

# Convolutional Neural Networks

## Lecture-5

# Machine Learning

“A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.”

**Tom Mitchell, 1997**

# Machine Learning

- Want my computer to learn where the bananas are (T) in a bowl of fruits (E); Performance measure (P): color



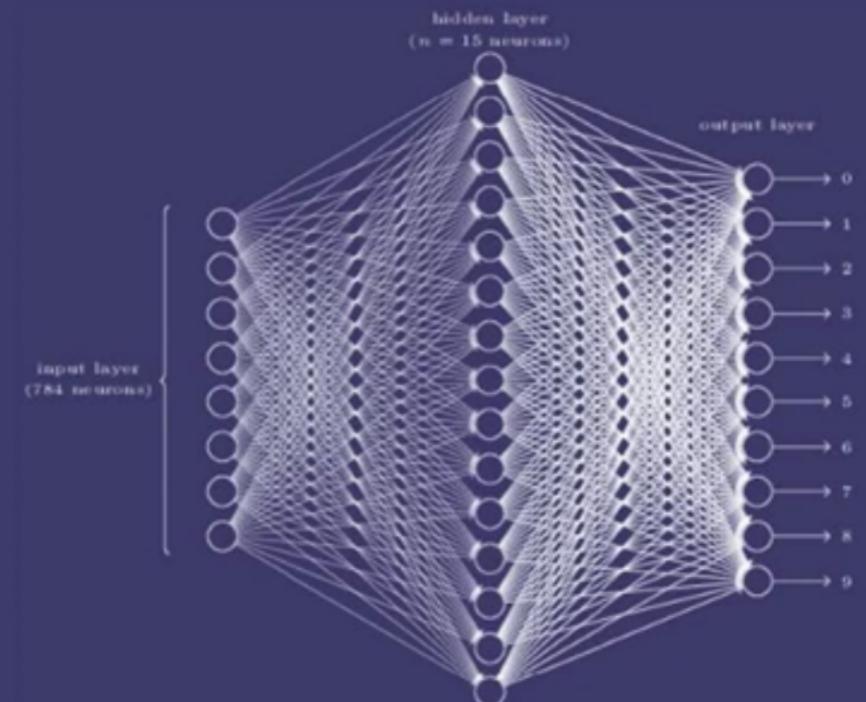
# Neural Network

## Training phase

**Take pairs of input data and desired output that has been collected beforehand**

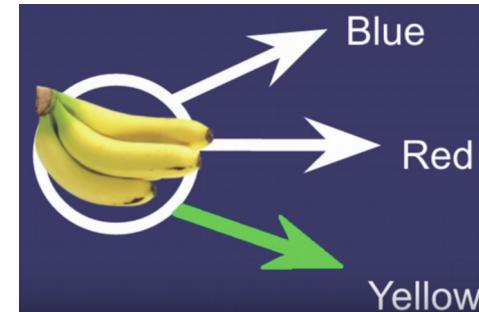
- ✓ Input layer: An image
- ✓ Hidden layer: Characteristics of the image (provides confidence)
- ✓ Output layer: Tags for that image

## Adjacent Neural Networks

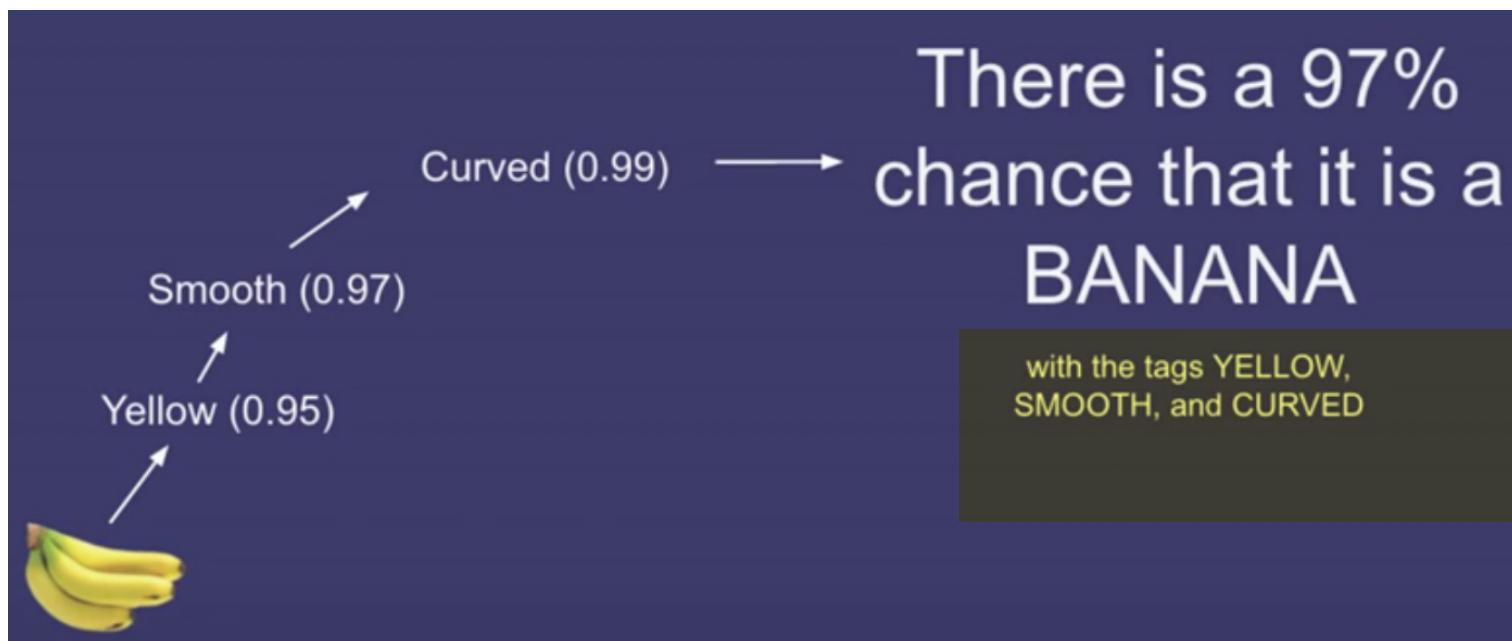


# Neural Network

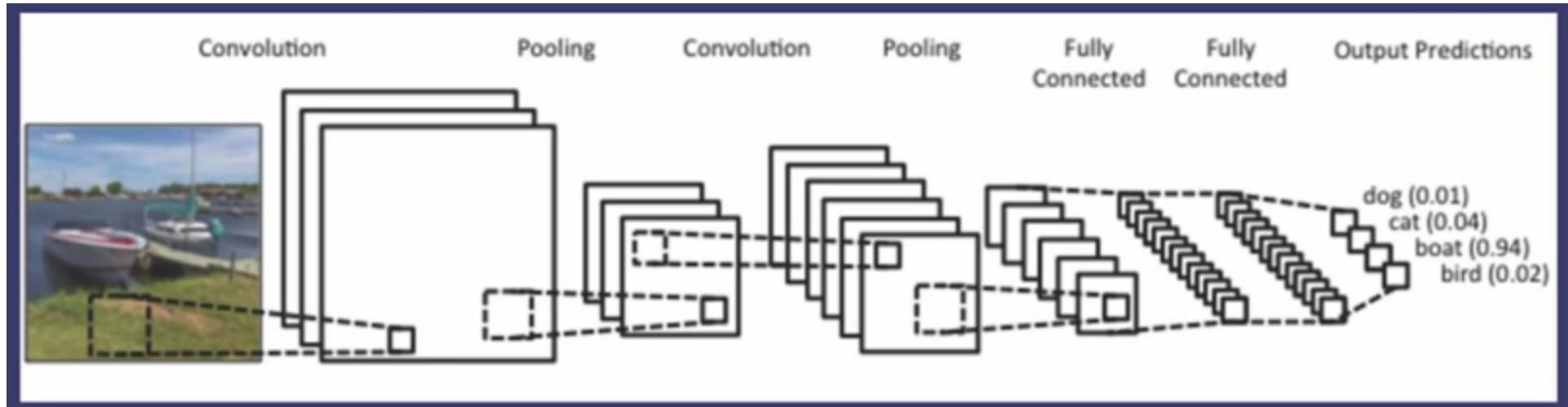
- ✓ Hidden layer: Color characteristics of the image (provides confidence)



- ✓ Cumulative confidence



# Convolution Neural Networks



## Definitions:

- ✓ **Local Receptive Fields:** a window on the input of pixels
- ✓ **Feature Map:** mapping from input layer to hidden layer
- ✓ **Shared Weights:** positive or negatives on a feature map
- ✓ **Pooling:** simplifying information from the feature map

