Week 03: Convolutional Neural Networks (CNNs)

Machine Learning 2

Dr. Hongping Cai

Topic 1:
Challenges for Visual Recognition

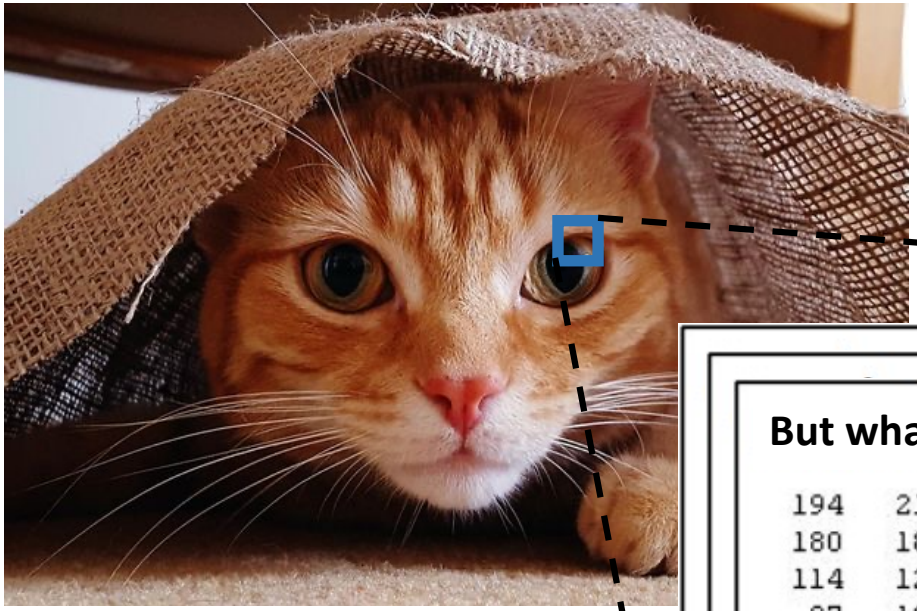
http://www.image-net.org/challenges/LSVRC/

- Annual competition on:
  - Image Classification
  - Single-object localization
  - Object detection
- **ImageNet** dataset
  - 1.4 million annotated images
  - 1000 object classes

The best classification performance in 2011 (before deep learning):

25.8% error

Why is visual recognition so difficult?



An image is a matrix of numbers [0,255],
i.e., 1024 x 800 x 3 for a color image.

But what the computer sees:

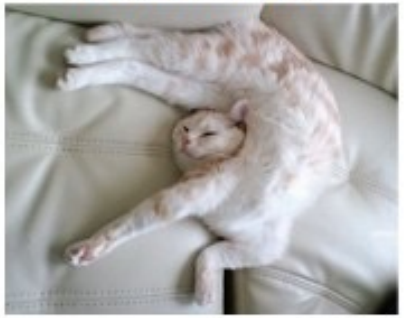
194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

Why is visual recognition so difficult?

Viewpoint variation

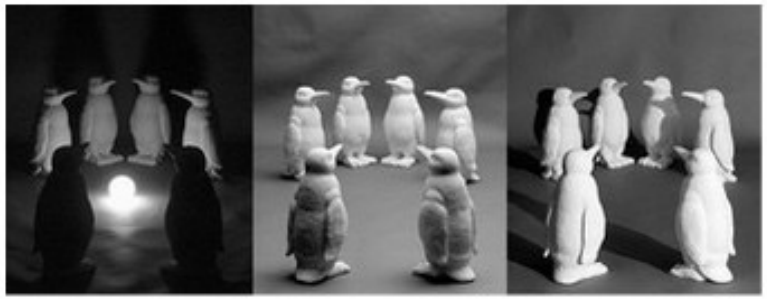
Scale variation

Deformation



Occlusion

Illumination conditions

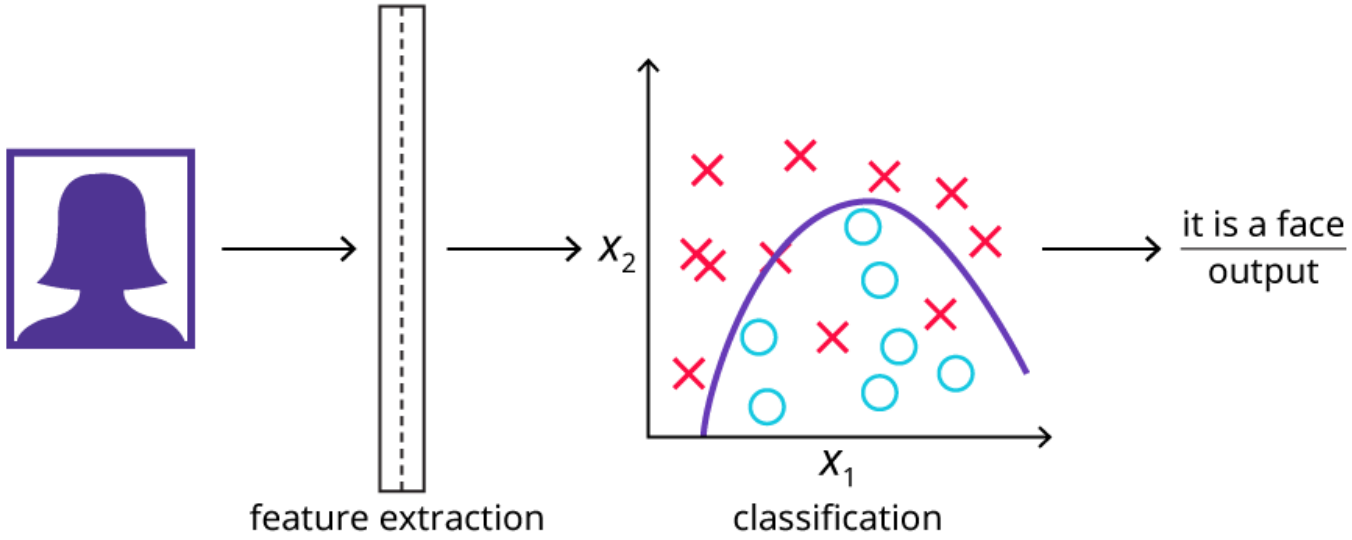


Background clutter

Intra-class variation

Image from: https://cs231n.github.io/classification/

Image classification before deep learning



Things changed since 2012 …

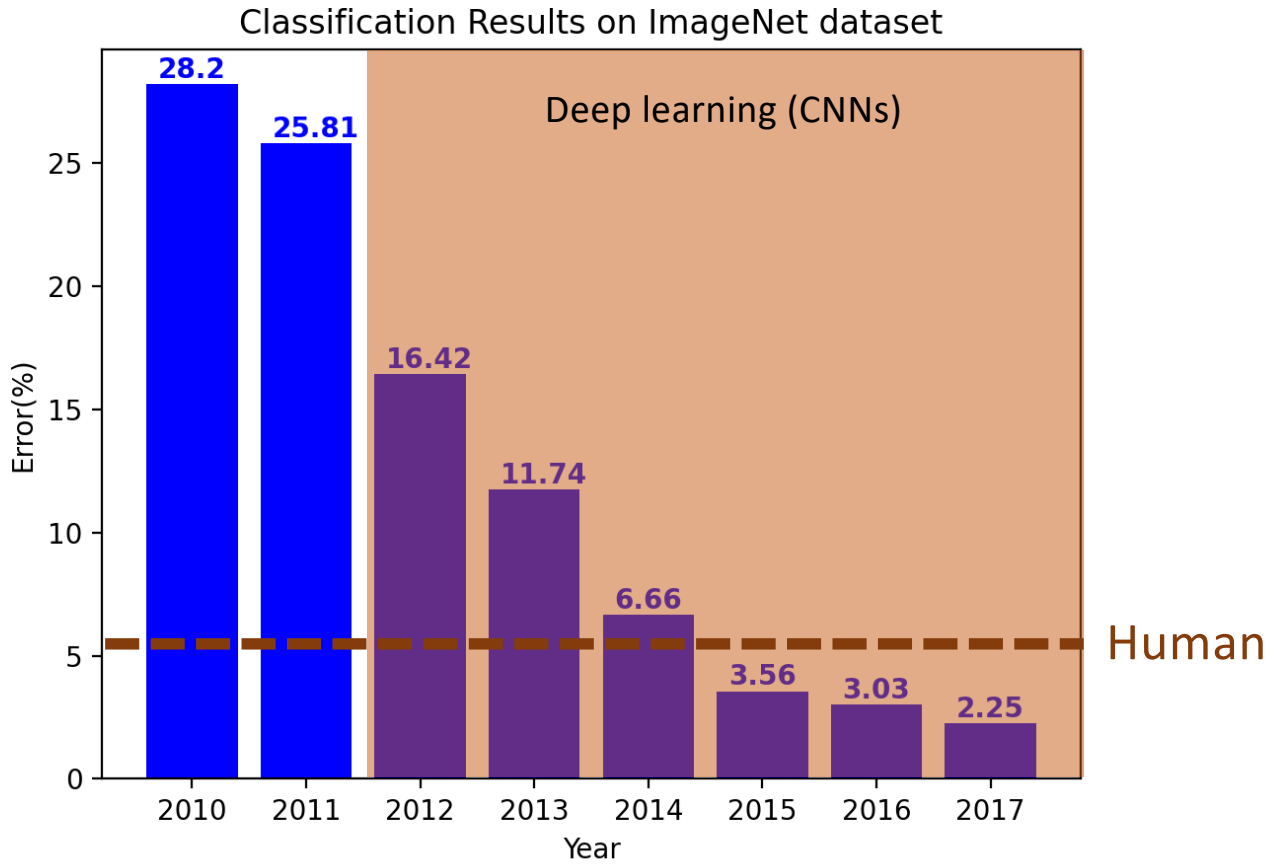
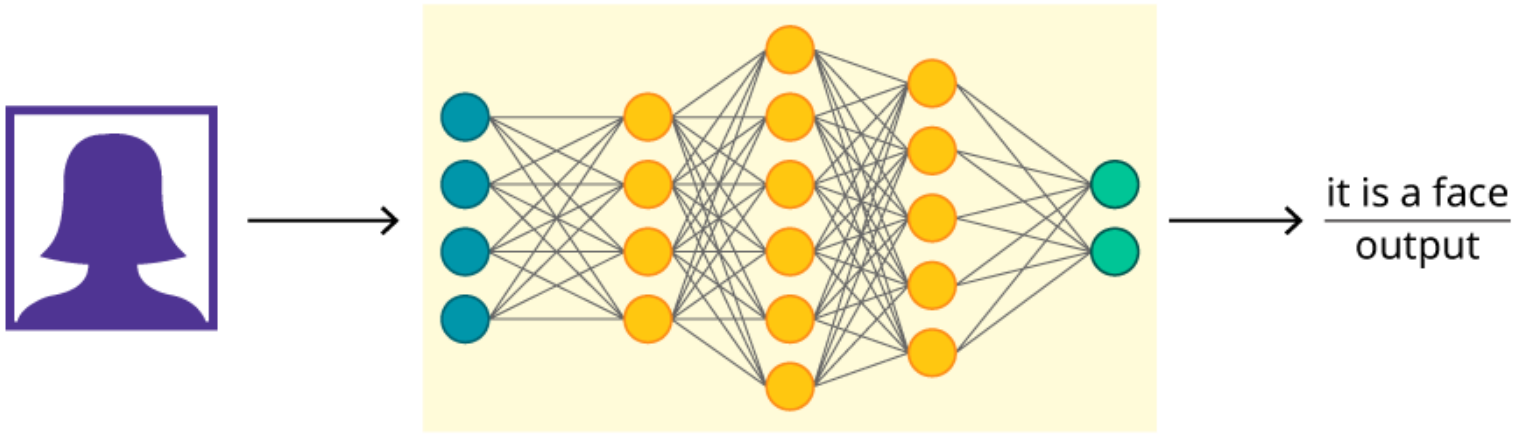


Image classification with deep learning



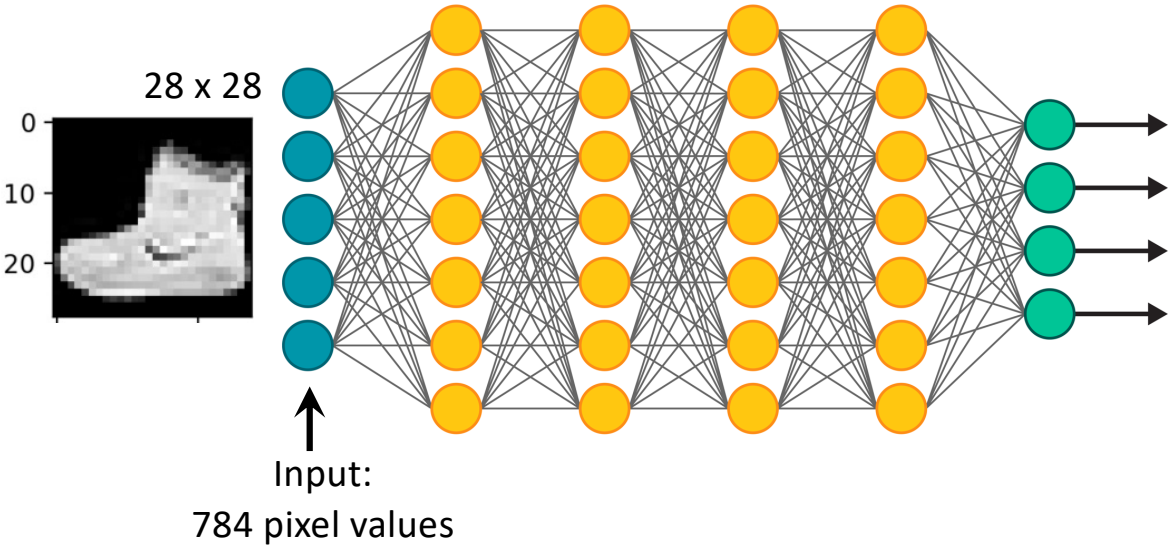
Reference for Topic 1

- Video lecture by Alexander Amini: MIT course on Convolutional Neural Networks: https://www.youtube.com/watch?v=iaSUYvmCekI
- https://towardsdatascience.com/deep-learning-for-image-classification-why-its-challenging-where-we-ve-been-and-what-s-next-93b56948fcef

Topic 2:
Convolution Operation

Can we use MLP to classify images?

