Convolutional Neural Network

©Dr. Min Chi
mchi@ncsu.edu

AI Academy

Computer Vision is Hard

Viewpoint variation

Scale variation

Deformation

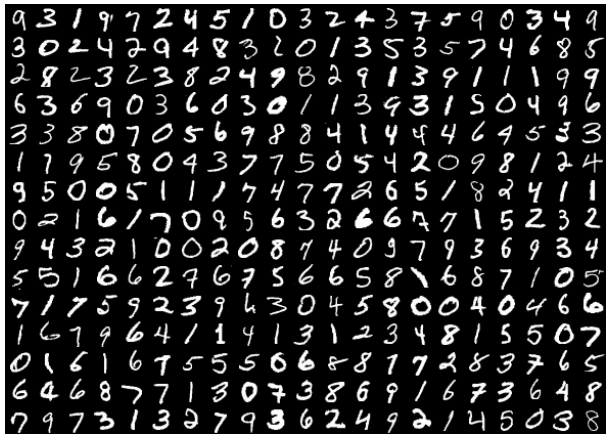
Occlusion

Illumination conditions

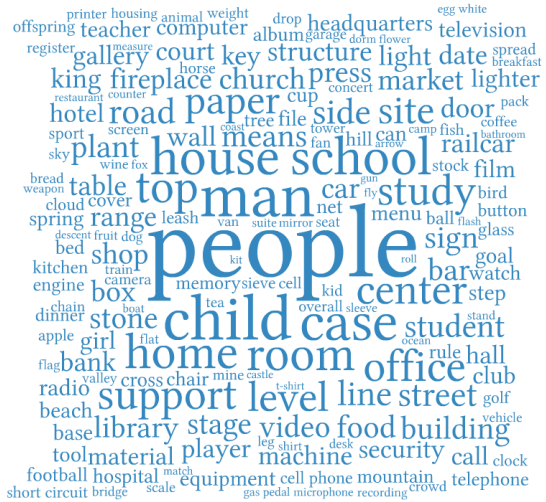
Background clutter

Intra-class variation

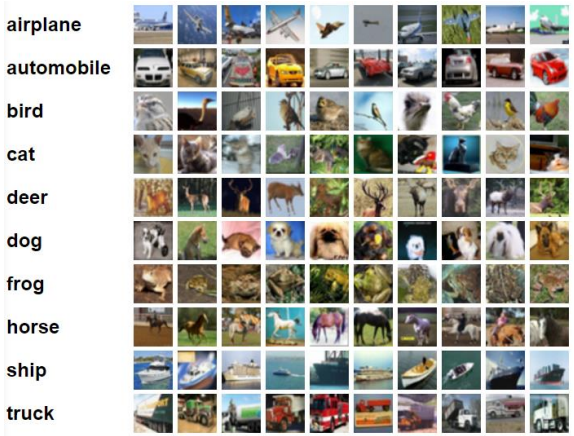
Famous Computer Vision Datasets



MNIST: handwritten digits



ImageNet: WordNet hierarchy



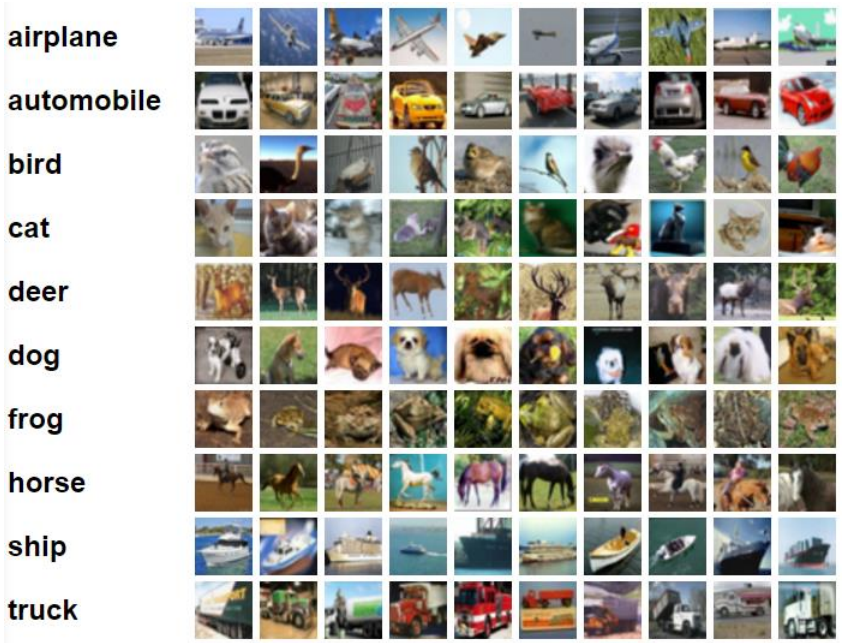
CIFAR-10(0): tiny images

Places: natural scenes

AI Academy

NC STATE

Let's Build an Image Classifier for CIFAR-10



test image					training image					pixel-wise absolute value differences				
56	32	10	18	-	10	20	24	17	=	46	12	14	1	→ 456
90	23	128	133		8	10	89	100		82	13	39	33	
24	26	178	200		12	16	178	170		12	10	0	30	
2	0	255	220		4	32	233	112		2	32	22	108	

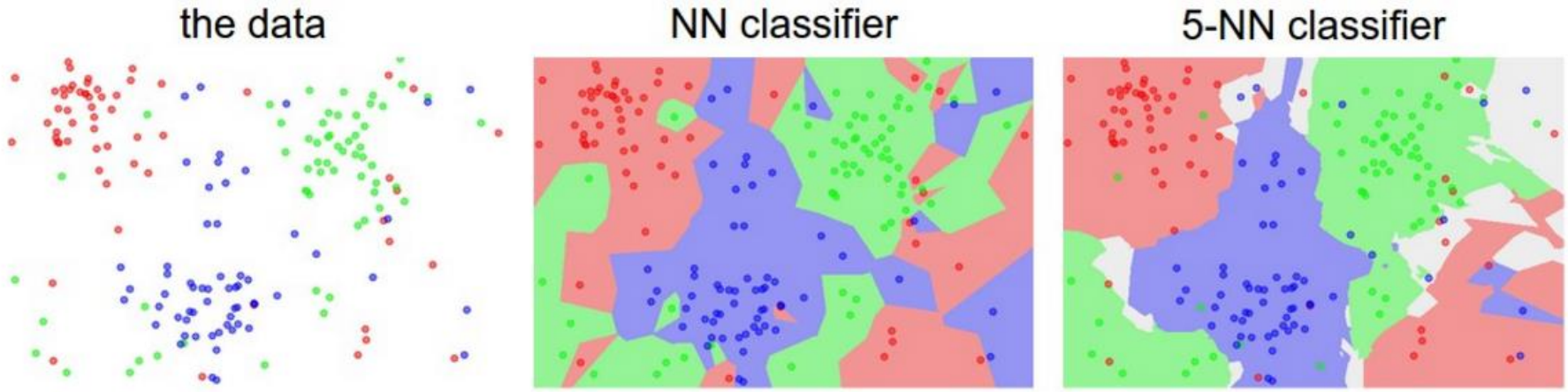
Let's Build an Image Classifier for CIFAR-10

test image					training image					pixel-wise absolute value differences				
56	32	10	18	-	10	20	24	17	=	46	12	14	1	→ 456
90	23	128	133		8	10	89	100		82	13	39	33	
24	26	178	200		12	16	178	170		12	10	0	30	
2	0	255	220		4	32	233	112		2	32	22	108	

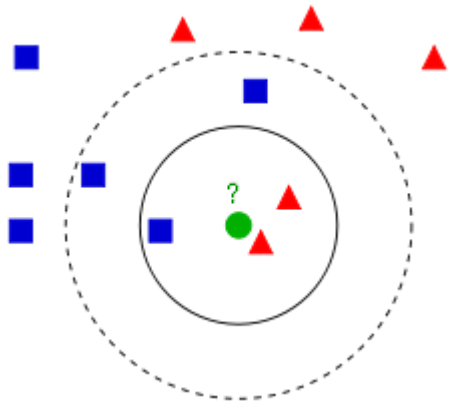
Accuracy

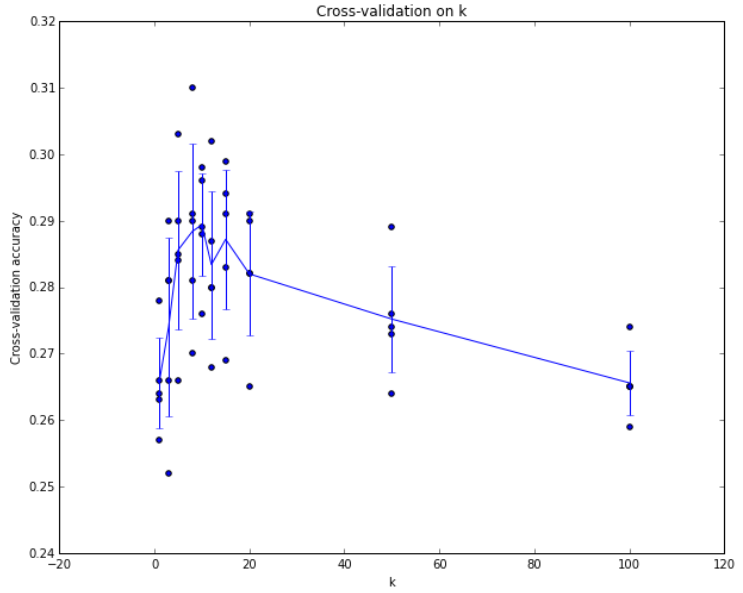
Random: **10%**

K-Nearest Neighbors: Generalizing the Image-Diff Classifier[6]



Tuning (hyper)parameters:

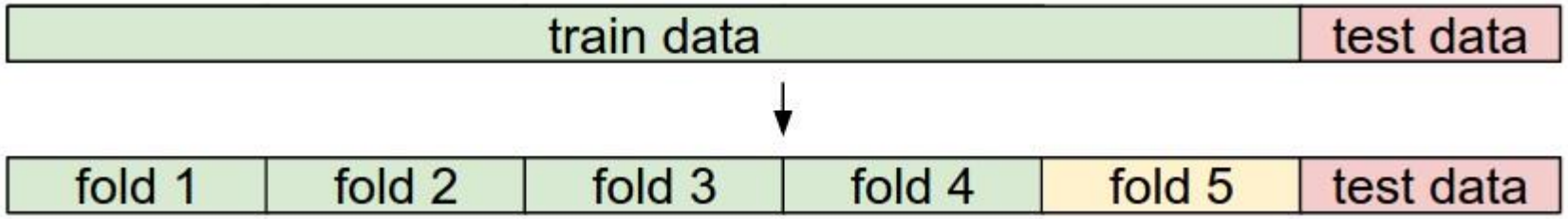


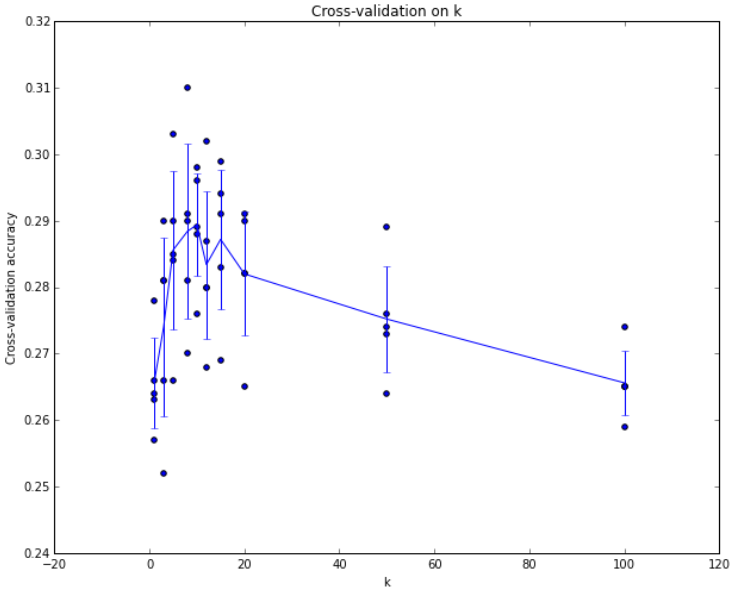


AI Academy

NC STATE

K-Nearest Neighbors: Generalizing the Image-Diff Classifier





Accuracy

Random: **10%**

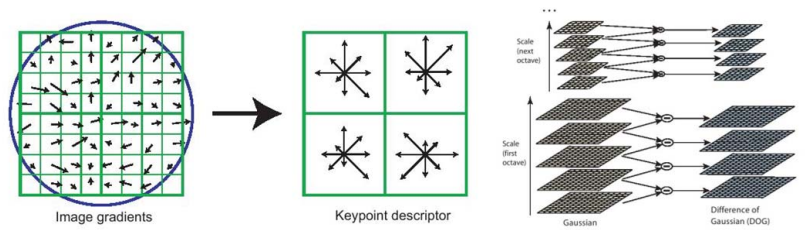
7-Nearest Neighbors: **~30%**

Human: **~94%**

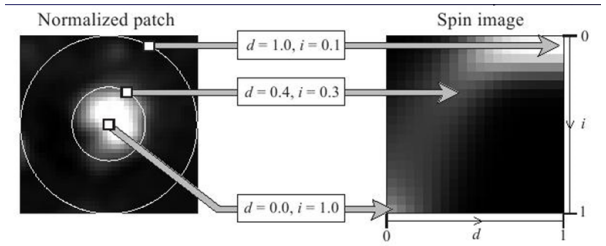
…

Convolutional Neural Networks: **~95%**

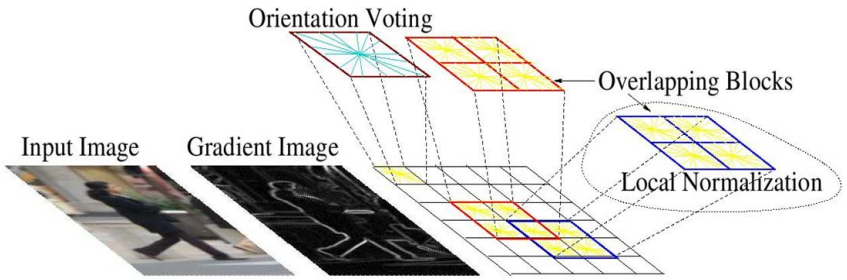
Grant challenges in computer vision: feature engineering

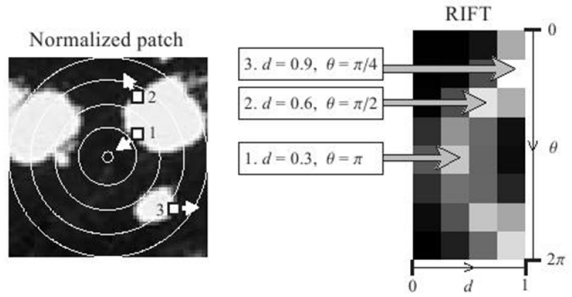


SIFT



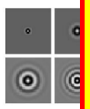
Spin image



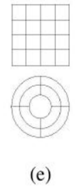


Drawbacks of feature engineering

1. Needs expert knowledge
2. Time consuming hand-tuning



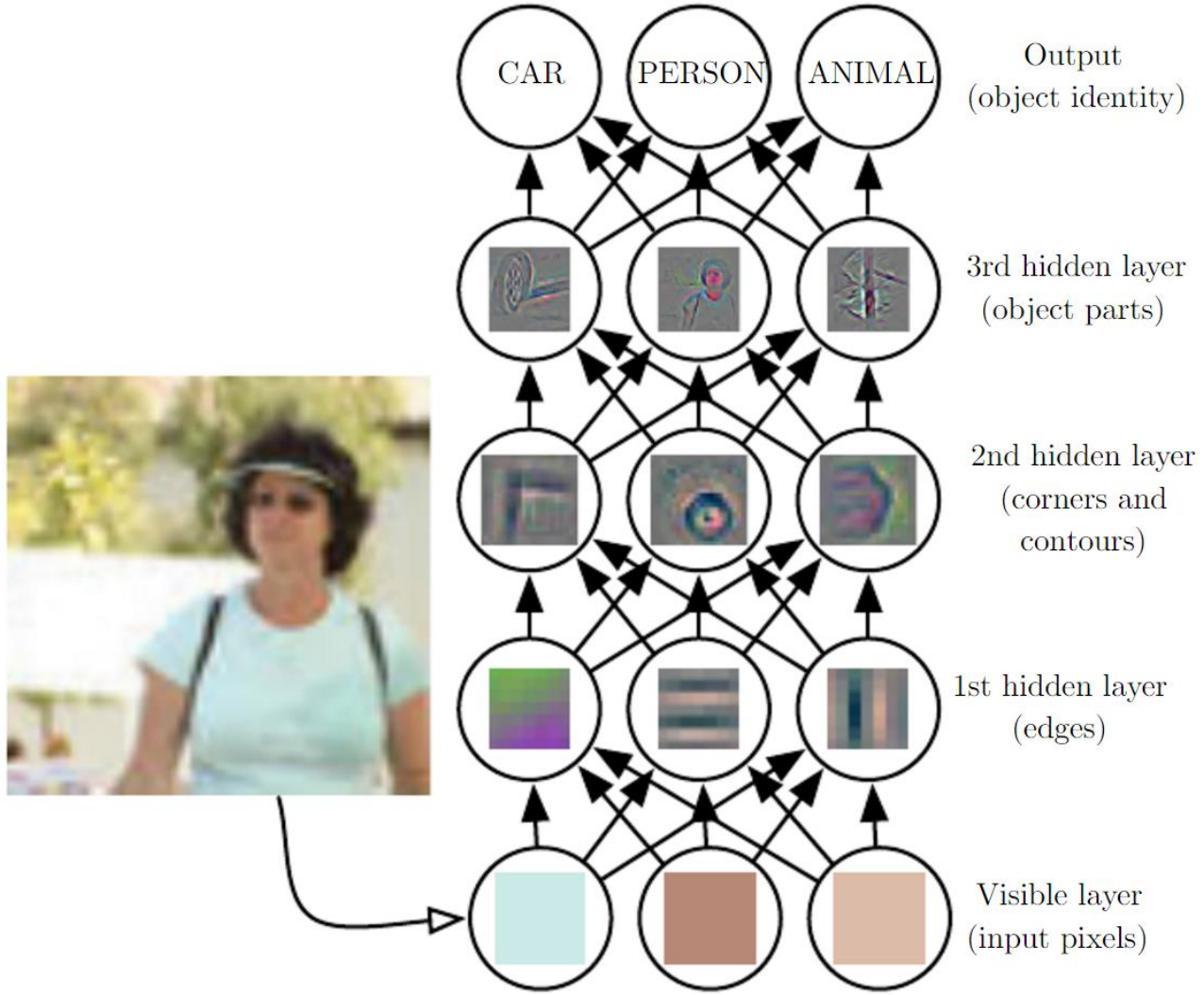
Textons

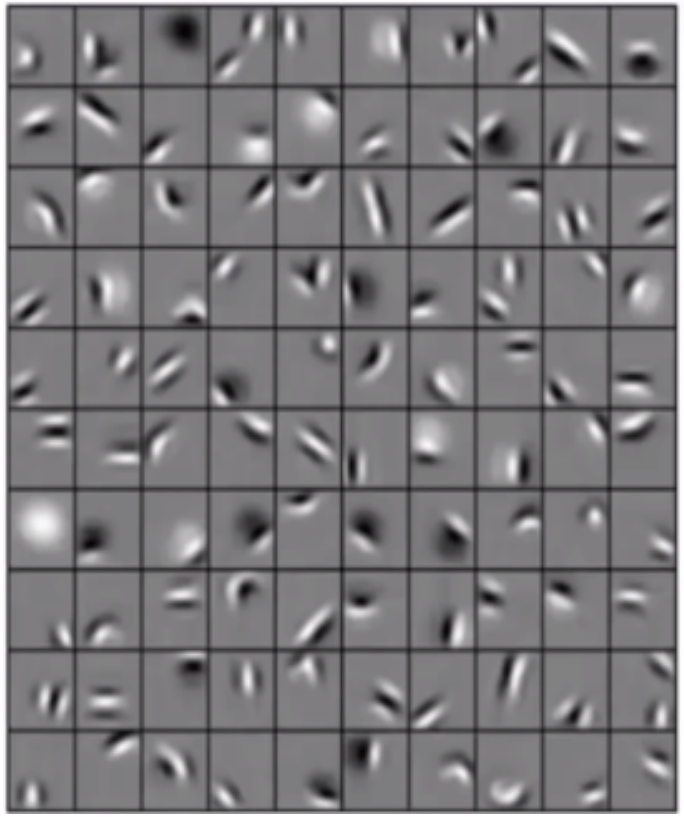


(e)

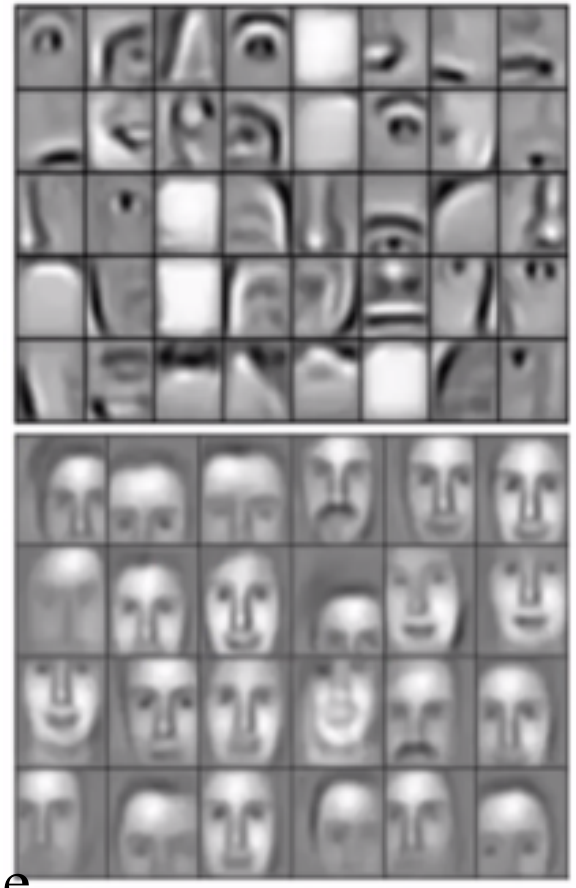
e and Ng

Deep Learning is **Representation Learning**

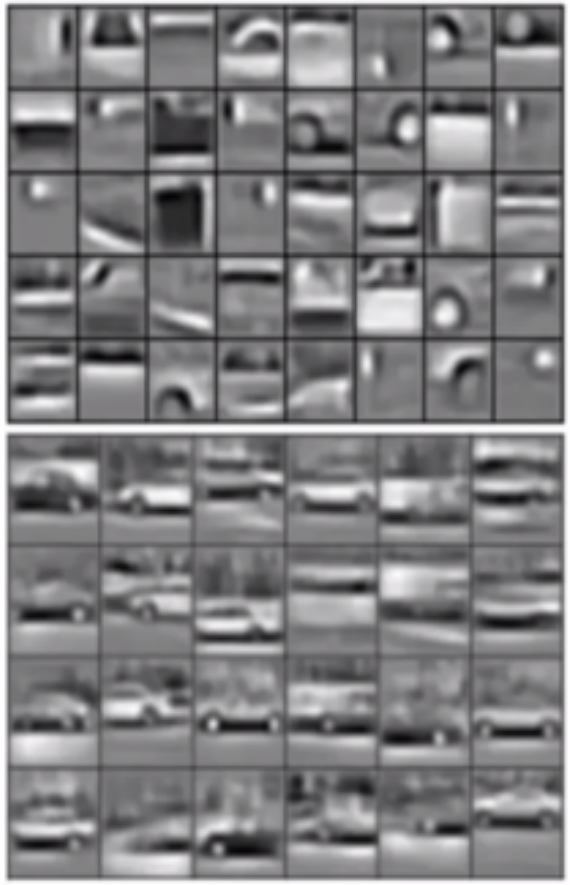




Faces

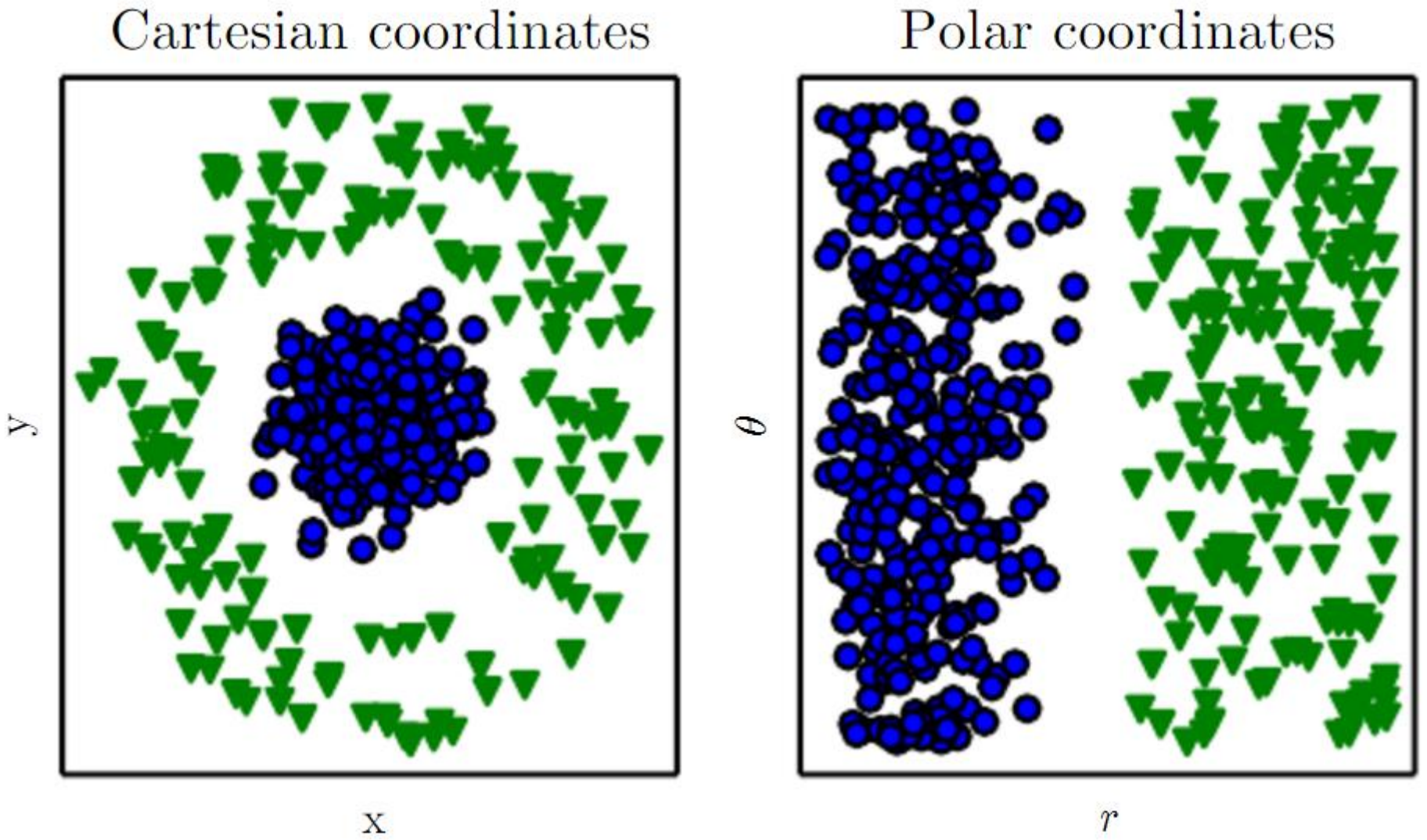


Cars

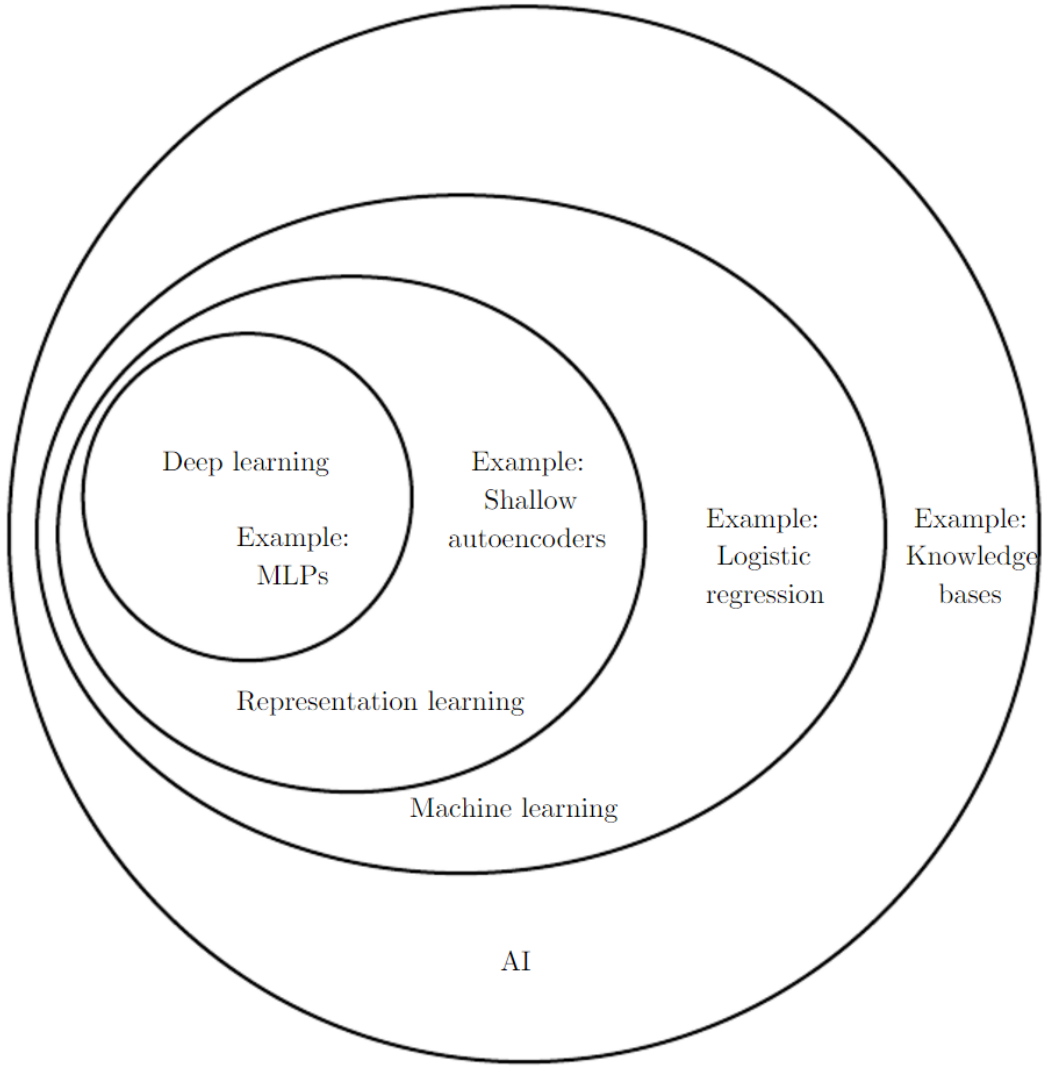


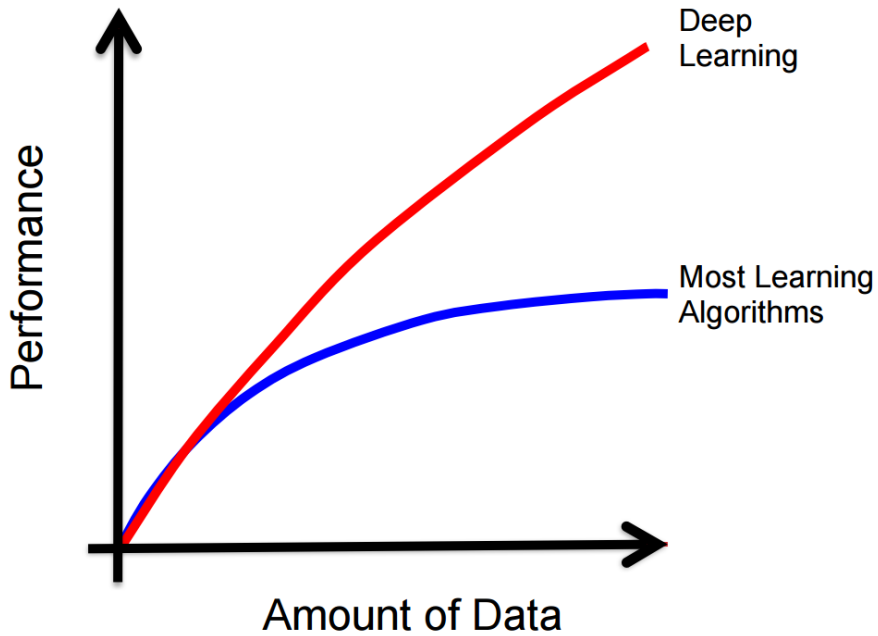
Convolutional Deep Belief Networks
for Scalable Unsupervised Learning of Hierarchical Representations;
Honglak Lee, Roger Grosse
Rajesh Ranganath, Andrew Ng