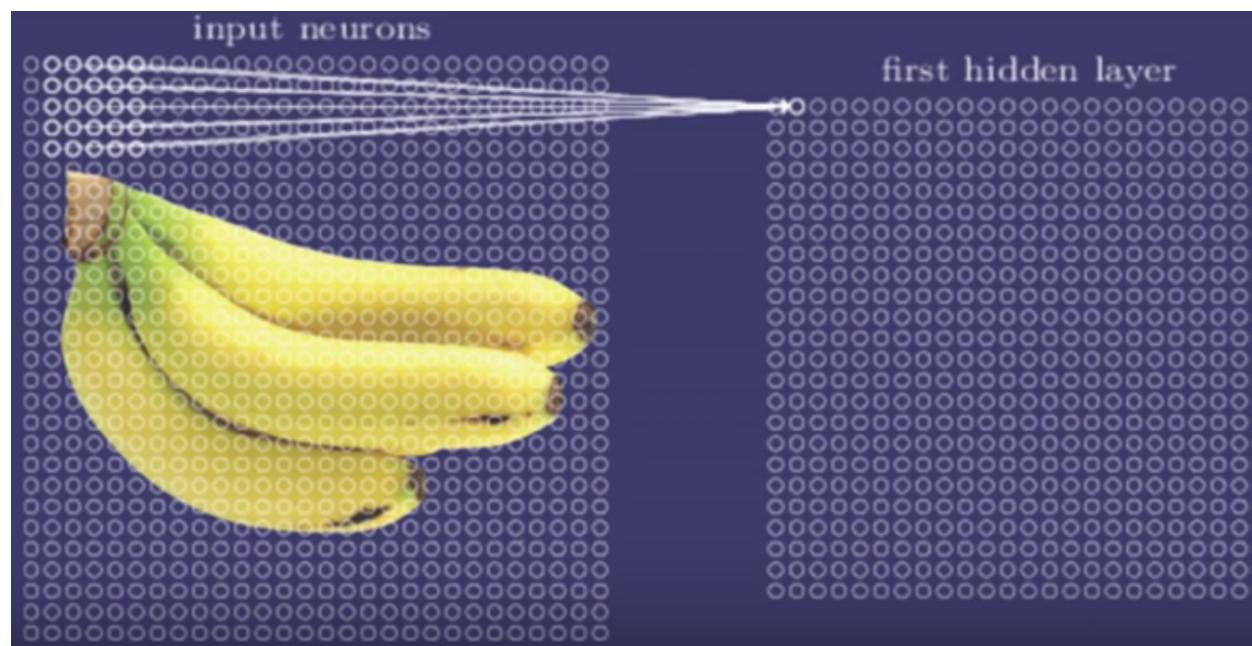
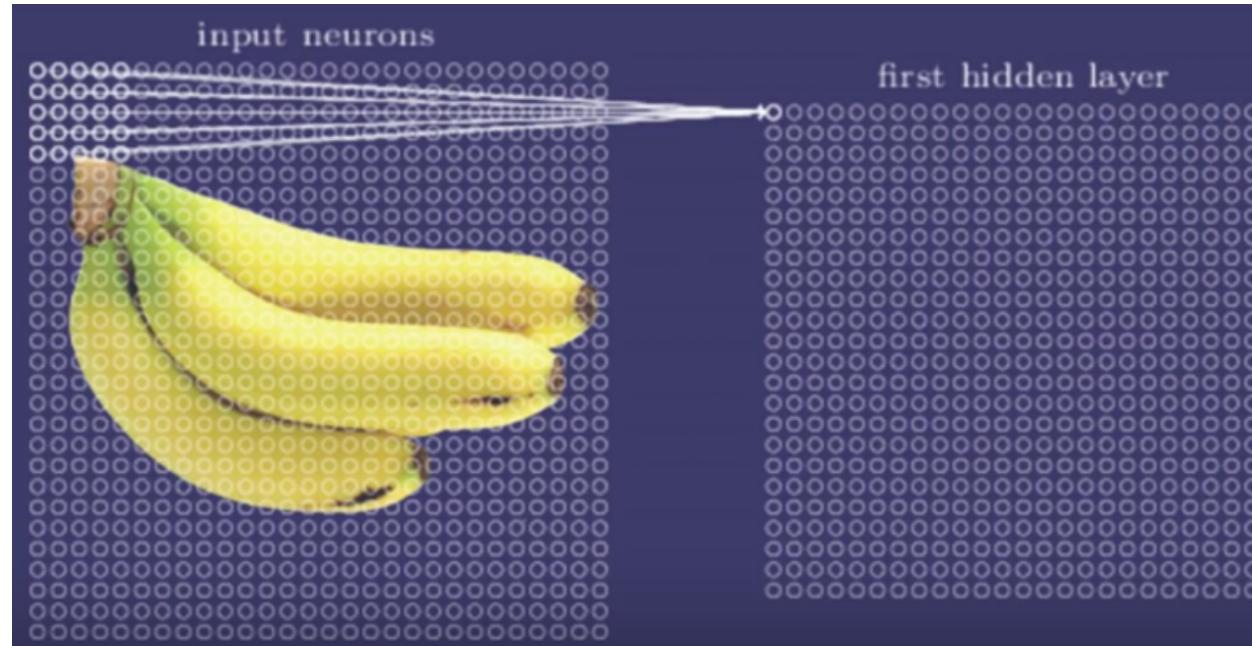
# Convolution Neural Networks



- ✓ **Local Receptive Fields:** a window on the input of pixels

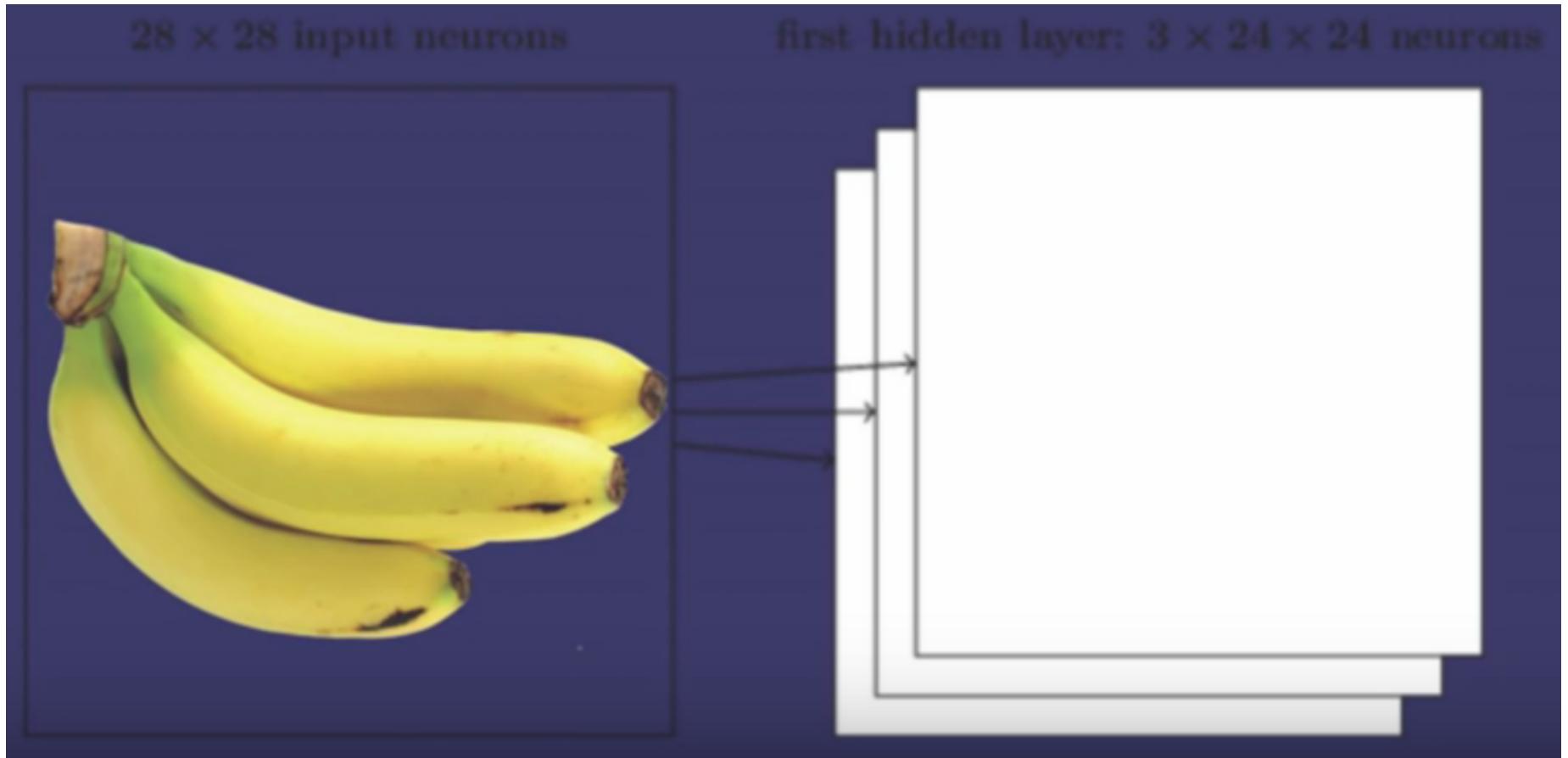


# Convolution Neural Networks



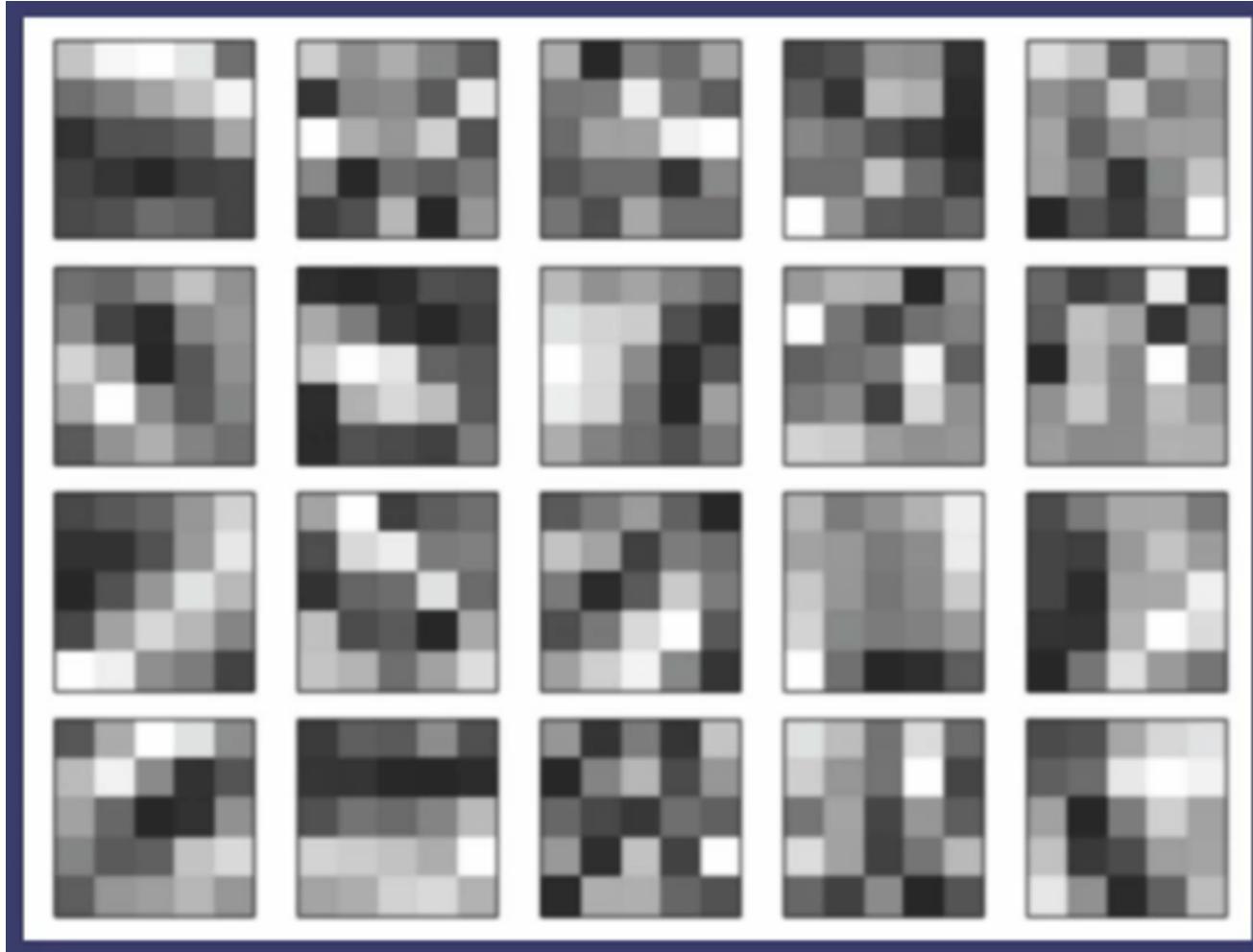
- ✓ **Feature Map:** (definition of one characteristic) mapping from input layer to hidden layer

# Convolution Neural Networks



- ✓ **3 feature map (each feature map defines one characteristic) comprising 3 hidden layers, performing the convolution.**

