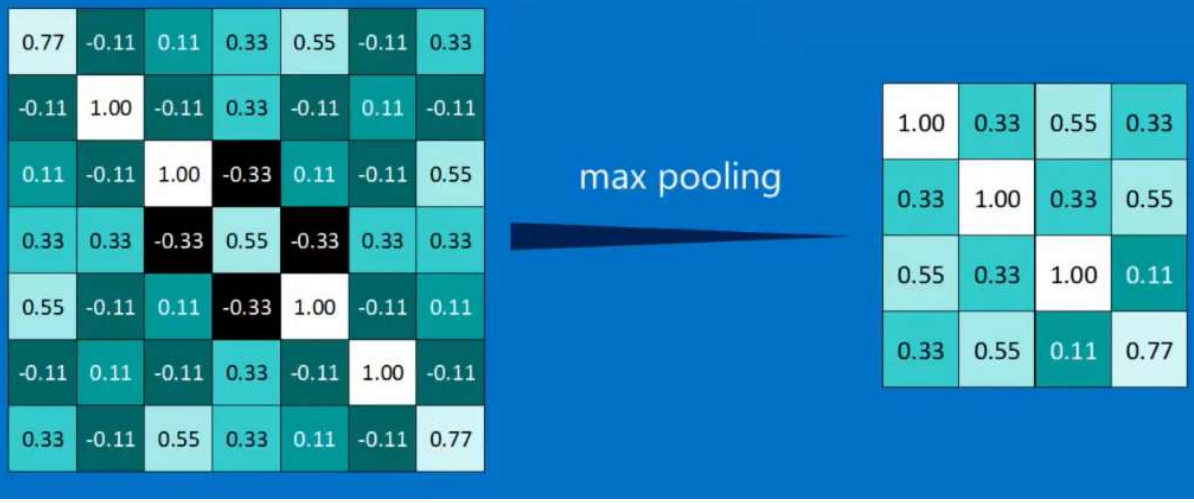
Pooling: Shrinking the image stack

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

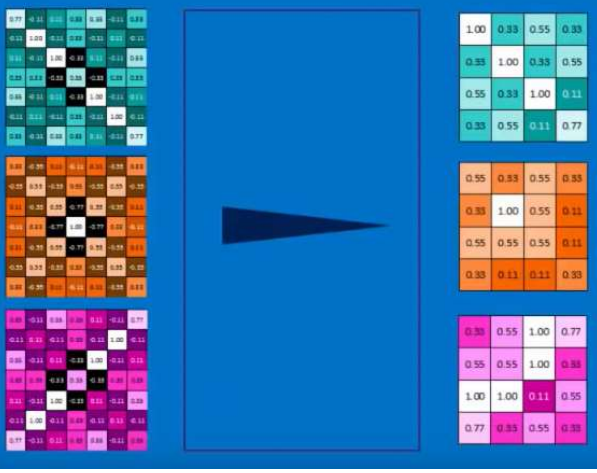


Pooling



Pooling layer

A stack of images becomes a stack of smaller images.



Normalization

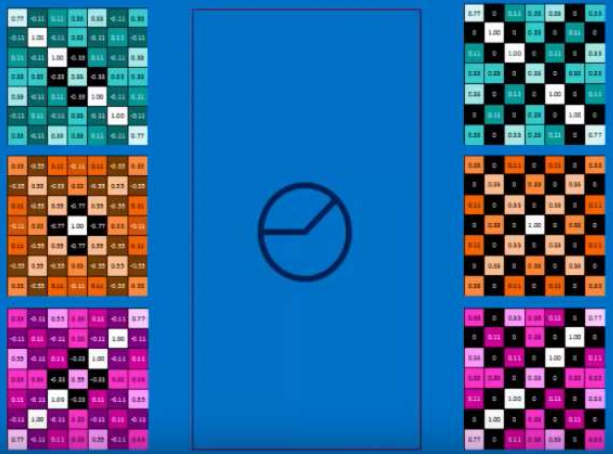
Keep the math from breaking by tweaking each of the values just a bit.

Change everything negative to zero.