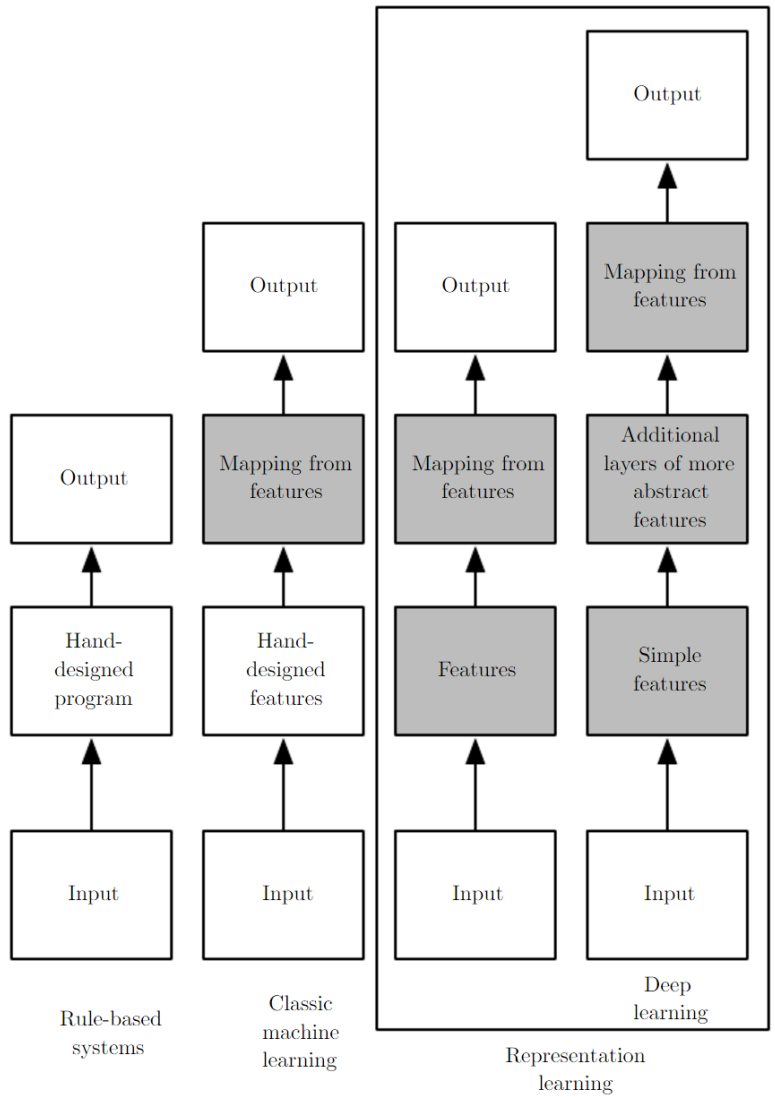
Representation Matters



Deep Learning is **Representation Learning**

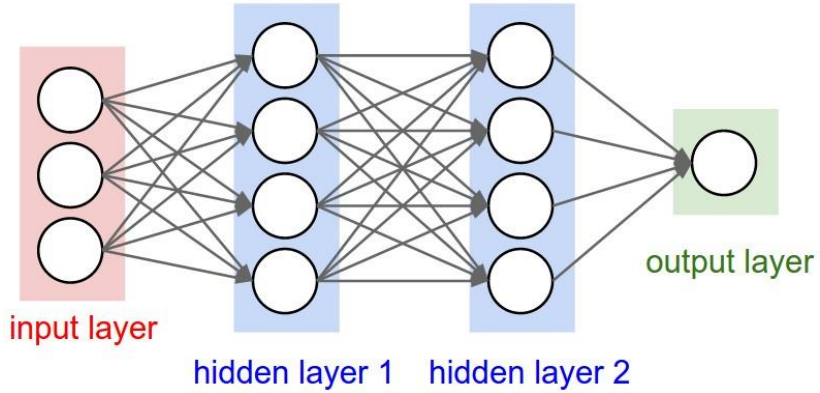



Performance
Amount of Data
Deep Learning
Most Learning Algorithms

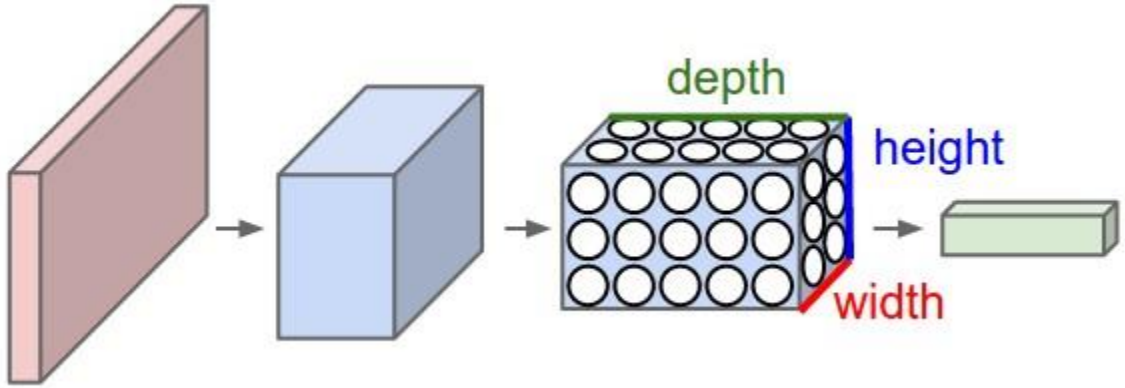

Output
Hand-designed program
Input
Output
Mapping from features
Hand-designed features
Input
Output
Mapping from features
Features
Input
Output
Mapping from features
Additional layers of more abstract features
Simple features
Input
Rule-based systems
Classic machine learning
Representation learning
Deep learning

Convolutional Neural Networks

Regular neural network (fully connected):



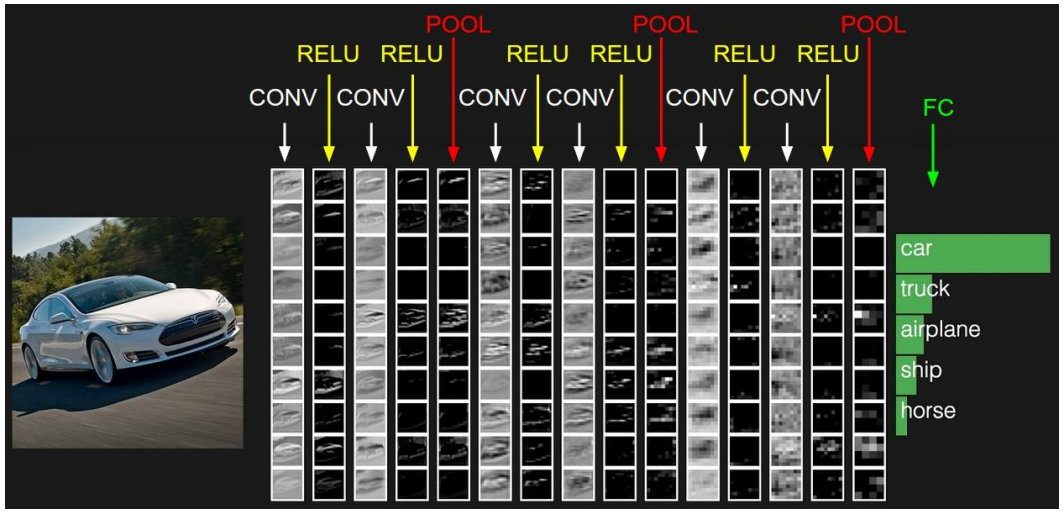
Convolutional neural network:



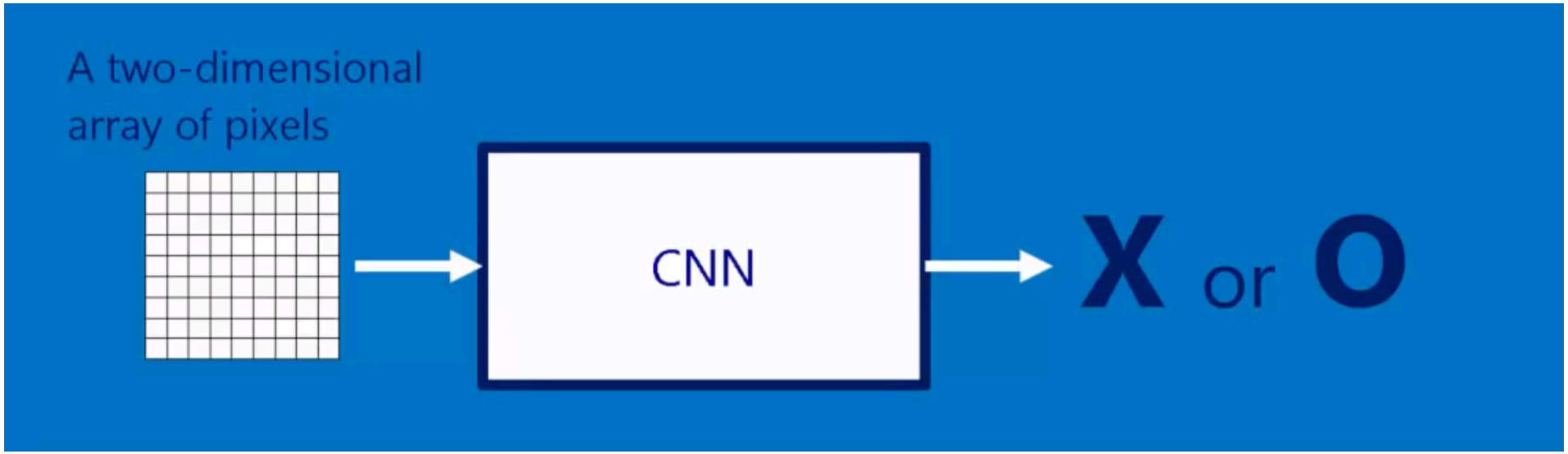
Each layer takes a 3d volume, produces 3d volume with some smooth function that may or may not have parameters.

Convolutional Neural Networks: Layers

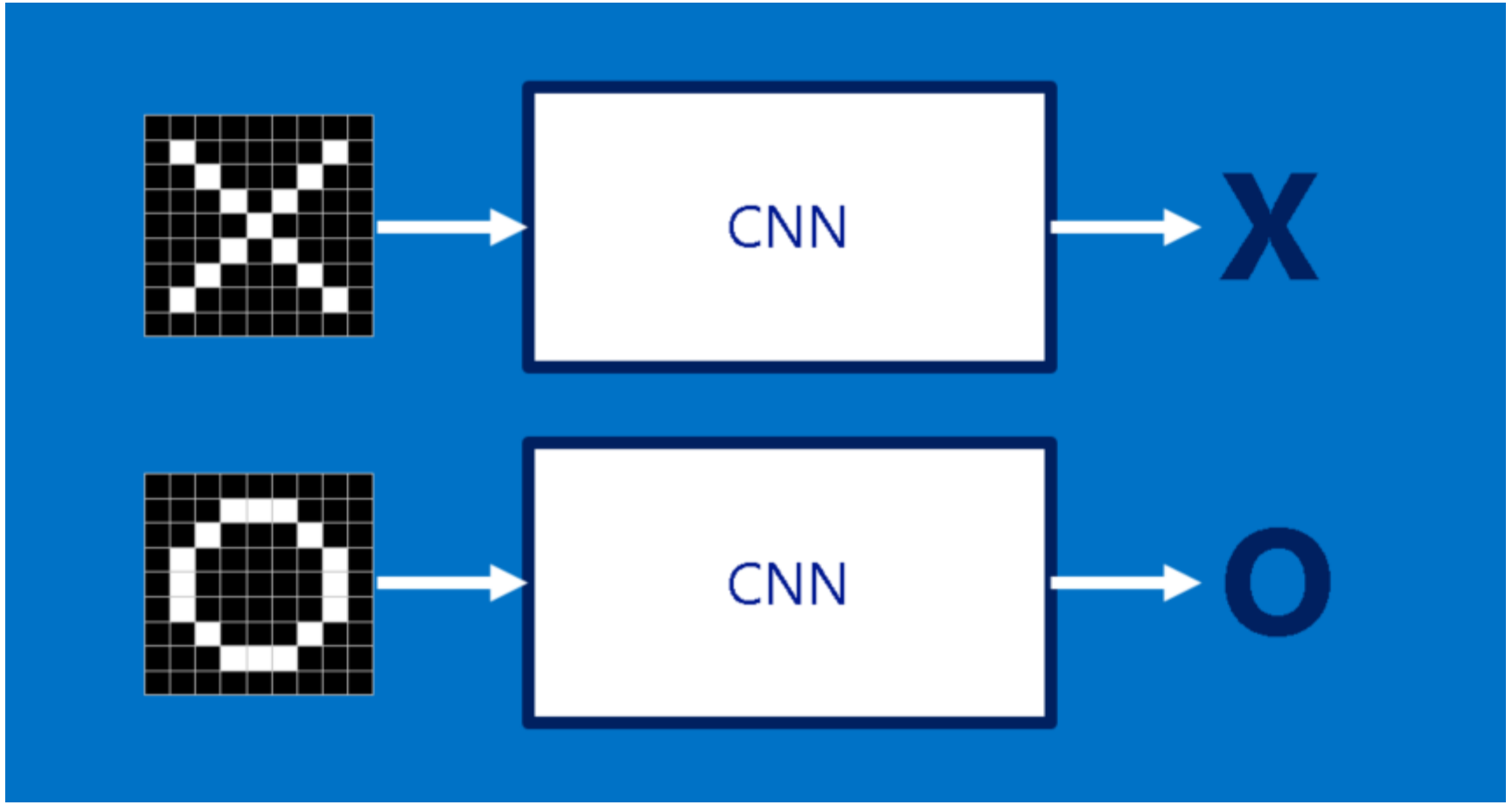
- **INPUT**
- **CONV** layer
- **RELU** layer
- **POOL** layer
- **FC** (i.e. fully-connected) layer



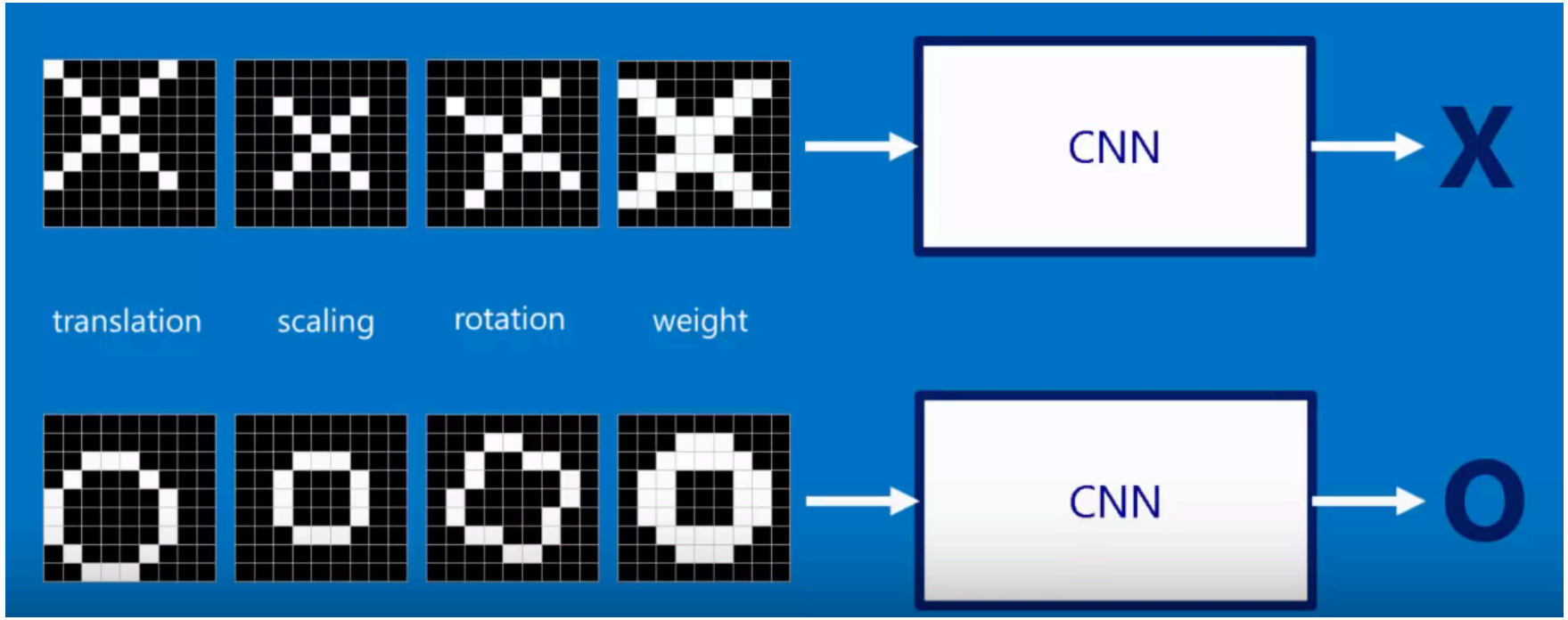
A Toy Example: X's or O's
Says whether a picture is of an X or an O



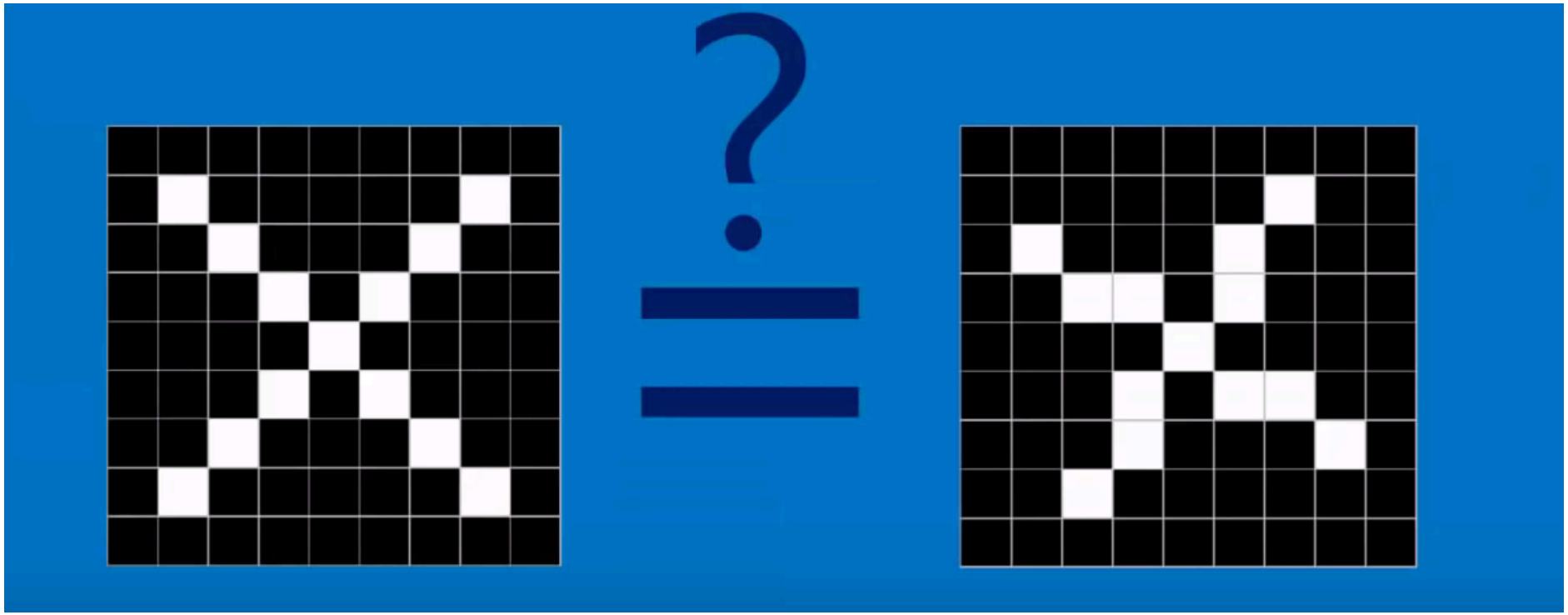
For Example

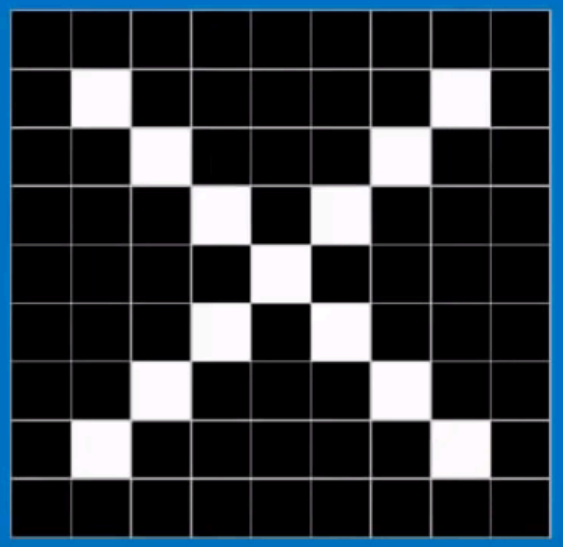


Trickier Cases

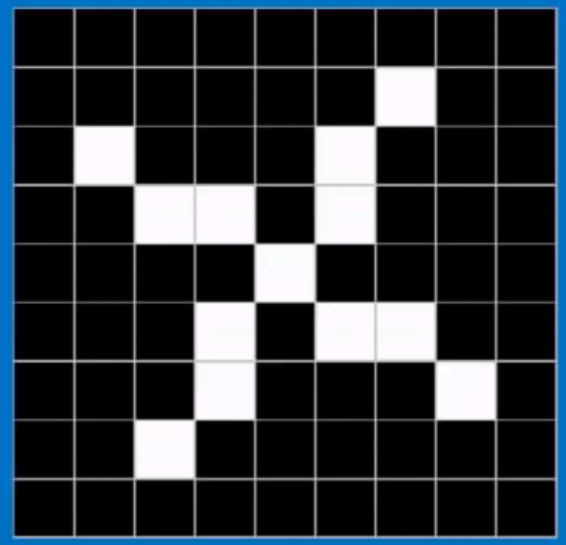


Deciding is hard for computer





?

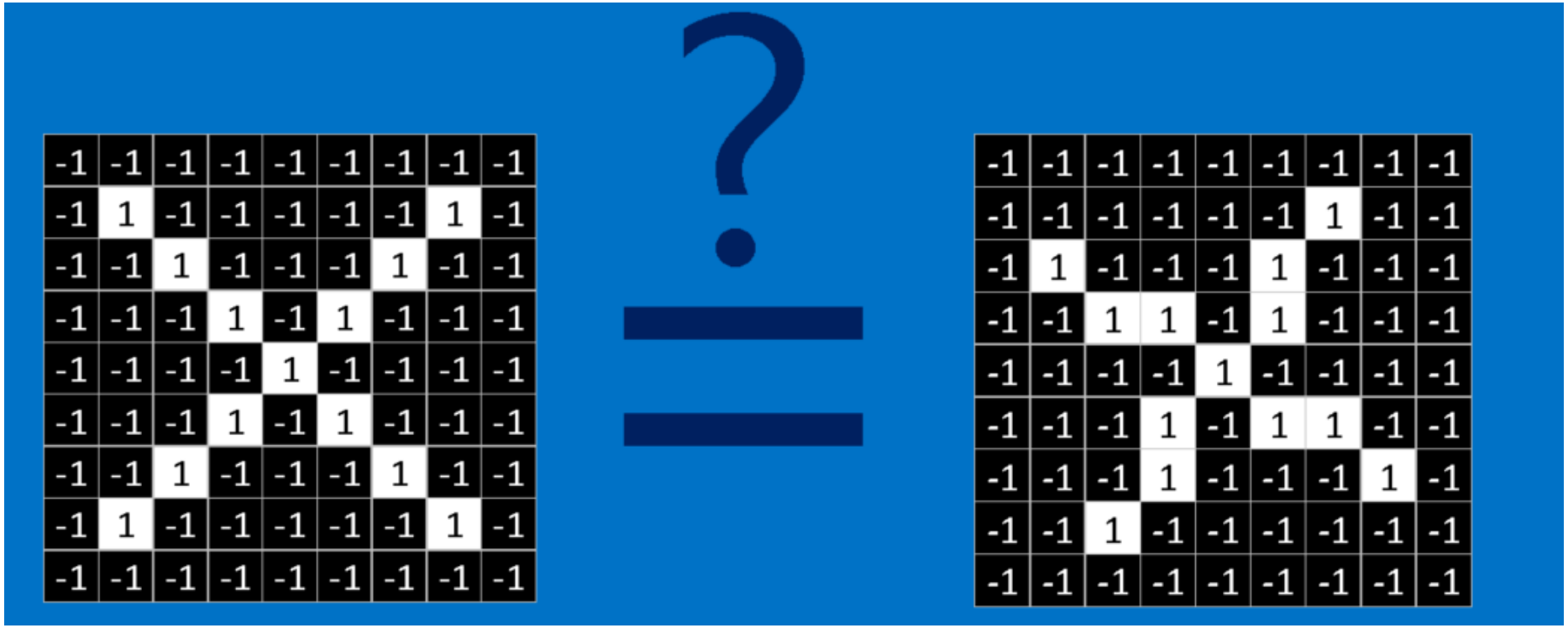


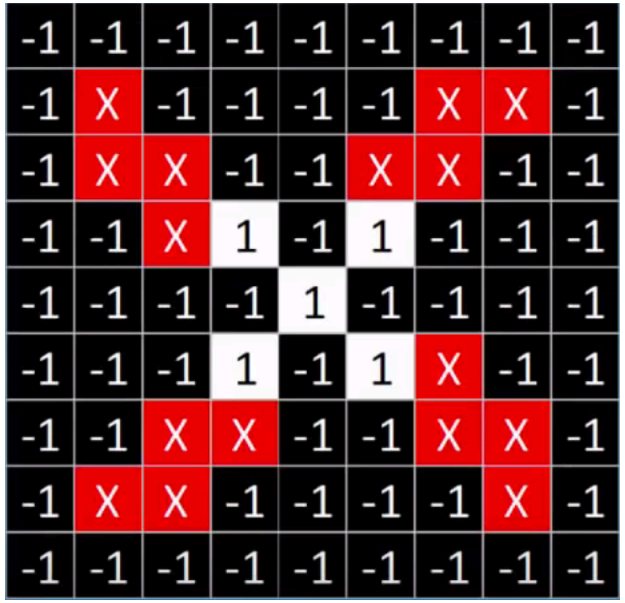
?

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

?