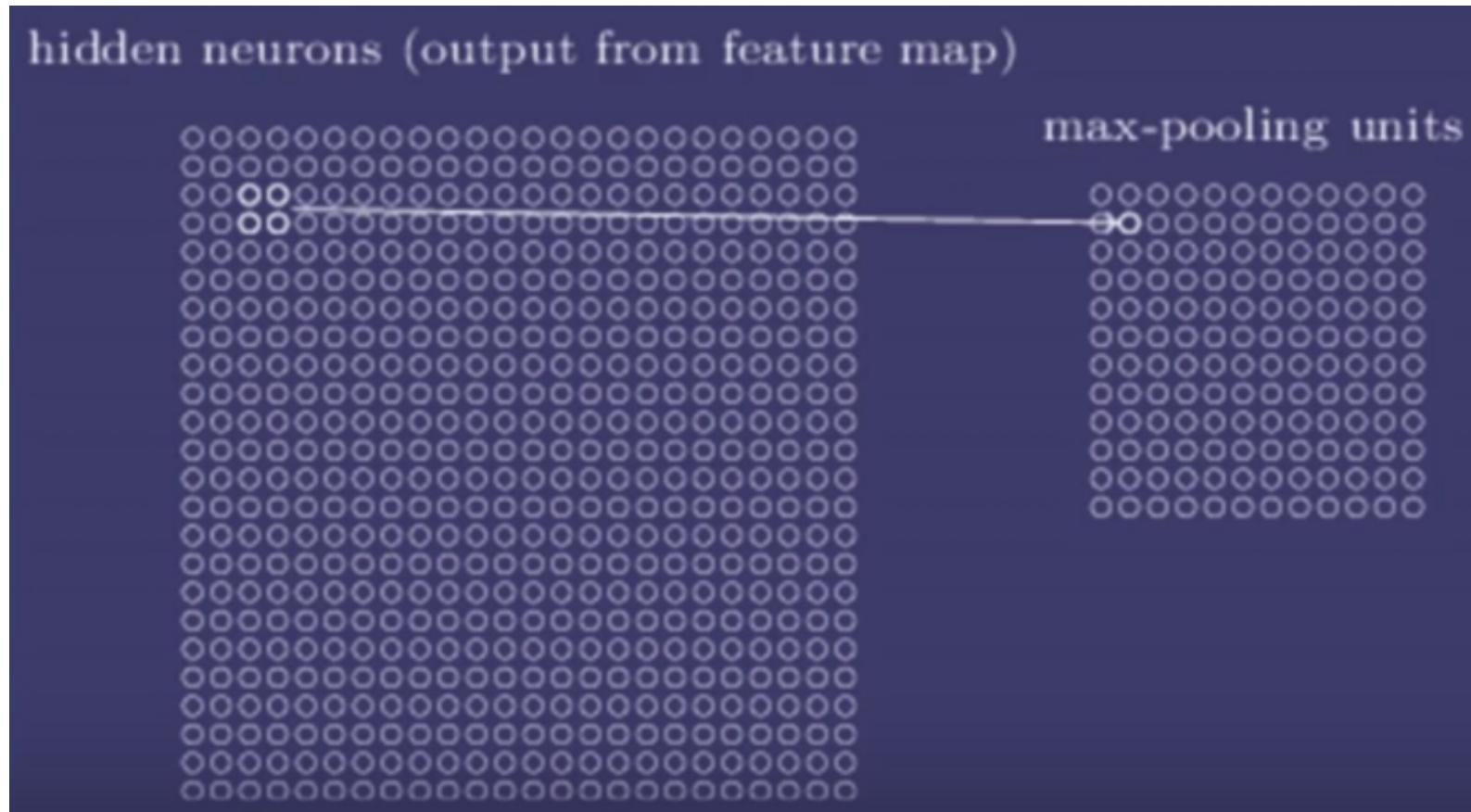
# Convolution Neural Networks



## Actual look of convolution:

- ✓ A convolution: a whole hidden layer
- ✓ Each box is a feature map; each square in each of the boxes is a neuron
- ✓ Darker represents more positive; lighter represents more negative

# Convolution Neural Networks



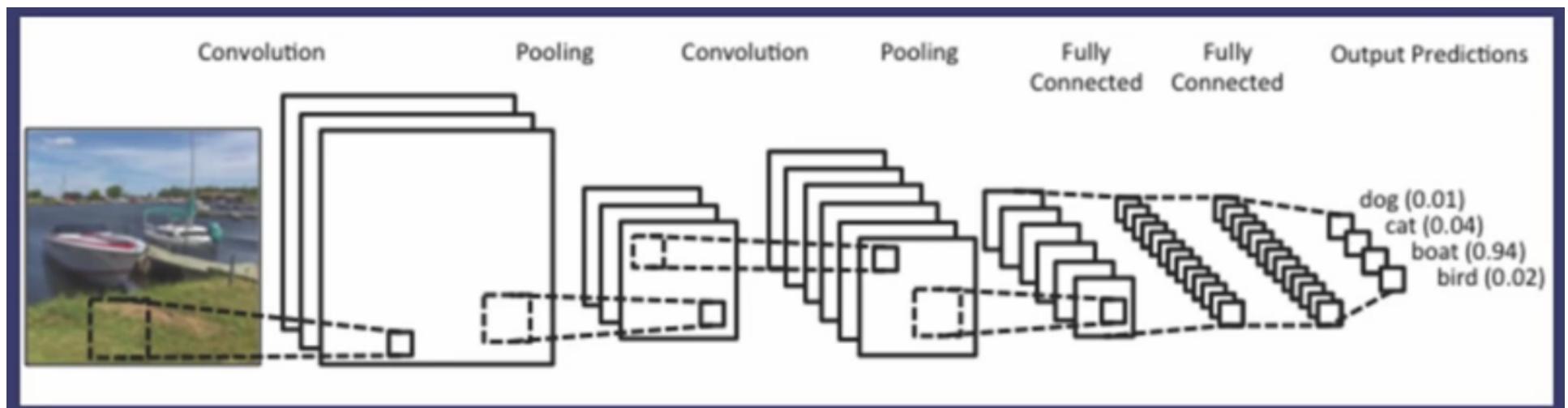
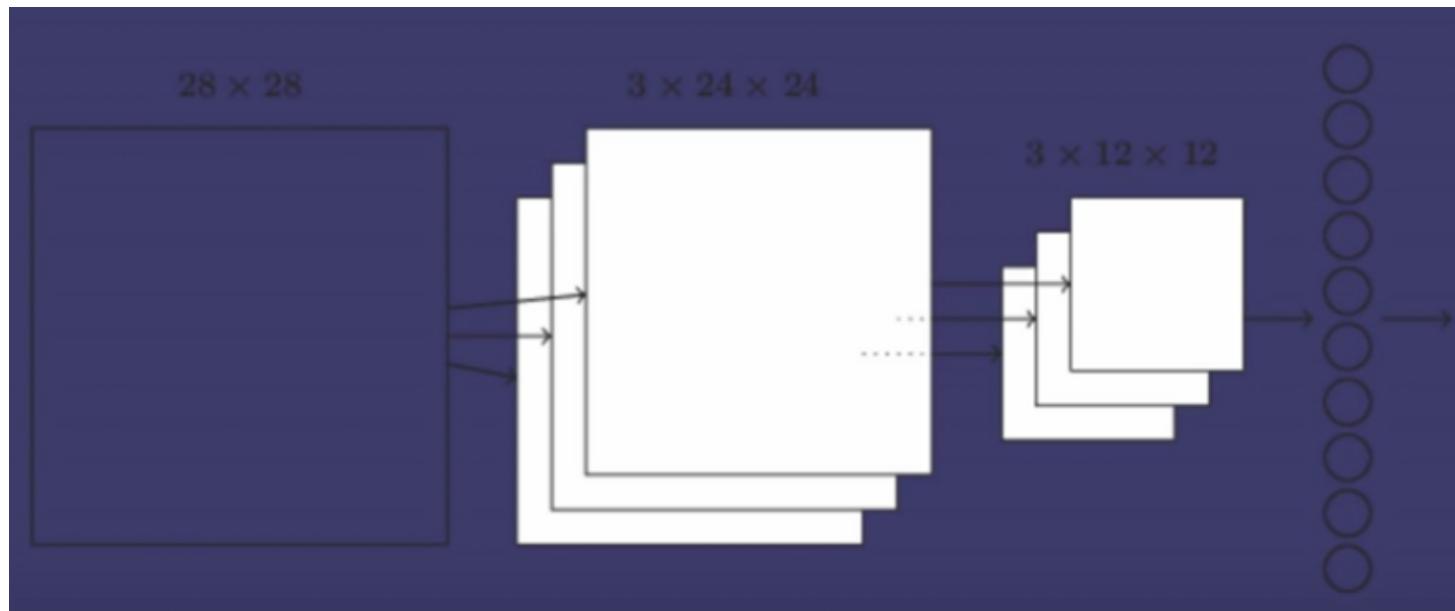
## Pooling:

- ✓ simplifying information from the feature map
- ✓ Condensed all of the hidden neurons

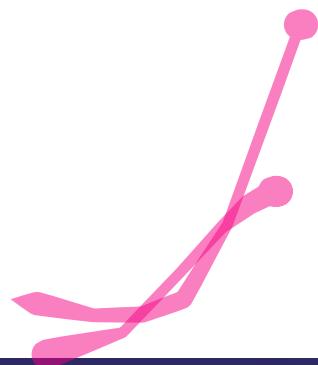
# Convolution Neural Networks

## Overall:

- ✓ Focusing on characteristics in different locations;
- ✓ working on smaller regions in separate thread, which is fast



# Convolution Neural Networks



**Summary:**

## Convolution vs. Adjacent

- Convolution: Fast to train, multiple items found
- Adjacent: No recognition of spacial structure, great for finding a single item
- Both: Create multilayer neural network