Rectified Linear Units (ReLUs)

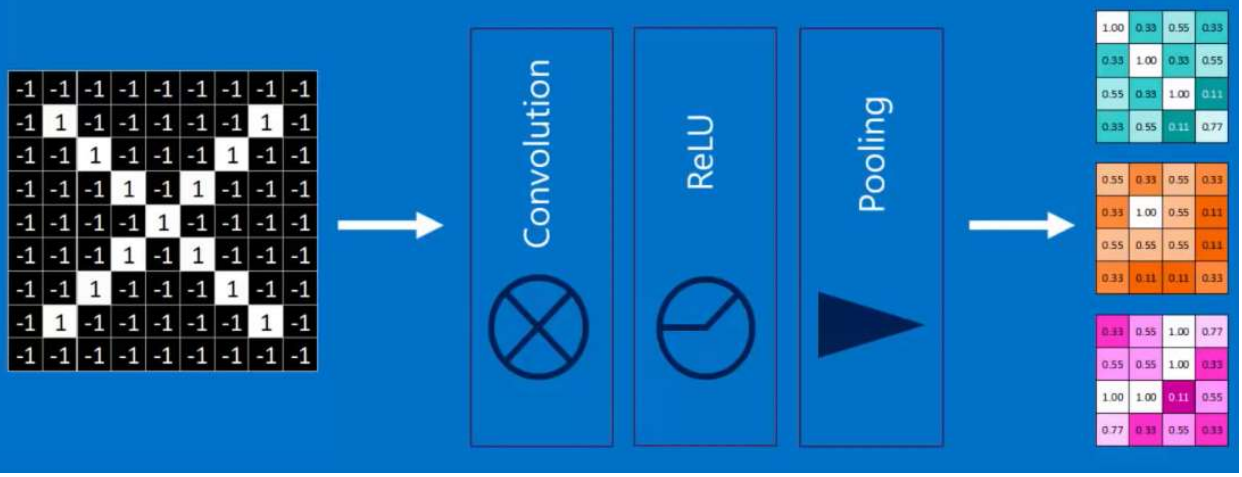
ReLU layer

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



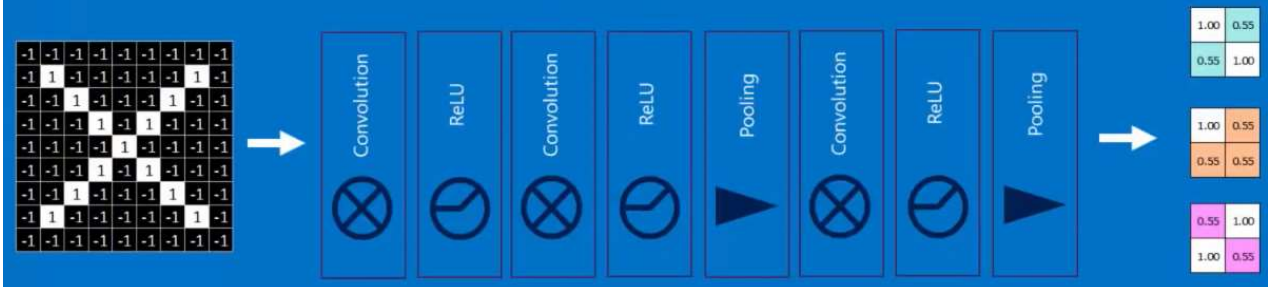
Layers get stacked

The output of one becomes the input of the next.



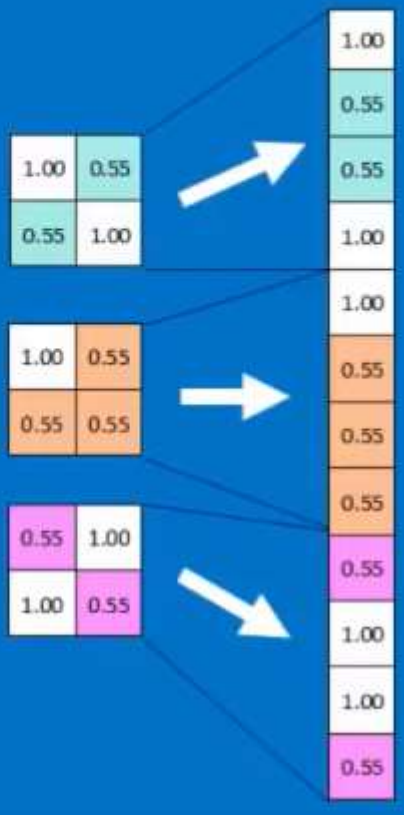
Deep stacking

Layers can be repeated several (or many) times.