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


-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
?
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 1 -1 -1 -1
-1 -1 1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 1 -1 -1
-1 -1 -1 1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

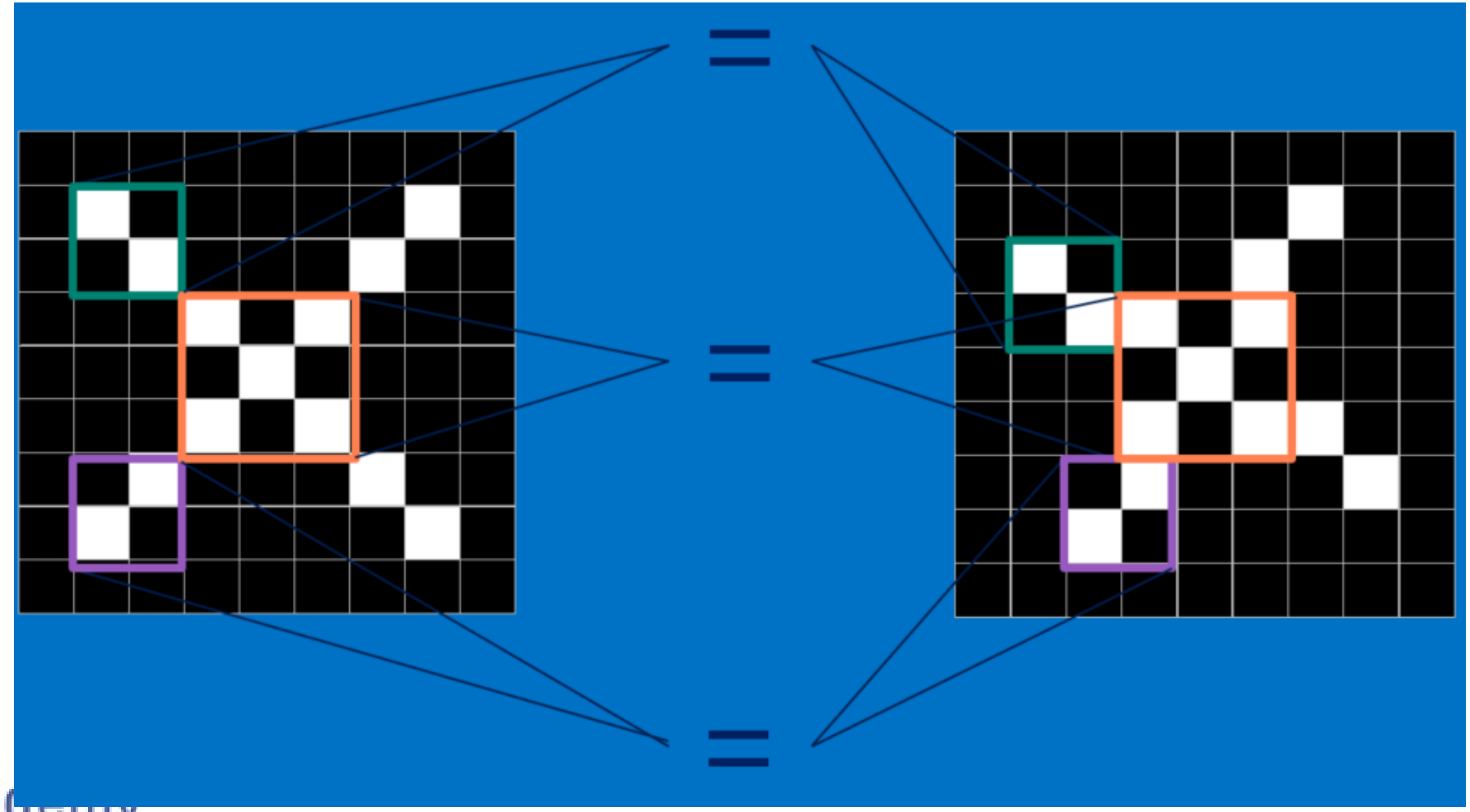

-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 X -1 -1 -1 -1 X X -1
-1 X X -1 -1 X X -1 -1
-1 -1 X 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 X -1 -1
-1 -1 X X -1 -1 X X -1
-1 X X -1 -1 -1 -1 X -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

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

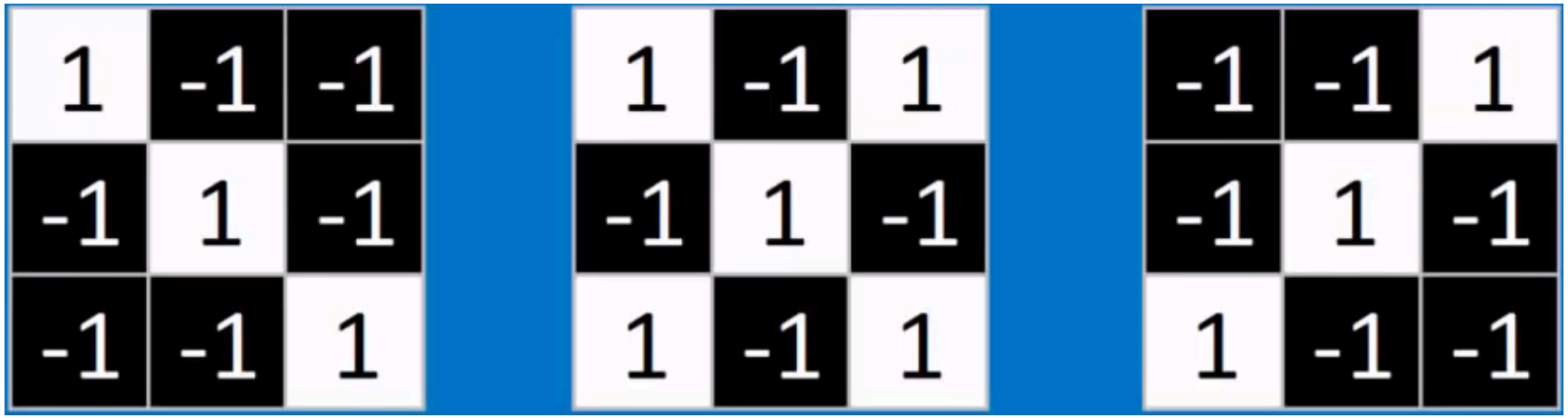
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

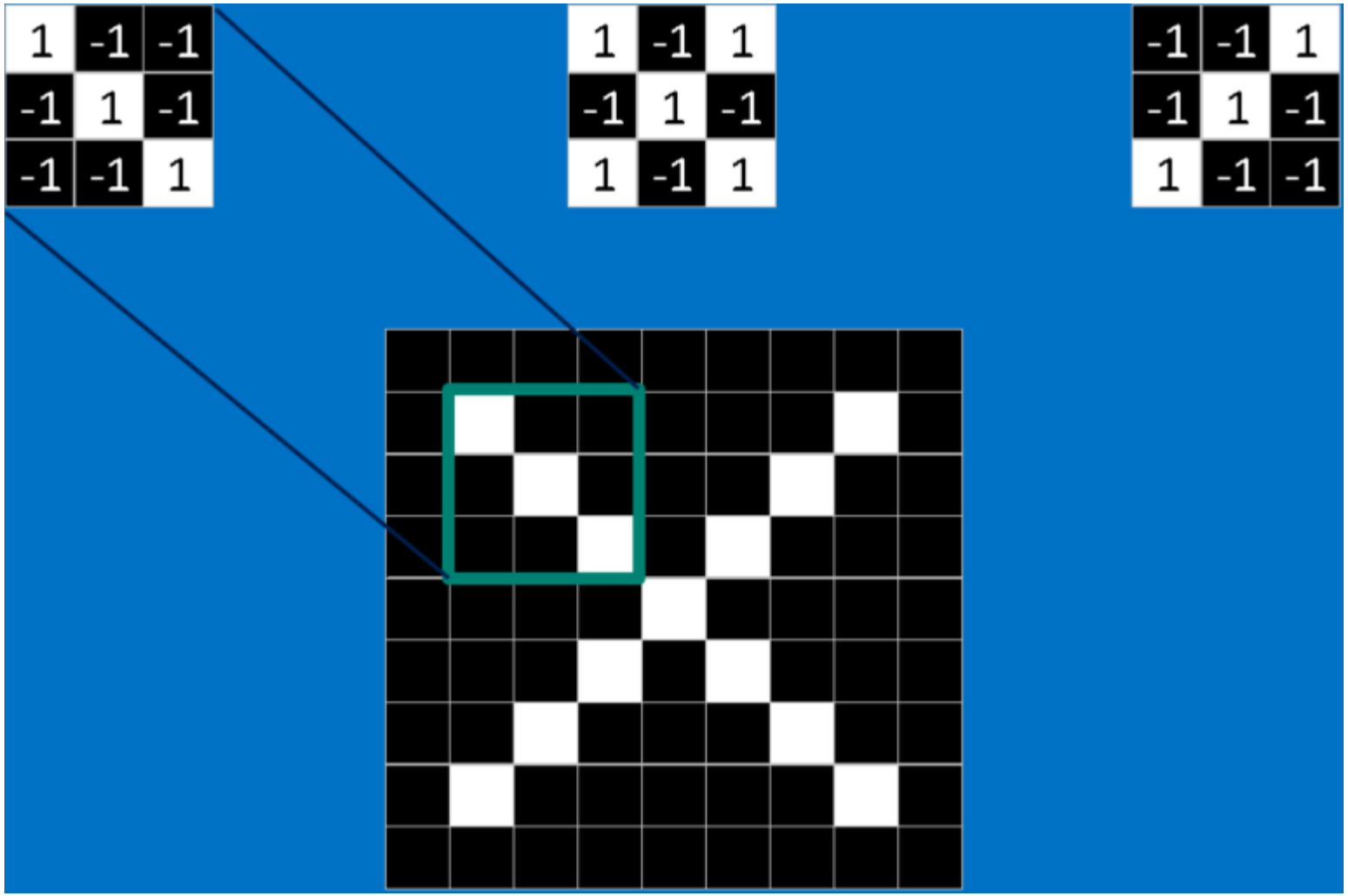
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	1	-1	-1	-1
-1	-1	1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	1	-1	-1
-1	-1	-1	1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