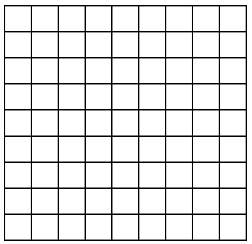
- Yes, we can.
- But,
  - Spatial information is removed.
  - Too many parameters for fully-connected layers

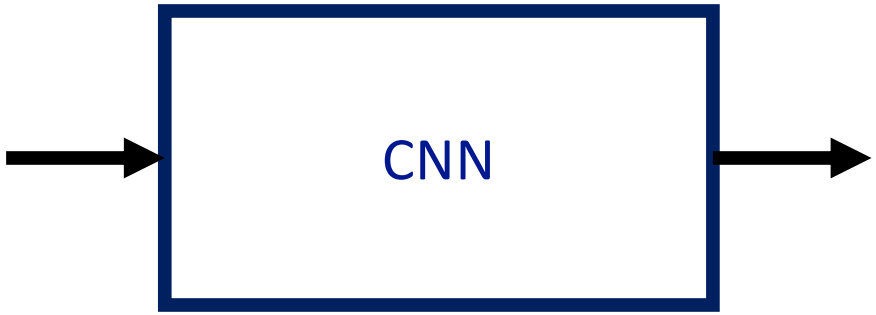


Q: If the input is a color image of size 200 x 200, the second layer is 300 neurons, how may weights needed for connecting the first two layers?
A: 36,000,300 (=200 x 200 x 3 x 300 + 300)!

A simple example: X's and O's

A two-dimensional array of pixels

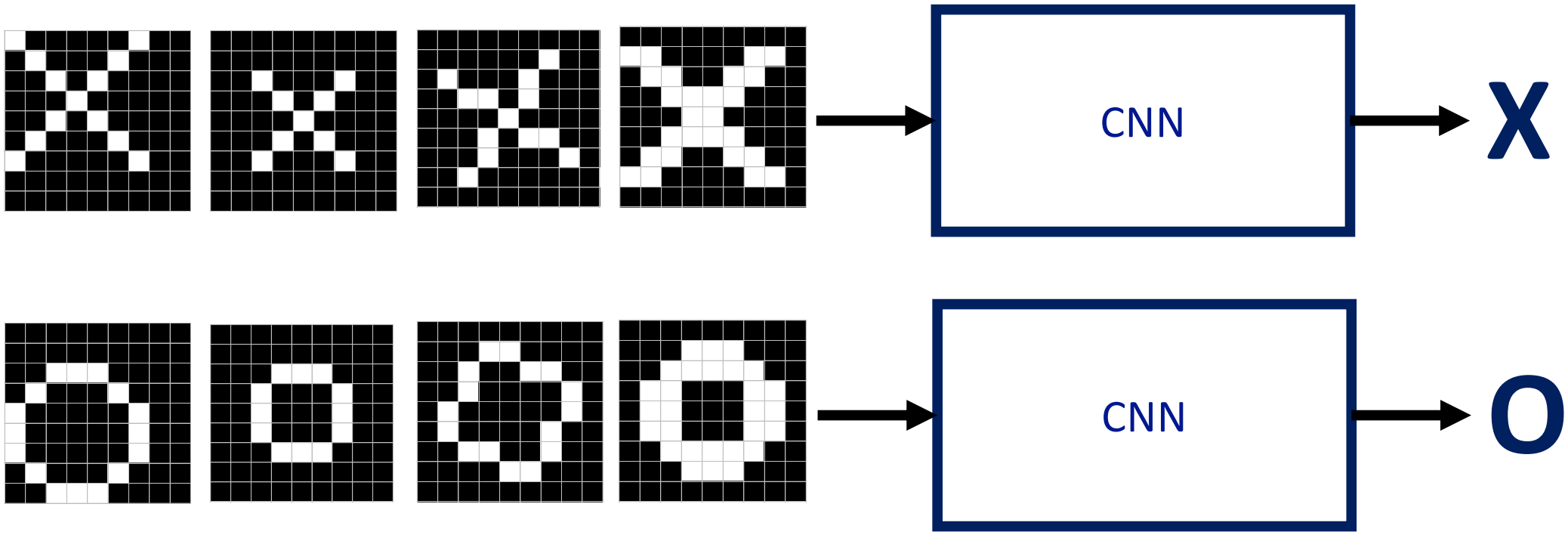




X or **O?**

Example from: https://www.youtube.com/watch?v=FmpDIaiMIeA

A simple example: X's and O's

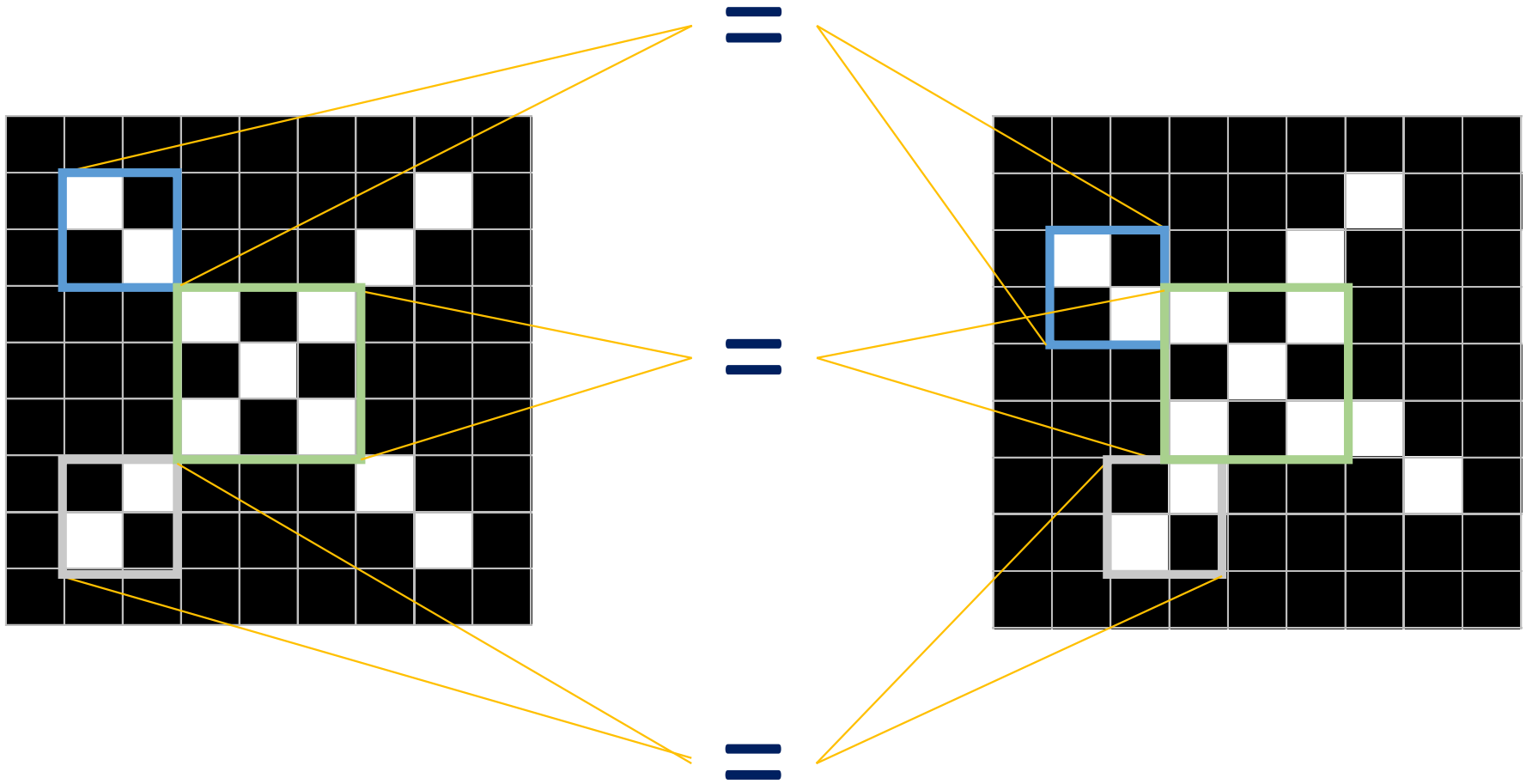


A simple example: X's and O's

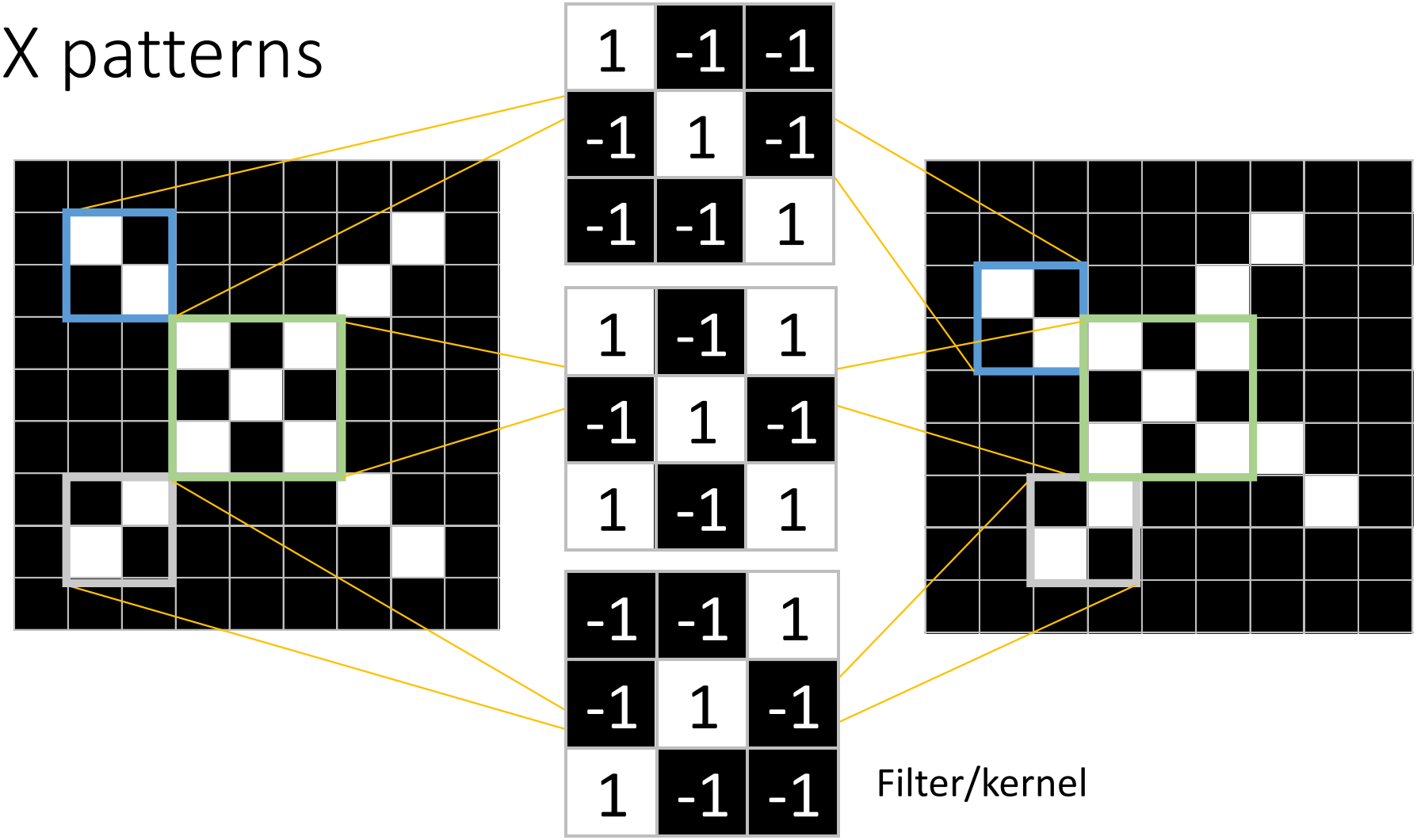
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