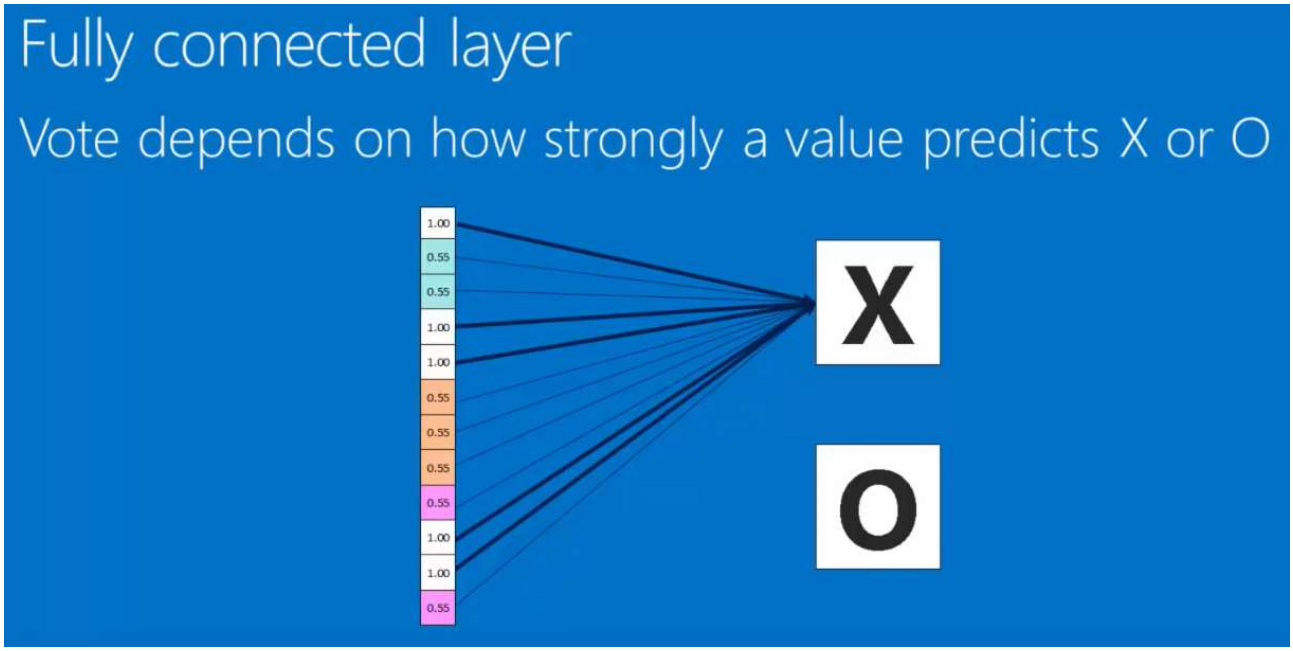
Fully connected layer

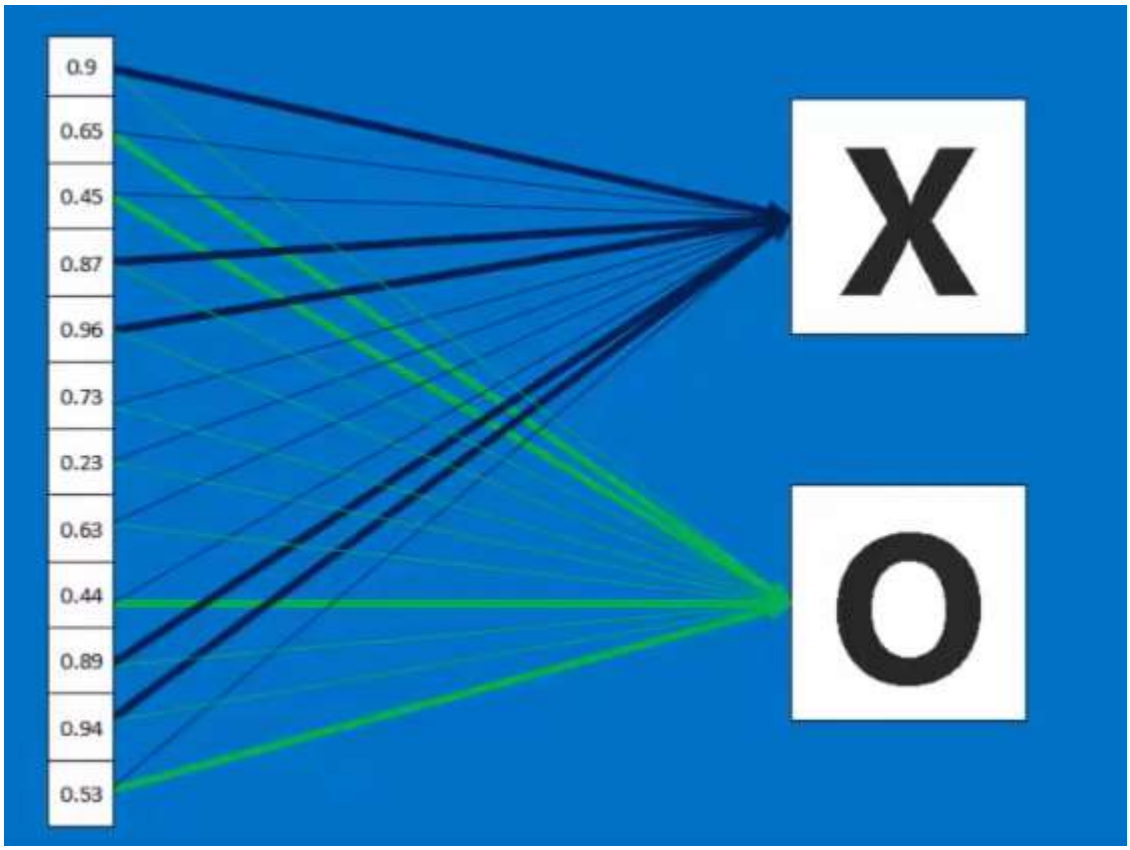
Every value gets a vote