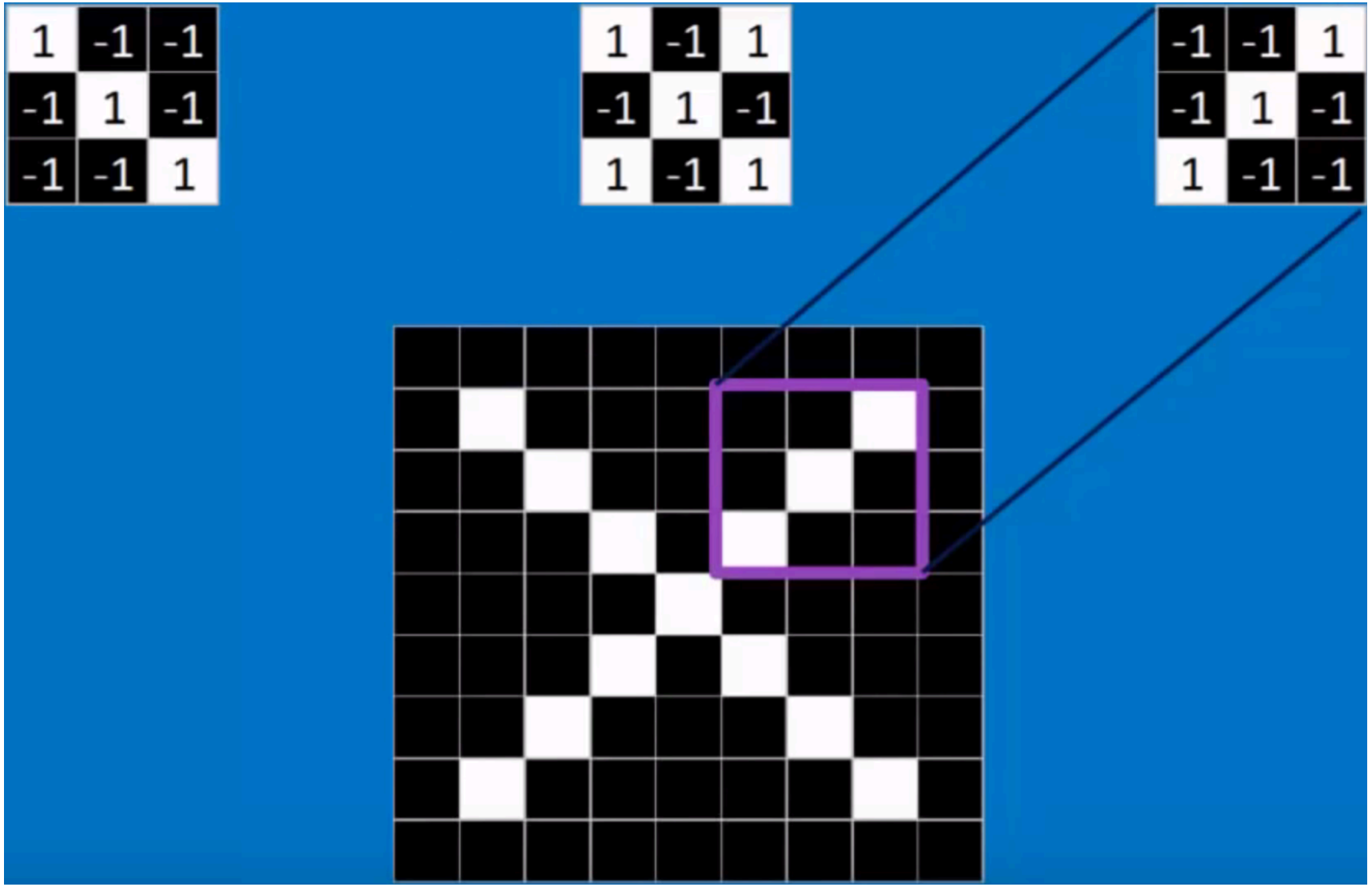
CNNs match pieces of the Image

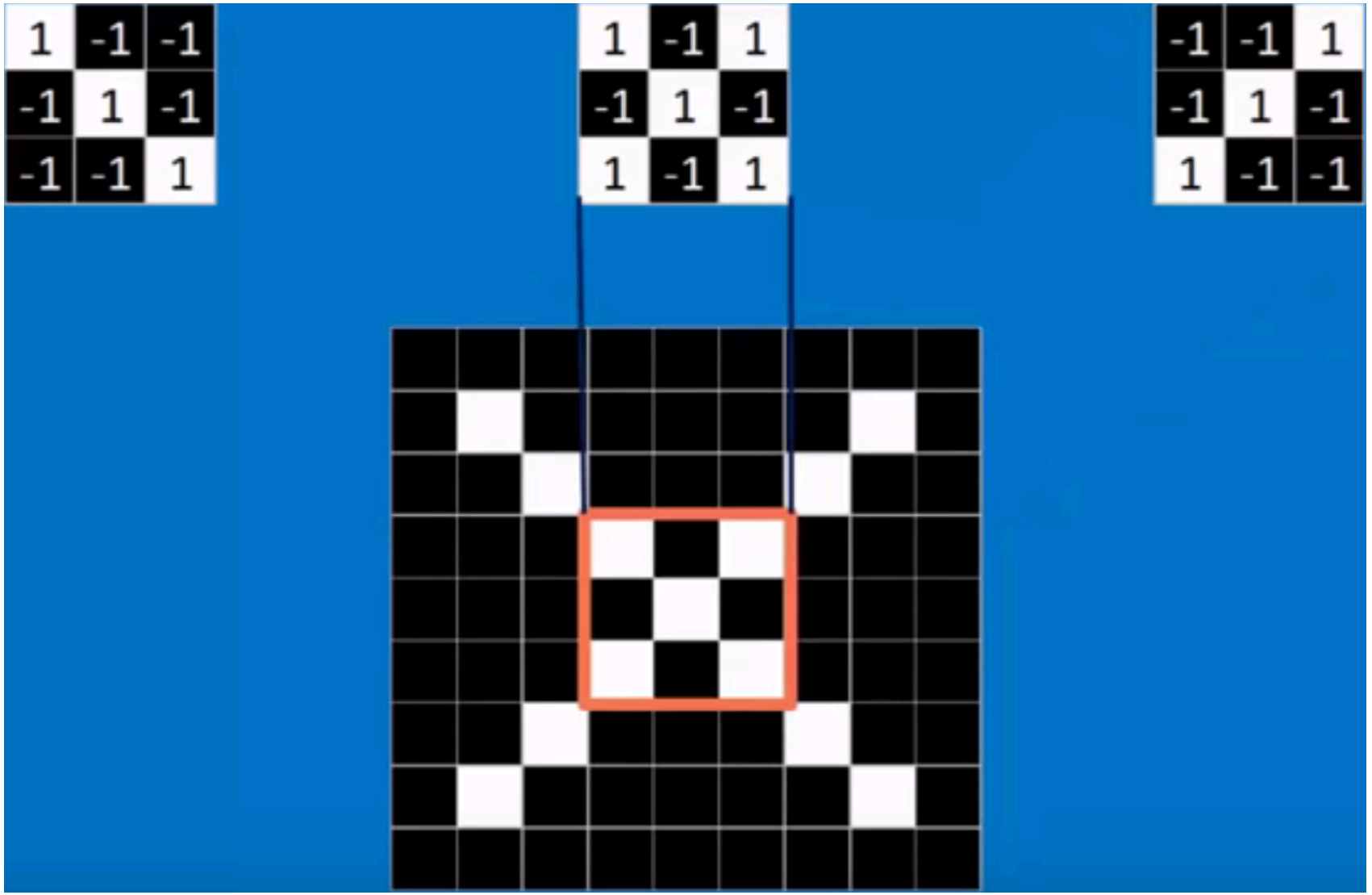


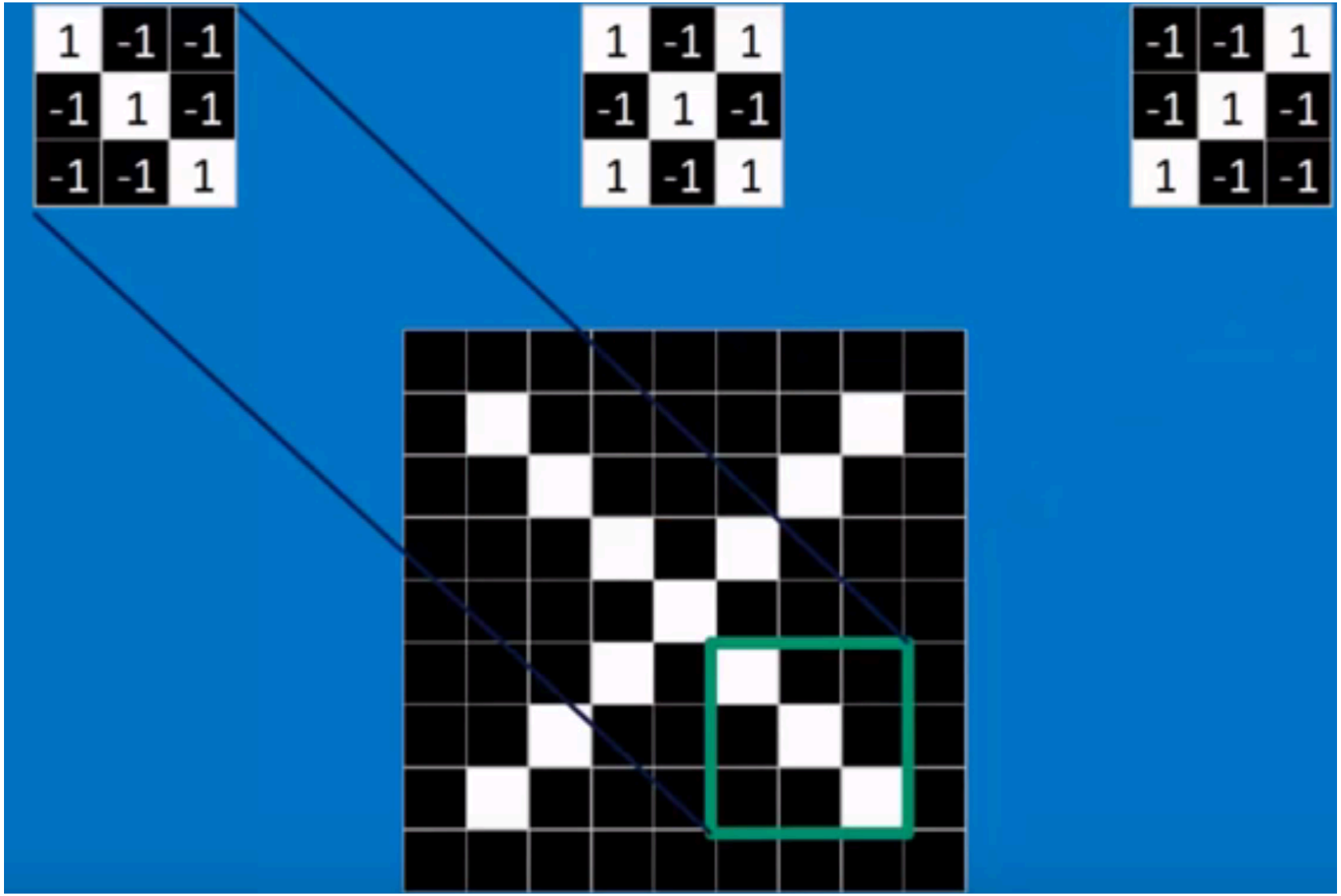
Features Match Pieces of the Image

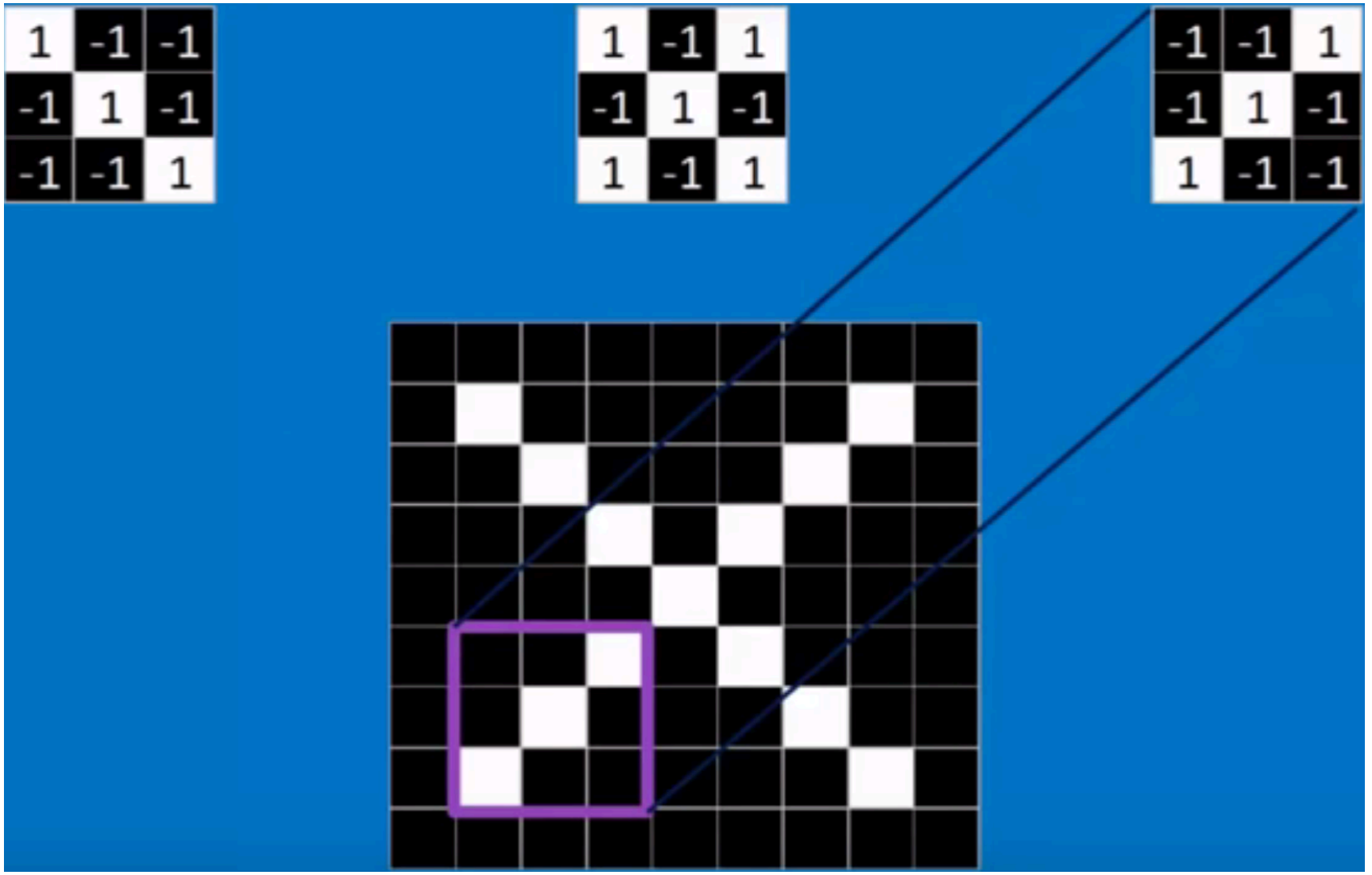



1 -1 -1
-1 1 -1
-1 -1 1
1 -1 1
-1 1 -1
1 -1 1
-1 -1 1
-1 1 -1
1 -1 -1

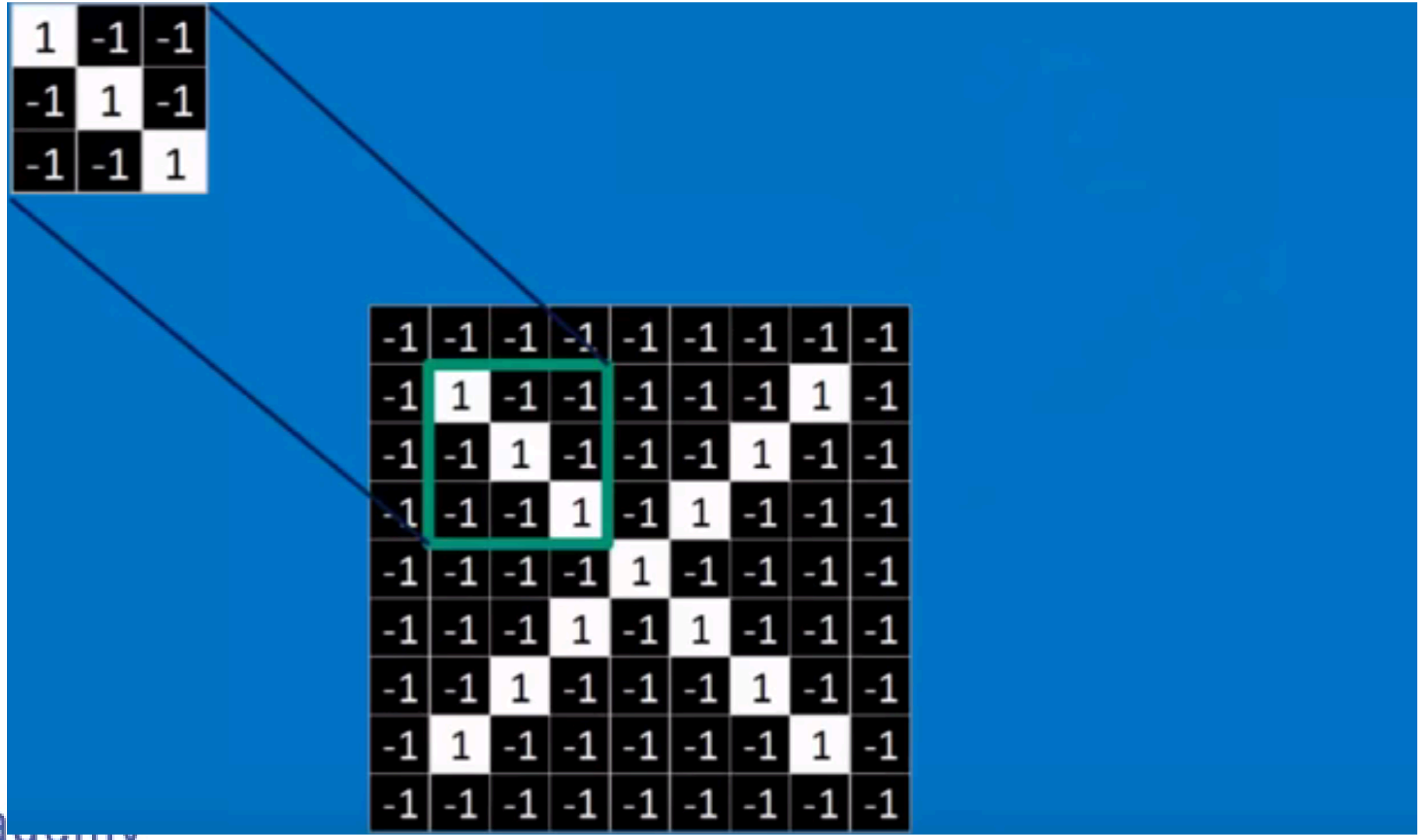

1 -1 -1
-1 1 -1
-1 -1 1
1 -1 1
-1 1 -1
1 -1 1
-1 -1 1
-1 1 -1
1 -1 -1


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


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


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

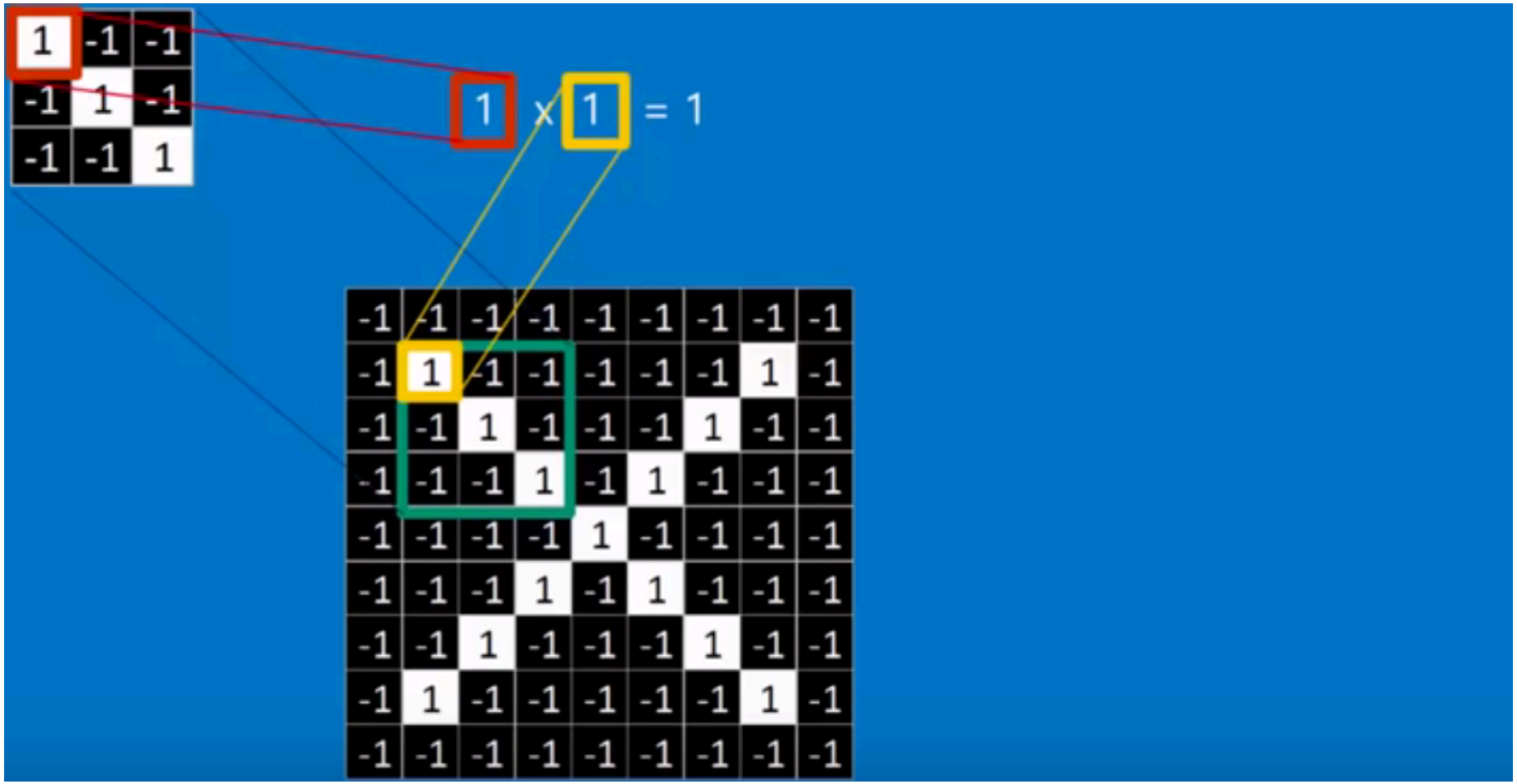
Filtering: The math behind the match



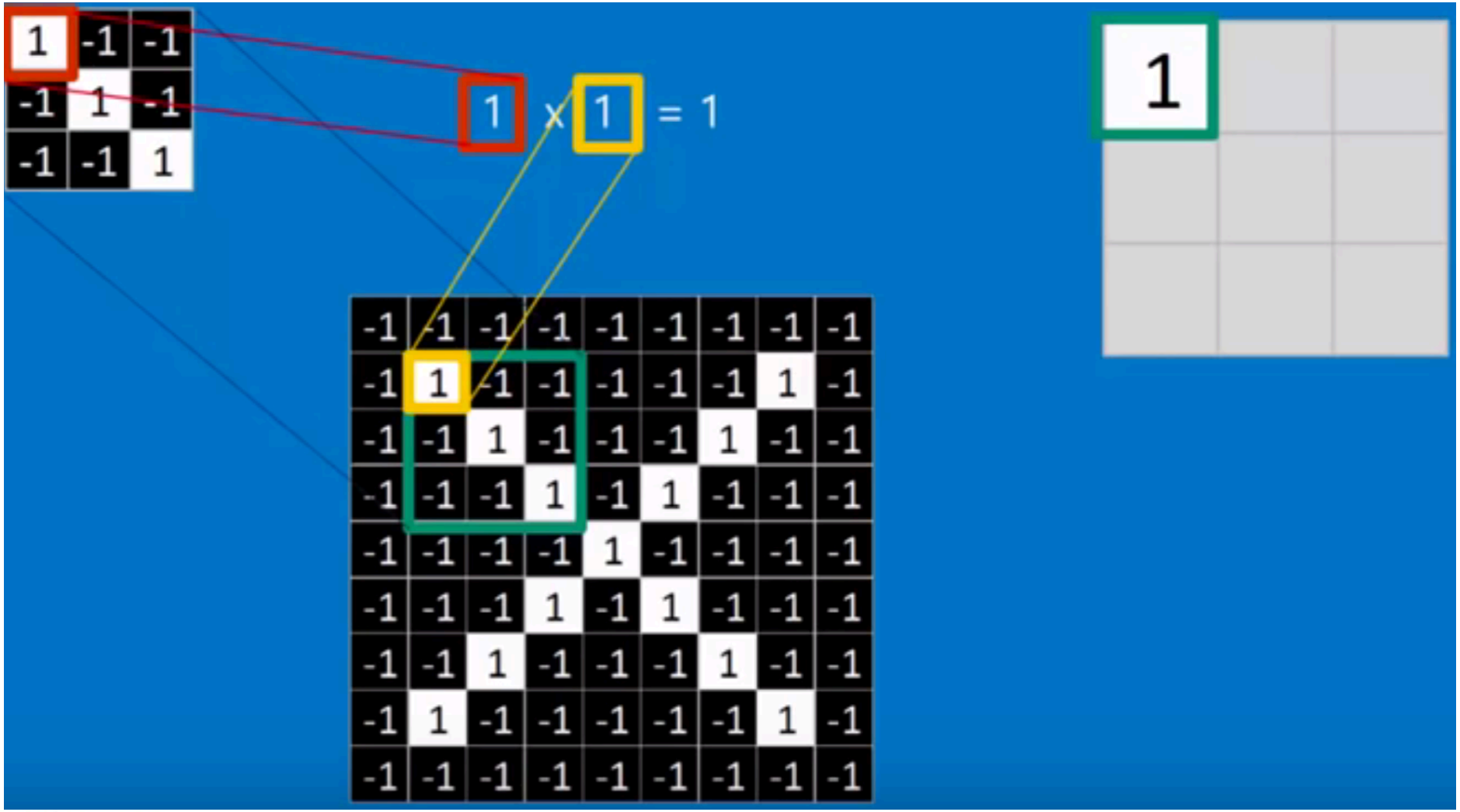
Filtering: The math behind the match

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

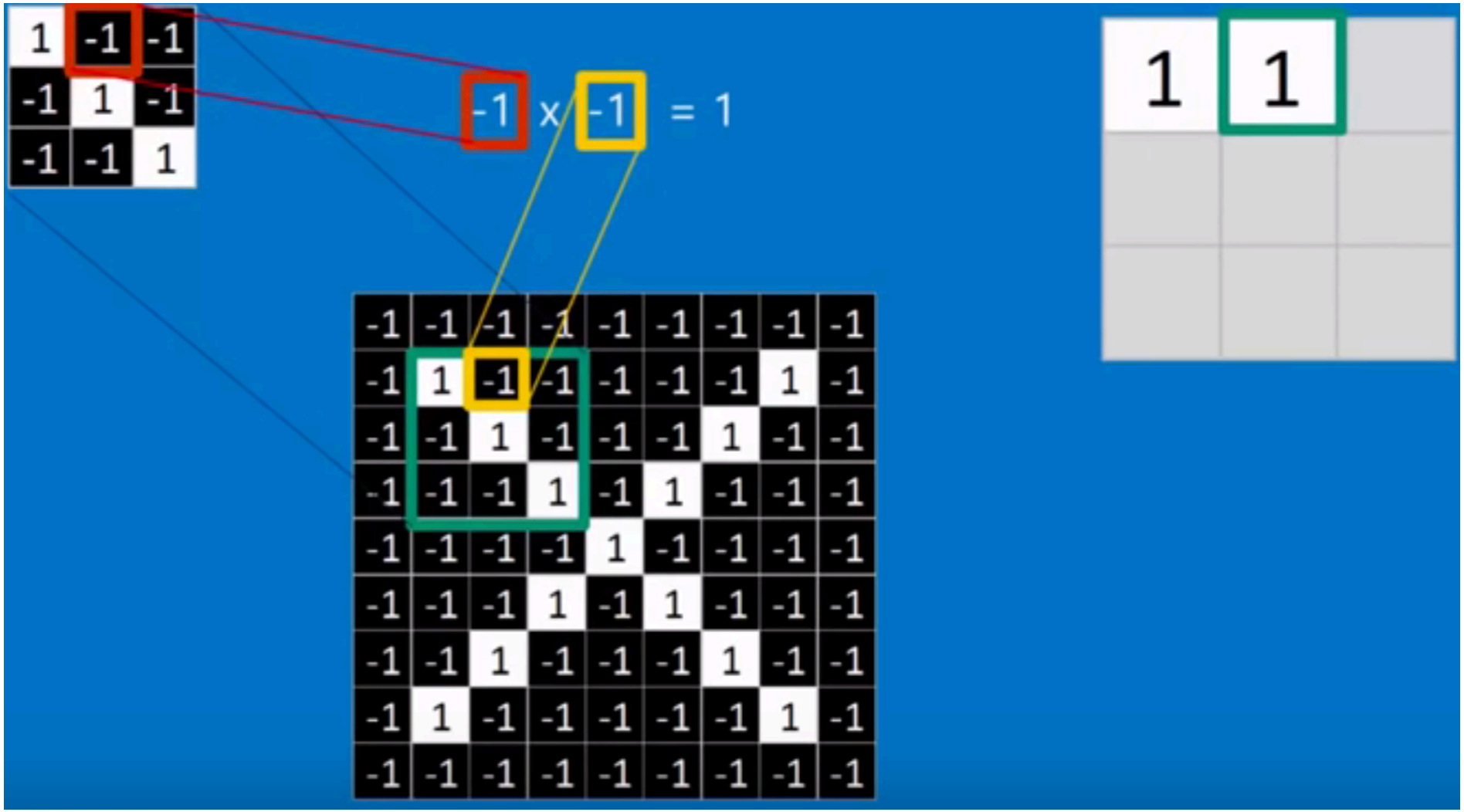
Filtering: The math behind the match

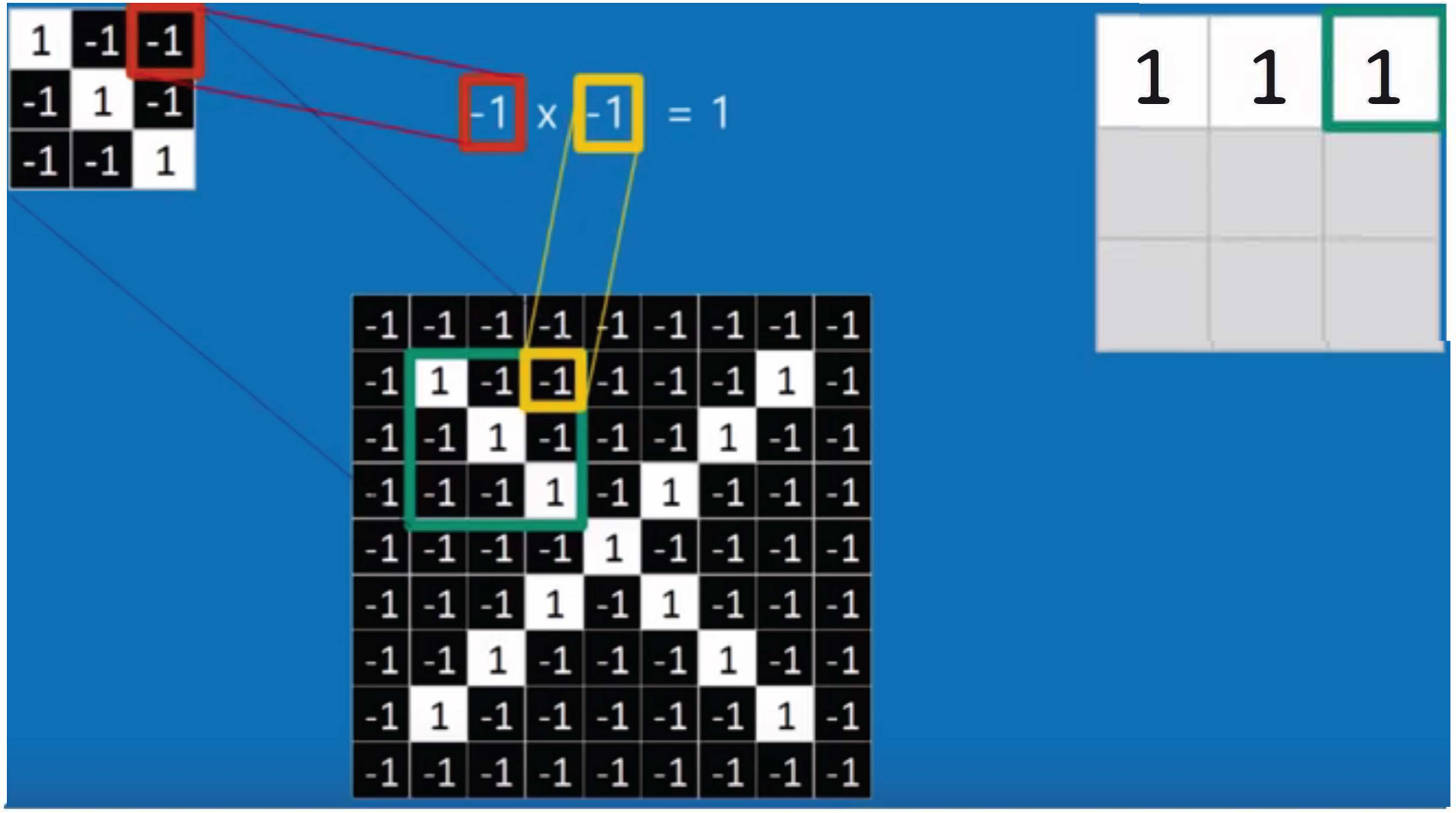


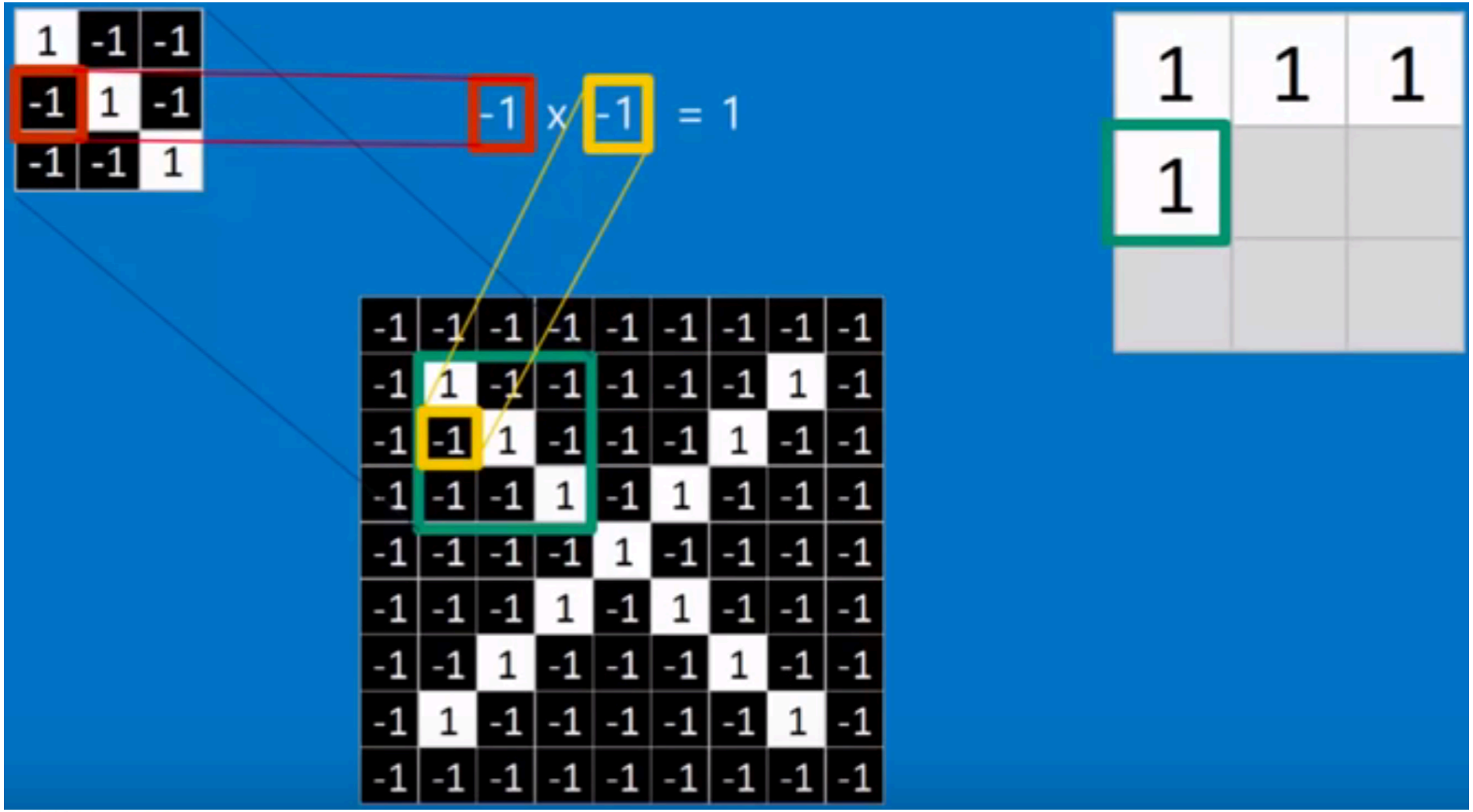
Filtering: The math behind the match

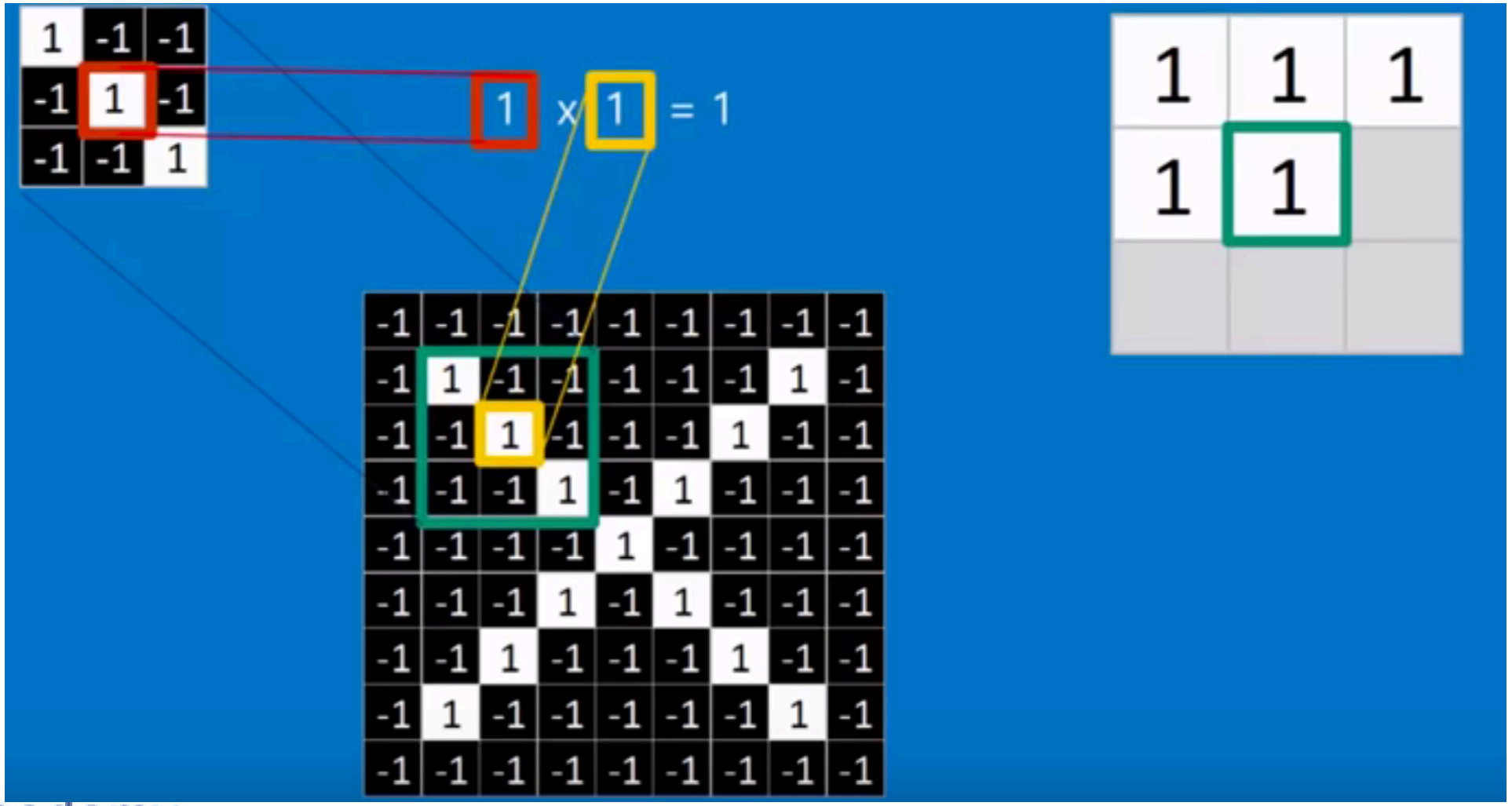


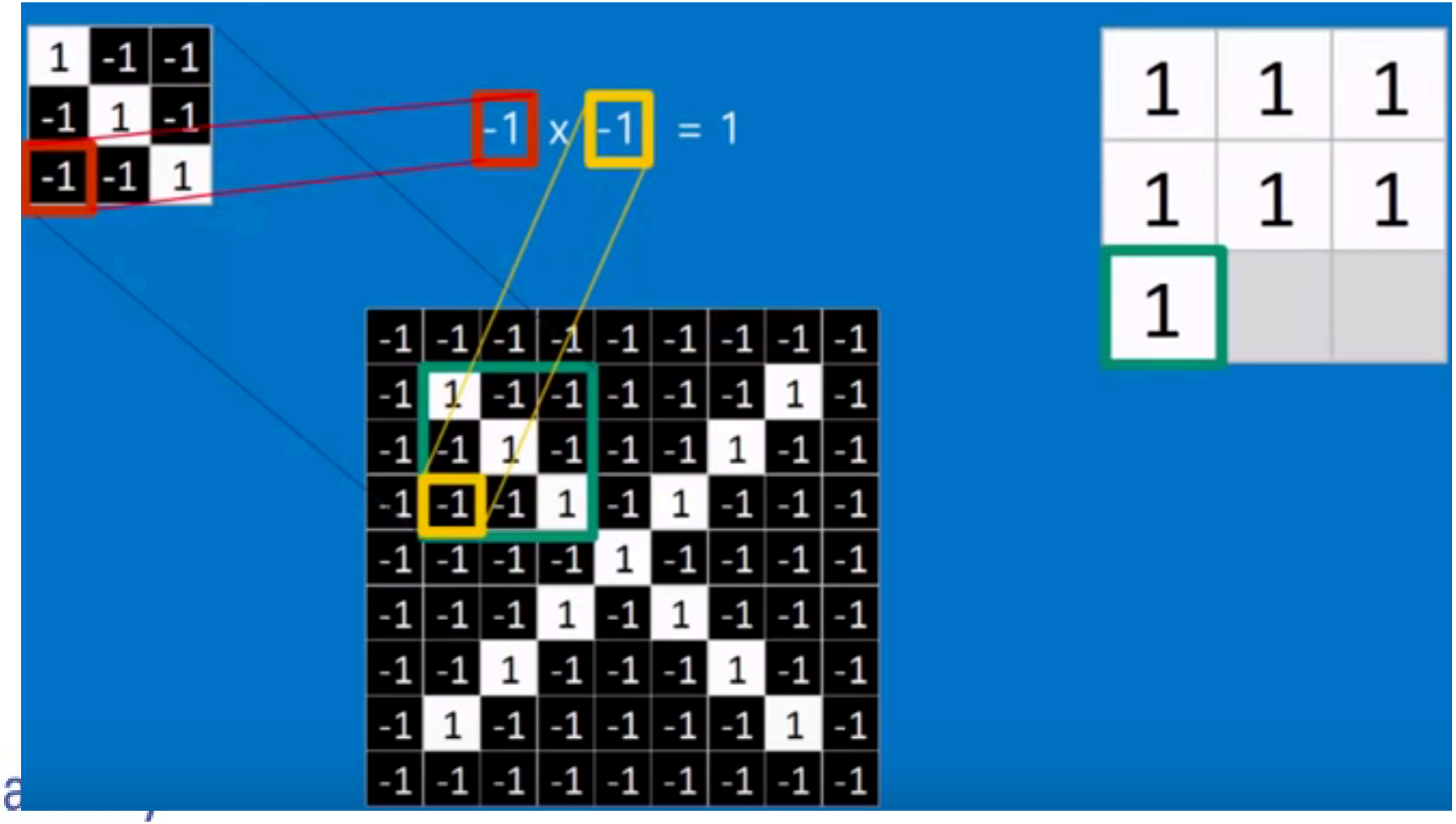
Filtering: The math behind the match

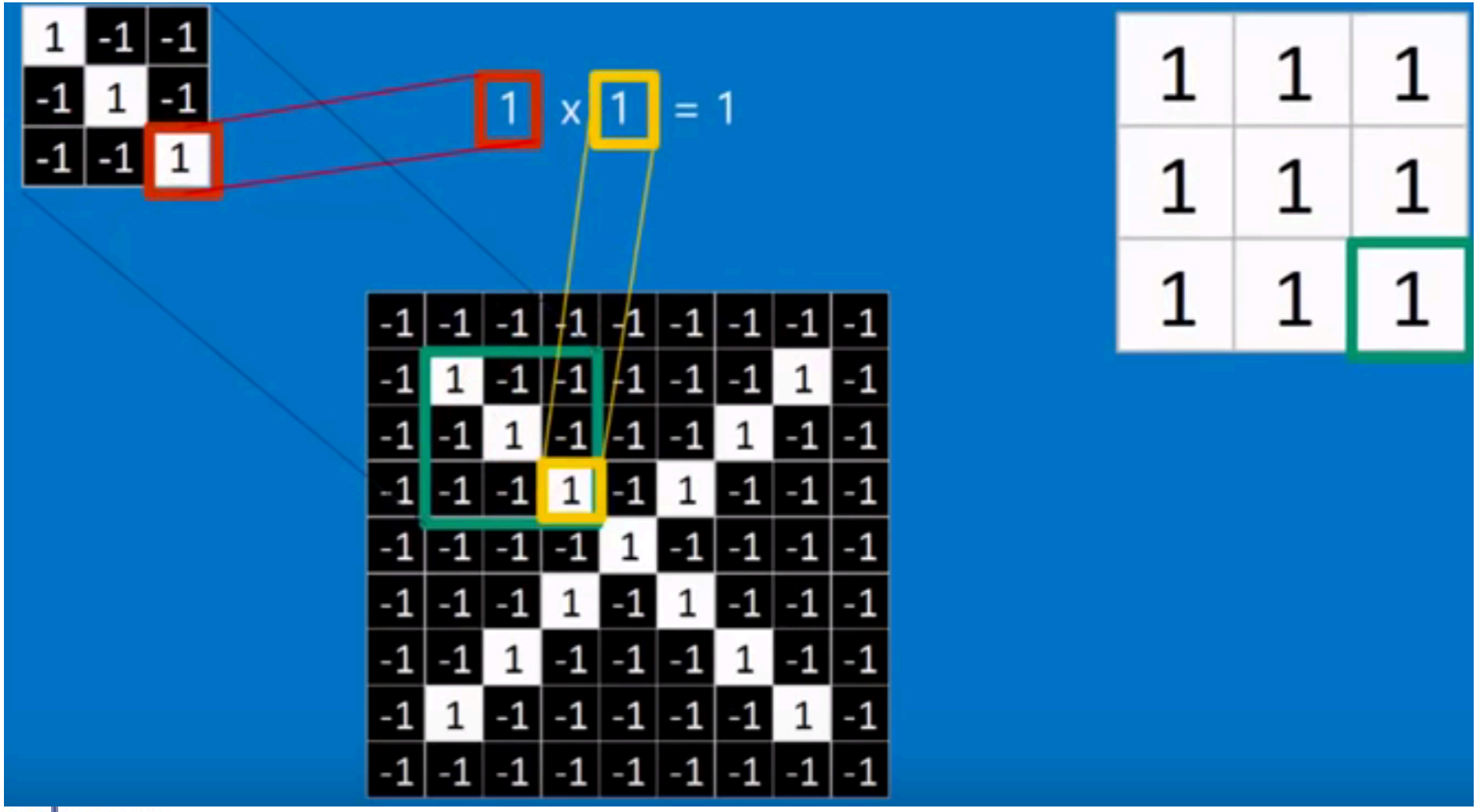