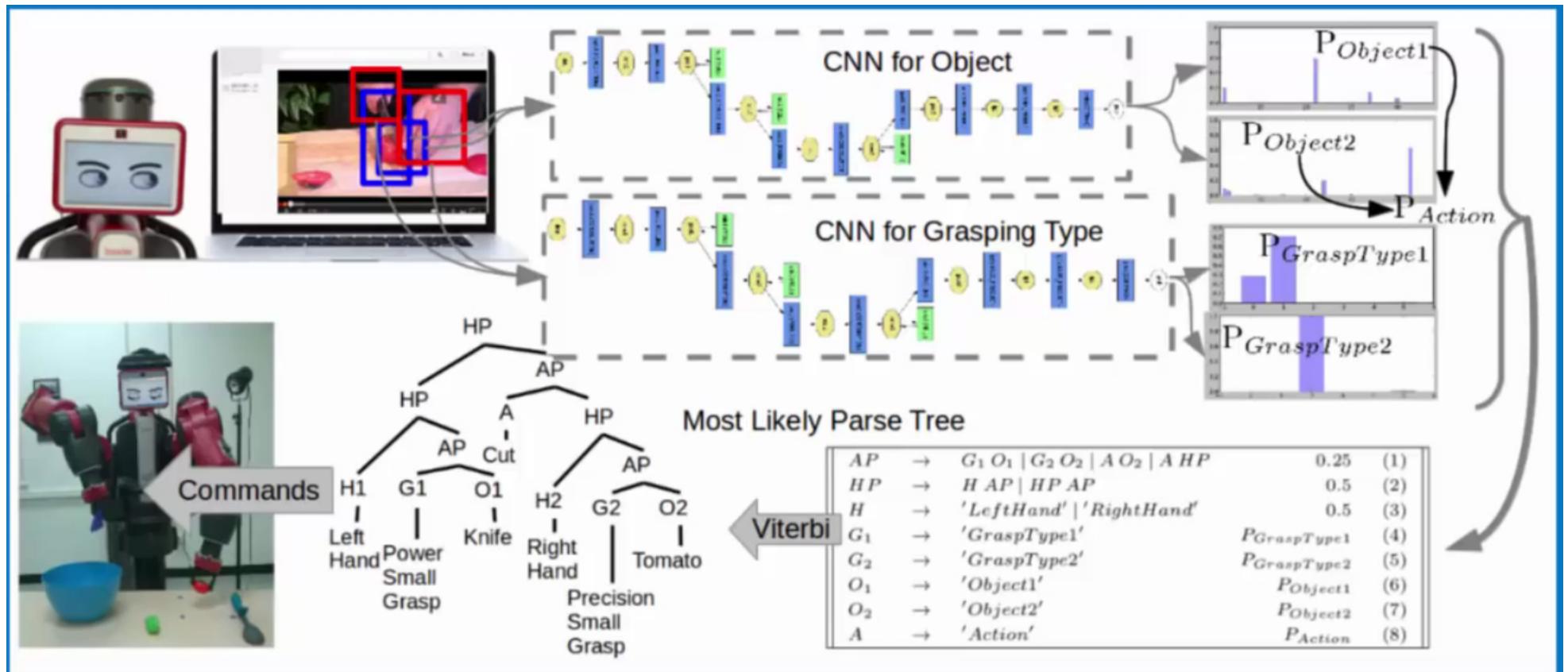


# Convolution Neural Networks

**Extra ...**

# Convolution Neural Networks

- ✓ Robot learns to cook by watching youtube videos



# Convolution Neural Networks

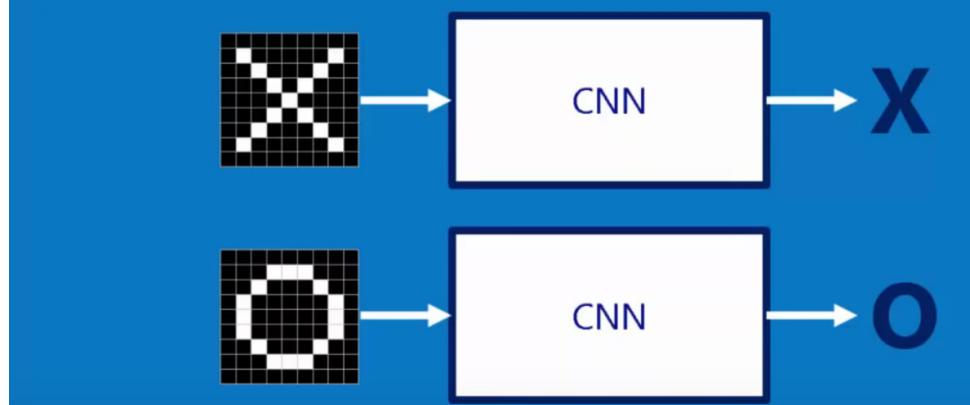
A toy ConvNet: X's and O's

Says whether a picture is of an X or an O

A two-dimensional  
array of pixels

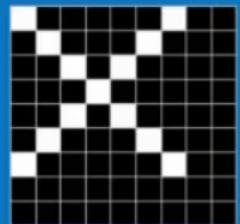


For example

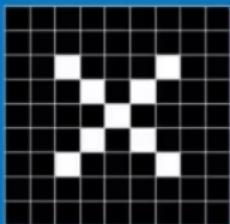


# Convolution Neural Networks

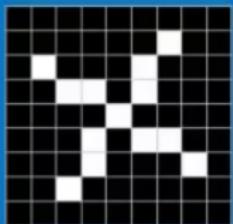
Trickier cases



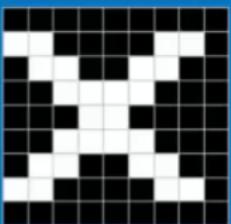
translation



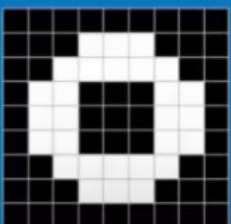
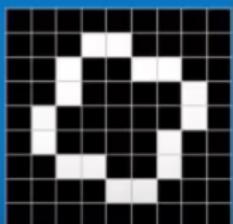
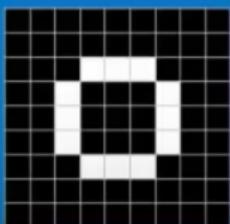
scaling



rotation

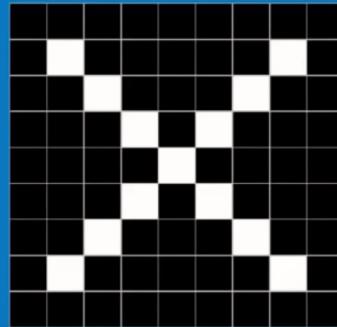


weight

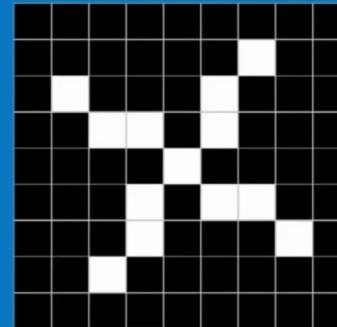


# Convolution Neural Networks

Deciding is hard



?



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

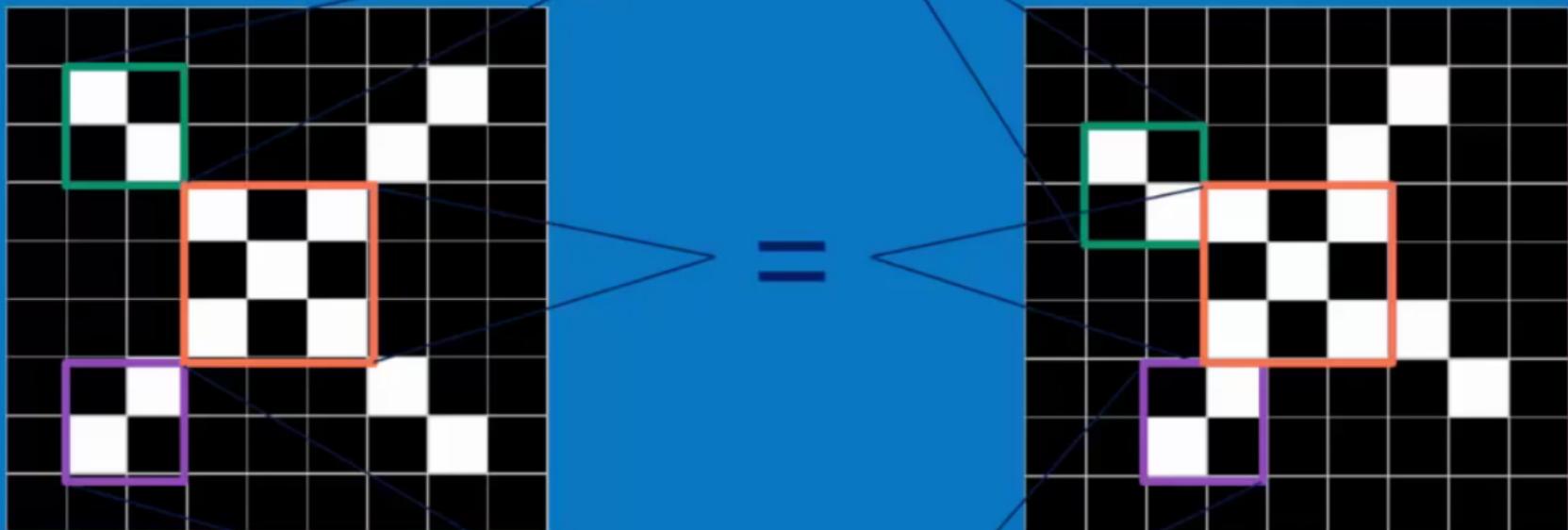
?

-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

-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	-1	-1	-1
-1	-1	-1	1	-1	-1	1	X	-1
-1	-1	X	X	-1	-1	X	X	-1
-1	X	X	-1	-1	-1	-1	X	-1

# Convolution Neural Networks

ConvNets match pieces of the image



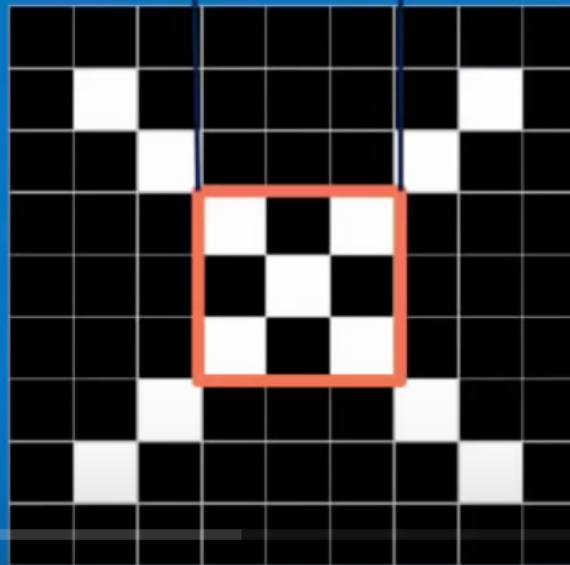
# Convolution Neural Networks

Features match pieces of the image

$$\begin{matrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{matrix}$$

$$\begin{matrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{matrix}$$

$$\begin{matrix} -1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & -1 \end{matrix}$$



# Convolution Neural Networks

Filtering: The math behind the match

$$\begin{matrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{matrix}$$

$$\begin{matrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 \end{matrix}$$

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.

# Convolution Neural Networks

Filtering: The math behind the match

$$\begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix}$$

$$-1 \times -1 = 1$$

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \end{bmatrix}$$



5:44 / 26:13



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.