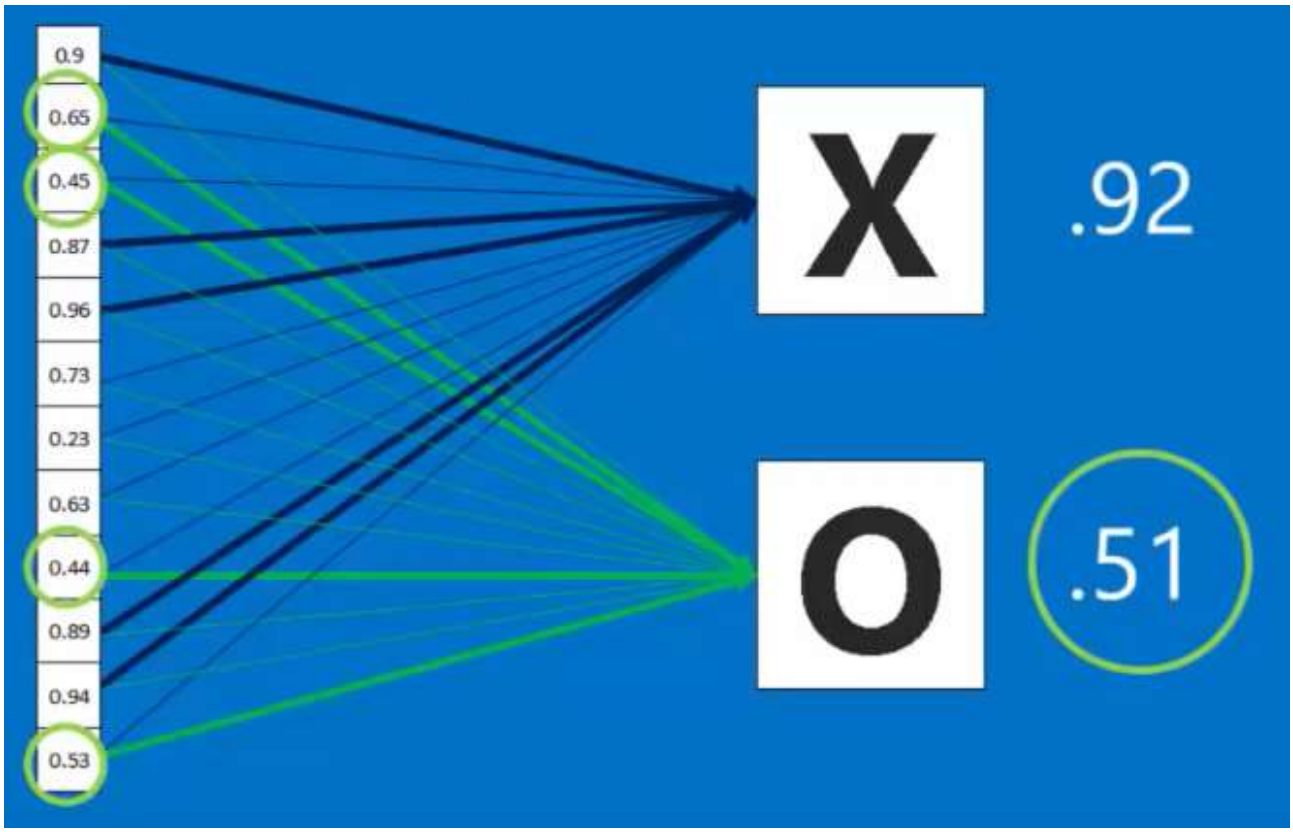


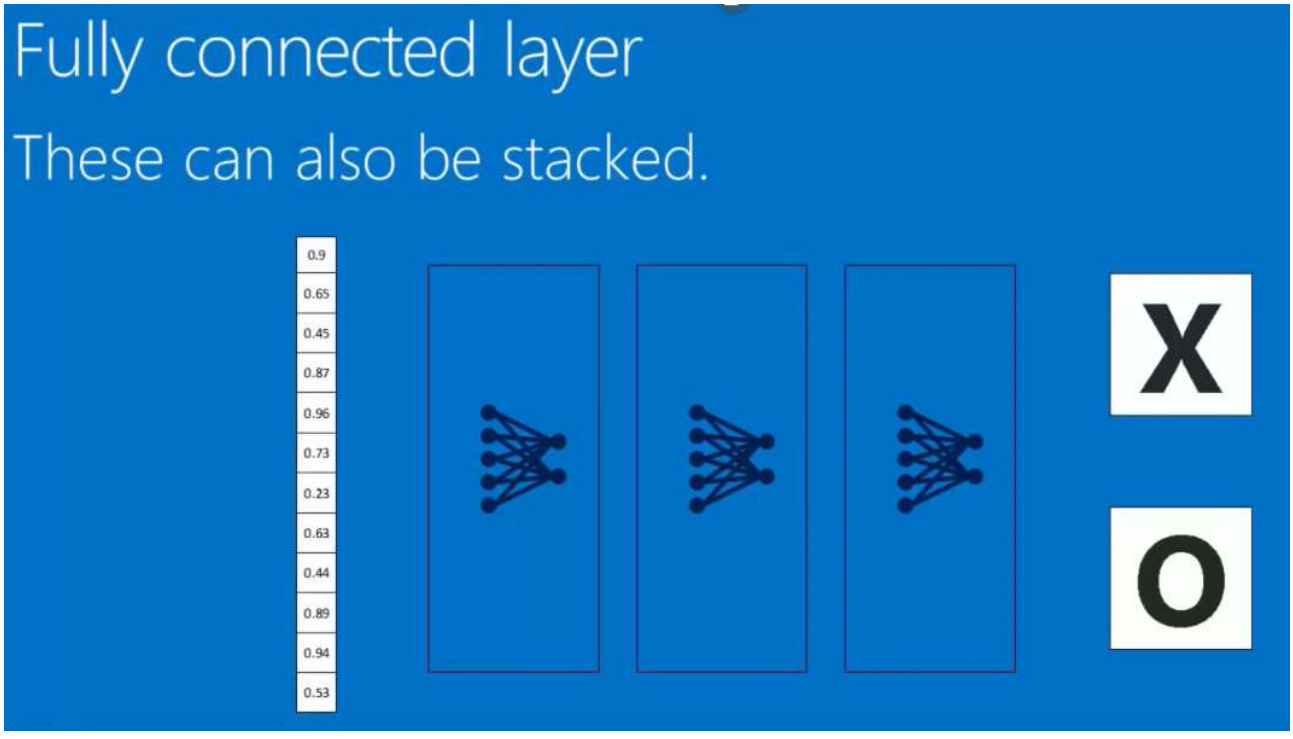
Fully connected layer

Vote depends on how strongly a value predicts X or O