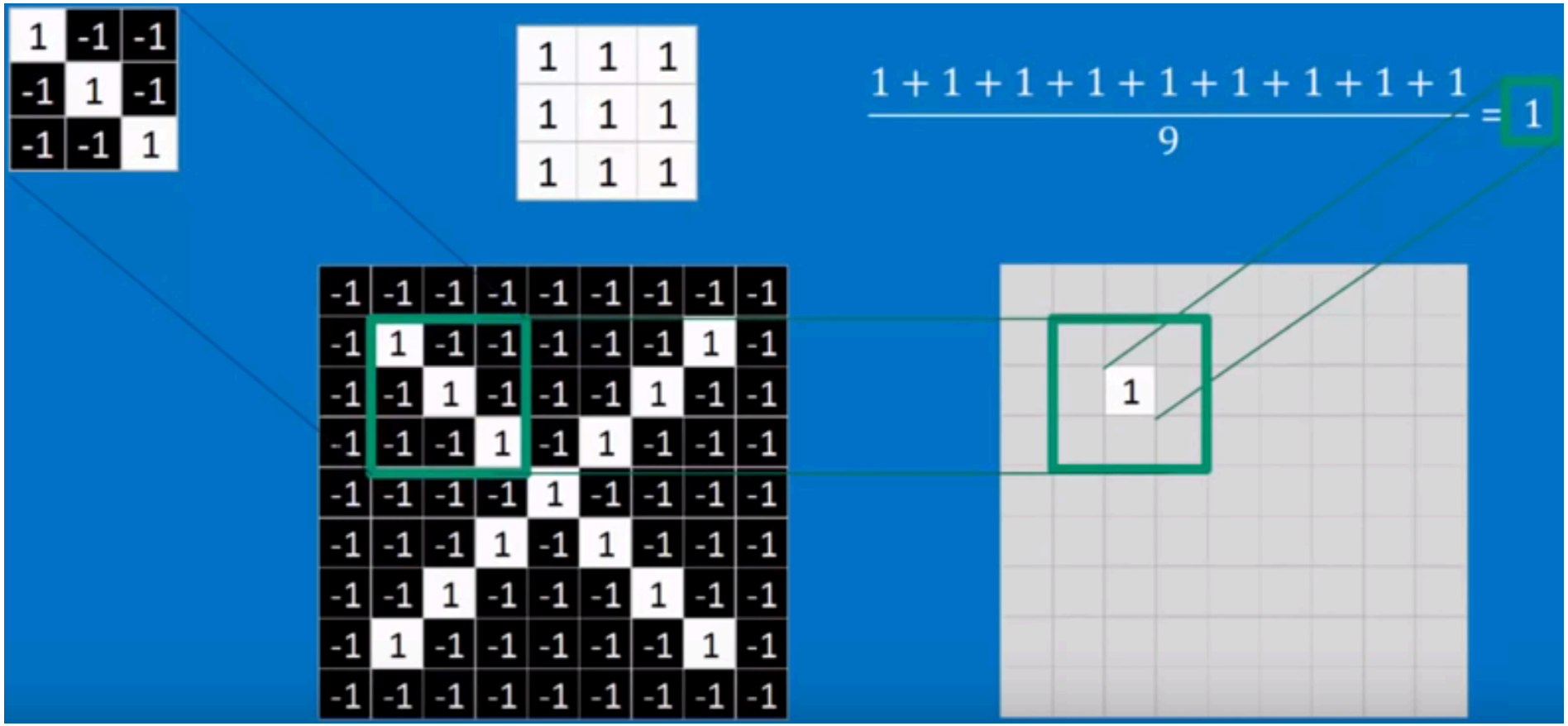

1 -1 -1
-1 1 -1
-1 -1 1
-1 x -1 = 1
1 1 1

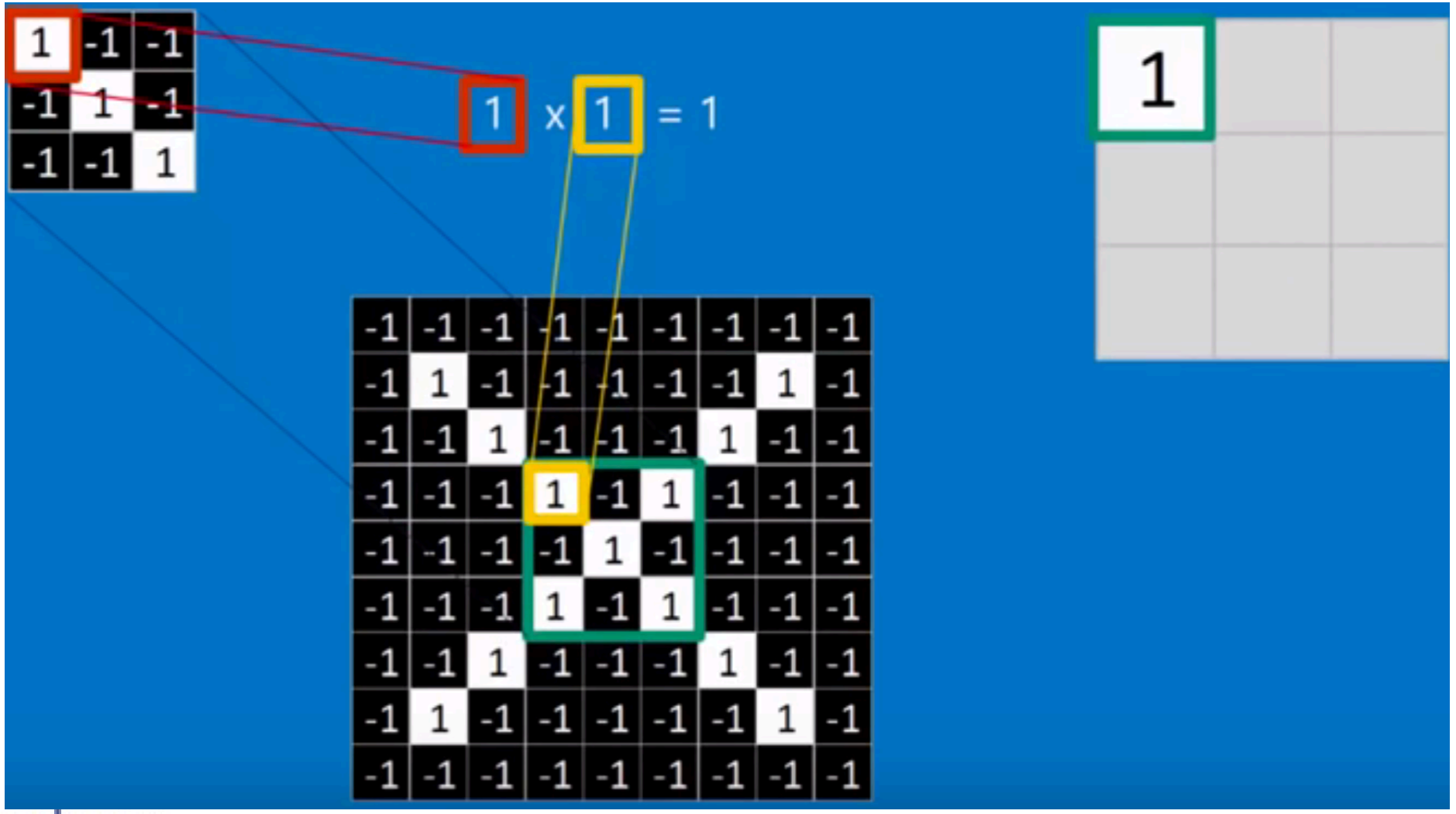

1 -1 -1
-1 1 -1
-1 -1 1
-1 x -1 = 1
1 1 1
1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

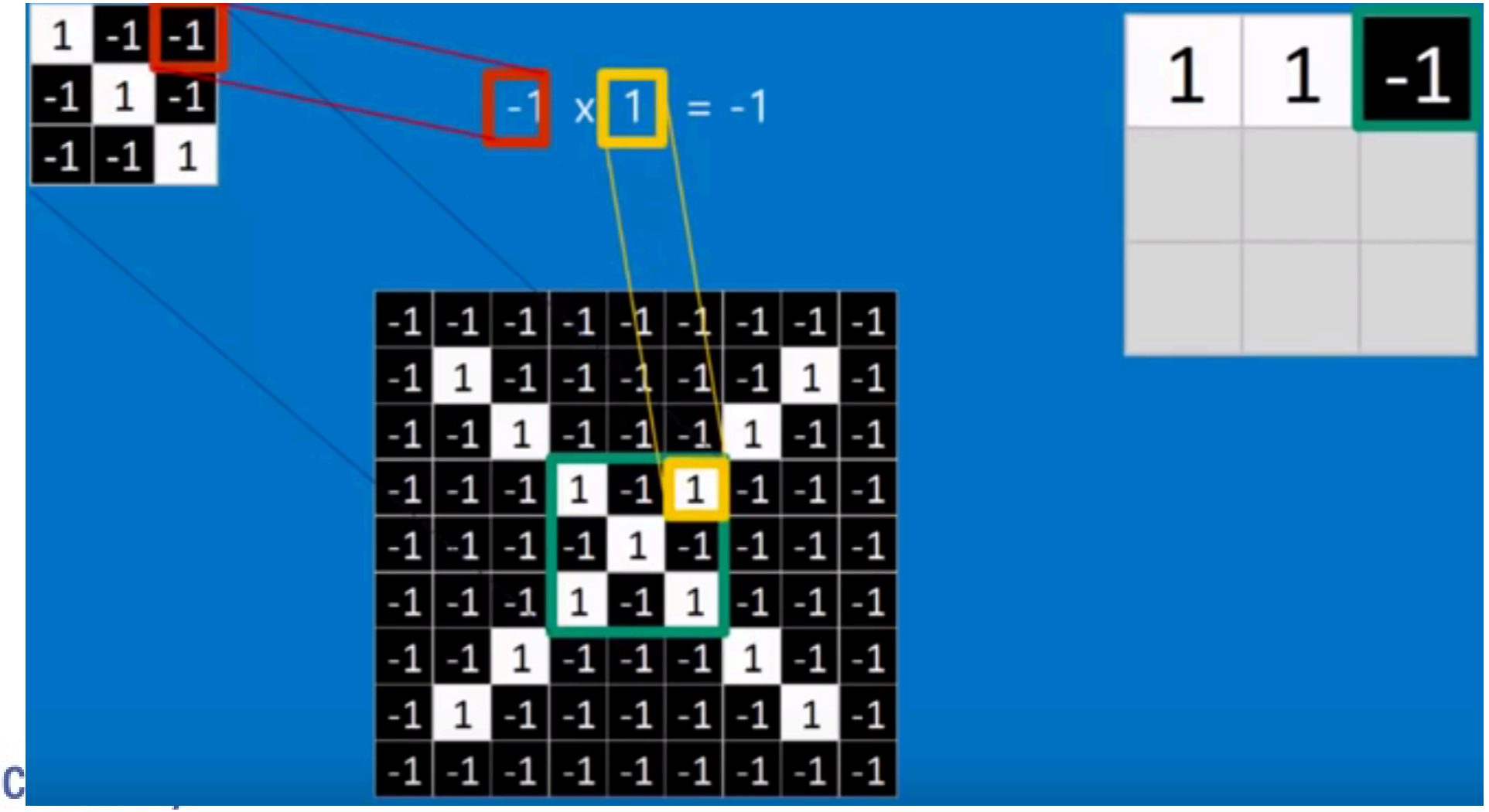

1 -1 -1
-1 1 -1
-1 -1 1
1 x 1 = 1
1 1 1
1 1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

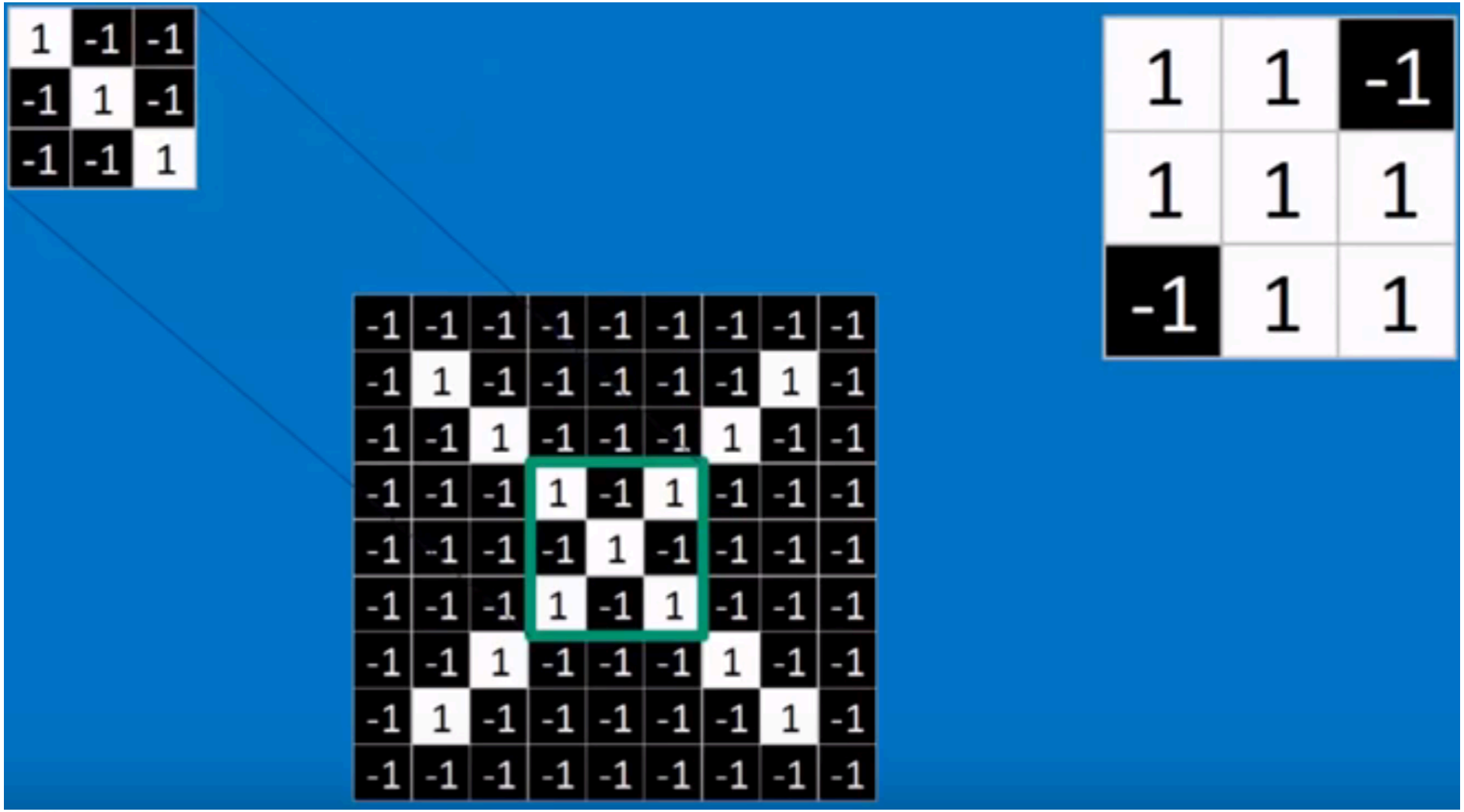

1 -1 -1
-1 1 -1
-1 -1 1
-1 x -1 = 1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
1 1 1
1 1 1
1

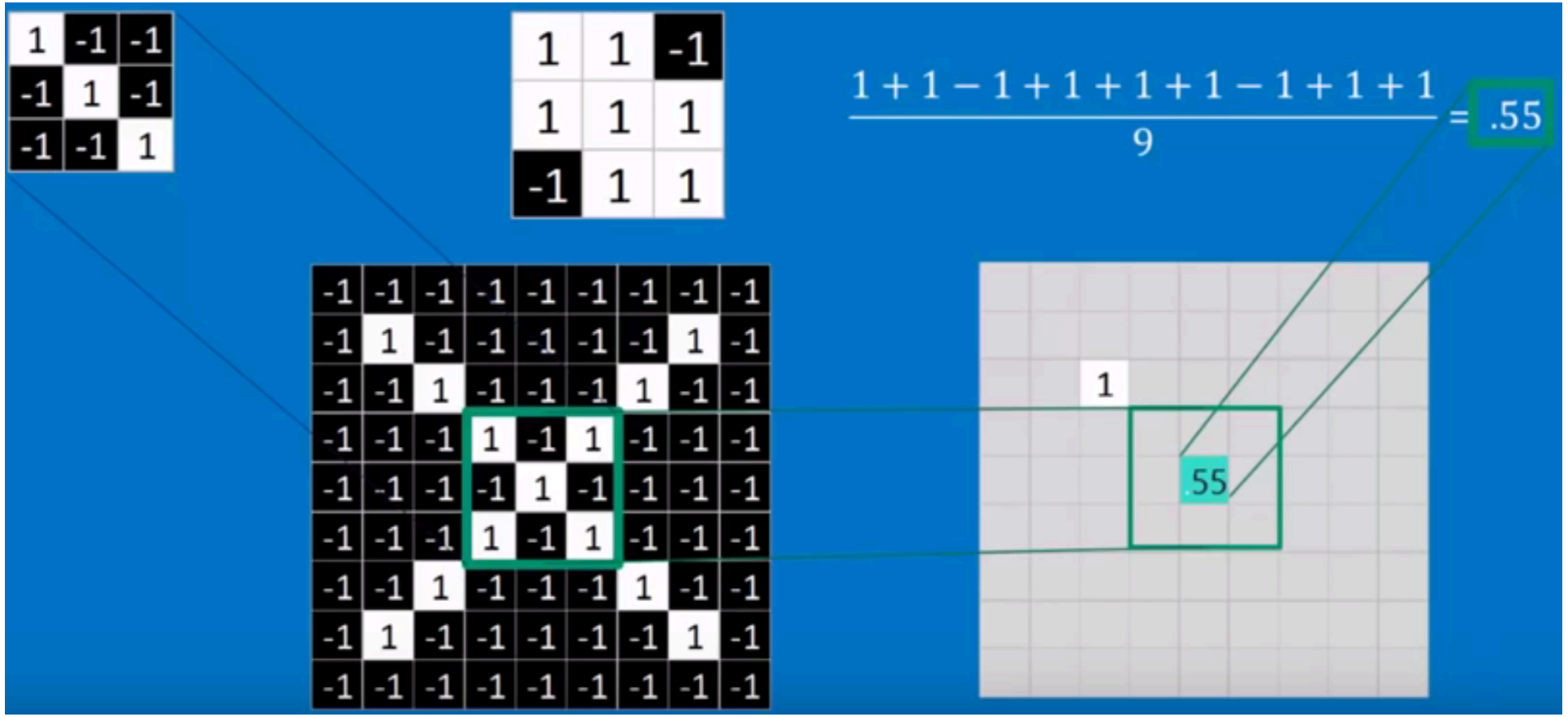

1 -1 -1
-1 1 -1
-1 -1 1
1 x 1 = 1
1 1 1
1 1 1
1 1 1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1


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

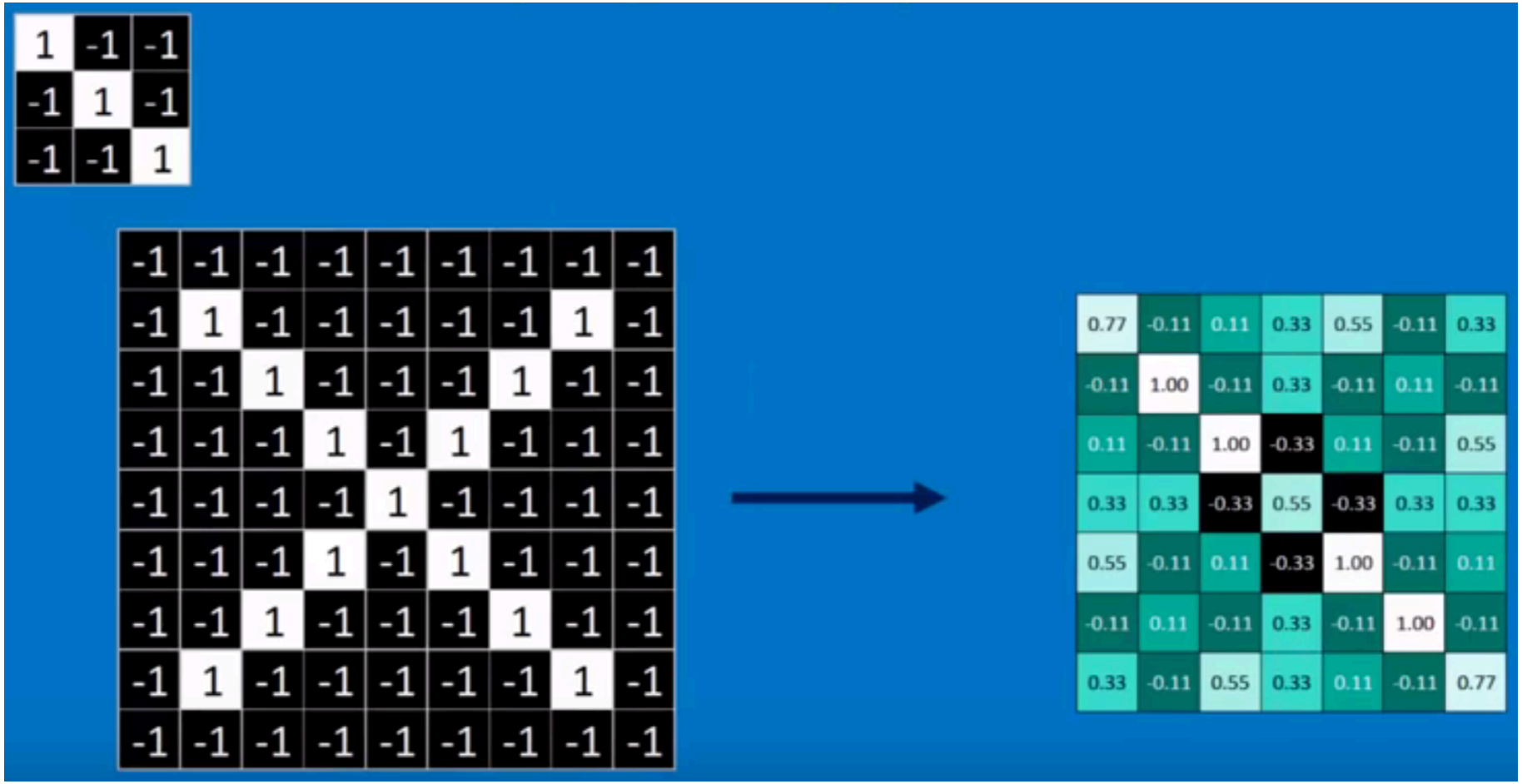

1 -1 -1
-1 1 -1
-1 -1 1
1 x 1 = 1
1


1 -1 -1
-1 1 -1
-1 -1 1
-1 x 1 = -1
1 1 -1

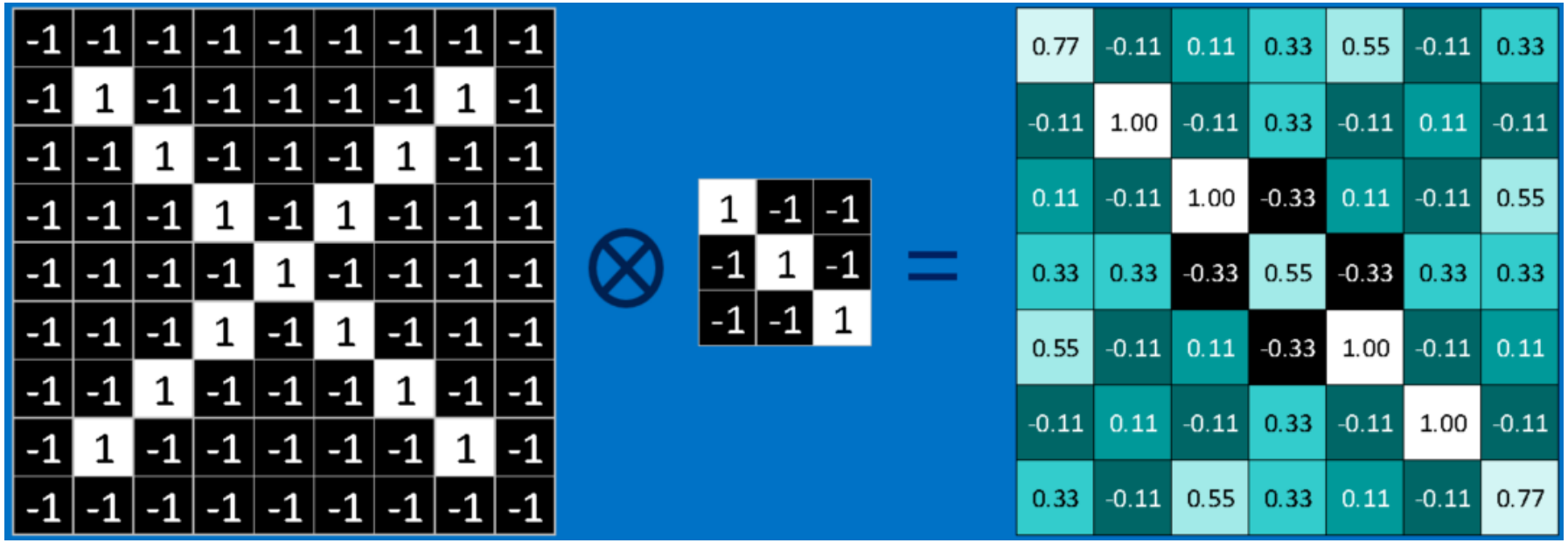

1 -1 -1
-1 1 -1
-1 -1 1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
1 1 -1
1 1 1
-1 1 1


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

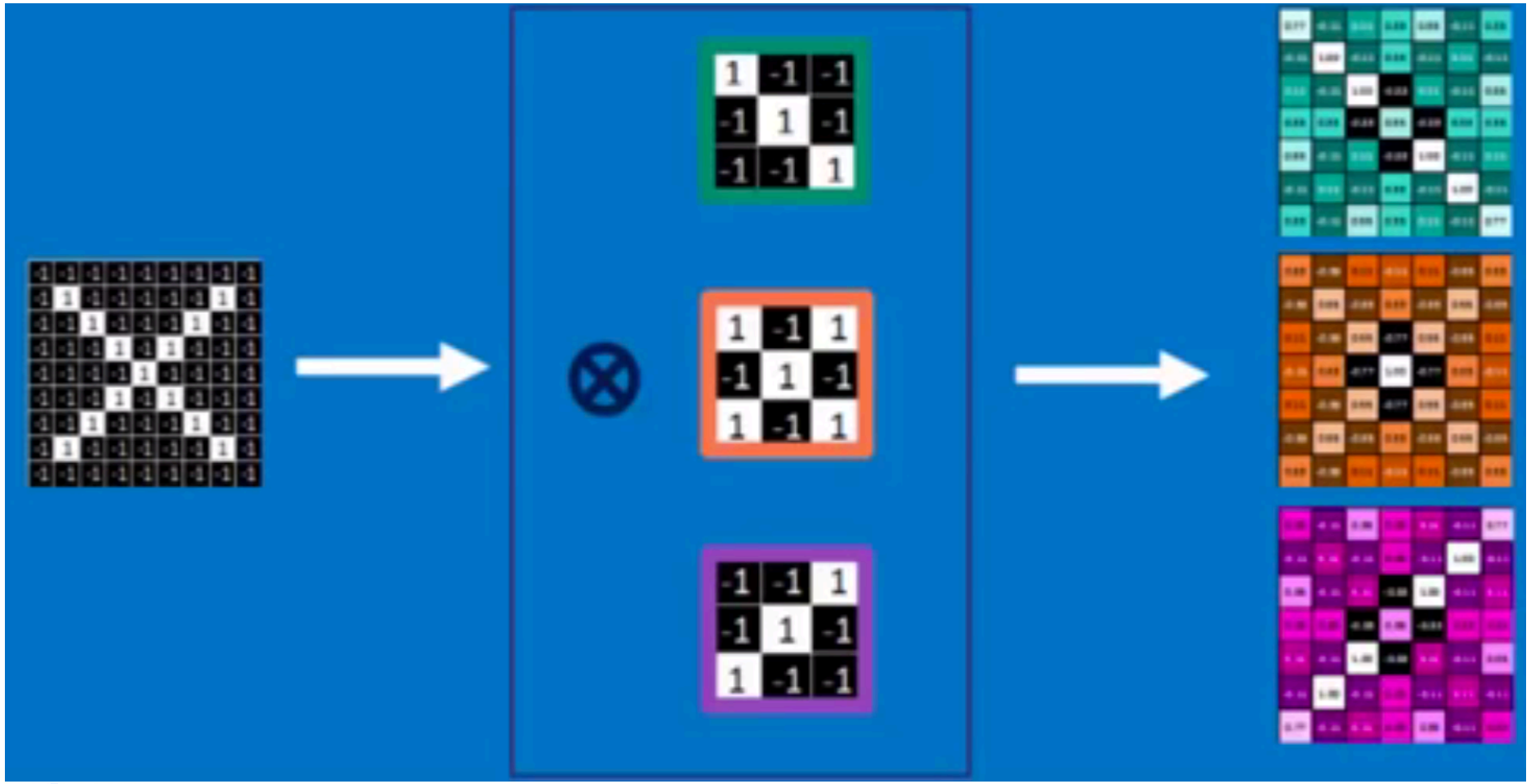
Convolution: Trying every possible match.



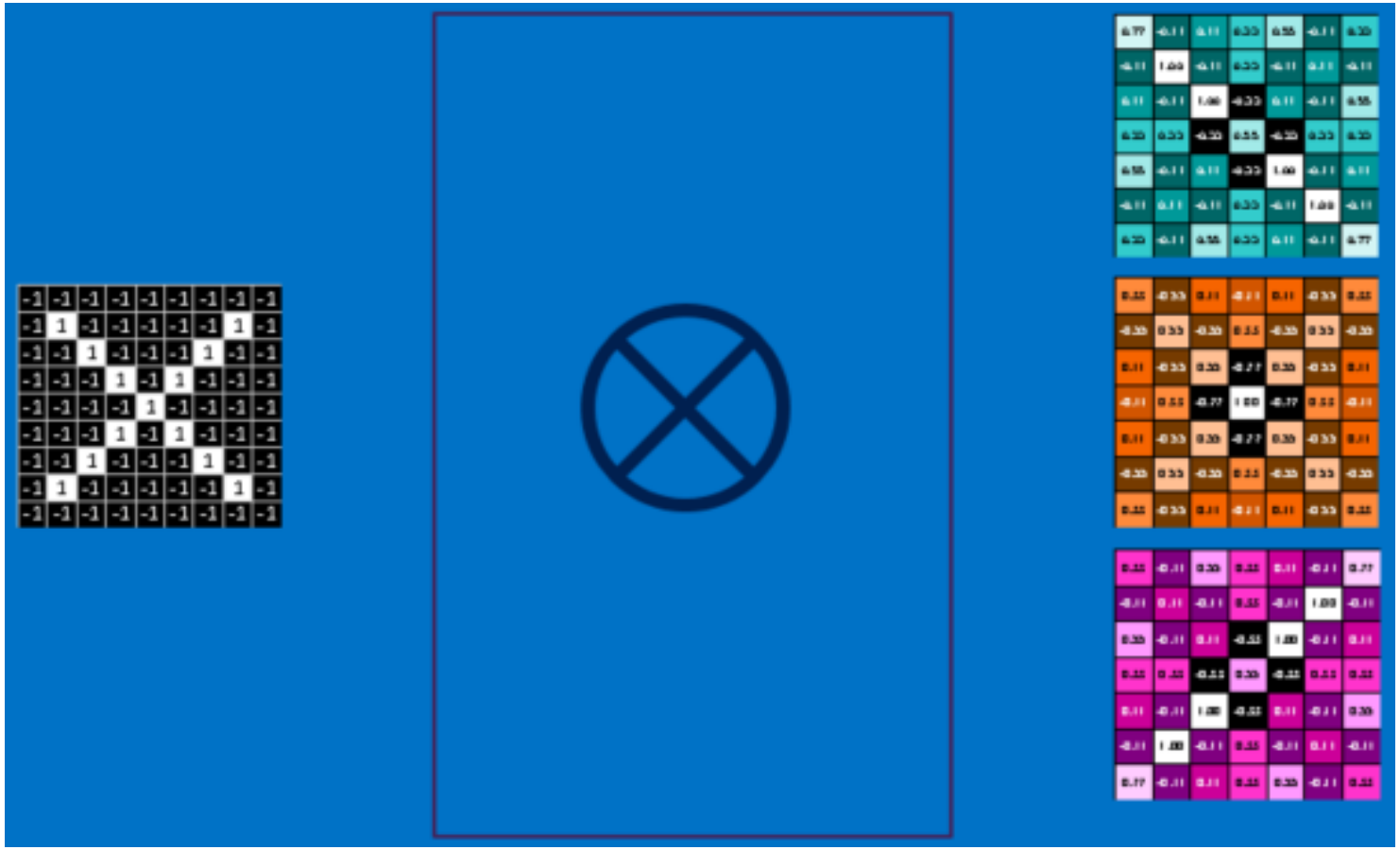
Convolution: Trying every possible match.