# Convolution Neural Networks

Filtering: The math behind the match

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

1	1	1
1	1	1
1	1	1

$$\frac{1 + 1 + 1 + 1 + 1 + 1 + 1 + 1}{9} = 1$$

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

1
---



6:05 / 26:13



- Convoluton: feature map

# Convolution Neural Networks

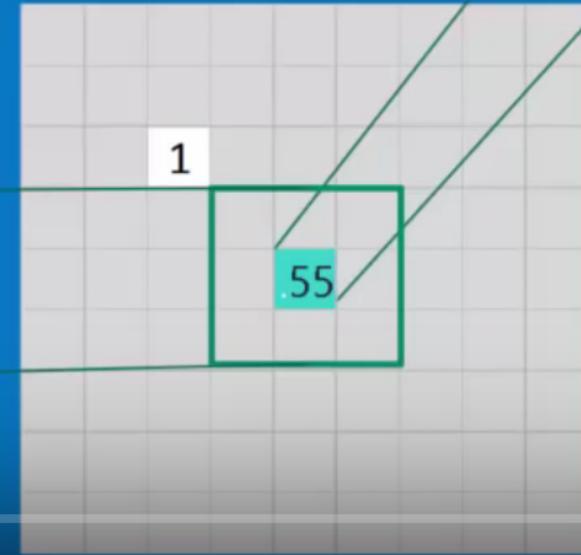
## Filtering: The math behind the match

$$\begin{matrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{matrix}$$

$$\begin{matrix} 1 & 1 & -1 \\ 1 & 1 & 1 \\ -1 & 1 & 1 \end{matrix}$$

$$\frac{1 + 1 - 1 + 1 + 1 + 1 - 1 + 1 + 1}{9} = .55$$

$$\begin{matrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{matrix}$$



- Convoluton: feature map

# Convolution Neural Networks

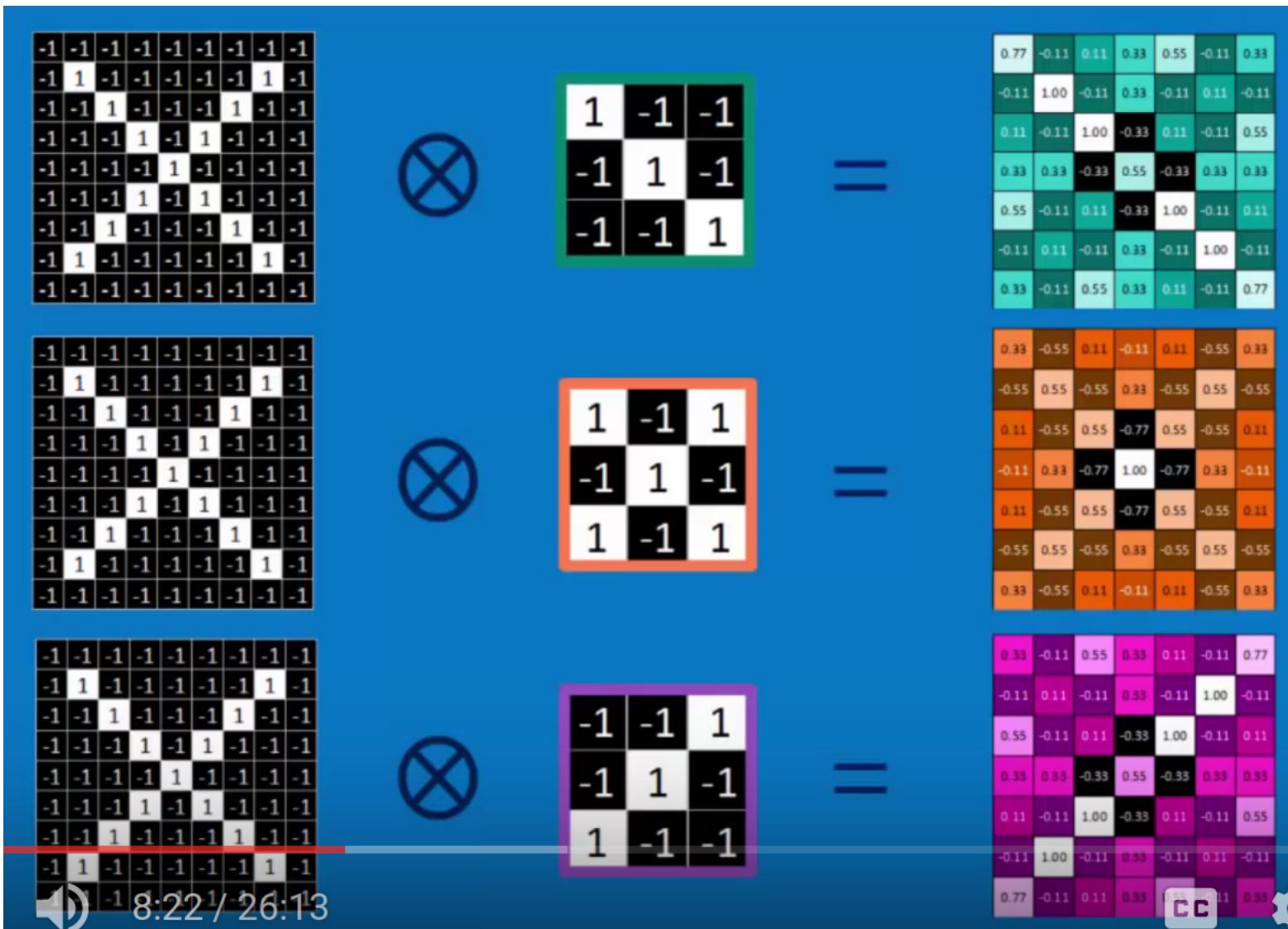
Convolution: Trying every possible match

$$\begin{matrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & \boxed{1} & -1 & -1 & -1 & -1 & -1 & \boxed{1} & -1 \\ -1 & -1 & \boxed{1} & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & \boxed{1} & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & \boxed{1} & -1 & -1 & -1 \\ -1 & -1 & -1 & \boxed{1} & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & \boxed{1} & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & \boxed{1} & -1 & -1 & -1 & -1 & -1 & \boxed{1} & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{matrix} \otimes \begin{matrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{matrix} = \begin{matrix} 0.77 & -0.11 & 0.11 & 0.33 & 0.55 & -0.11 & 0.33 \\ -0.11 & 1.00 & -0.11 & 0.33 & -0.11 & 0.11 & -0.11 \\ 0.11 & -0.11 & 1.00 & -0.33 & 0.11 & -0.11 & 0.55 \\ 0.33 & 0.33 & -0.33 & 0.55 & -0.33 & 0.33 & 0.33 \\ 0.55 & -0.11 & 0.11 & 1.00 & -0.11 & 0.11 & 0.11 \\ -0.11 & 0.11 & -0.11 & 0.33 & -0.11 & 1.00 & -0.11 \\ 0.33 & -0.11 & 0.55 & 0.33 & 0.11 & -0.11 & 0.77 \end{matrix}$$

In a nutshell  
ConvNets are great at finding patterns and using them to classify. 1.00

25:24

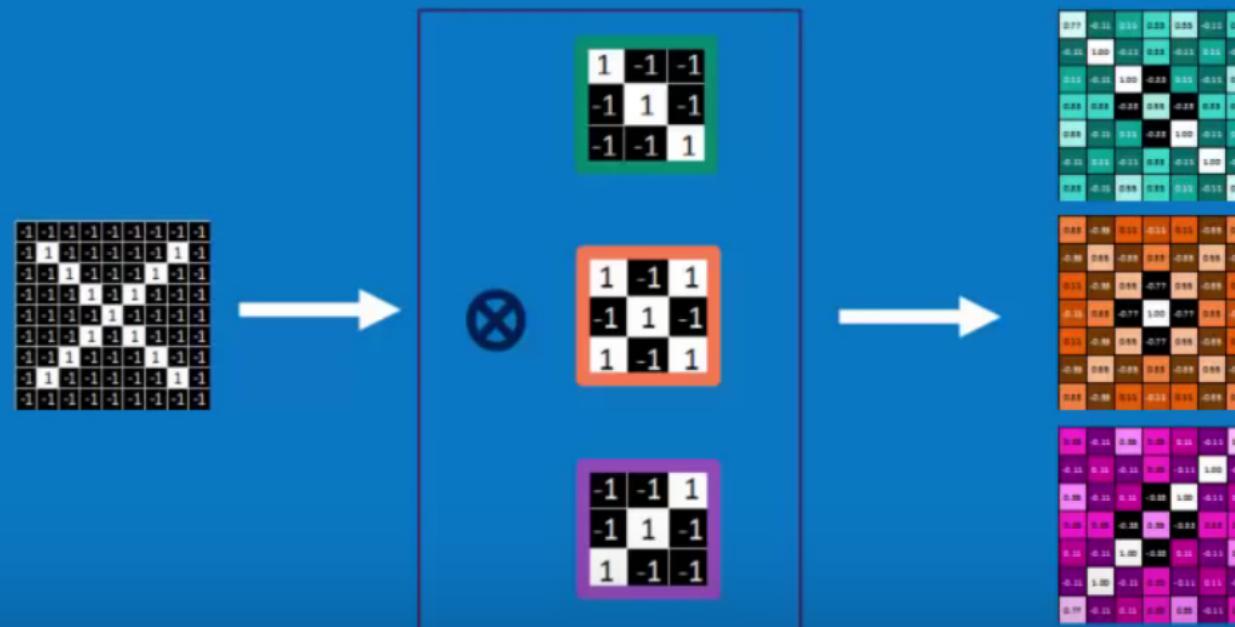
# Convolution Neural Networks



# Convolution Neural Networks

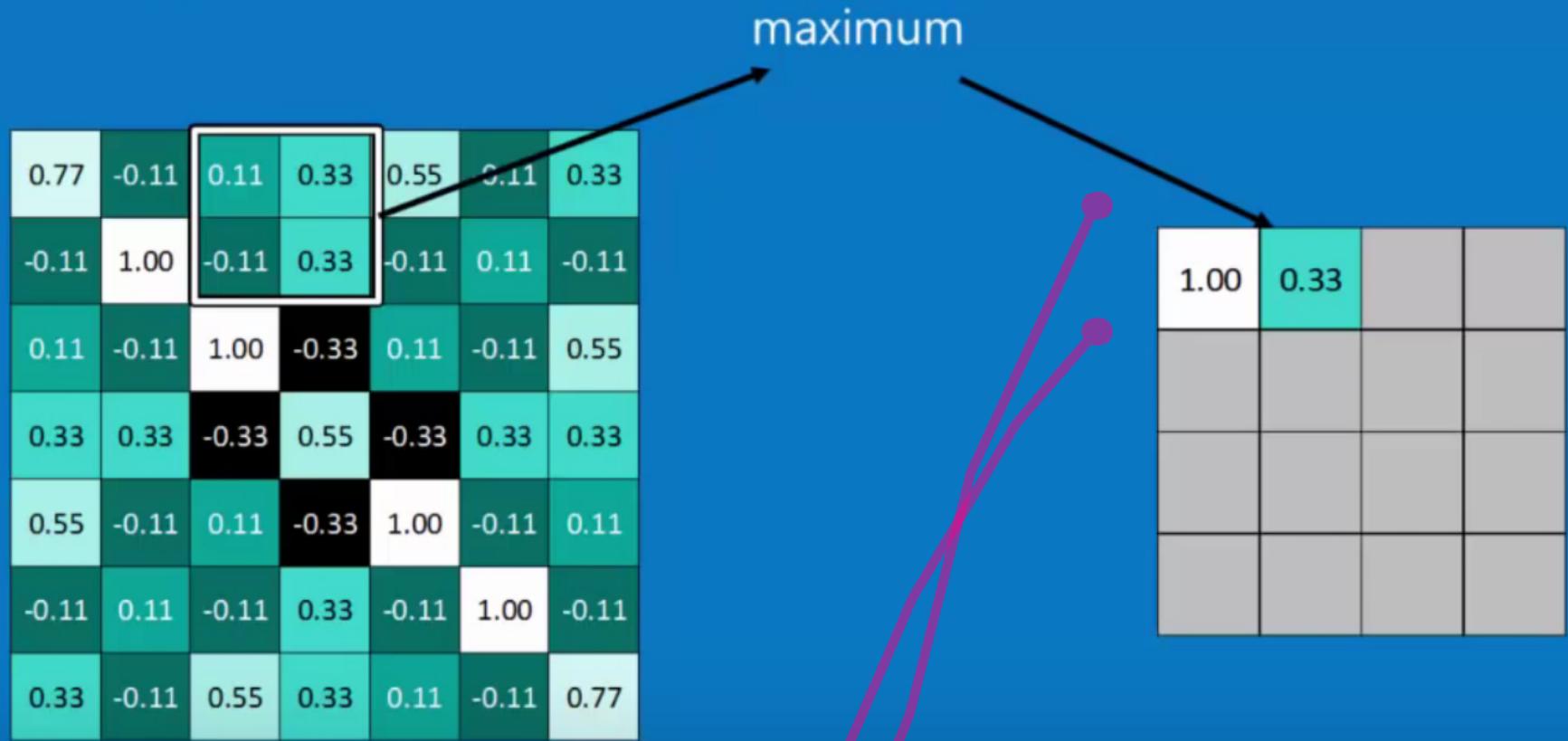
## Convolution layer

One image becomes a stack of filtered images



# Convolution Neural Networks

## Pooling



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.