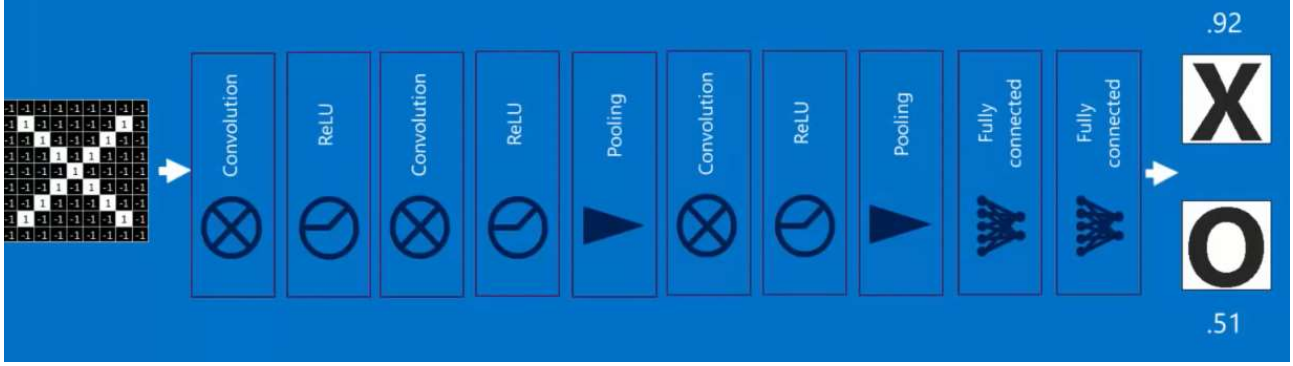



0.9
0.65
0.45
0.87
0.96
0.73
0.23
0.63
0.44
0.89
0.94
0.53
X
.92
O
.51