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

Common local patterns



Filters to detect X patterns



Filtering: The math behind the match

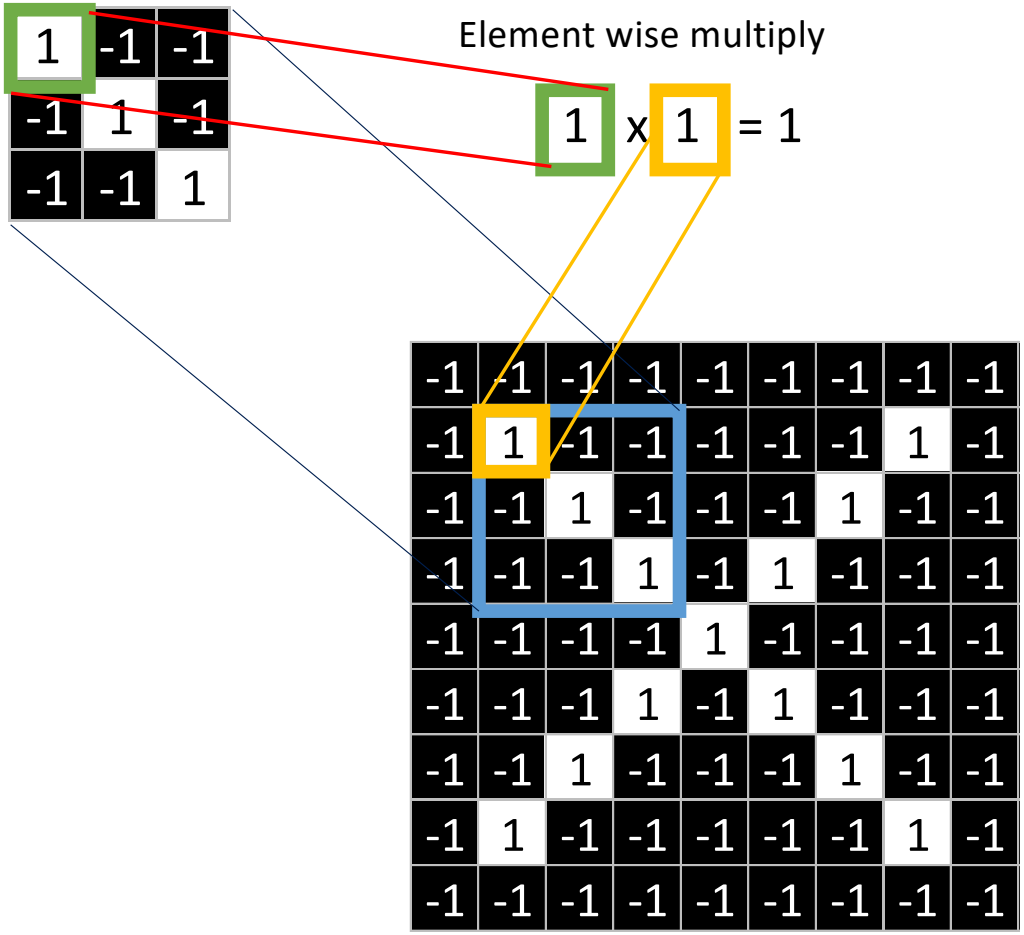
h1	h2	h3
h4	h5	h6
h7	h8	h9

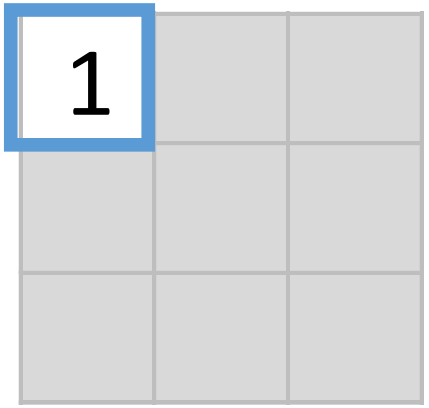
Filter/kernel

$$y=(x1*h1+x2*h2+\ldots+x9*h9)/9$$

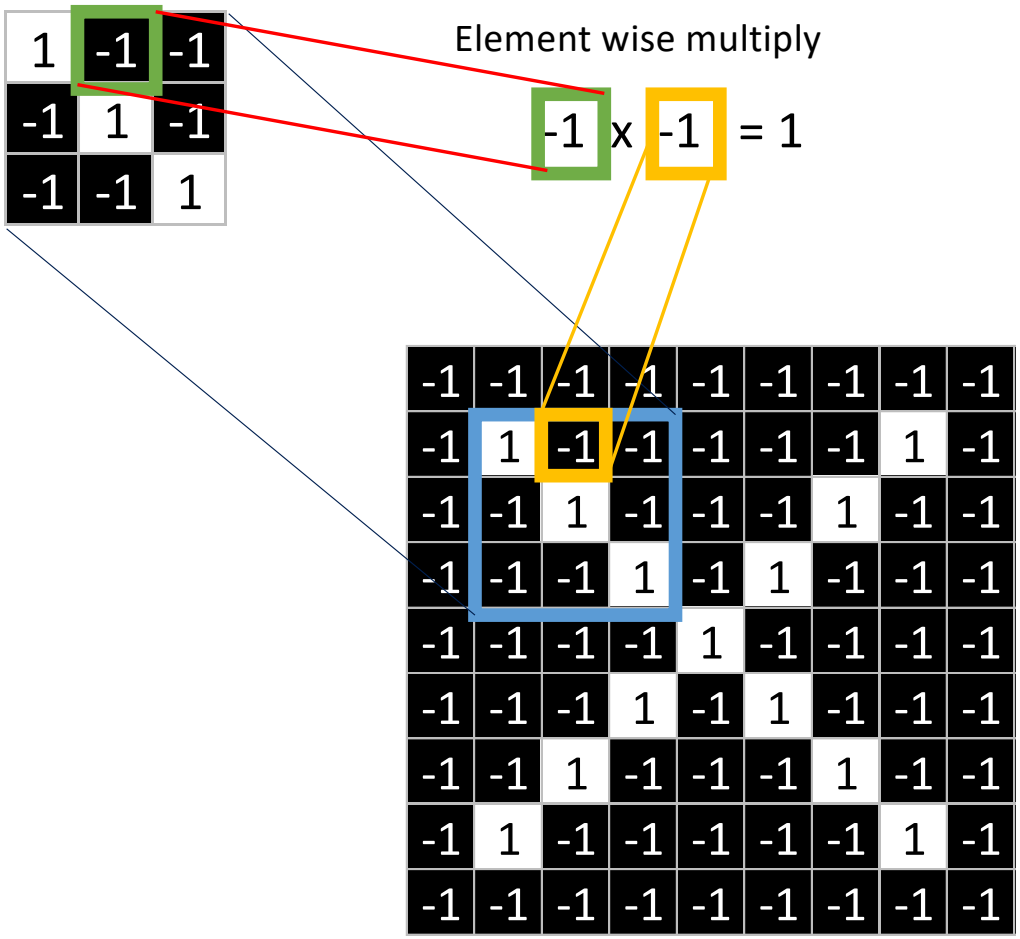
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	x1	x2	x3	-1	-1	-1	1	-1
-1	x4	x5	x6	-1	-1	1	-1	-1
-1	x7	x8	x9	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