# Convolution Neural Networks

## Pooling

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

max pooling

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

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.

# Convolution Neural Networks

## Pooling layer

A stack of images becomes a stack of smaller images.

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

0.22	-0.33	0.11	-0.11	0.11	-0.33	0.22
-0.33	0.33	-0.33	0.33	-0.33	0.33	-0.33
0.11	-0.22	0.22	-0.77	0.22	-0.22	0.11
0.22	0.33	-0.77	1.00	-0.77	0.33	-0.11
0.11	-0.33	0.33	-0.77	0.33	-0.55	0.11
-0.33	0.22	-0.33	0.22	-0.22	0.22	-0.33
0.33	-0.33	0.11	-0.11	0.11	-0.33	0.22

0.44	-0.11	0.33	0.33	0.11	-0.11	0.77
0.11	0.11	-0.11	0.33	0.11	1.00	-0.11
0.33	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.11	-0.33	0.33	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.11
0.11	1.00	-0.11	0.33	-0.11	0.11	0.11
0.77	-0.11	0.11	0.33	0.33	-0.11	0.11



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

0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

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

25:24

# Convolution Neural Networks

## 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

### Normalization

Keep the math from breaking by tweaking each of the values just a bit.  
Change everything negative to zero.

# Convolution Neural Networks

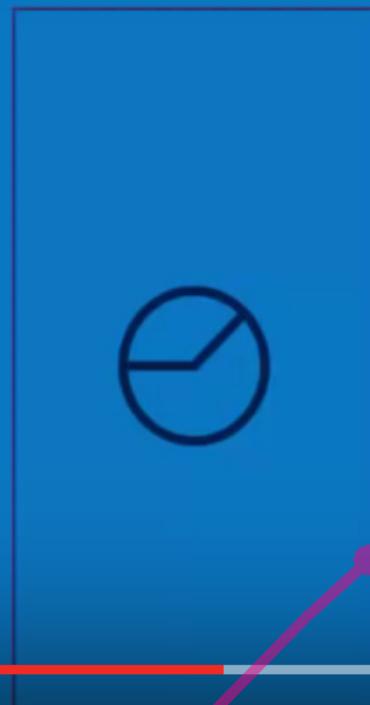
## ReLU layer

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

0.77	-0.11	0.11	0.11	0.11	0.11	0.11	0.11
-0.11	1.00	-0.11	0.11	-0.11	0.11	-0.11	
0.11	-0.11	1.00	-0.11	0.11	-0.11	0.11	
0.11	0.11	-0.11	0.11	-0.11	0.11	0.11	
0.11	0.11	0.11	1.00	-0.11	0.11	0.11	
0.11	0.11	0.11	0.11	1.00	-0.11	0.11	
0.11	0.11	0.11	0.11	0.11	1.00	-0.11	
0.11	0.11	0.11	0.11	0.11	0.11	1.00	

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

0.11	-0.11	0.11	0.11	-0.11	0.11	0.11	0.11
-0.11	0.11	-0.11	0.11	-0.11	1.00	-0.11	
0.11	-0.11	0.11	-0.11	0.11	-0.11	1.00	-0.11
0.11	0.11	-0.11	-0.11	0.11	-0.11	0.11	1.00
0.11	0.11	1.00	-0.11	0.11	-0.11	0.11	0.11
0.11	0.11	0.11	-0.11	0.11	-0.11	0.11	1.00
0.11	0.11	0.11	0.11	1.00	-0.11	0.11	0.11
0.11	0.11	0.11	0.11	0.11	1.00	-0.11	0.11



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

0.33	0	0.33	0	0.33	0	0.33
0	0.33	0	0.33	0	0.33	0
0.33	0	0.33	0	0.33	0	0.33
0	0.33	0	0.33	0	0.33	0
0.33	0	0.33	0	0.33	0	0.33
0	0.33	0	0.33	0	0.33	0
0.33	0	0.33	0.33	0.33	0	0.33

0.33	0	0.33	0.33	0.33	0	0.33
0	0.33	0	0.33	0.33	0	0.33
0.33	0	0.33	0	0.33	0	0.33
0	0.33	0	0.33	0	0.33	0
0.33	0	0.33	0	0.33	0	0.33
0	0.33	0	0.33	0	0.33	0
0.33	0	0.33	0.33	0.33	0	0.33

## Normalization

Keep the math from breaking by tweaking each of the values just a bit.  
Change everything negative to zero.

# Convolution Neural Networks

Layers get stacked

The output of one becomes the input of the next.

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



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

0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

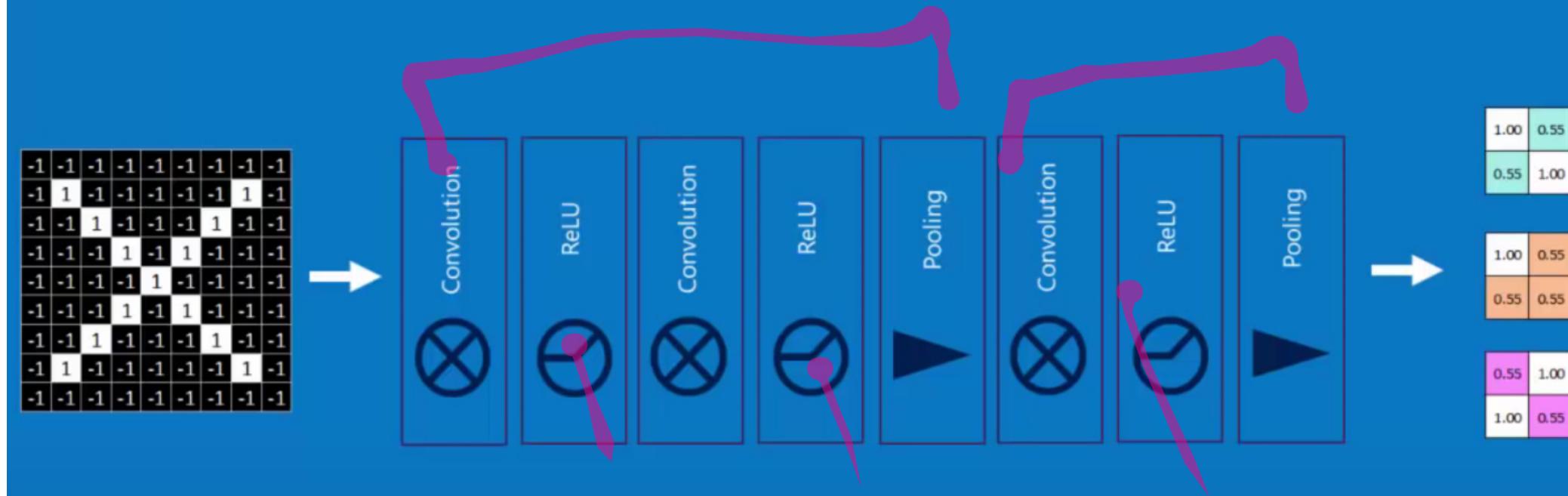
  

0.33	0.55	1.00	0.77
0.55	0.55	1.00	0.33
1.00	1.00	0.11	0.55
0.77	0.33	0.55	0.33

# Convolution Neural Networks

Deep stacking

Layers can be repeated several (or many) times.



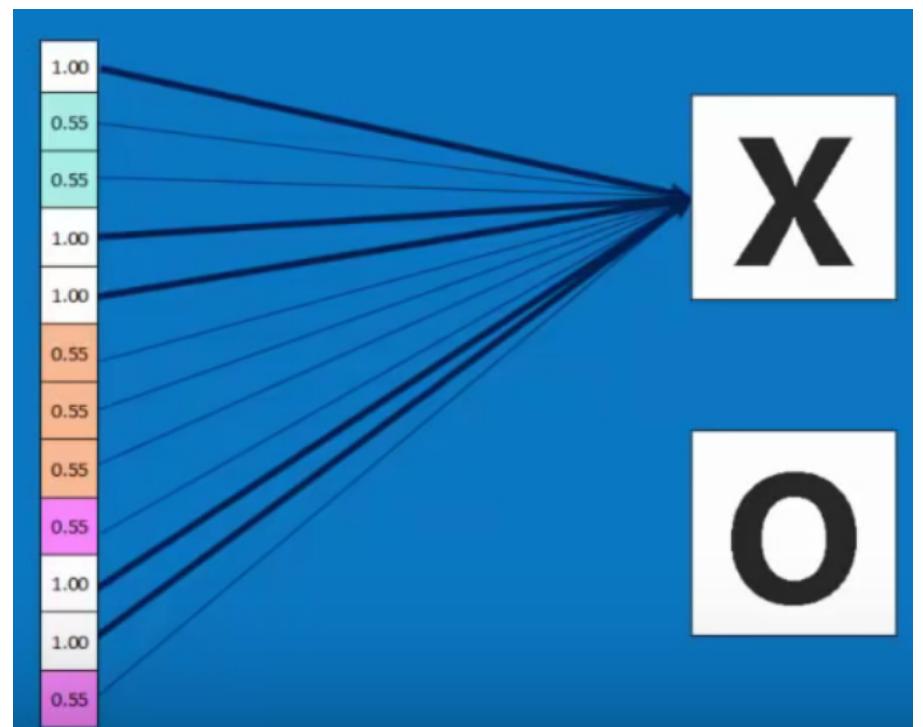
- ✓ Convolution layer: filtering
- ✓ Pooling layer: smaller