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

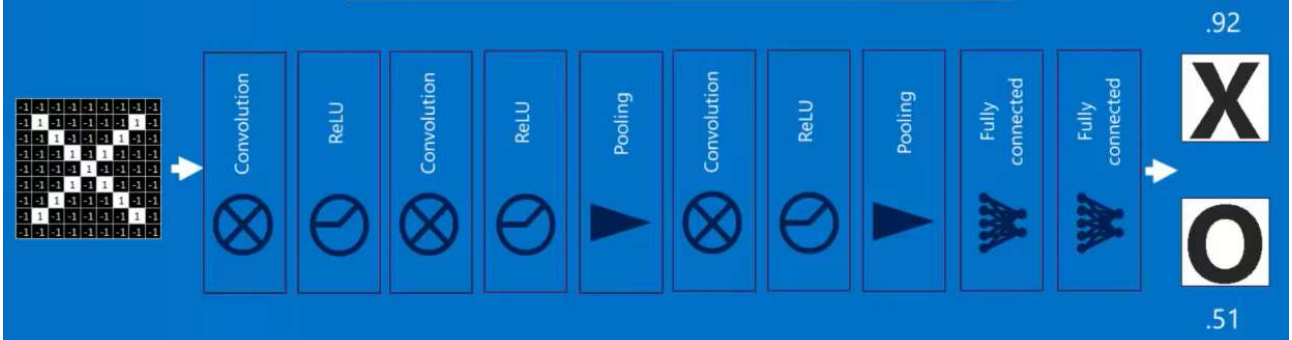
A: Backpropagation

Backprop

Error = right answer – actual answer