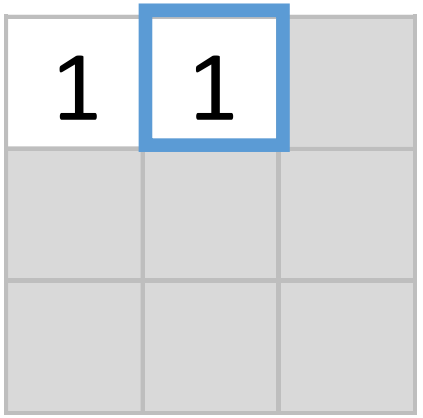
Filtering: The math behind the match



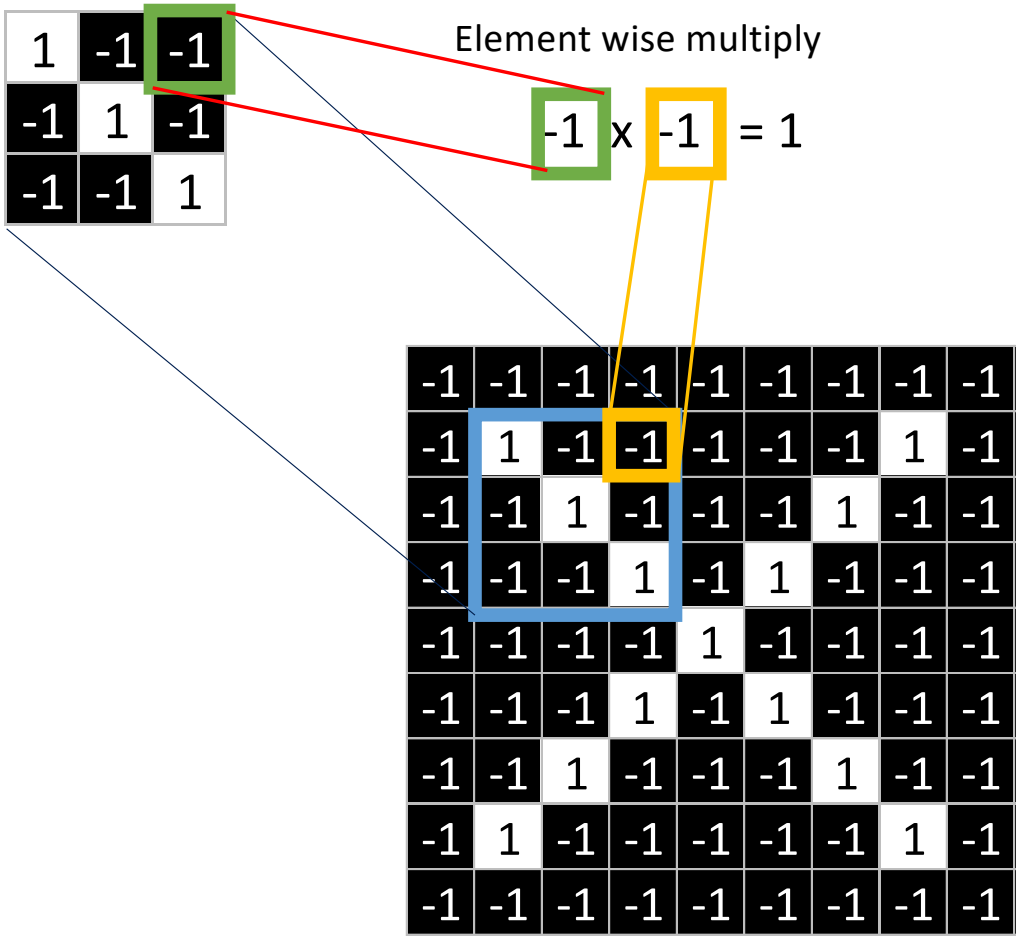


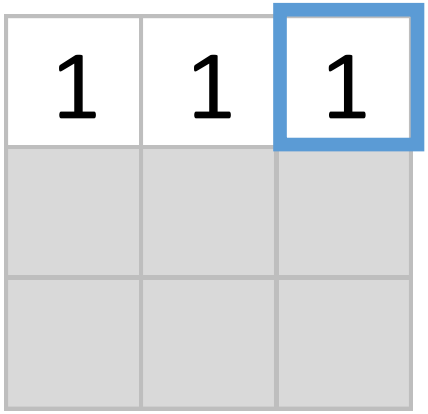
Filtering: The math behind the match



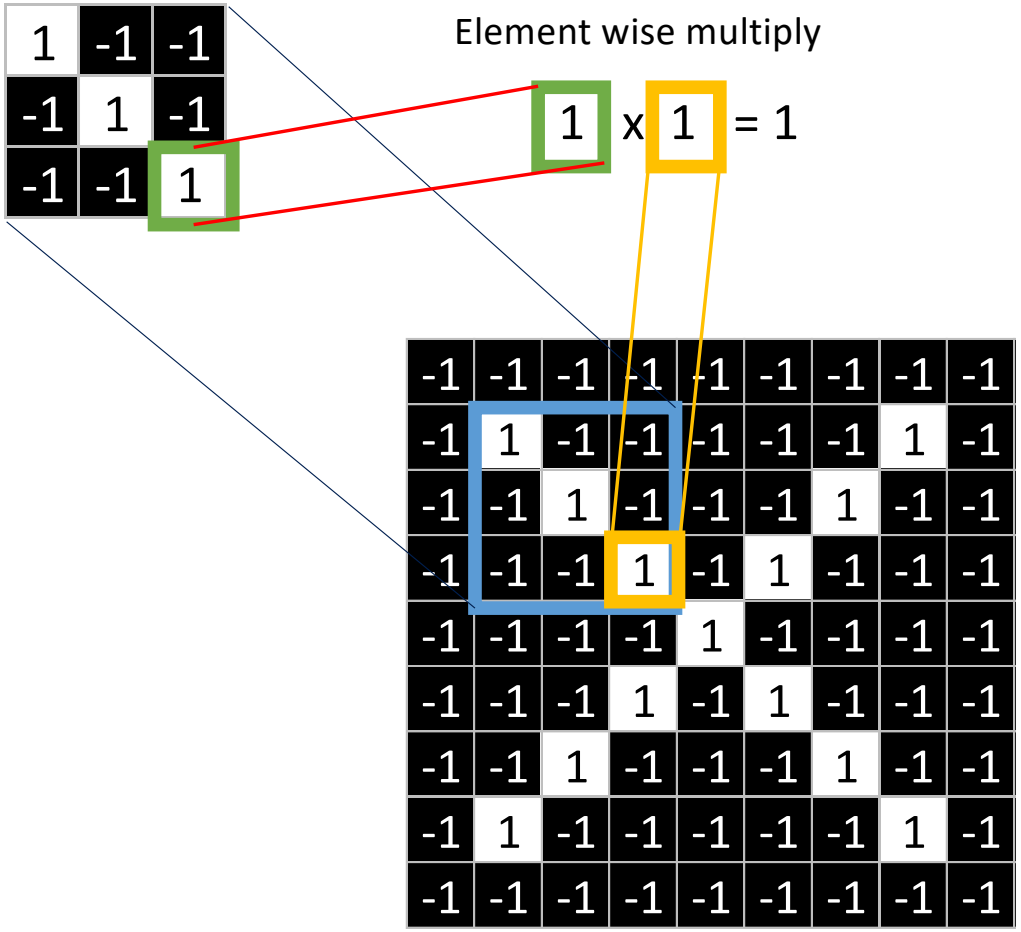


Filtering: The math behind the match



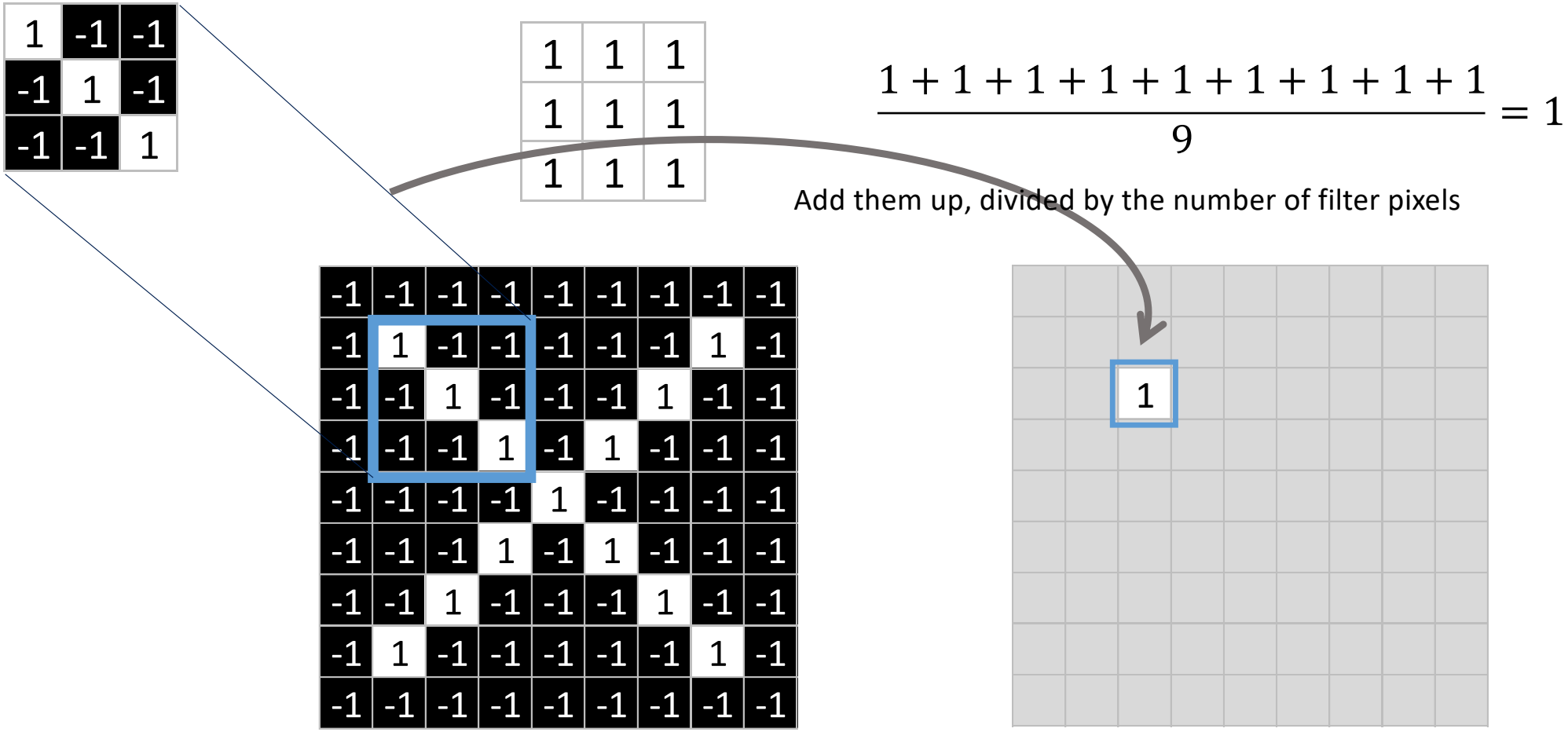


Filtering: The math behind the match

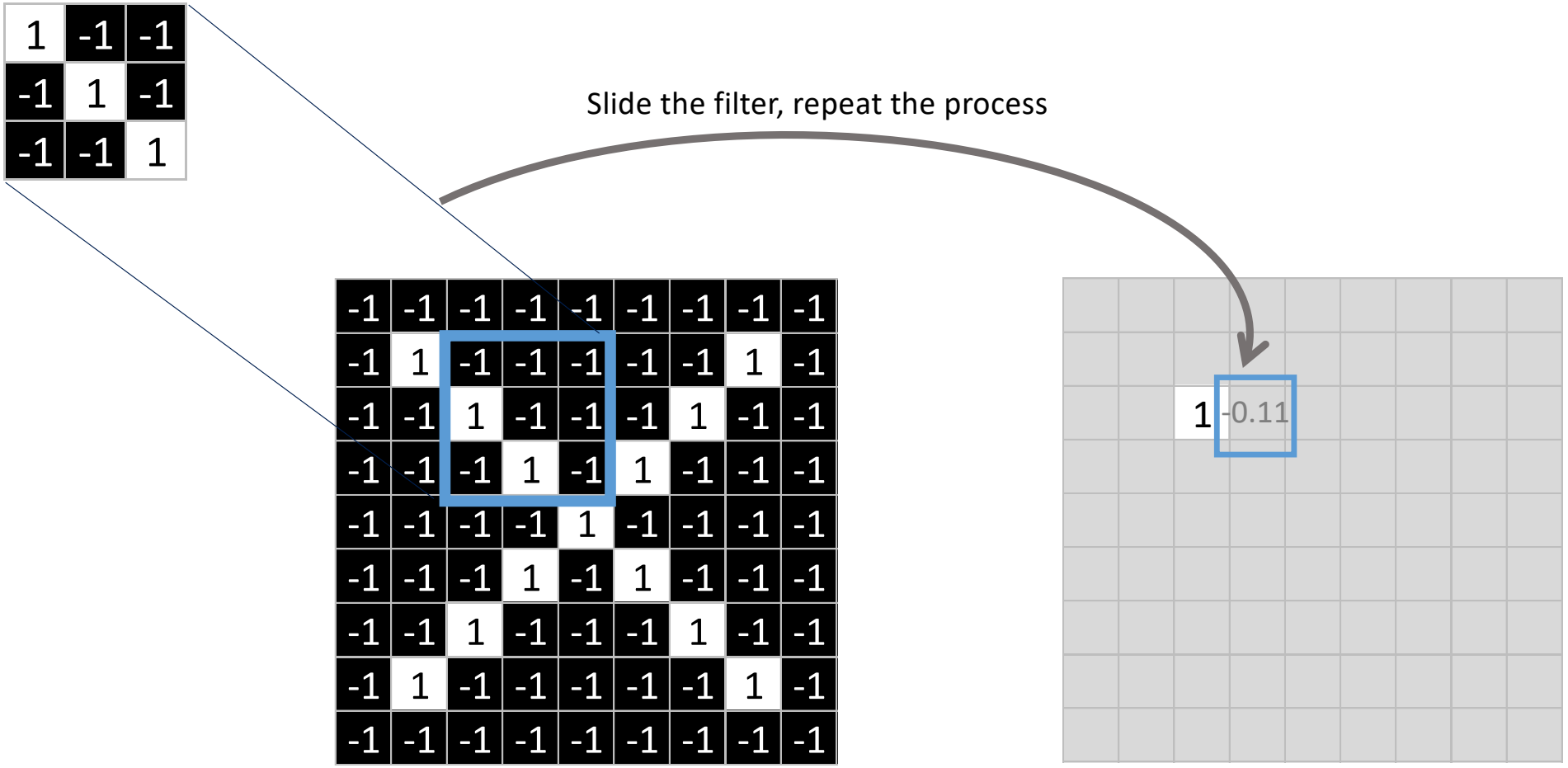


1	1	1
1	1	1
1	1	1

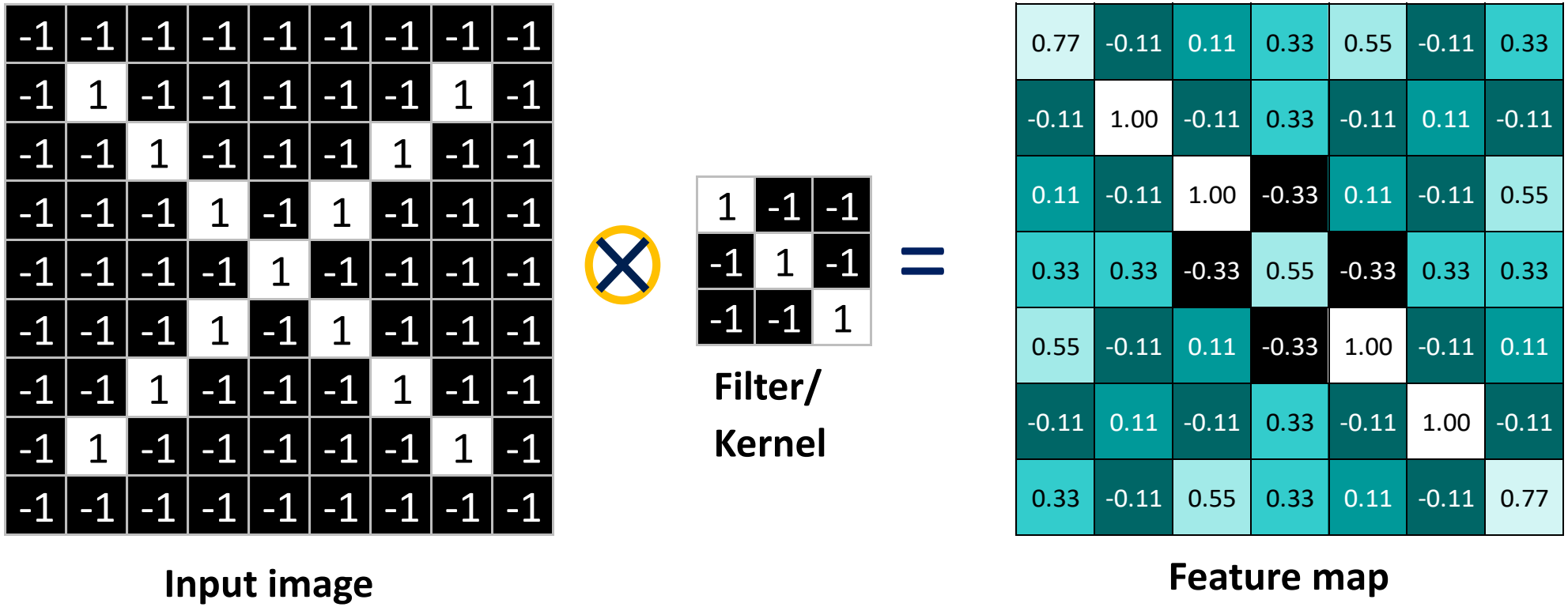
Filtering: The math behind the match



Filtering: The math behind the match



Convolution operation

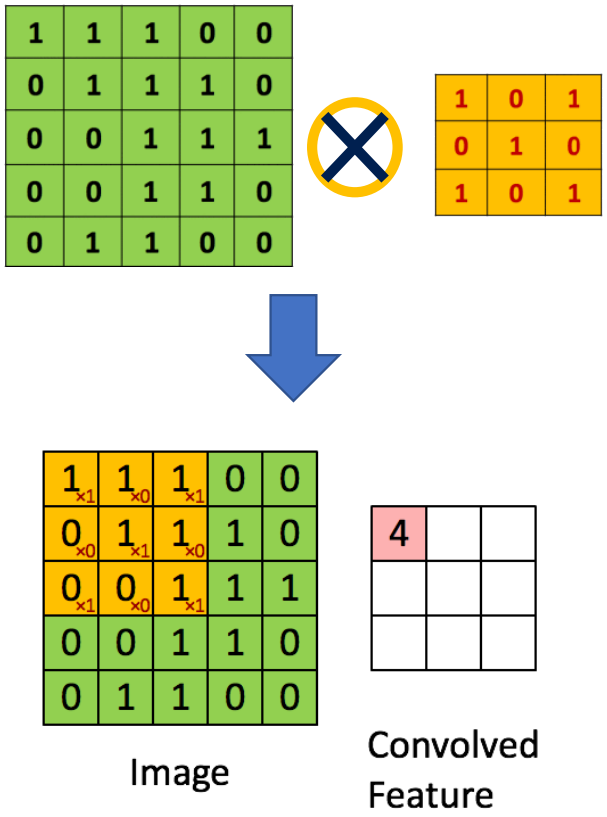


Input image

Filter/ Kernel

Feature map

Convolution operation



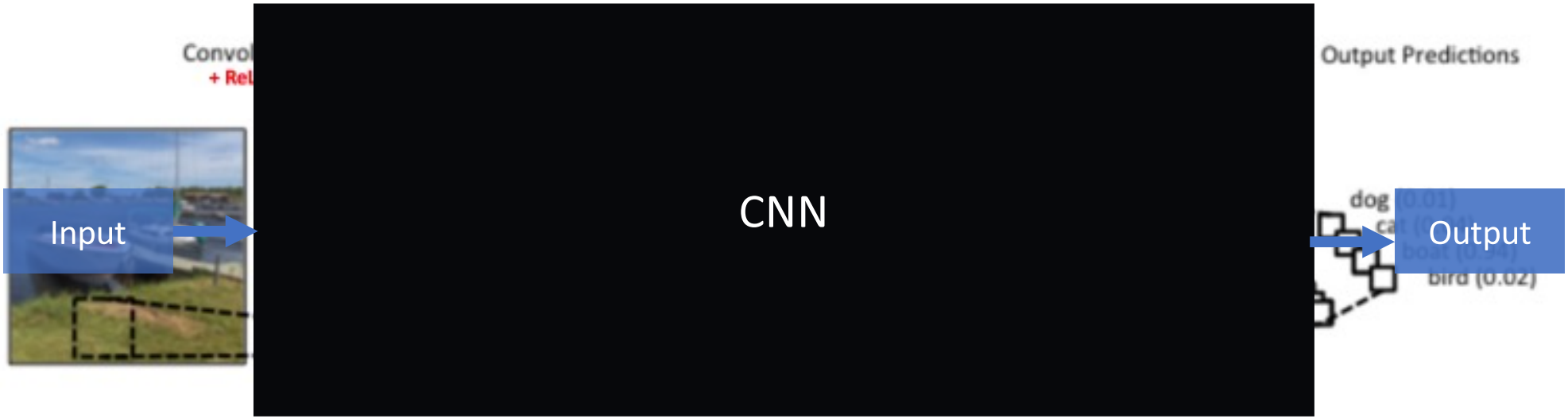
From: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Reference for Topic 2

- Video lecture by Brandon Rohrer: How Convolutional Neural Networks work: https://www.youtube.com/watch?v=FmpDIaiMIeA
- Video lecture by Alexander Amini: MIT course on Convolutional Neural Networks: https://www.youtube.com/watch?v=iaSUYvmCekI
- Blog by ujjwalkarn: An Intuitive Explanation of Convolutional Neural Networks: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Topic 3:
CNN: Convolutional Layer