# Convolution Neural Networks

Fully connected layer

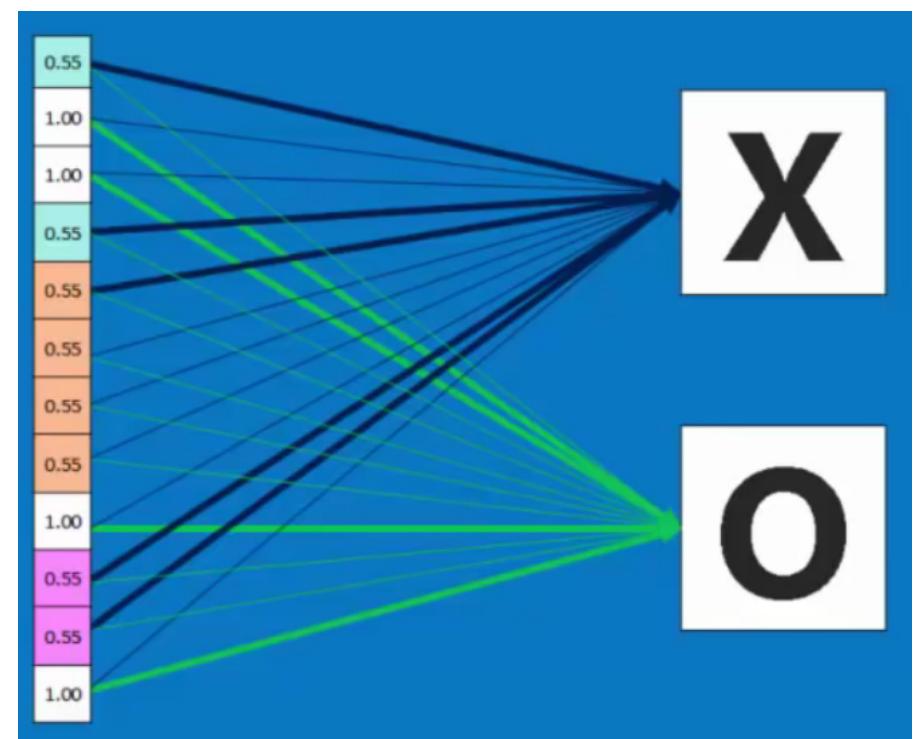
Every value gets a vote

Vote depends on how strongly a value predicts X or O



# Convolution Neural Networks

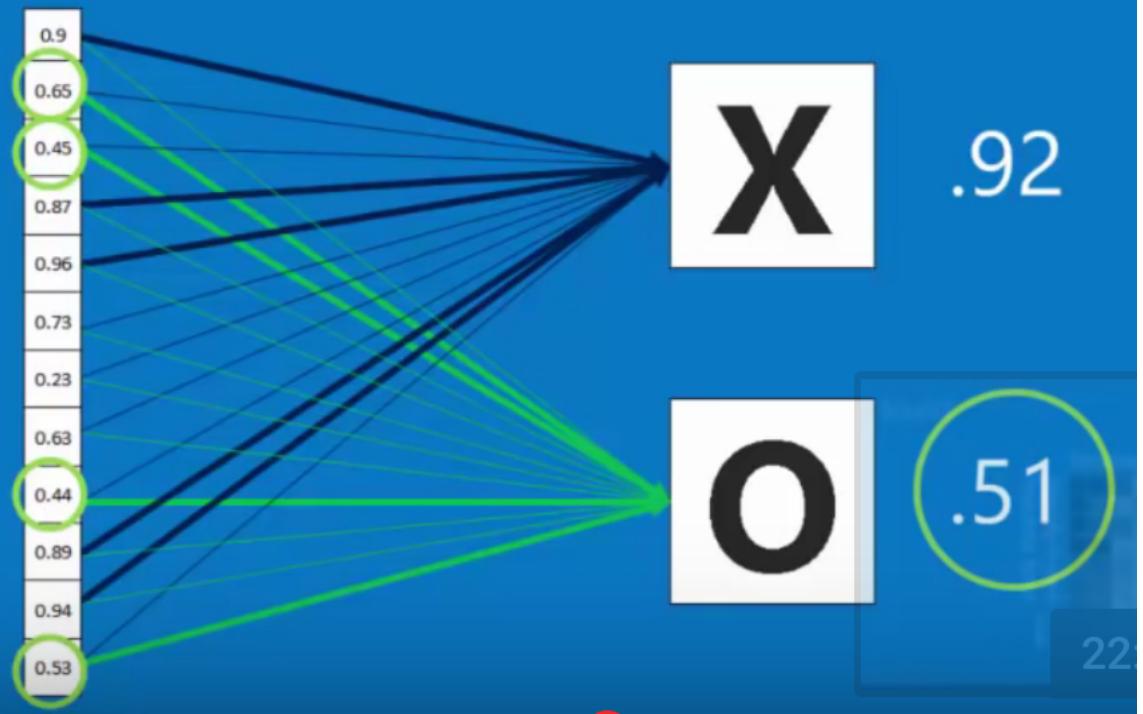
Vote depends on how strongly a value predicts X or O



# Convolution Neural Networks

Fully connected layer

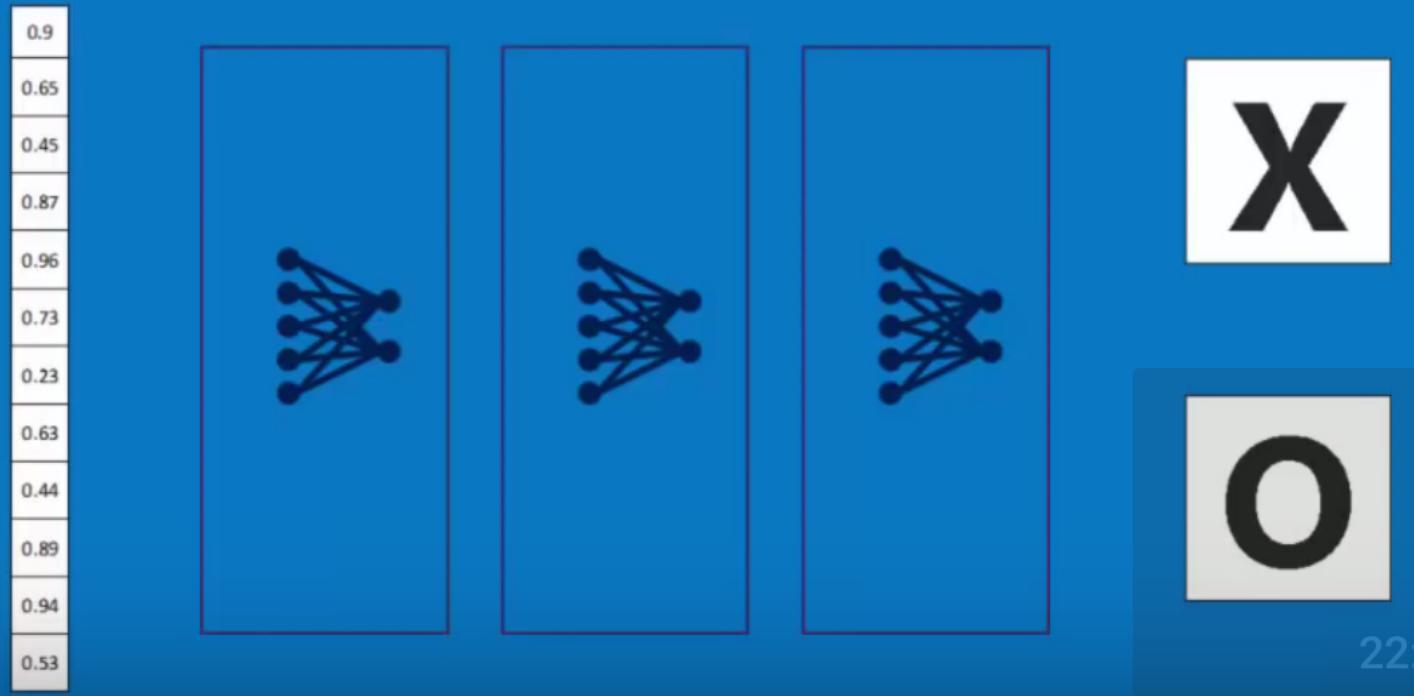
Future values vote on X or O



# Convolution Neural Networks

Fully connected layer

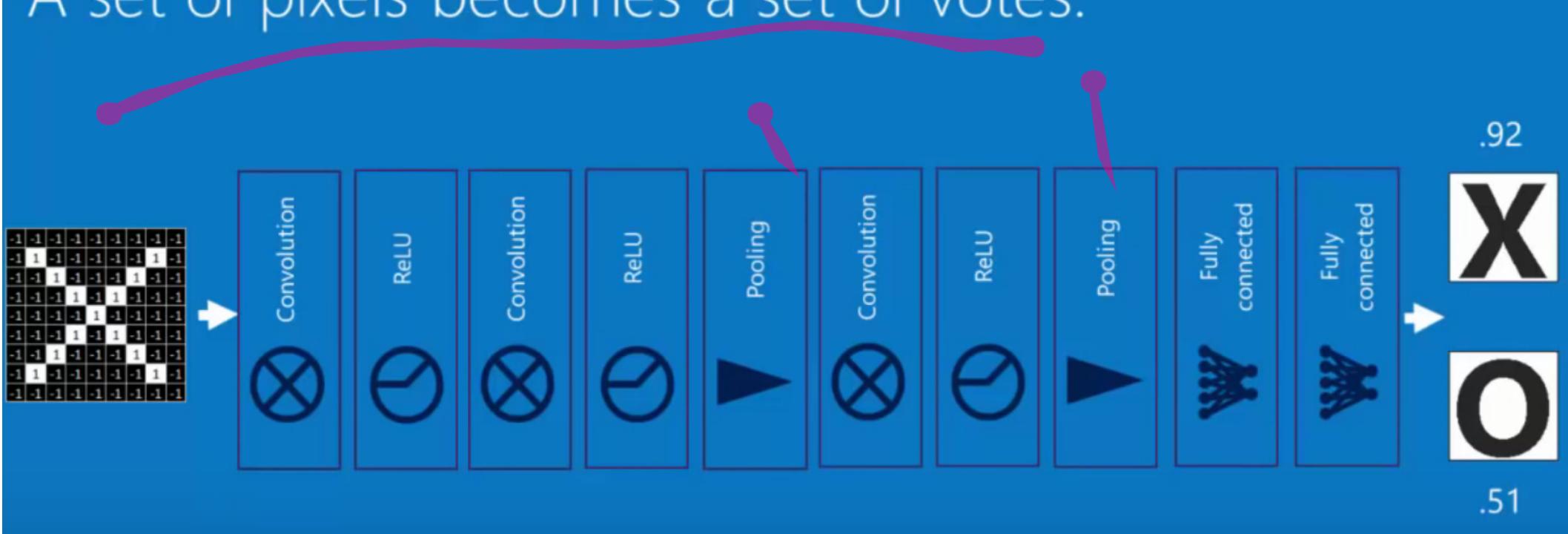
These can also be stacked.



# Convolution Neural Networks

Putting it all together

A set of pixels becomes a set of votes.



# Convolution Neural Networks

Learning

Q: Where do all the magic numbers come from?