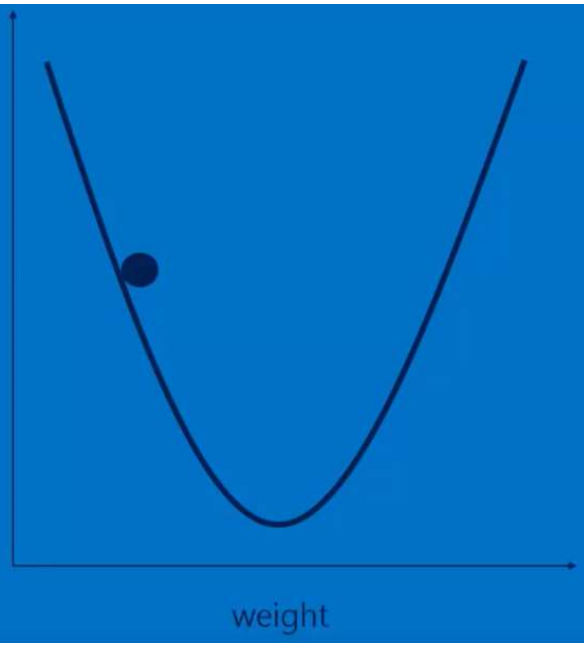
Backprop

	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

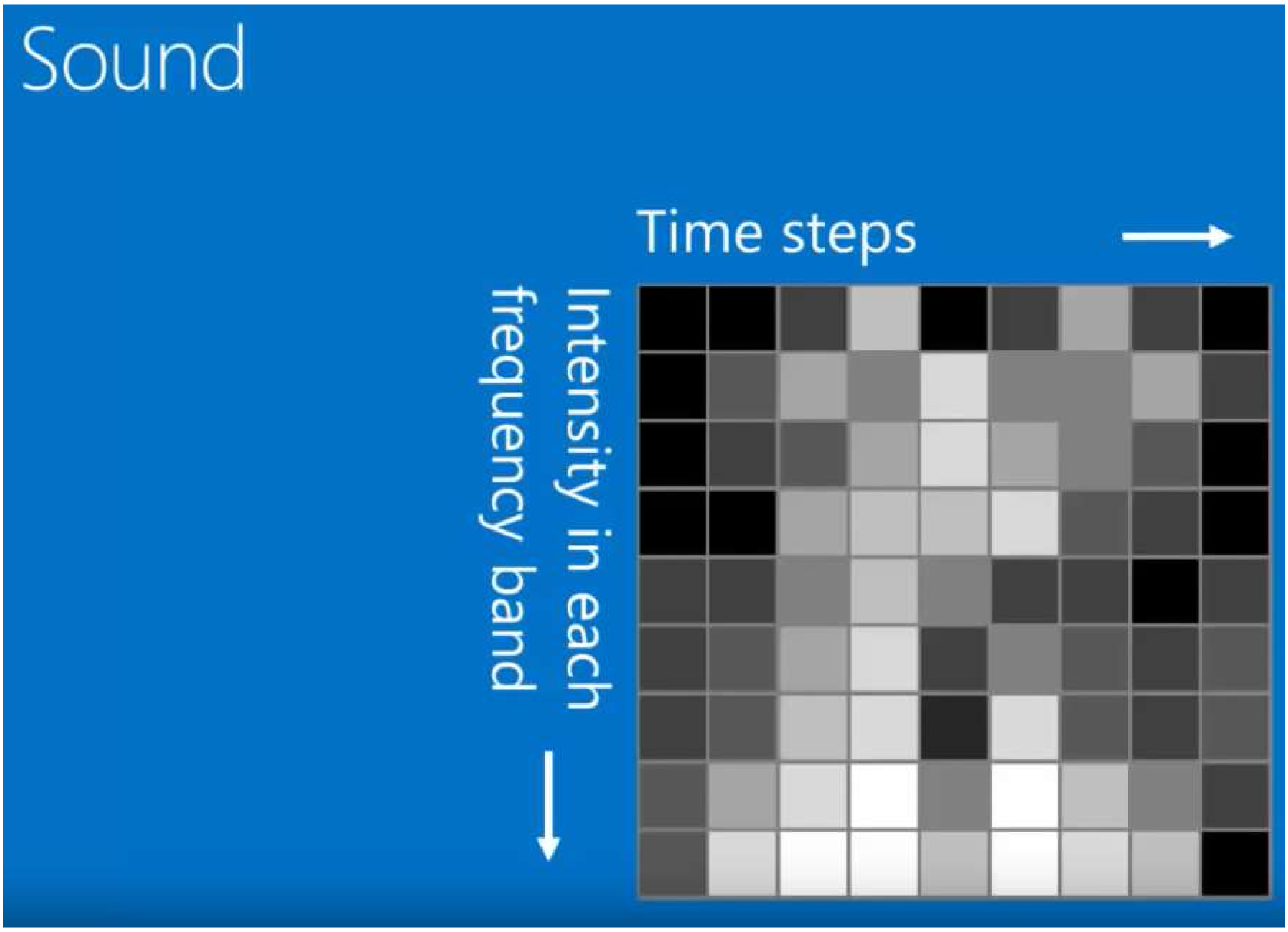