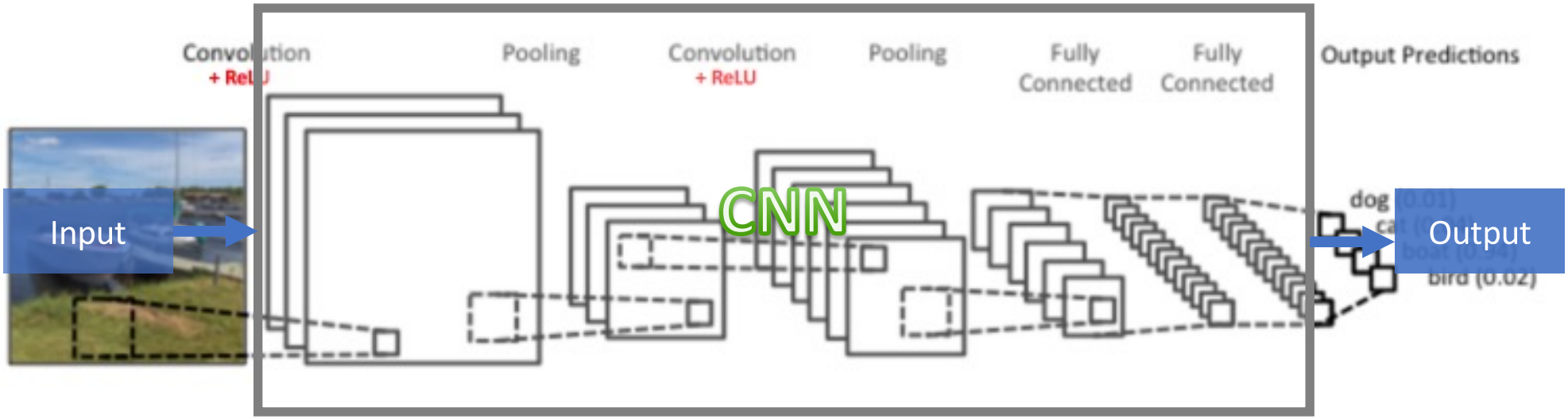
Image classification with CNN



Convolutional Neural Network (**CNN**, or **ConvNet**)

Image from: https://towardsdatascience.com/build-your-own-convolution-neural-network-in-5-mins-4217c2cf964f

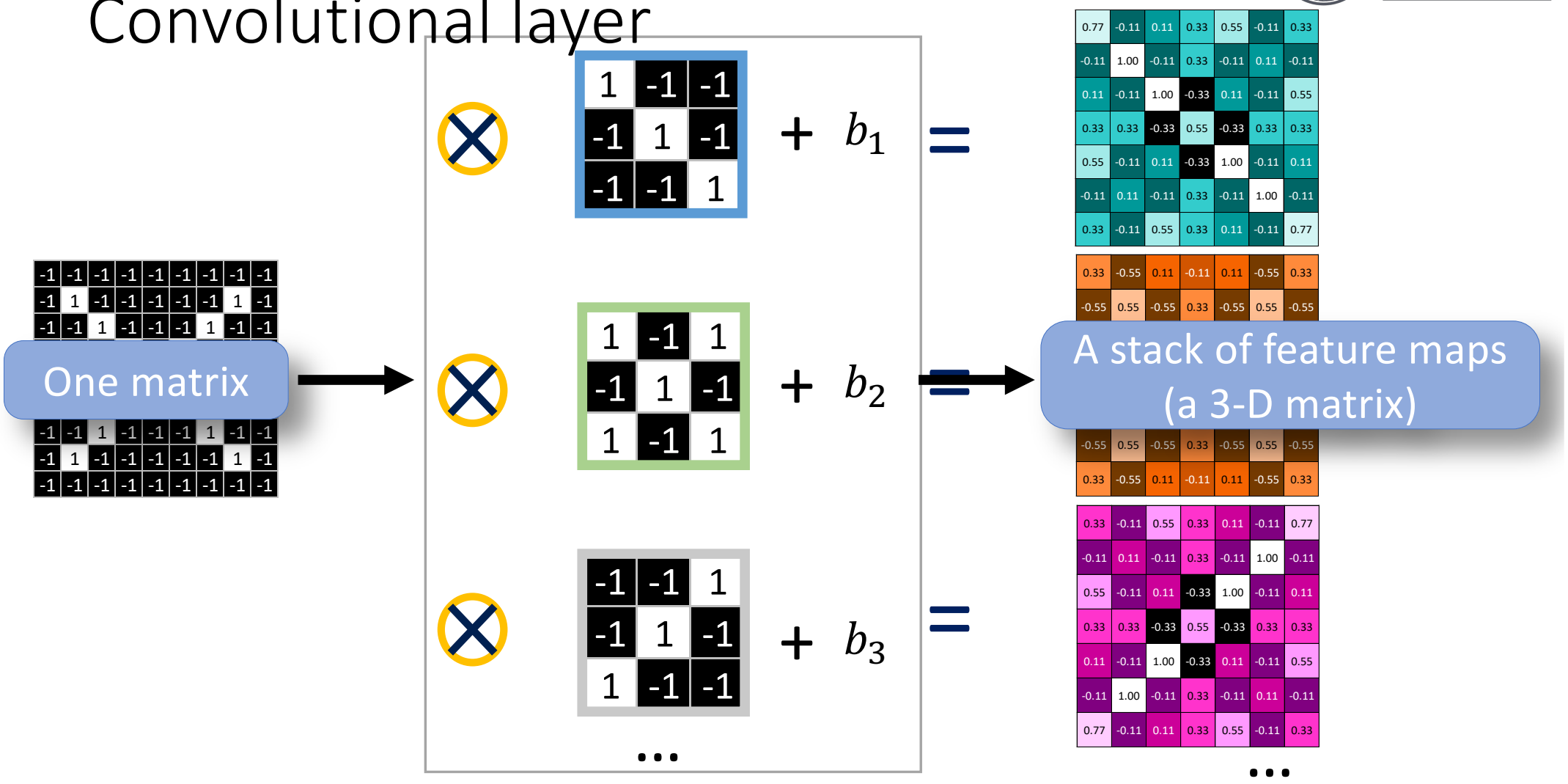
Image classification with CNN



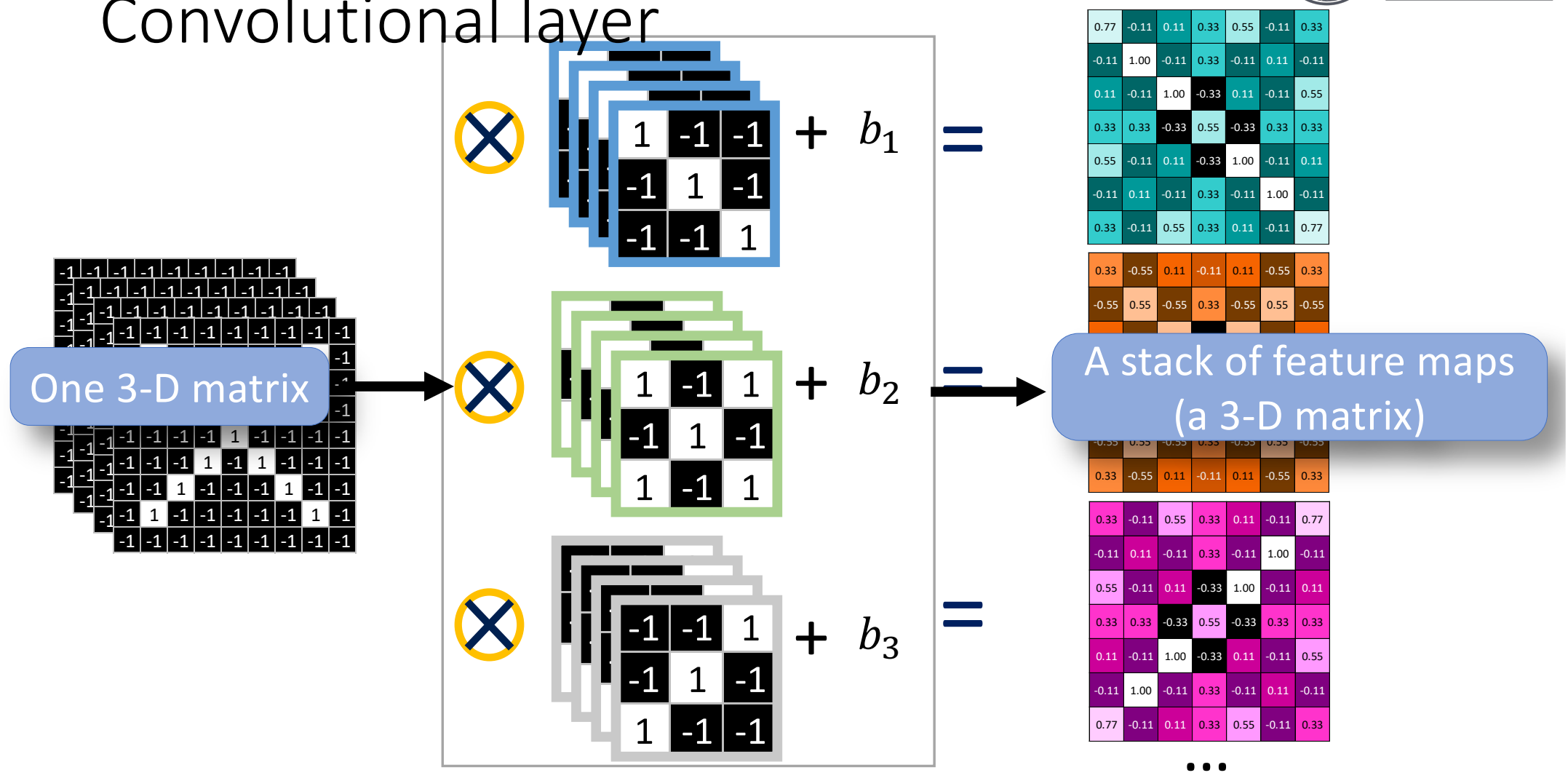
- Convolutional Layer
- ReLU Layer
- Pooling Layer
- Fully Connected Layer

Image from: https://towardsdatascience.com/build-your-own-convolution-neural-network-in-5-mins-4217c2cf964f

Convolutional layer



Convolutional layer

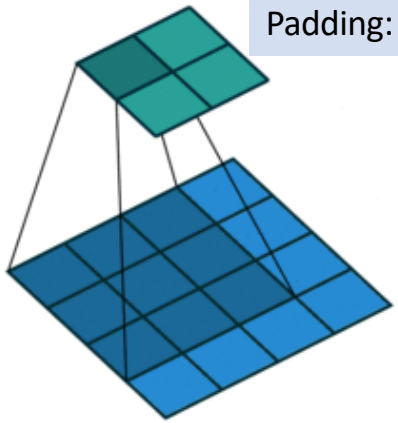


A few hyperparameters

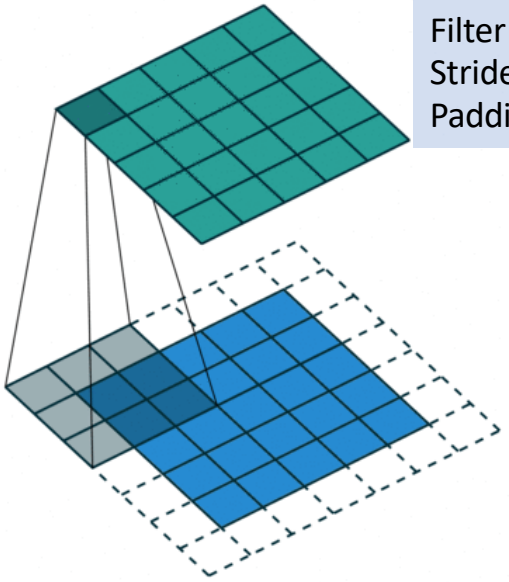
- **Receptive field** (F): the filter/kernel size
- **Stride** (S): the number of pixels by which we slide the filters
- **Zero-padding** (P): the number of zeros are padded at the border

Output (Green)

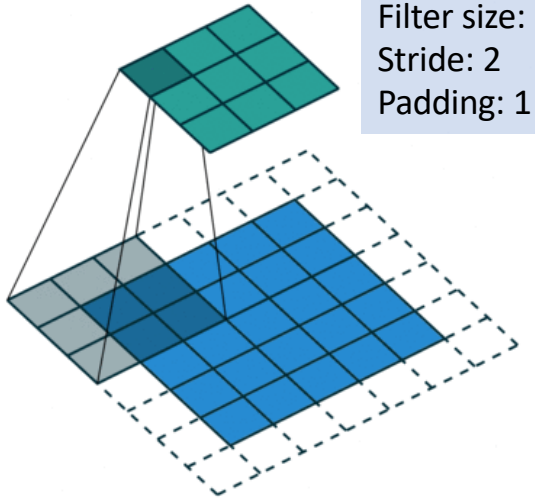
Input (Blue)



Filter size: 3x3
Stride: 1
Padding: 0



Filter size: 3x3
Stride: 1
Padding: 1

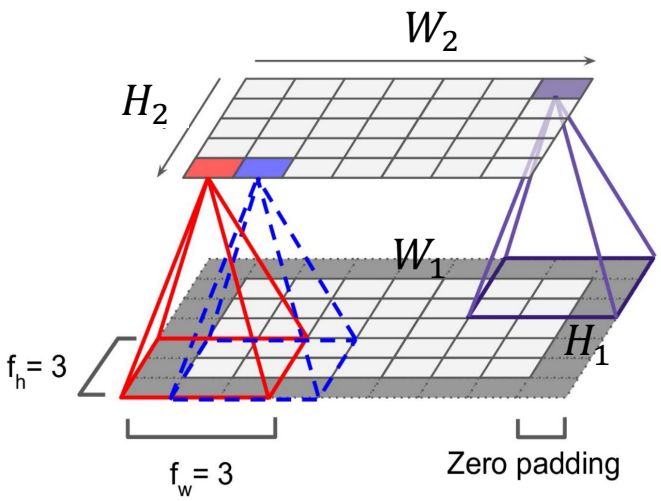


Filter size: 3x3
Stride: 2
Padding: 1

Animations from: https://github.com/vdumoulin/conv_arithmetic

A few hyperparameters

- **Receptive field** (F): the filter/kernel size
- **Stride** (S): the number of pixels by which we slide the filters
- **Zero-padding** (P): the number of zeros are padded at the border



$$W_2 = (7 - 3 + 2\times1)/1 + 1 = 7$$

Q: What is output feature map's size (W_2)?

A: $W_2 = (W_1 - F + 2P)/S + 1$

Image from: Aurelien Geron. Hands-On Machine Learning with Scikit-Learn and TensorFlow. O'Reilly. 2019.

The output volume

Given

- The volume of one layer: $W_1 \times H_1 \times D_1$
- Filter size: F
- Number of filters: K
- Stride: S
- Padding: P

Q: What is the volume of the next layer?