Features in convolutional layers

Voting weights in fully connected layers

A: Backpropagation

# Convolution Neural Networks

Backprop

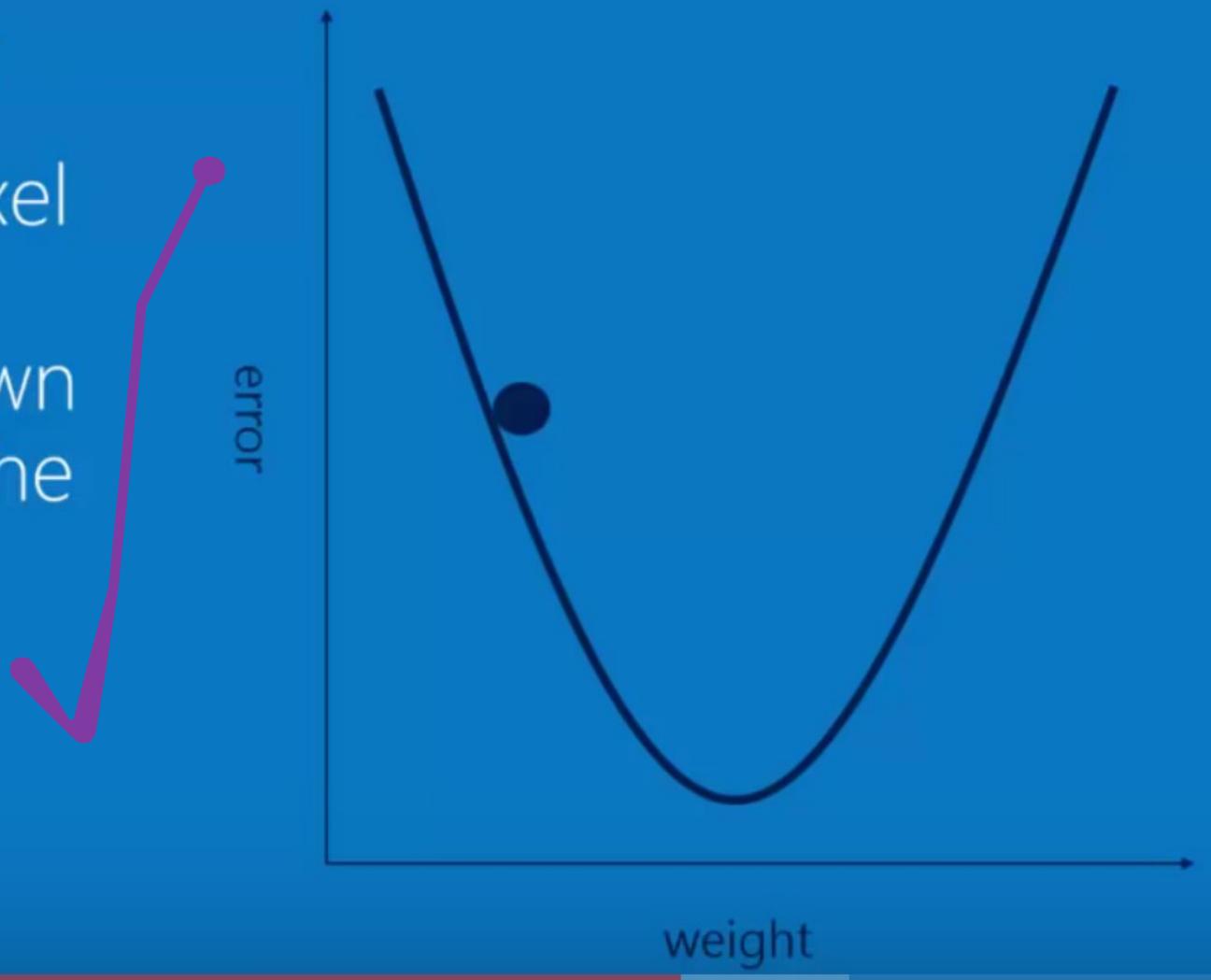
	Right answer	Actual answer	Error
X	1	0.92	0.08
O	0	0.51	0.49
Total			0.57



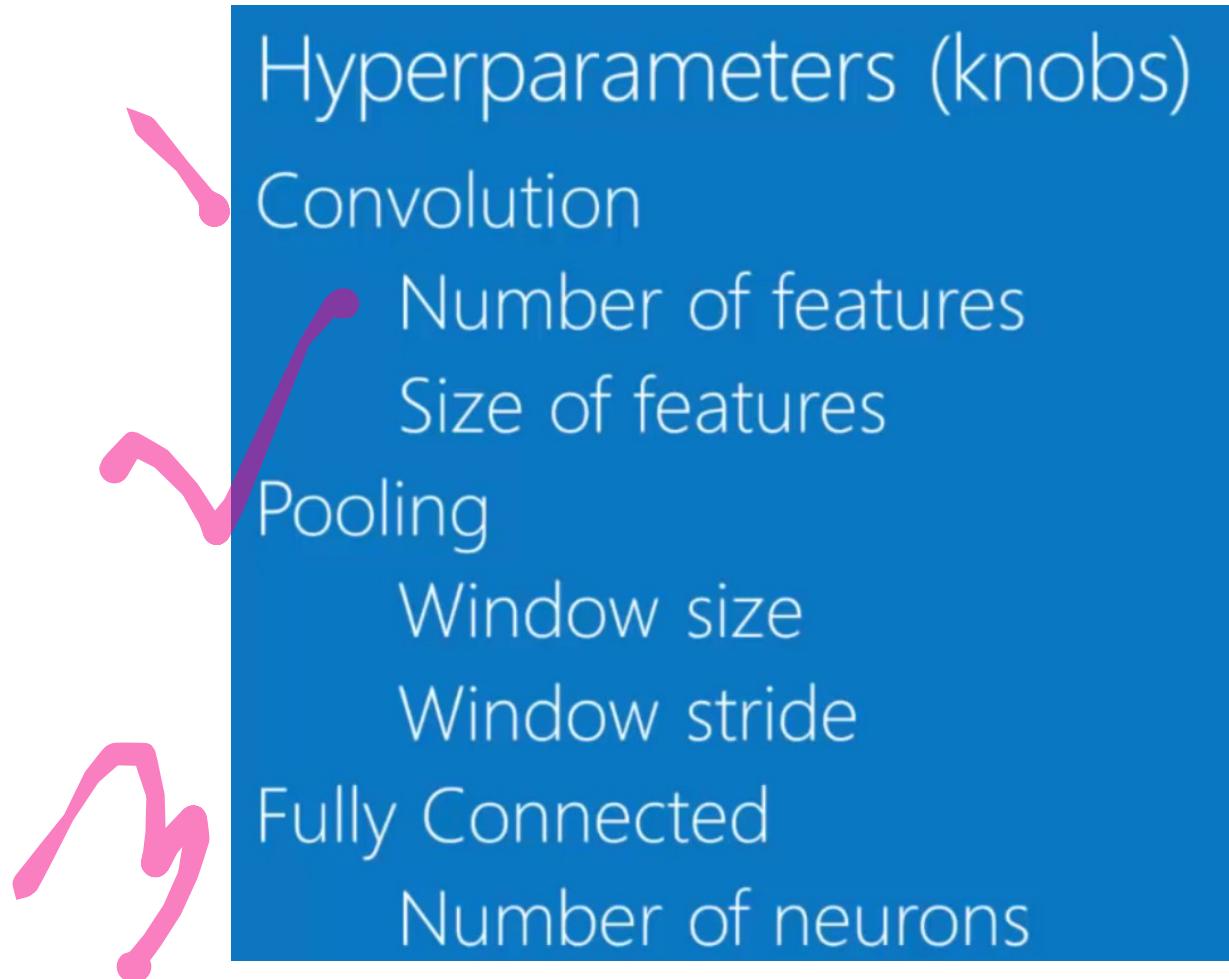
# Convolution Neural Networks

Gradient descent

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



# Convolution Neural Networks



Architecture

How many of each type of layer?  
In what order?

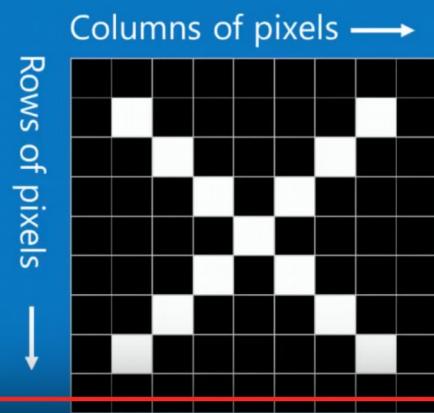
# Convolution Neural Networks

Not just images

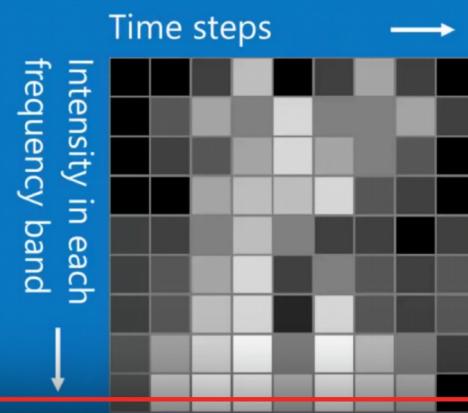
Any 2D (or 3D) data.

Things closer together are more closely related than things far away.

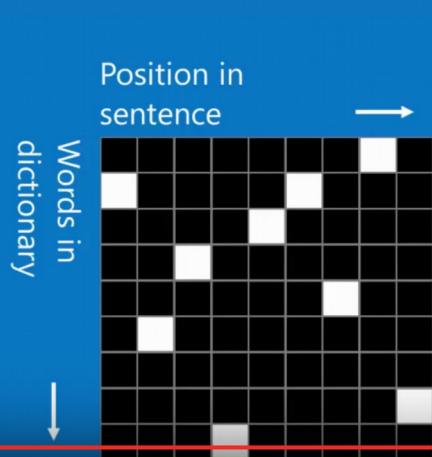
Images



Sound



Text



# Convolution Neural Networks

## Limitations

ConvNets only capture local "spatial" patterns in data.  
If the data can't be made to look like an image,  
ConvNets are less useful.

## Rule of thumb

- If your data is just as useful after swapping any of your columns with each other, then you can't use Convolutional Neural Networks.

## Customer data



Name, age,  
address, email,  
purchases,  
browsing activity,...

A	22	1A	a@a	1	aa	a1.a	123	aa1
B	33	2B	b@b	2	bb	b2.b	234	bb2
C	44	3C	c@c	3	cc	c3.c	345	cc3
D	55	4D	d@d	4	dd	d4.d	456	dd4
E	66	5E	e@e	5	ee	e5.e	567	ee5
F	77	6F	f@f	6	ff	f6.f	678	ff6
G	88	7G	g@g	7	gg	g7.g	789	gg7
H	99	8H	h@h	8	hh	h8.h	890	hh8
I	111	9I	i@i	9	ii	i9.i	901	ii9

# Convolution Neural Networks

In a nutshell

ConvNets are great at finding patterns and using them to classify images.

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