Fully Connected

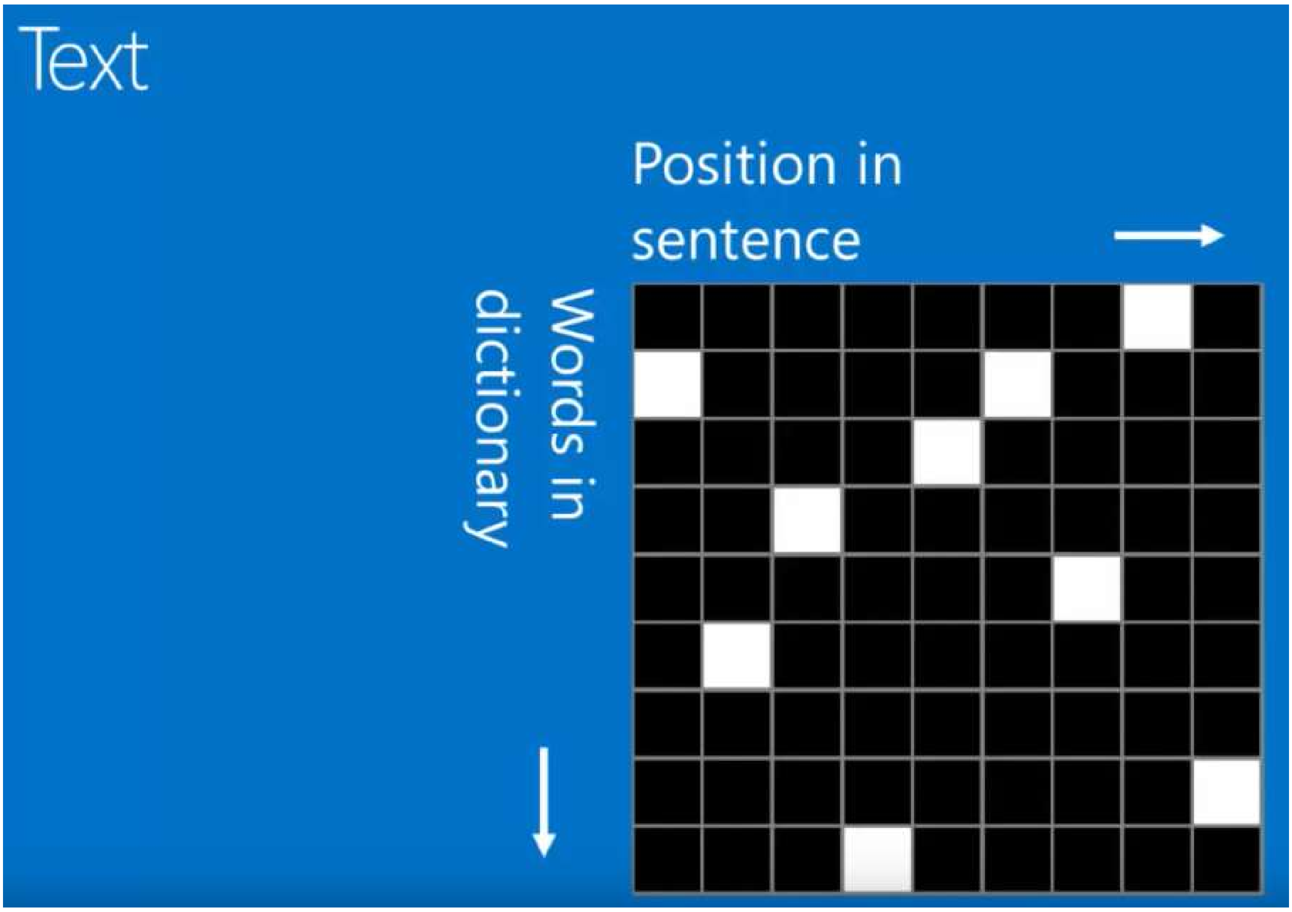
- Number of neurons

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