A: $W_2 = (W_1 - F + 2P)/S + 1$

$H_2 = (H_1 - F + 2P)/S + 1$

$D_2 = K$

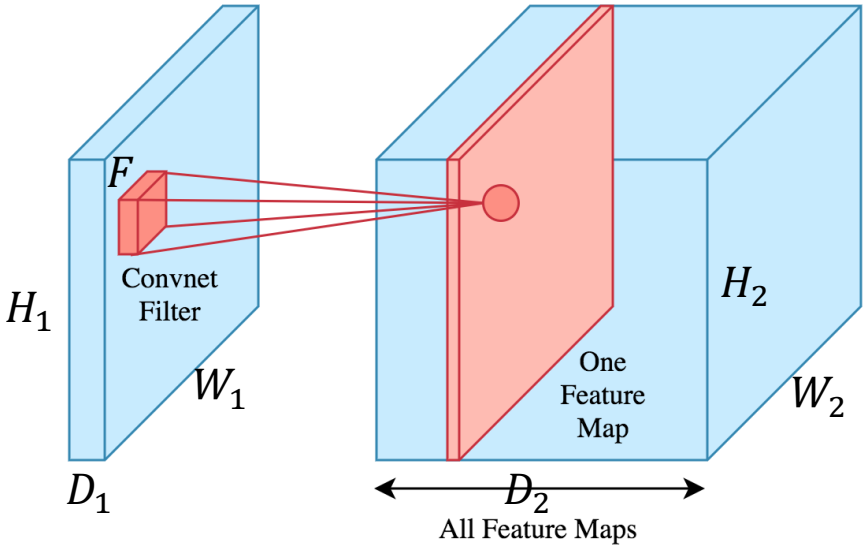


Image from: https://brilliant.org/wiki/convolutional-neural-network/

The output volume - Example

Example:

- The volume of input layer: $36 \times 36 \times 3$
- Filter size: $F = 11$
- Number of filters: $K = 9$
- Stride: $S = 1$
- Padding: $P = 0$

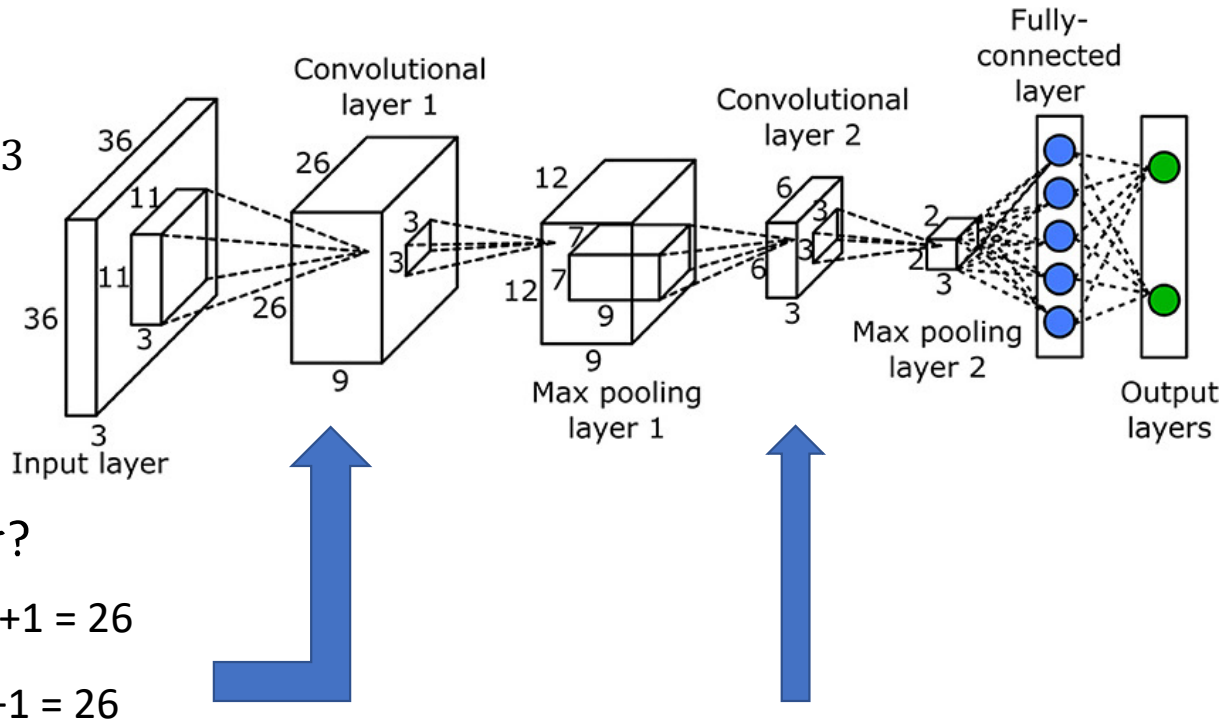
Q: What is the volume of the next layer?

Image from: https://brilliant.org/wiki/convolutional-neural-network/

The output volume - Example

Example:

- The volume of input layer: 36×36×3
- Filter size: $F = 11$
- Number of filters: $K = 9$
- Stride: $S = 1$
- Padding: $P = 0$



Q: What is the volume of the next layer?

A: $W_2 = (W_1 - F + 2P)/S + 1$ = (36-11+0)/1+1 = 26

$H_2 = (H_1 - F + 2P)/S + 1$ = (36-11+0)/1+1 = 26

$D_2 = K = 9$

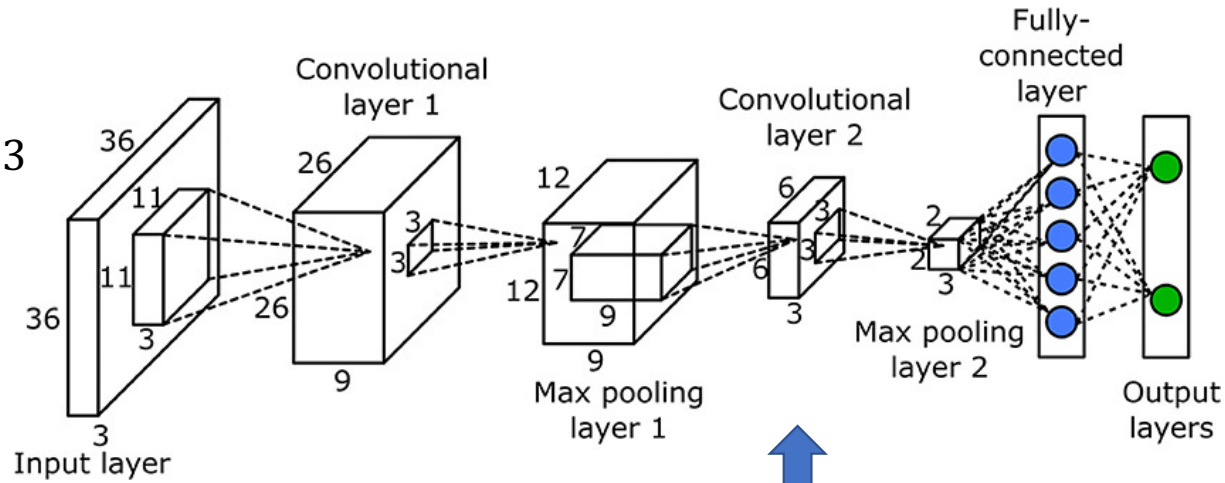
Q: Can you work out how this size 6 x 6 x 3 comes from?

Image from: https://brilliant.org/wiki/convolutional-neural-network/

How many parameters?

Example:

- The volume of input layer: 36×36×3
- Filter size: $F = 11$
- Number of filters: $K = 9$
- Stride: $S = 1$
- Padding: $P = 0$



Q: How many parameters needed for connecting the first two layers?

A: $K \cdot F \cdot F \cdot D_1 + K$ = 9 × 11 × 11 × 3 + 9 = 3,267

One bias term for each filter

Q: Can you work out how many parameters needed for producing Conv-2?

Image from: https://brilliant.org/wiki/convolutional-neural-network/

Summary of convolutional layer

- Use multiple filters to extract different local features
- Preserve the spatial relationship
- Local connectivity
- Parameter sharing → Use less number of parameters

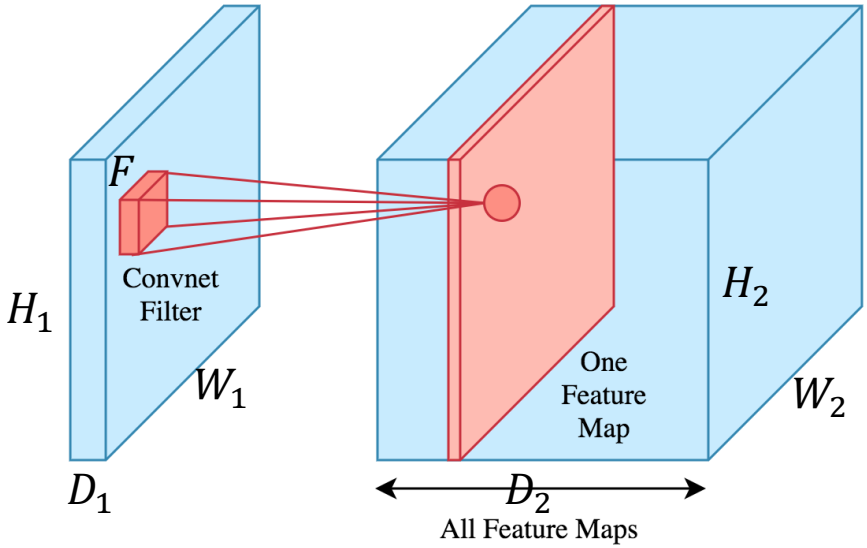


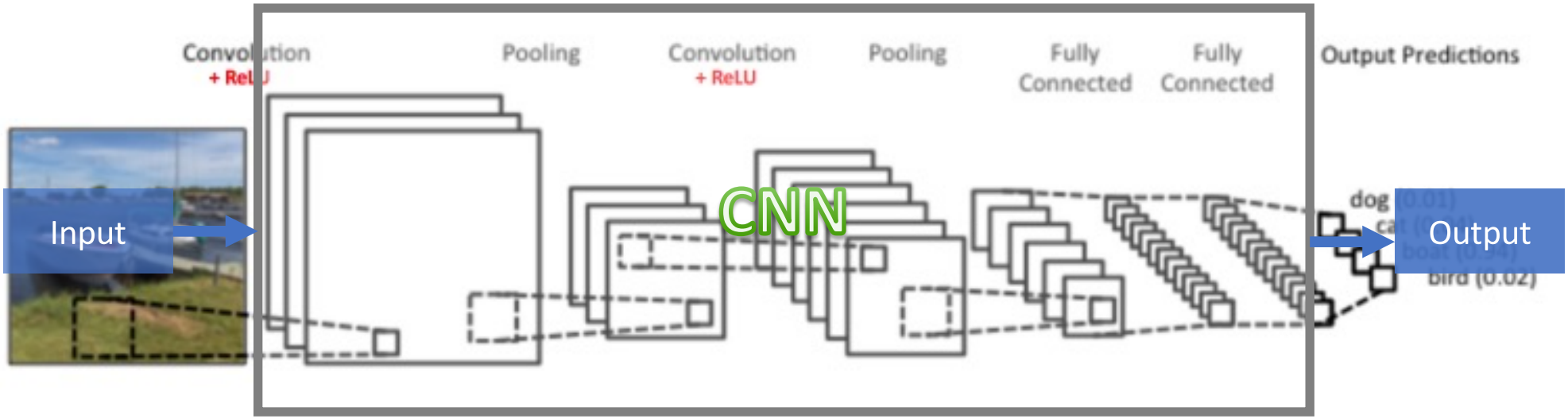
Image from: https://brilliant.org/wiki/convolutional-neural-network/

Reference for Topic 3

- Video lecture by Brandon Rohrer: How Convolutional Neural Networks work: https://www.youtube.com/watch?v=FmpDIaiMIeA
- Video lecture by Alexander Amini: MIT course on Convolutional Neural Networks: https://www.youtube.com/watch?v=iaSUYvmCekI
- Arden Dertat's Blog: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2
- Stanford course on CNNs: https://cs231n.github.io/convolutional-networks/

Topic 4:
CNN: ReLU Layer & Pooling Layer

Image recognition with CNN



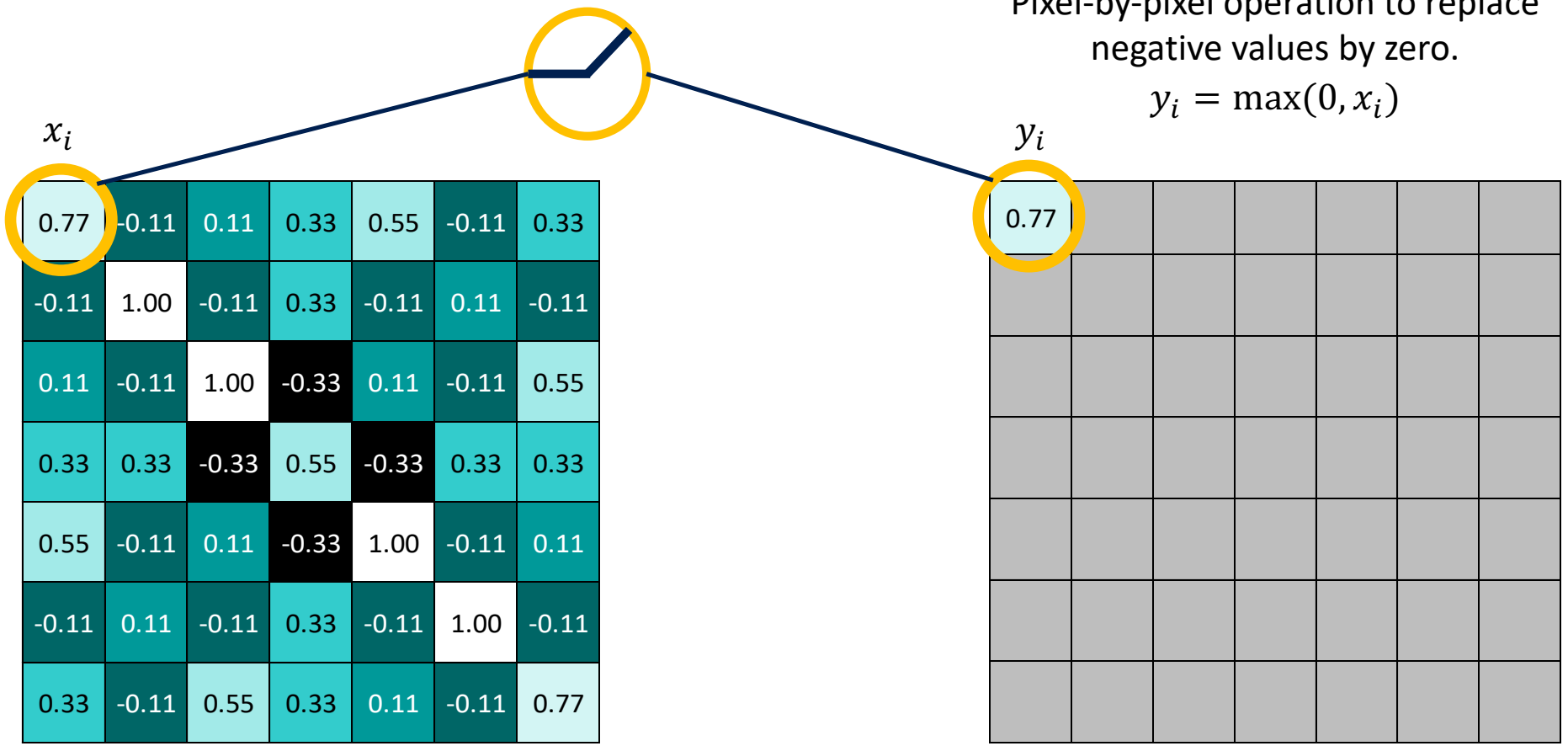
- Convolutional Layer
- ReLU Layer
- Pooling Layer
- Fully Connected Layer

Image from: https://towardsdatascience.com/build-your-own-convolution-neural-network-in-5-mins-4217c2cf964f

Rectified Linear Unit (ReLU)

Pixel-by-pixel operation to replace negative values by zero.