Convolution Layer: one image becomes a stack of filtered images



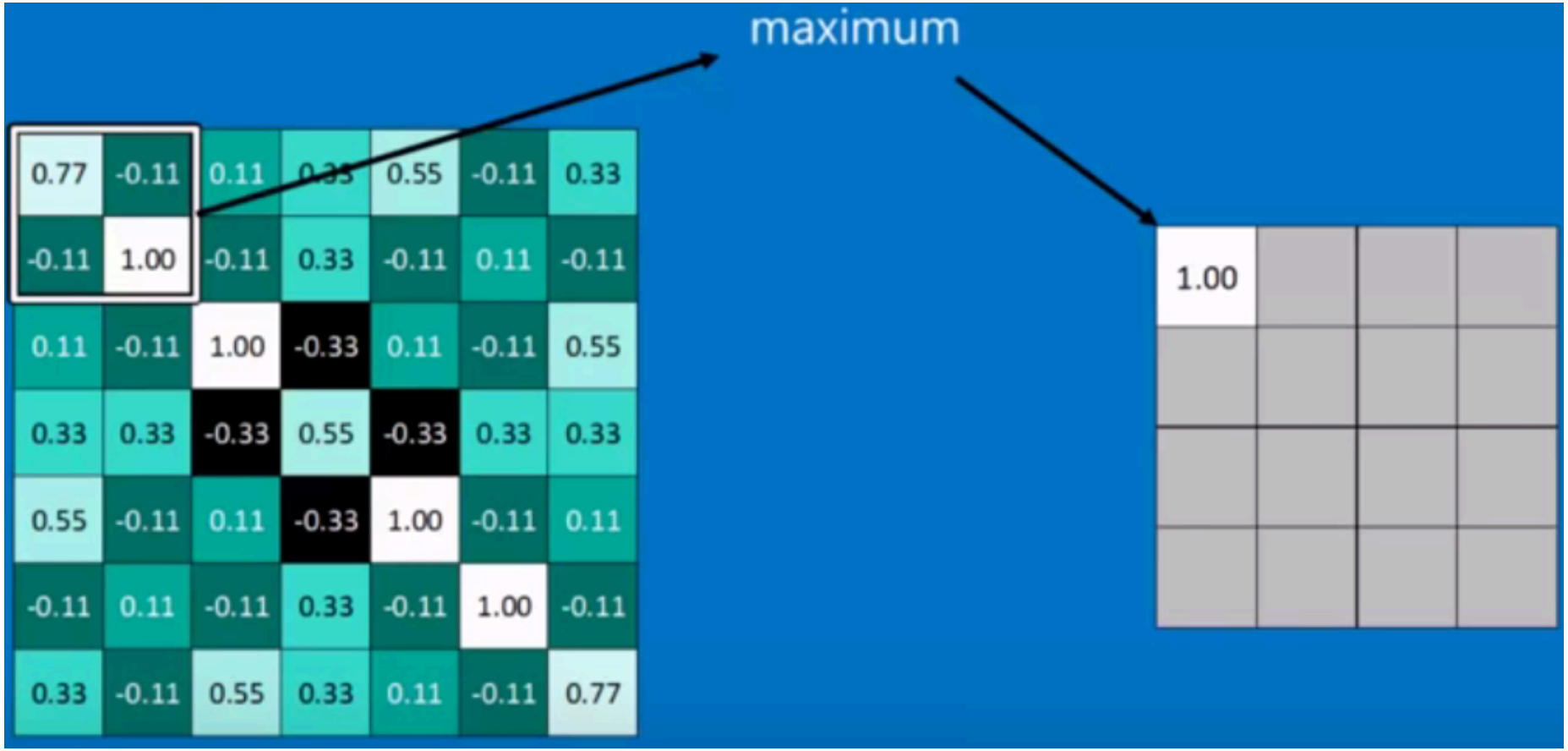
Convolution Layer: one image becomes a stack of filtered images



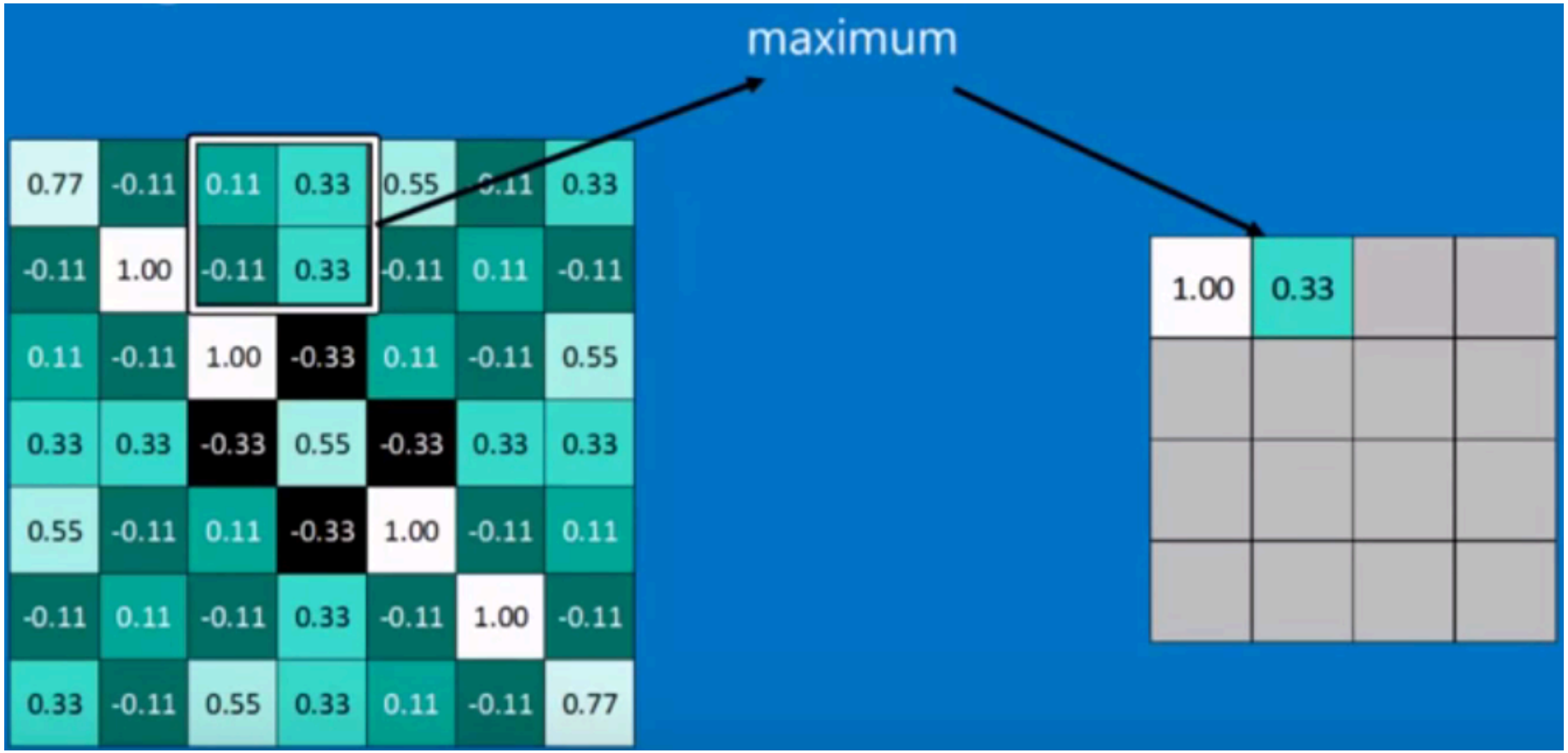
Pooling: Shrinking the Image Stack

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

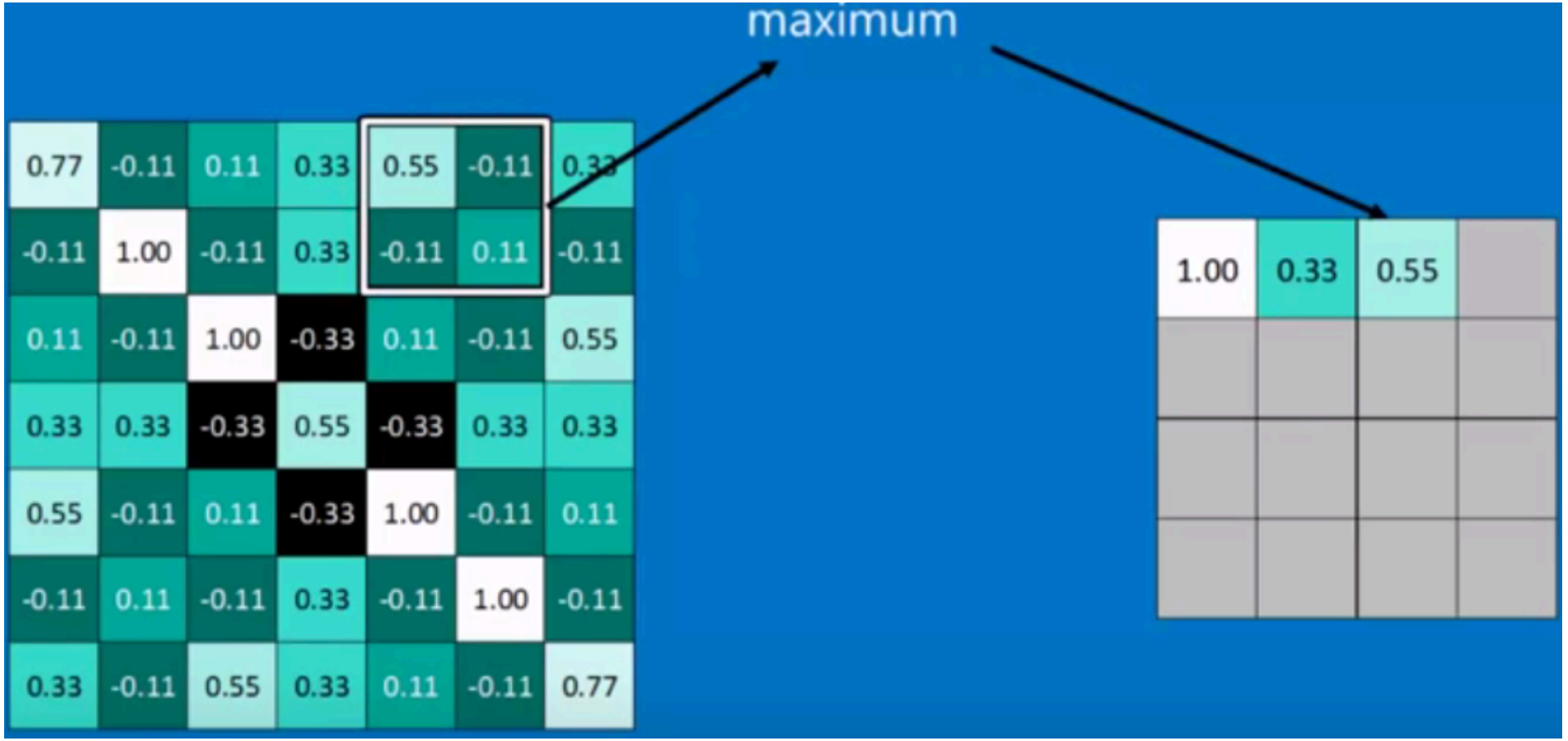
Pooling: Shrinking the Image Stack



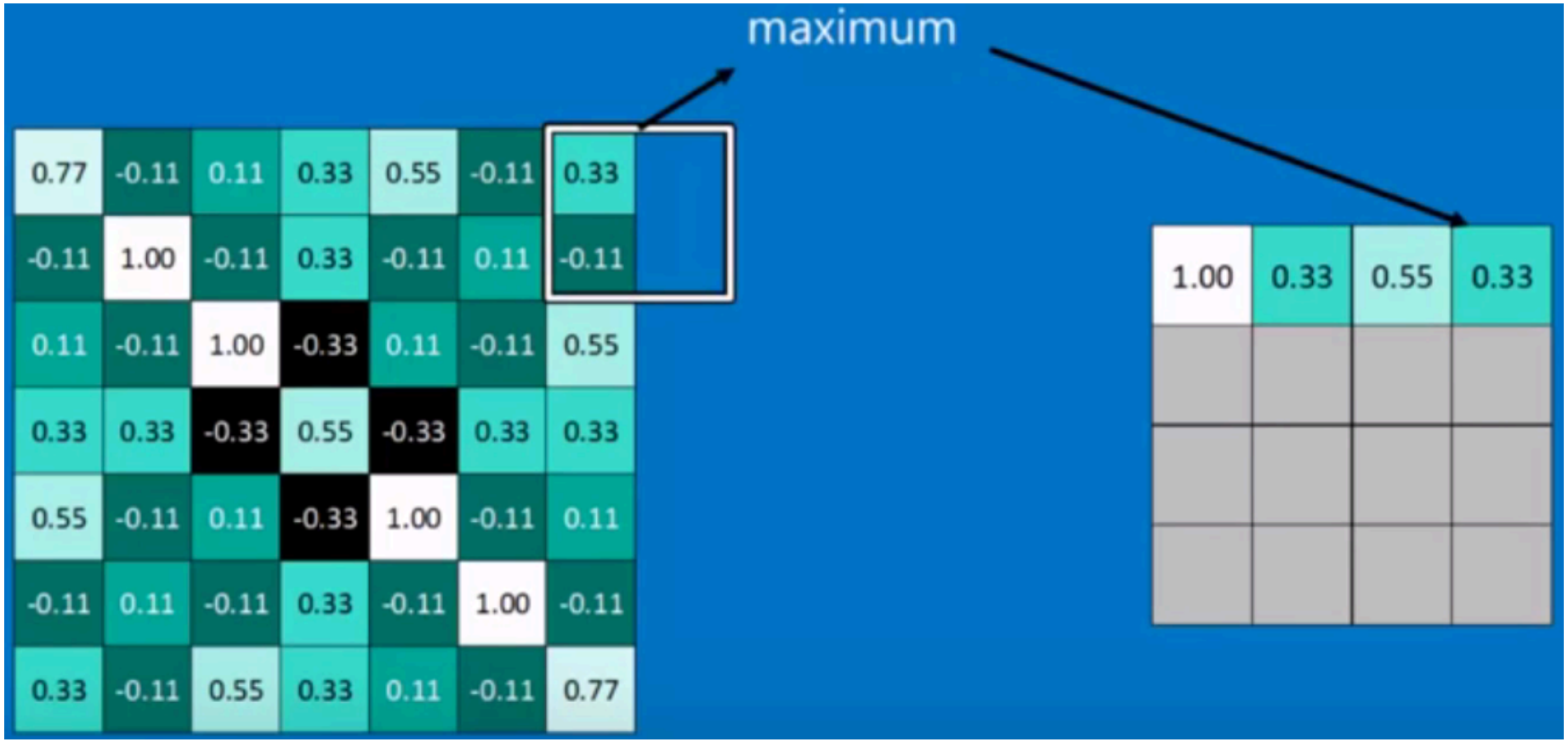
Pooling: Shrinking the Image Stack



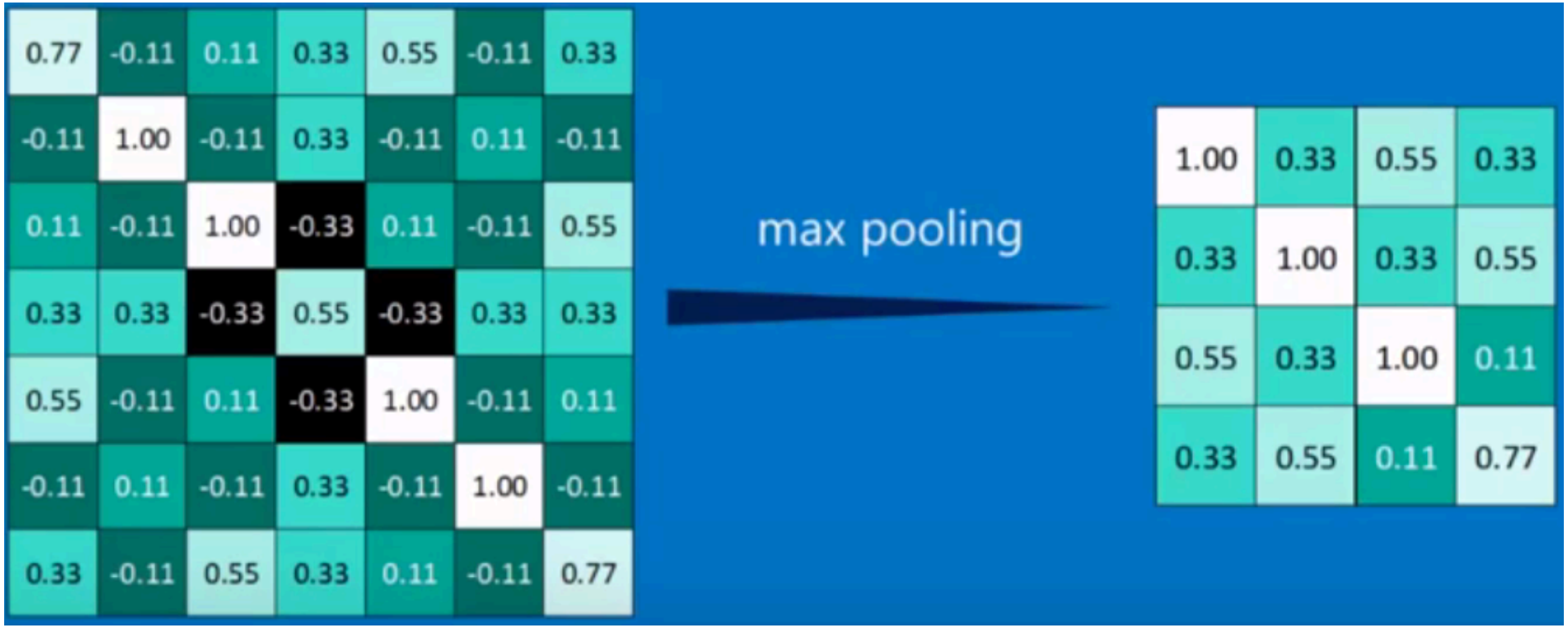
Pooling: Shrinking the Image Stack

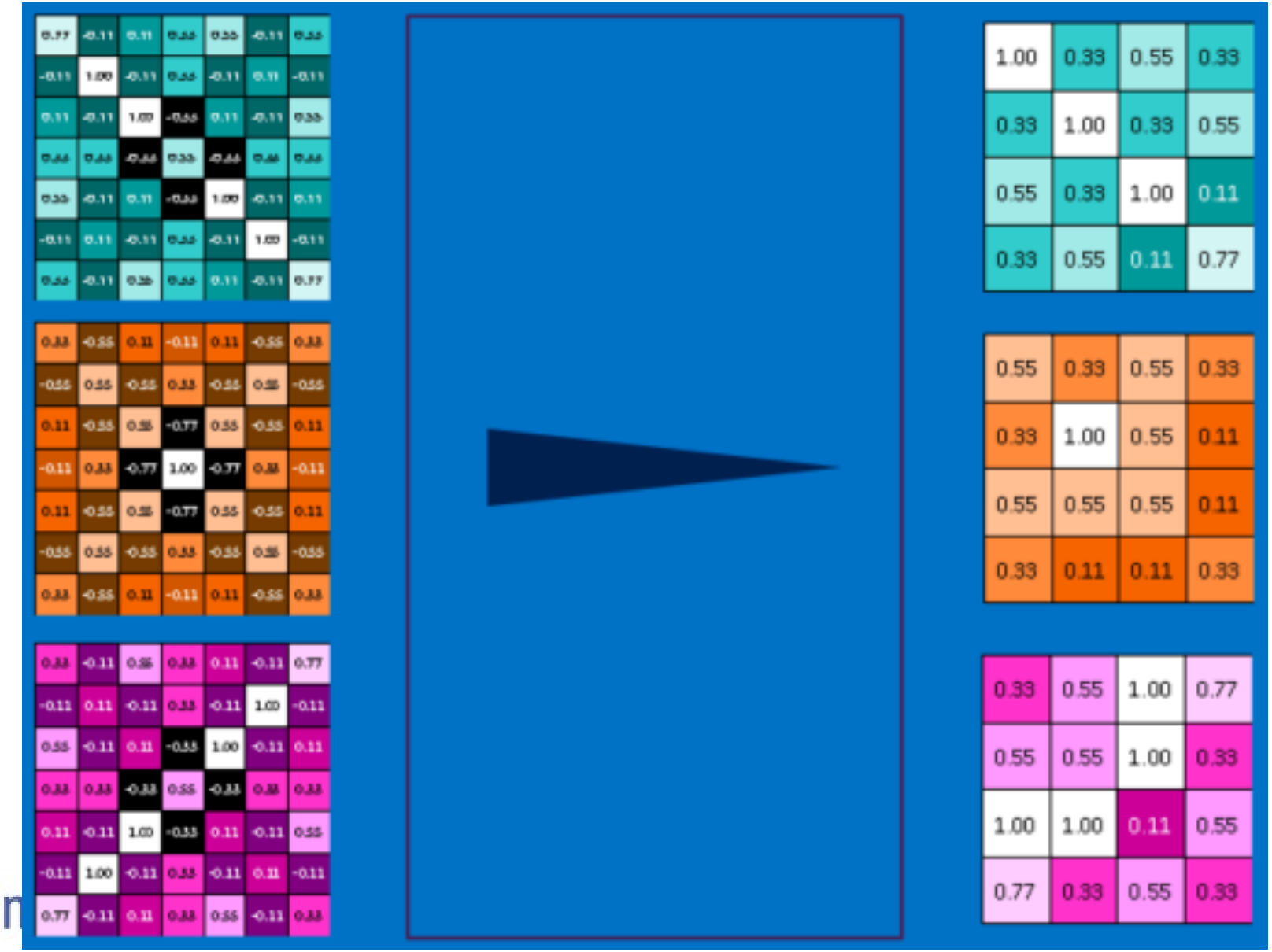


Pooling: Shrinking the Image Stack

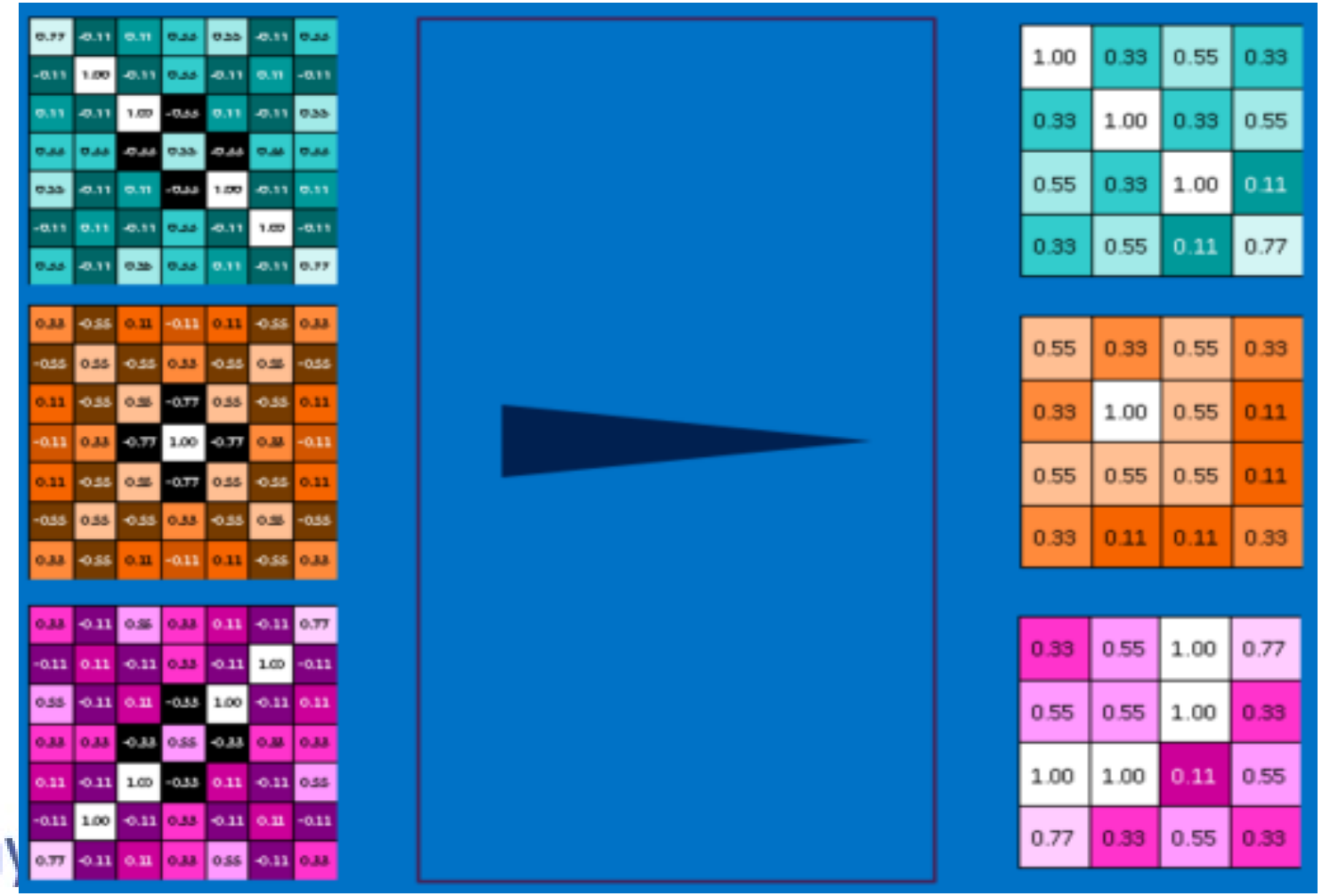


Pooling: Shrinking the Image Stack



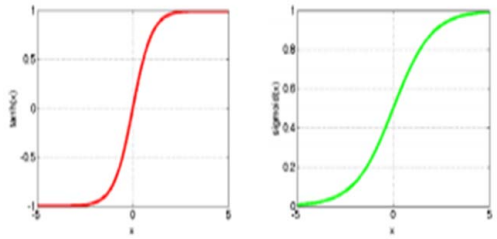

1.00 0.33 0.55 0.33
0.33 1.00 0.33 0.55
0.55 0.33 1.00 0.11
0.33 0.55 0.11 0.77
0.55 0.33 0.55 0.33
0.33 1.00 0.55 0.11
0.55 0.55 0.55 0.11
0.33 0.11 0.11 0.33
0.33 0.55 1.00 0.77
0.55 0.55 1.00 0.33
1.00 1.00 0.11 0.55
0.77 0.33 0.55 0.33

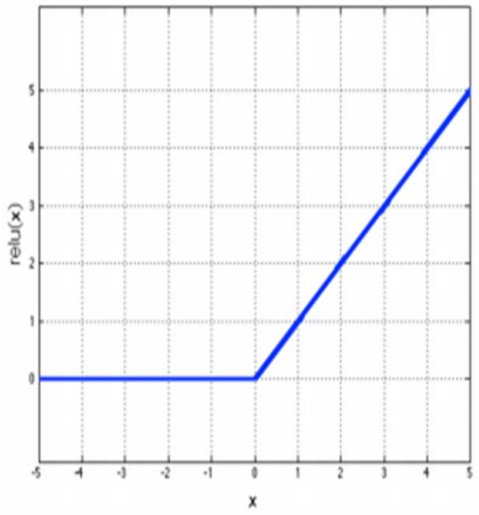
Pooling Layer: A stack of images becomes a stack of smaller Images.