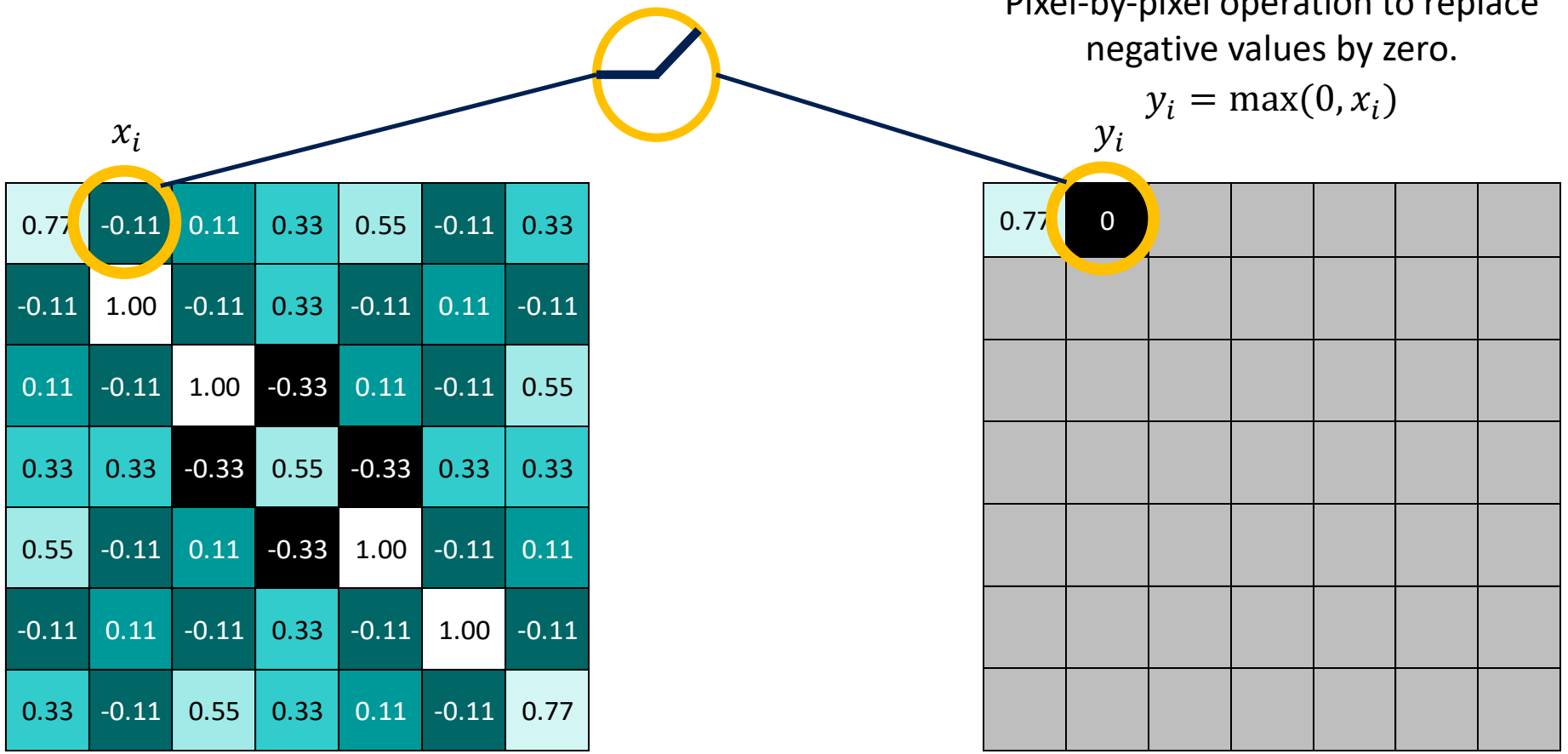
$$y_i = \max(0, x_i)$$



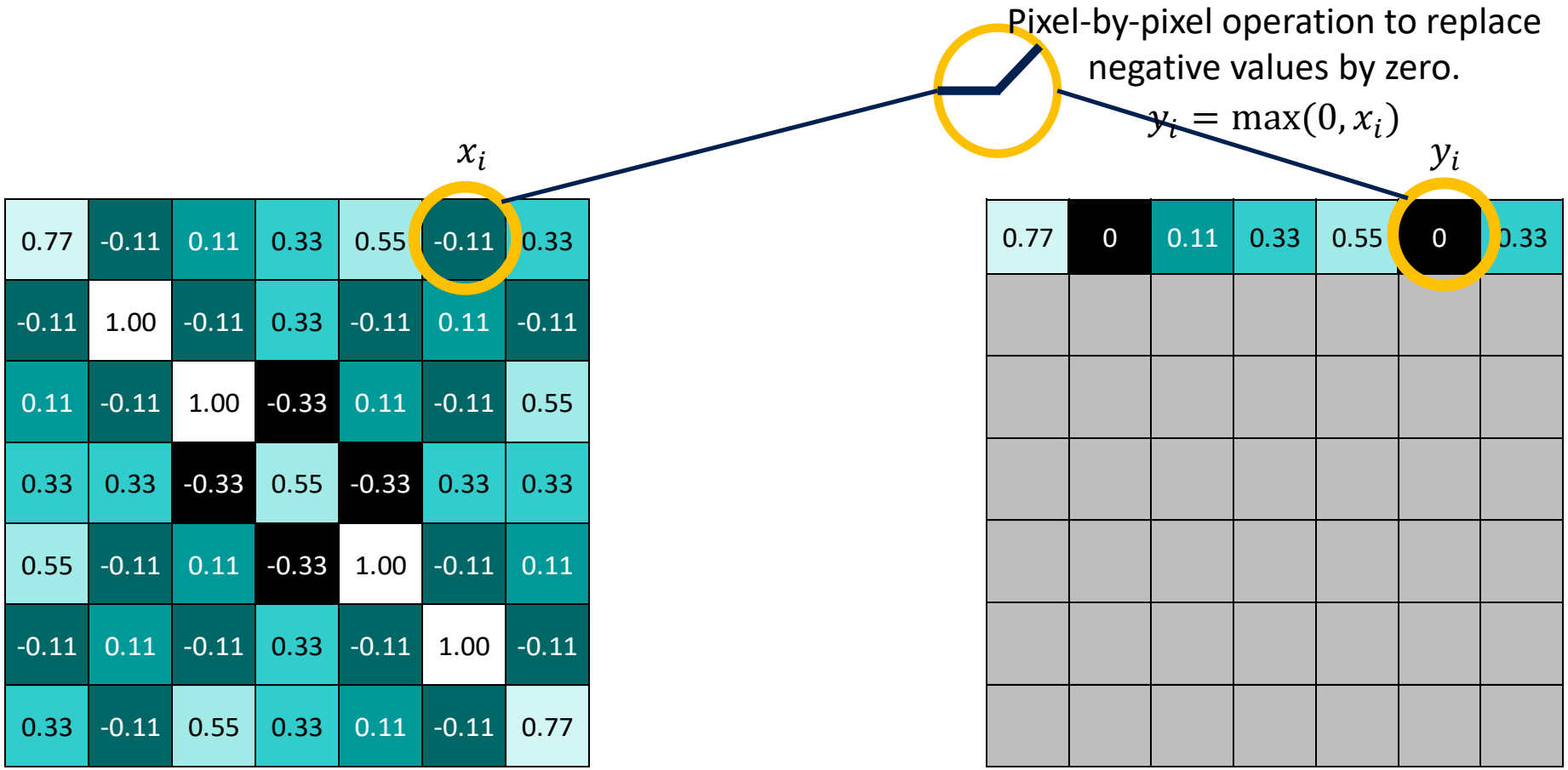
Rectified Linear Unit (ReLU)

Pixel-by-pixel operation to replace negative values by zero.

$$y_i = \max(0, x_i)$$



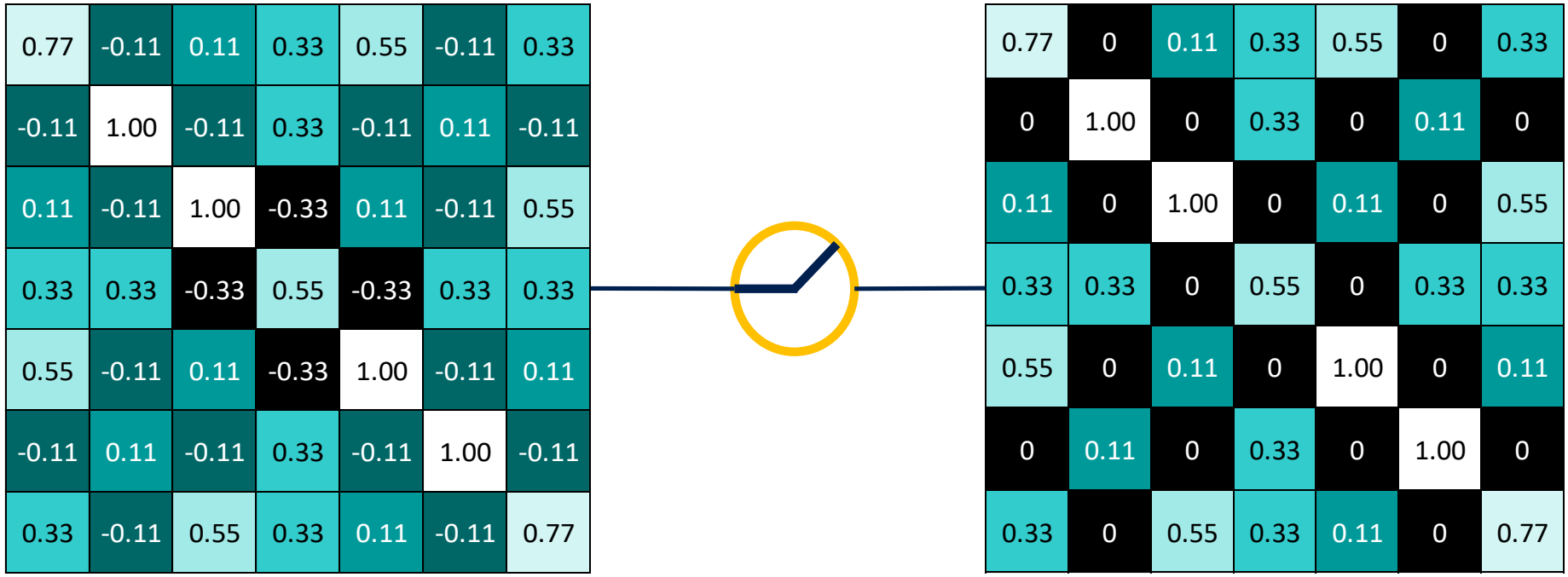
Rectified Linear Unit (ReLU)



Rectified Linear Unit (ReLU)

Pixel-by-pixel operation to replace negative values by zero.

$$y_i = \max(0, x_i)$$



Summary of ReLU layer

- Introduce non-linearity in the CNN network
- Apply after every convolutional operation