Nonlinerity

- Similar to NN, we need to introduce nonlinearity in CNN
  - Sigmoid
  - Tanh
  - RELU: Rectified Linear Units ->
    - Simplifies backpropagation
    - Makes learning faster
    - Avoids saturation issues

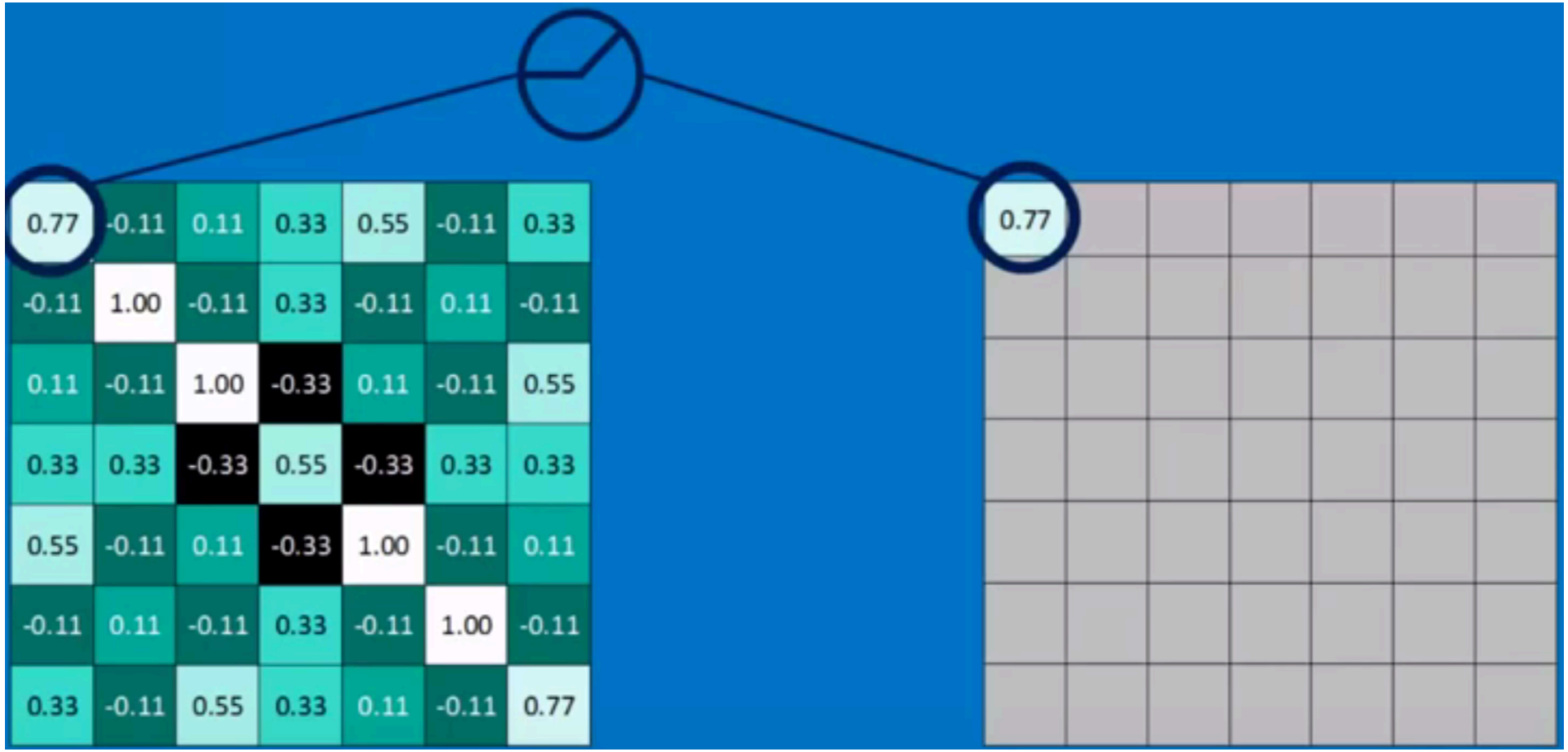




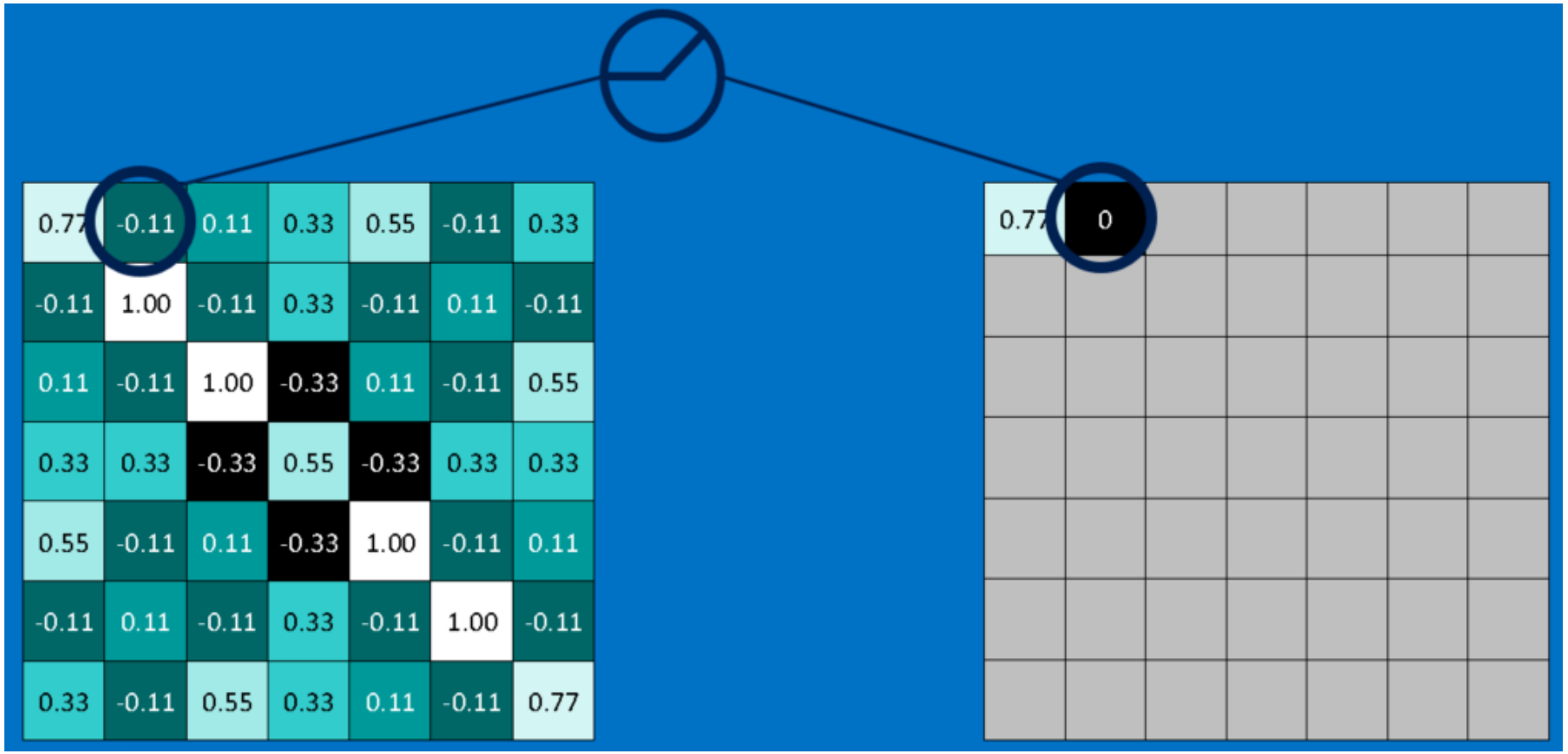
Nonlinerity

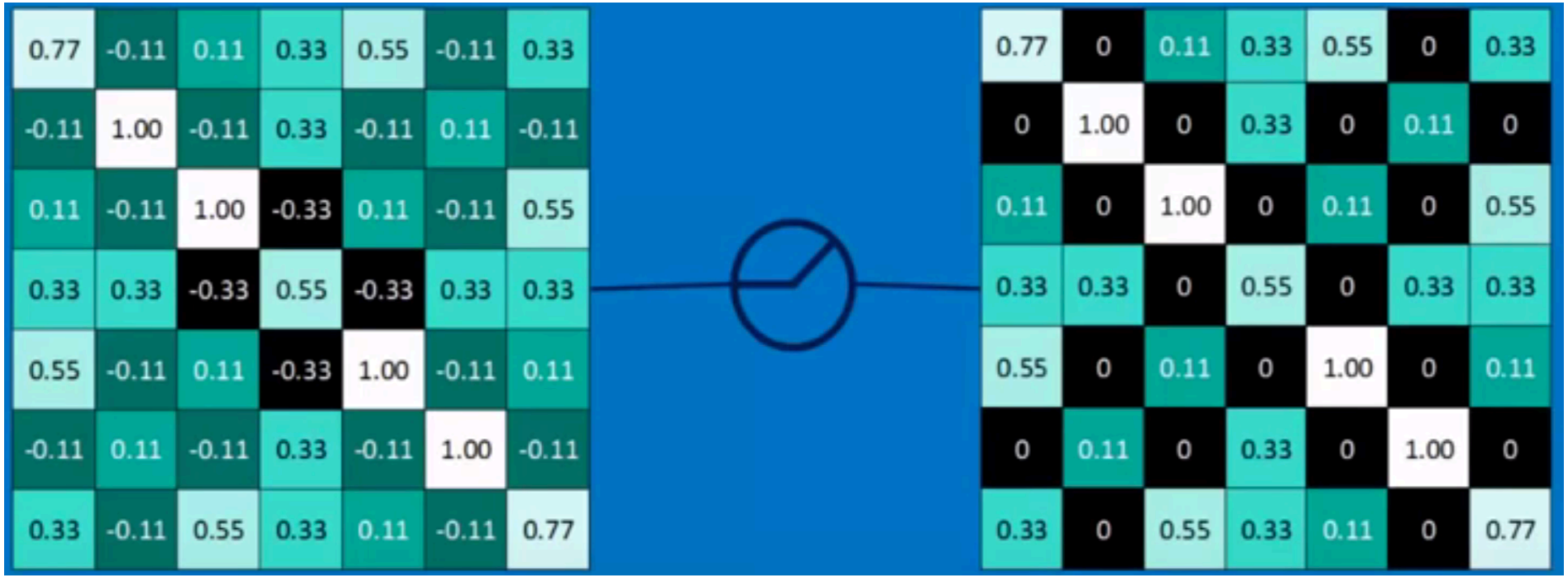
1. Keep the math from breaking by tweaking each of the values just a bit.
2. Change everything negative to be 0.

Rectified Linear Unites

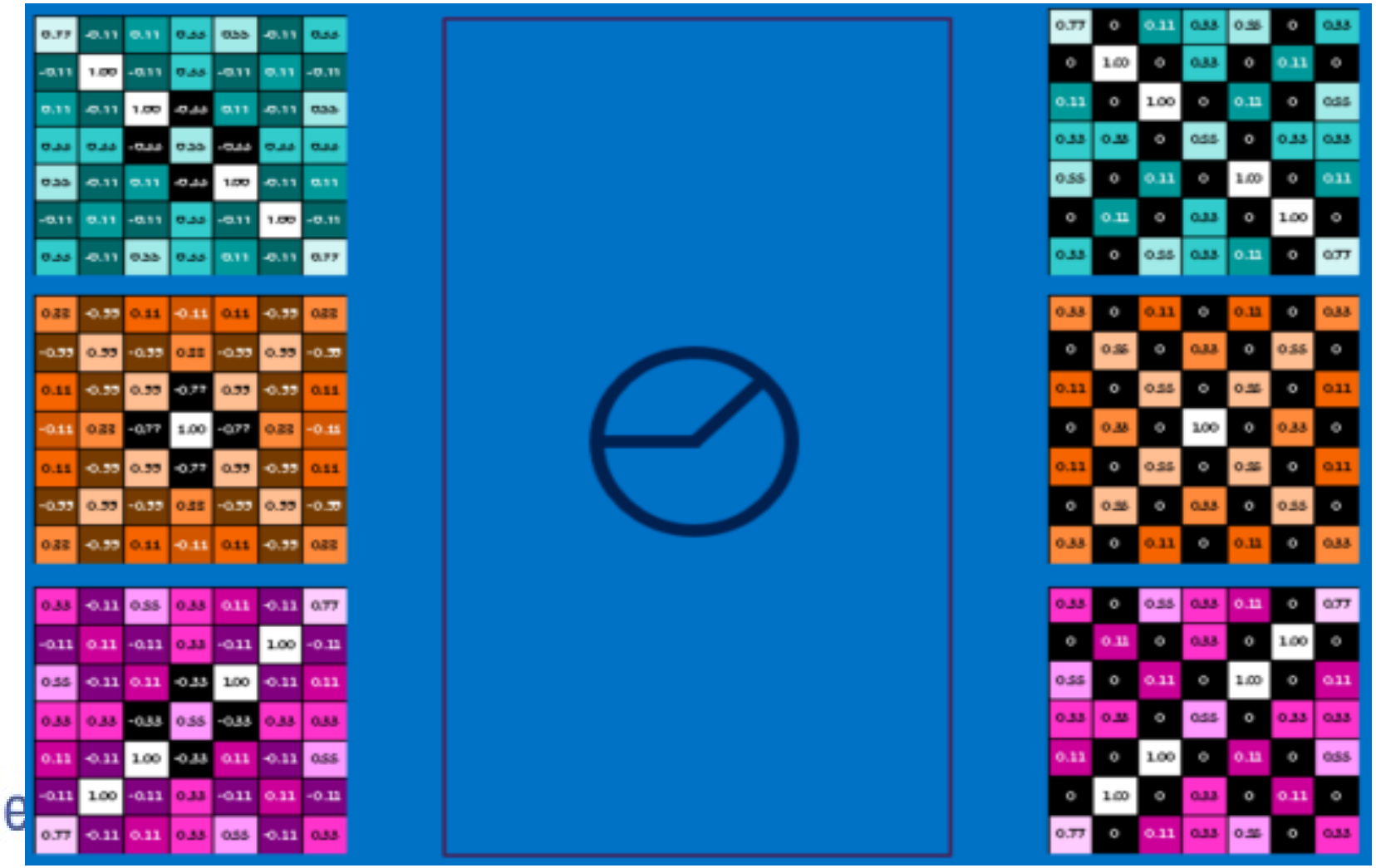


Rectified Linear Unites

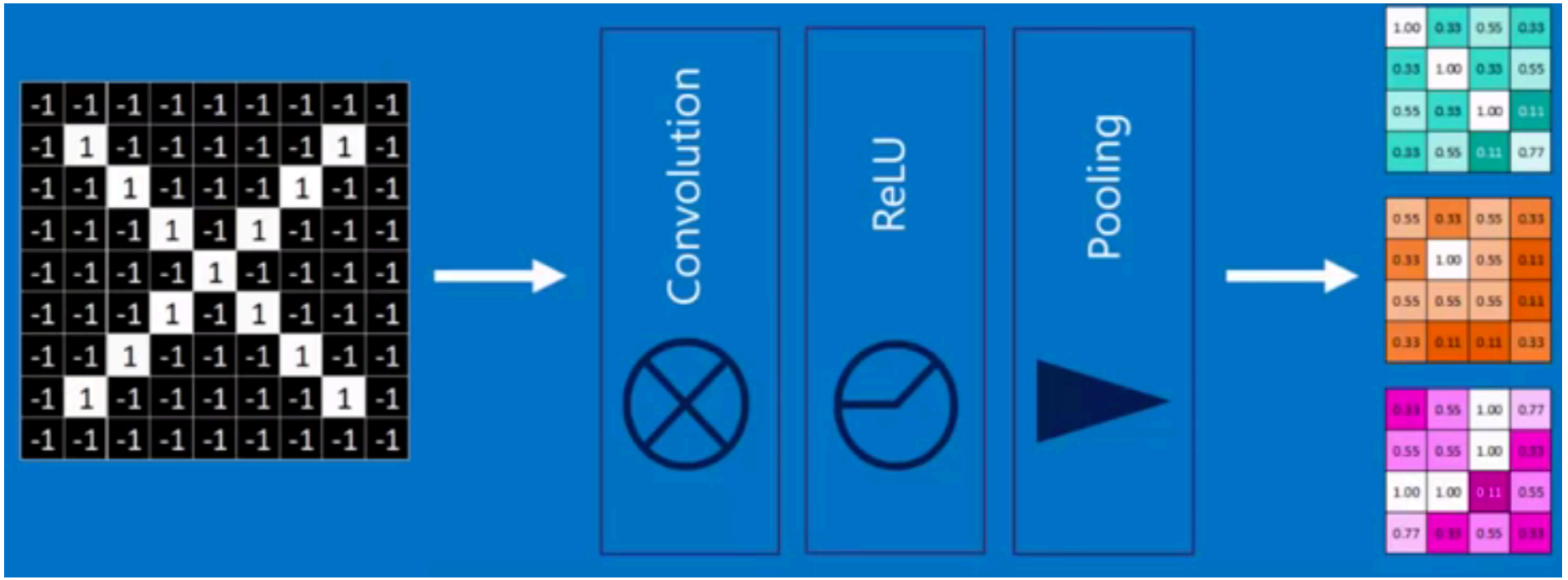



0.77 -0.11 0.11 0.33 0.55 -0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
0.77 0 0.11 0.33 0.55 0 0.33
0 1.00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

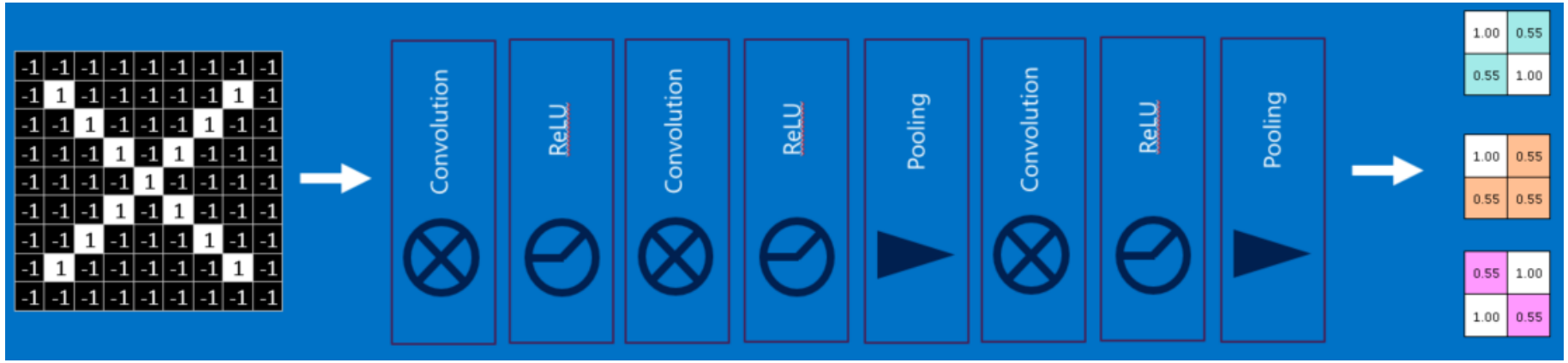
ReLU Layer: a stack of images becomes a stack of smaller images with no negative values.



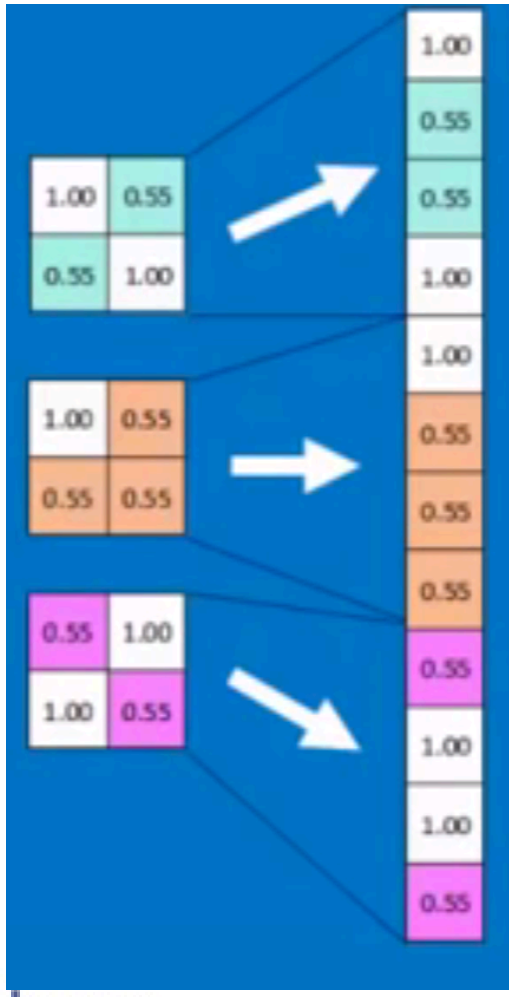
Layers get stacked. The output of one becomes the input of the next.

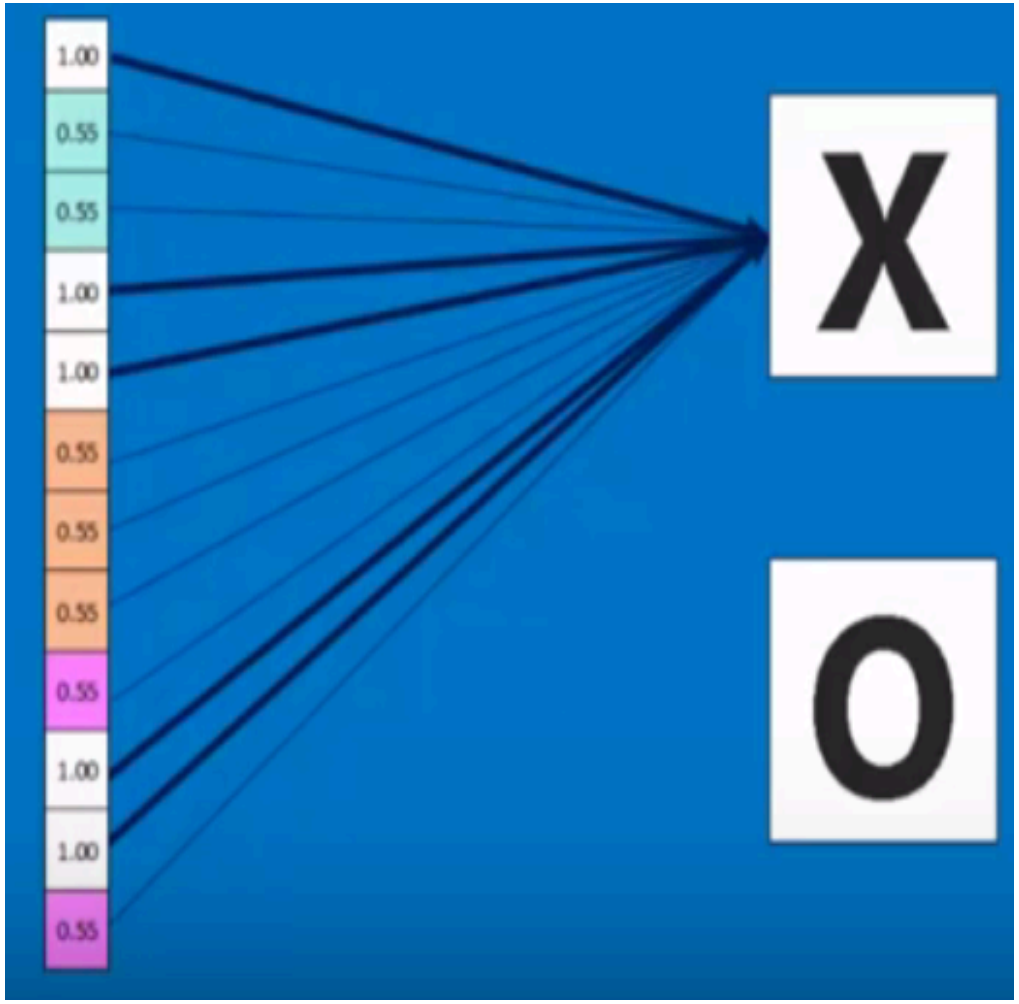


Deep Stacking
Layers can be repeated several (or many) times.

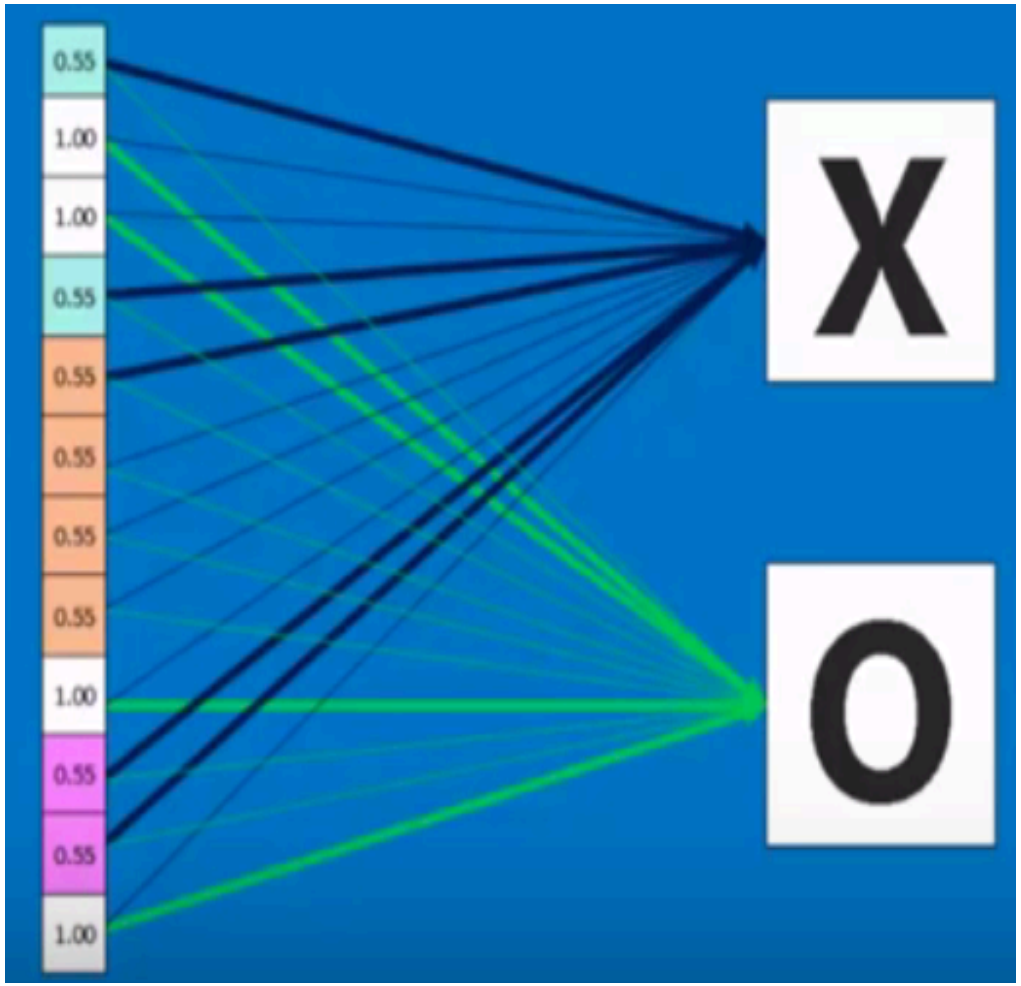


Fully connected Layer: Every values gets a vote

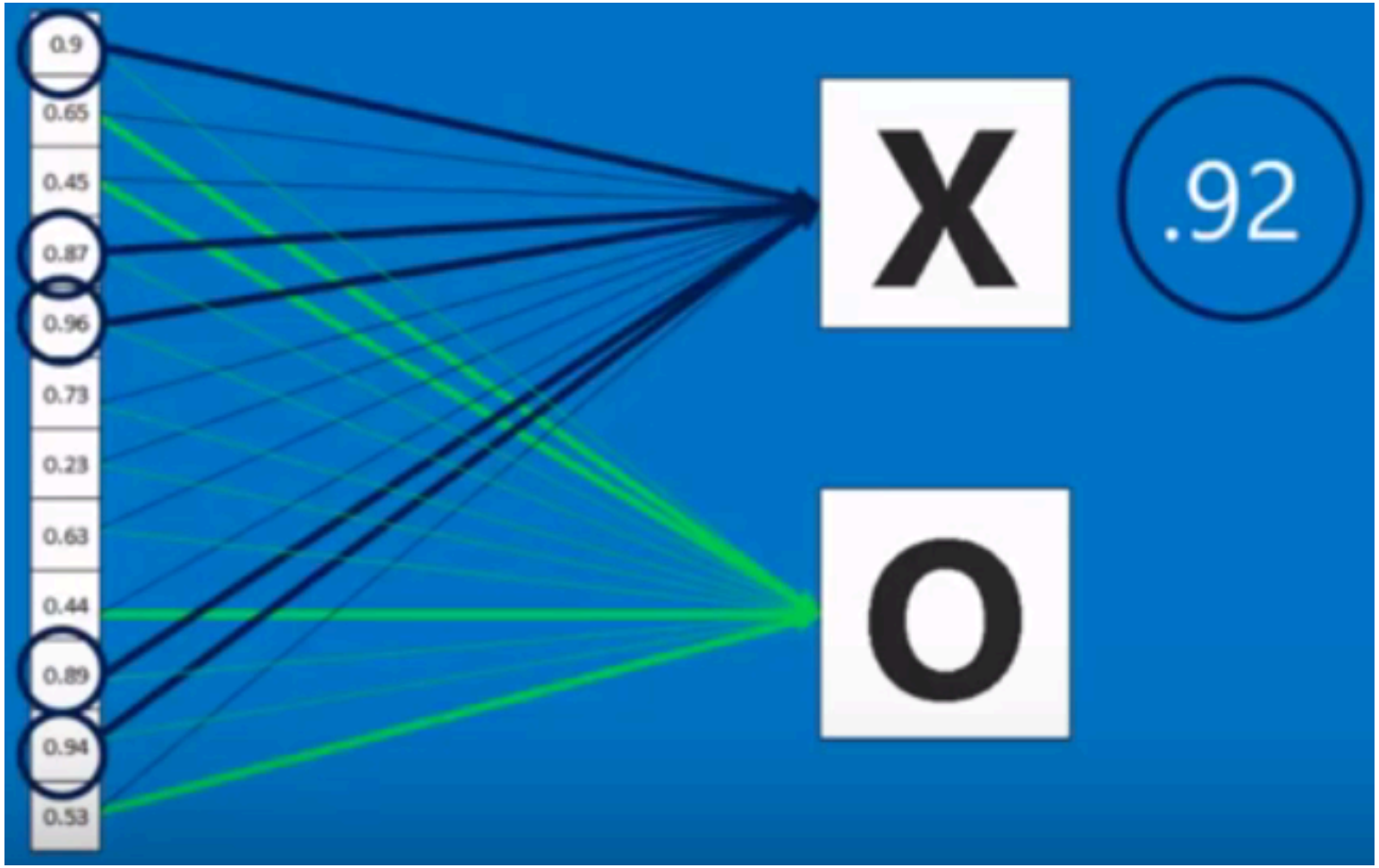


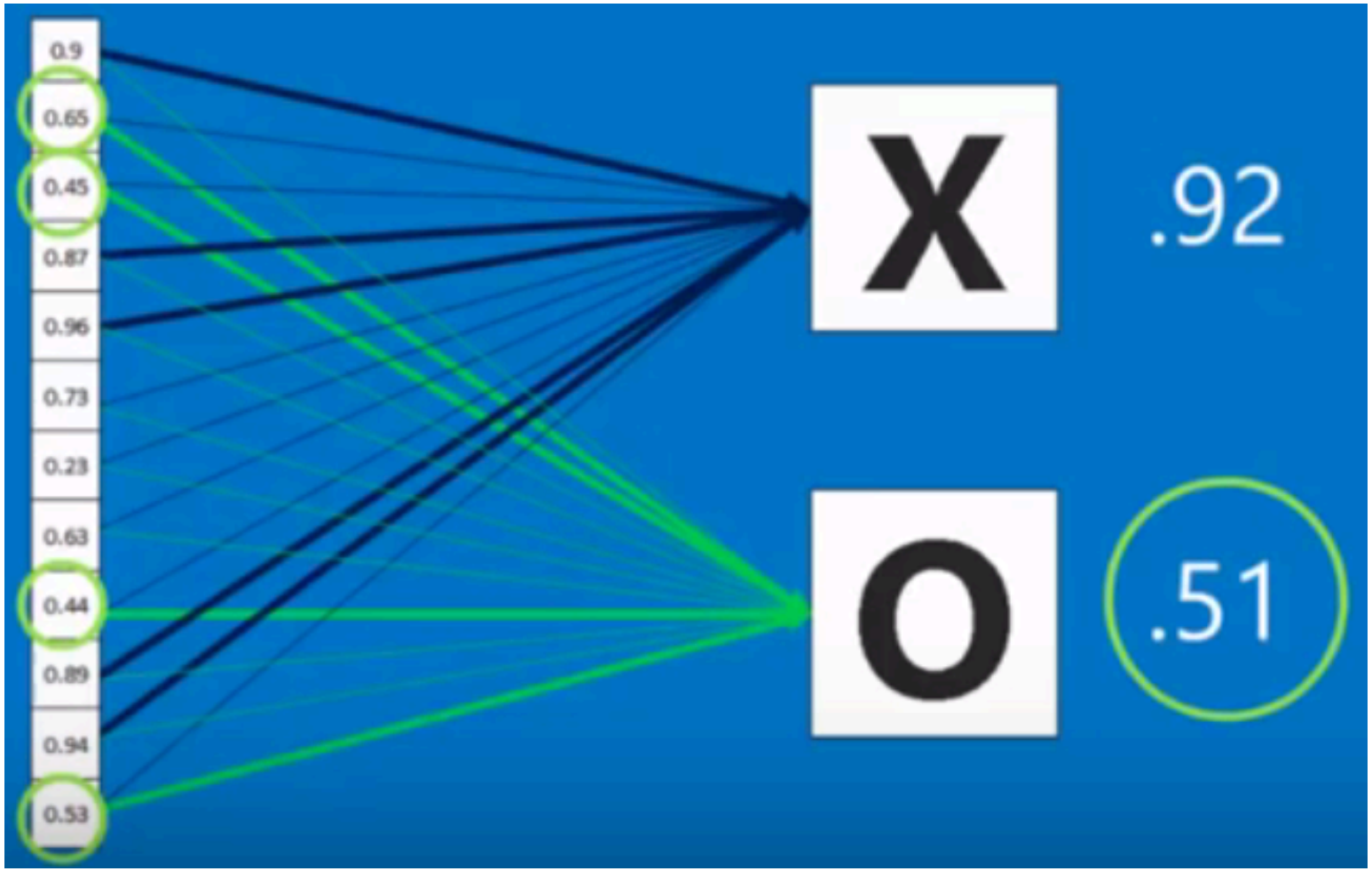


Fully connected Layer: Every values gets a vote

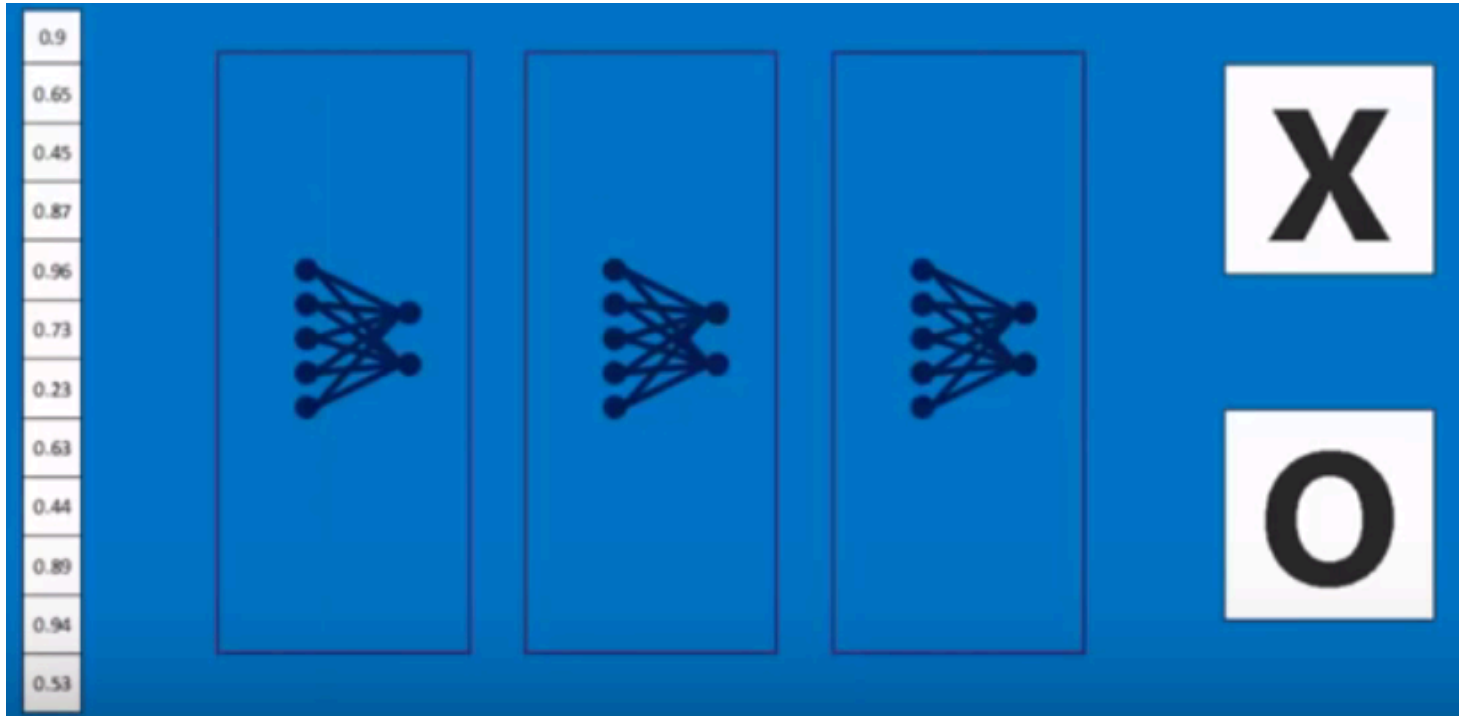


Future Values Vote on X or O.

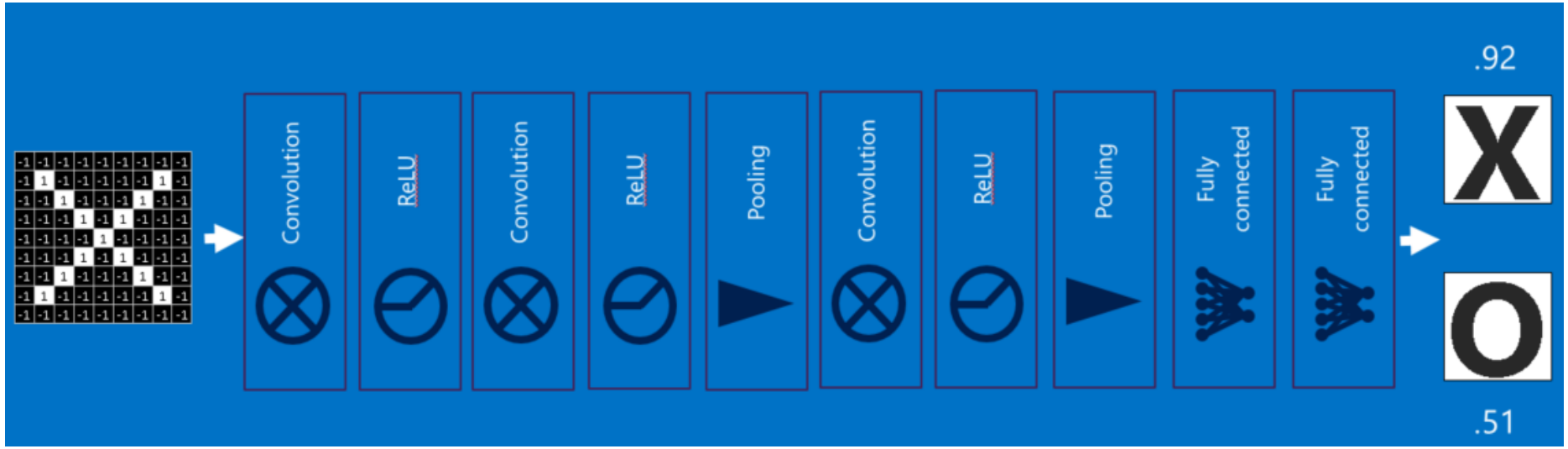



0.9
0.65
0.45
0.87
0.96
0.73
0.23
0.63
0.44
0.89
0.94
0.53
X
.92
O
.51

Fully connected layers: These can also be stacked.



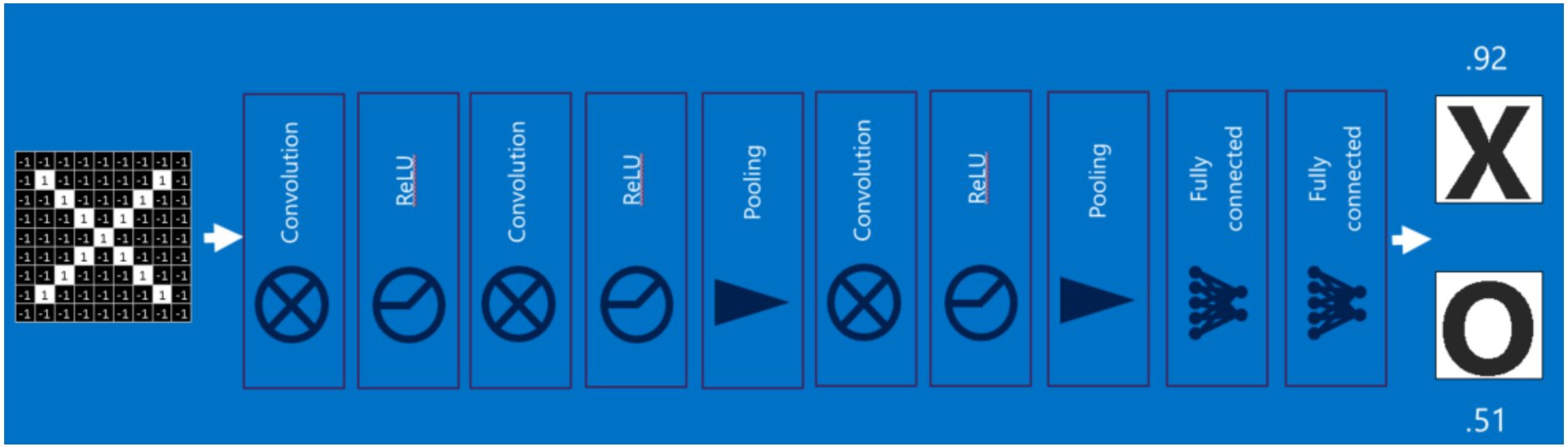
Putting it all together.



Learning

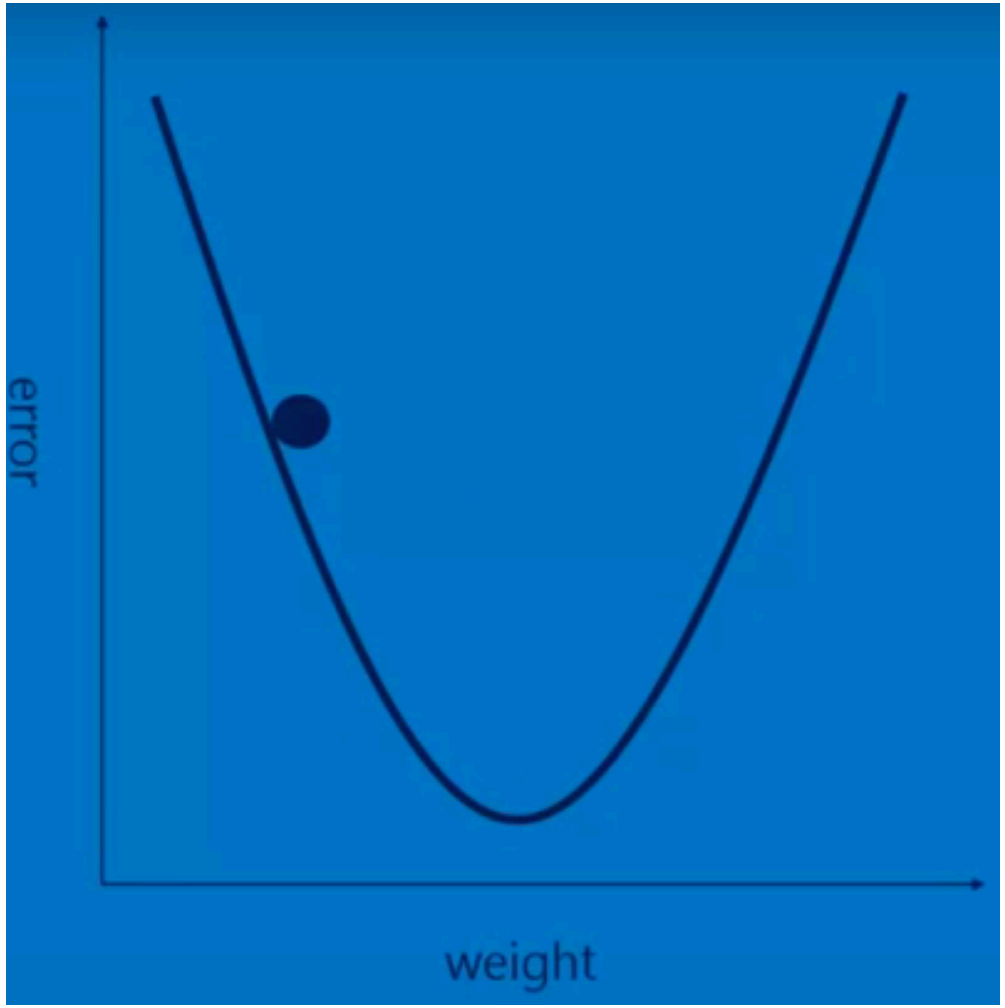
- Q: where do all the magic numbers come from?
  - Features in convolutional layers
  - Voting weights in fully connected layers.
- A: Backpropagation

BackProp: Error = (Predicted –Actual)2



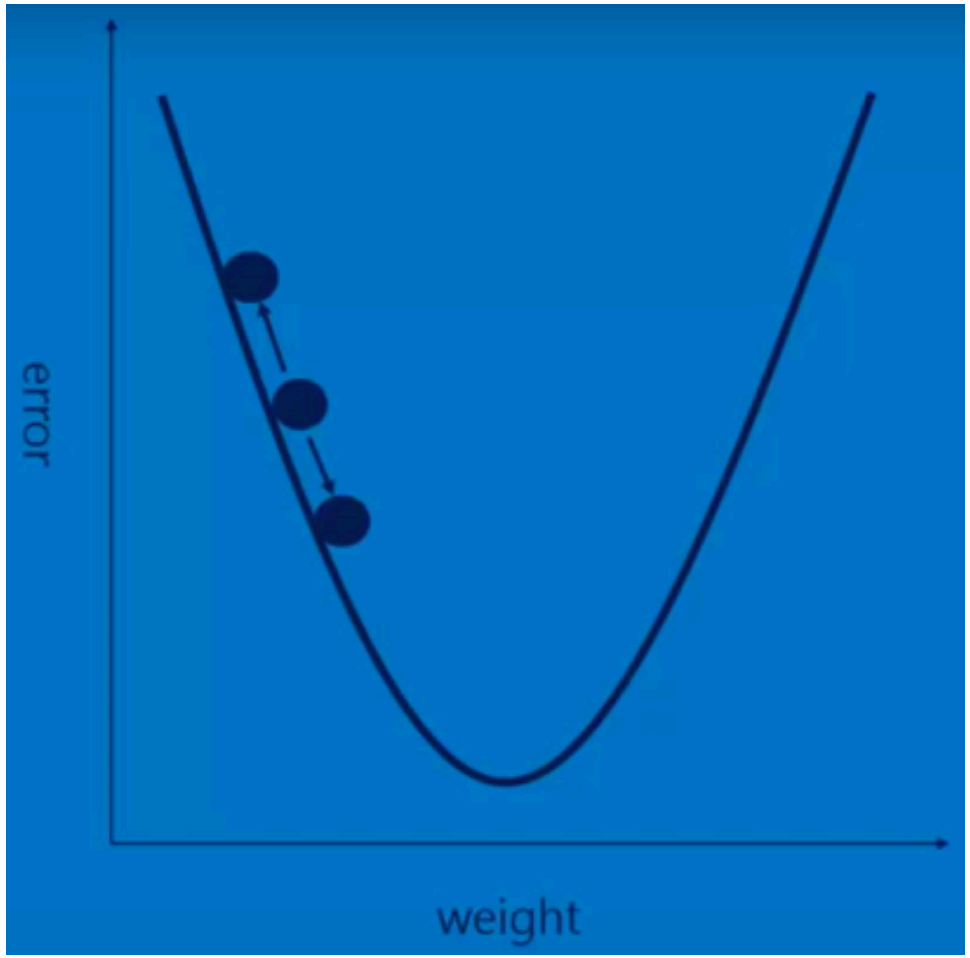
Gradient Descent

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



Gradient Descent

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



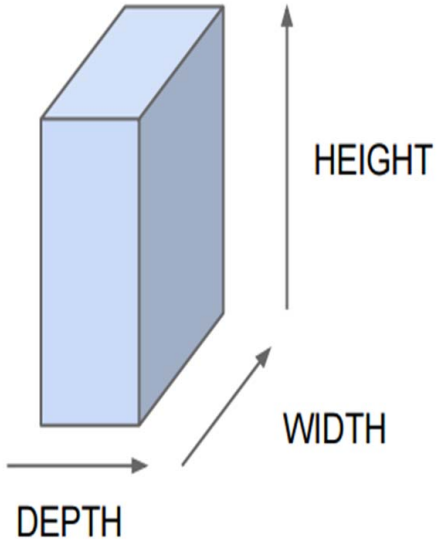
Hyper parameters

- 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?

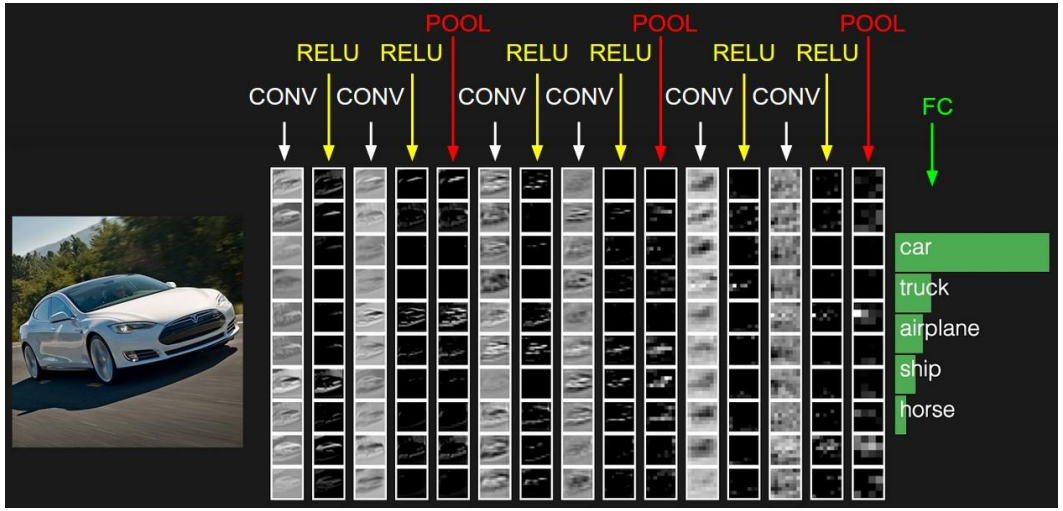
All Neural Net activations arranged in 3 dimensions



For example, a CIFAR-10 image is a 32*32*3 volume: 32 width, 32 height, 3 depth (RGB)

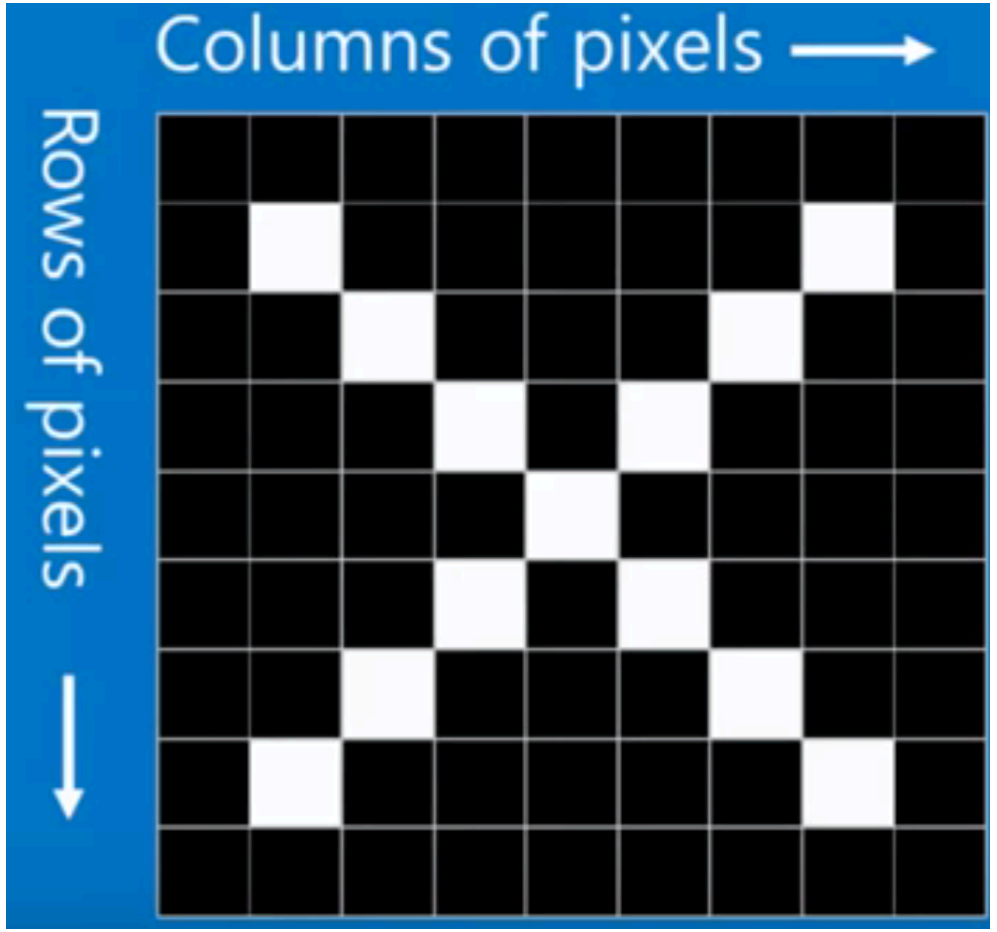
Convolutional Neural Networks: Layers

- **INPUT** [32x32x3] will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.
- **CONV** layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [32x32x12] if we decided to use 12 filters.
- **RELU** layer will apply an elementwise activation function, such as the *max(0,x)* thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).
- **POOL** layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].
- **FC** (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

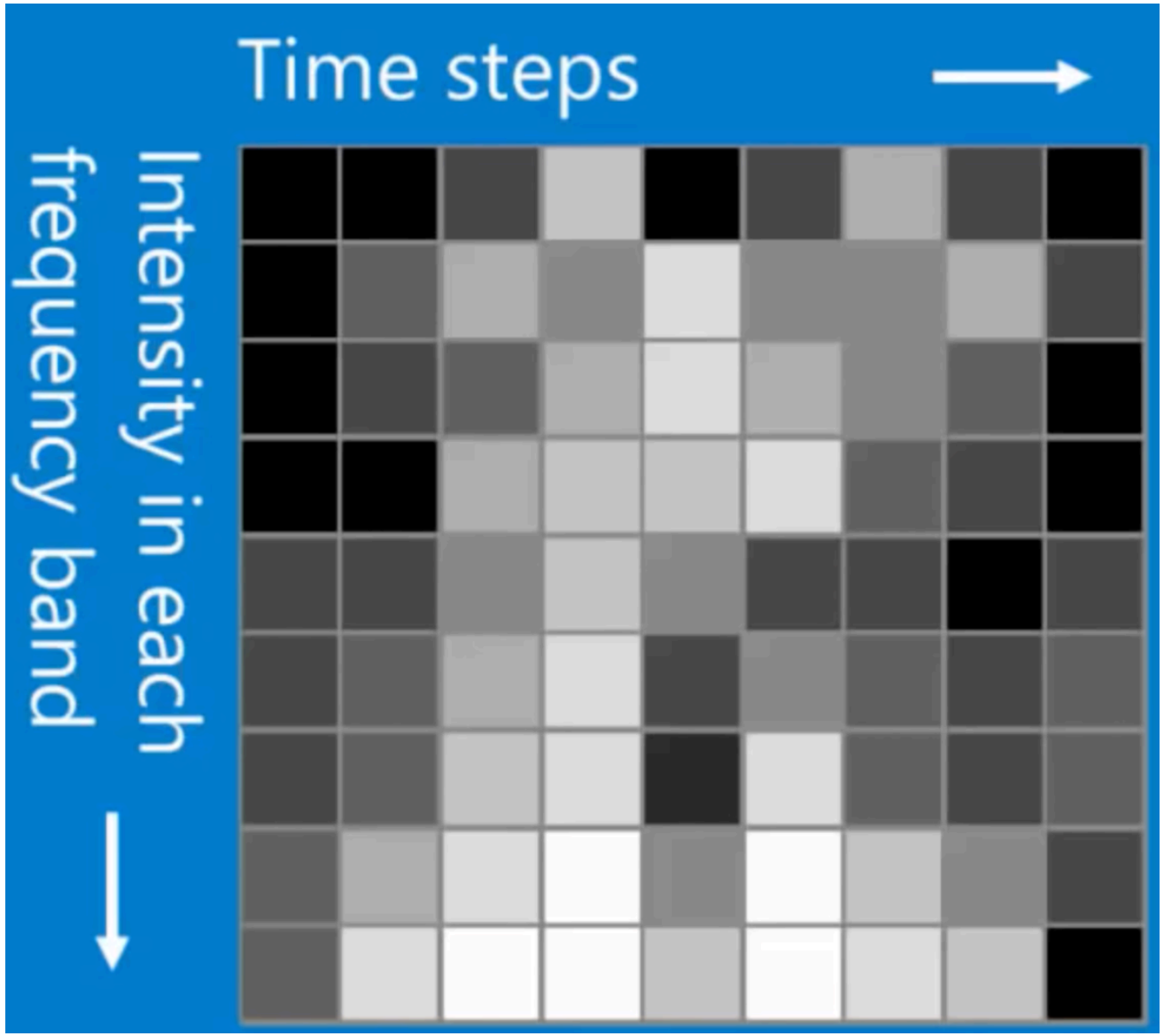


Not just images

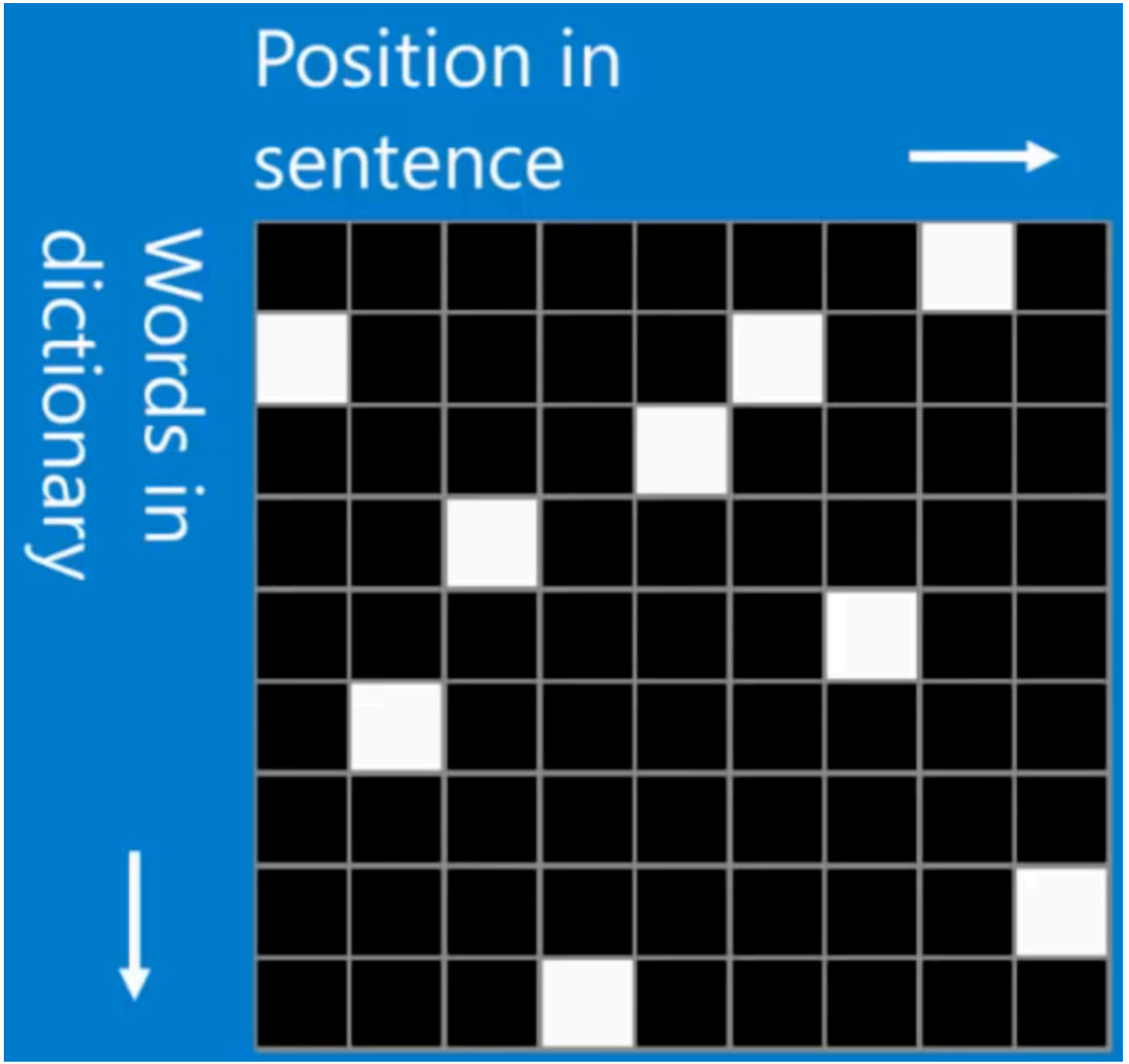
- Any 2D (or 3D) data
- Things closer together are more closely related than things far away.



Sound



Text



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.

Customer Data

Customers

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

Rule of thumb

If your data is just as useful after swapping any of your columns with each other, then we should not use CNN.

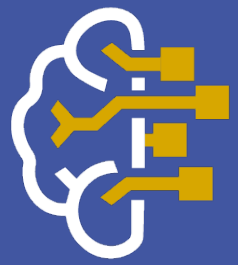
To home message: ConvNets are great at finding patterns and using them to classify images.

Stay Connected

Dr. Min Chi

Associate Professor
mchi@ncsu.edu
(919) 515-7825

AI Academy



AI Academy

go.ncsu.edu/aiacademy

NC STATE UNIVERSITY