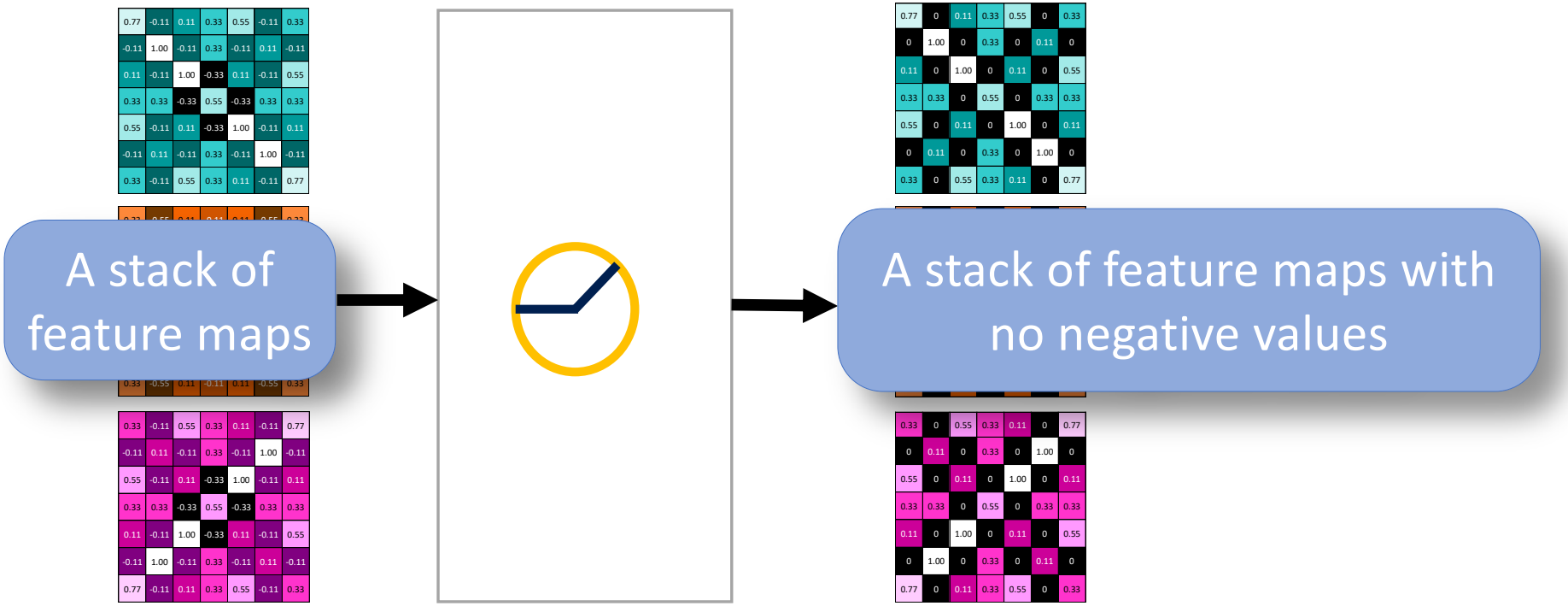
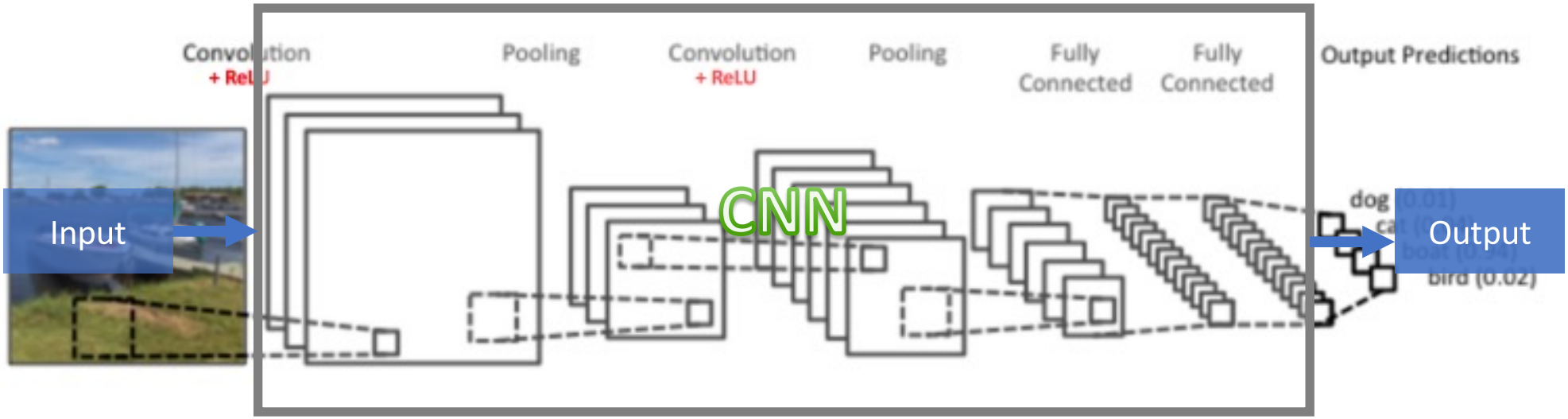


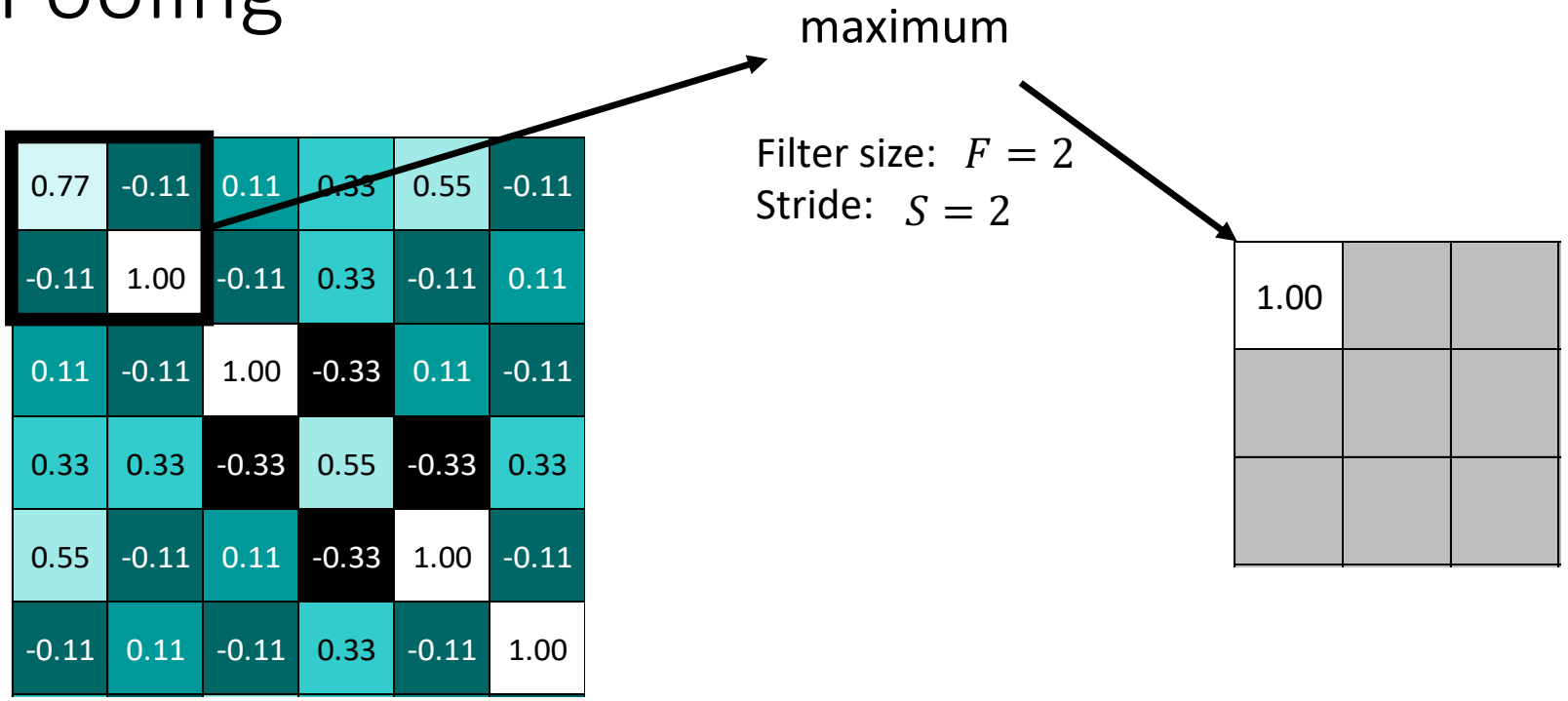
Image classification with CNN



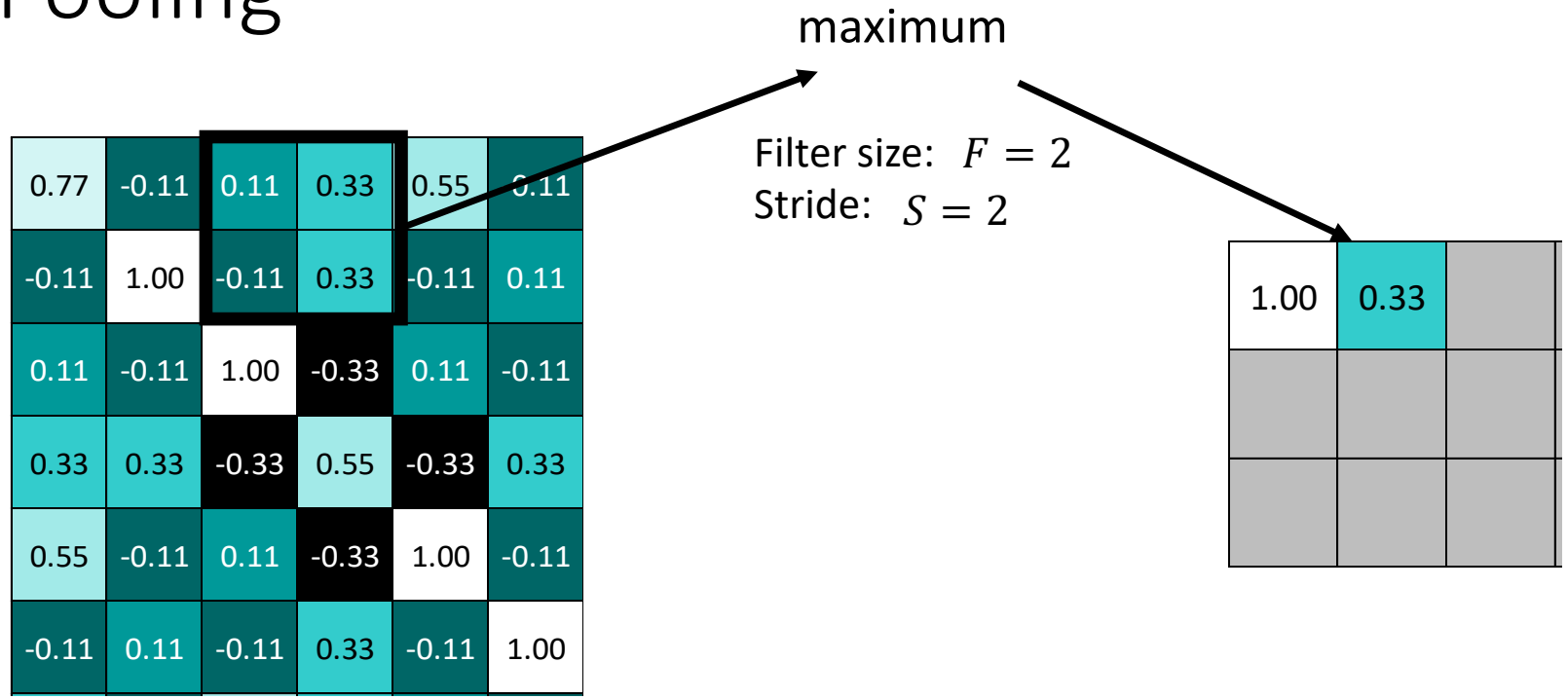
- Convolutional Layer
- ReLU Layer
- Pooling Layer
- Fully Connected Layer

Image from: https://towardsdatascience.com/build-your-own-convolution-neural-network-in-5-mins-4217c2cf964f

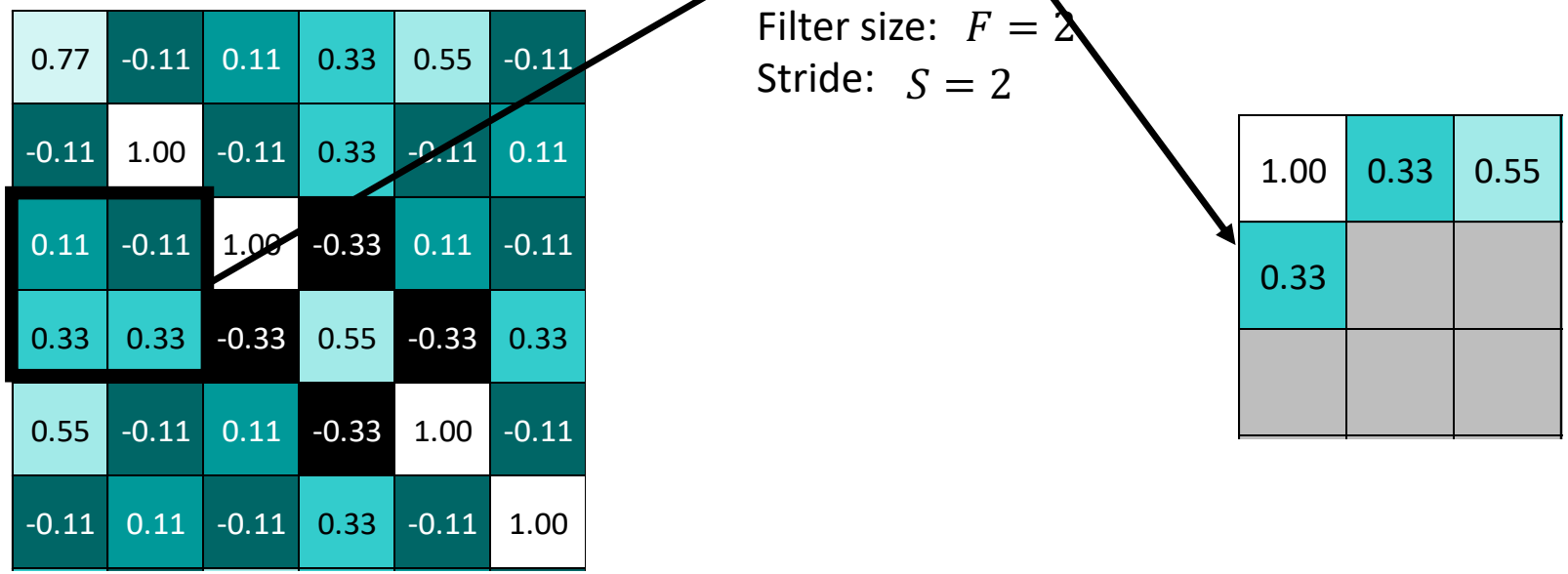
Pooling


maximum
Filter size: $F = 2$
Stride: $S = 2$
0.77 -0.11 0.11 0.33 0.55 -0.11
-0.11 1.00 -0.11 0.33 -0.11 0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11
0.33 0.33 -0.33 0.55 -0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00
1.00

Pooling


maximum
Filter size: $F = 2$
Stride: $S = 2$
0.77 -0.11 0.11 0.33 0.55 0.11
-0.11 1.00 -0.11 0.33 -0.11 0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11
0.33 0.33 -0.33 0.55 -0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00
1.00 0.33

Pooling

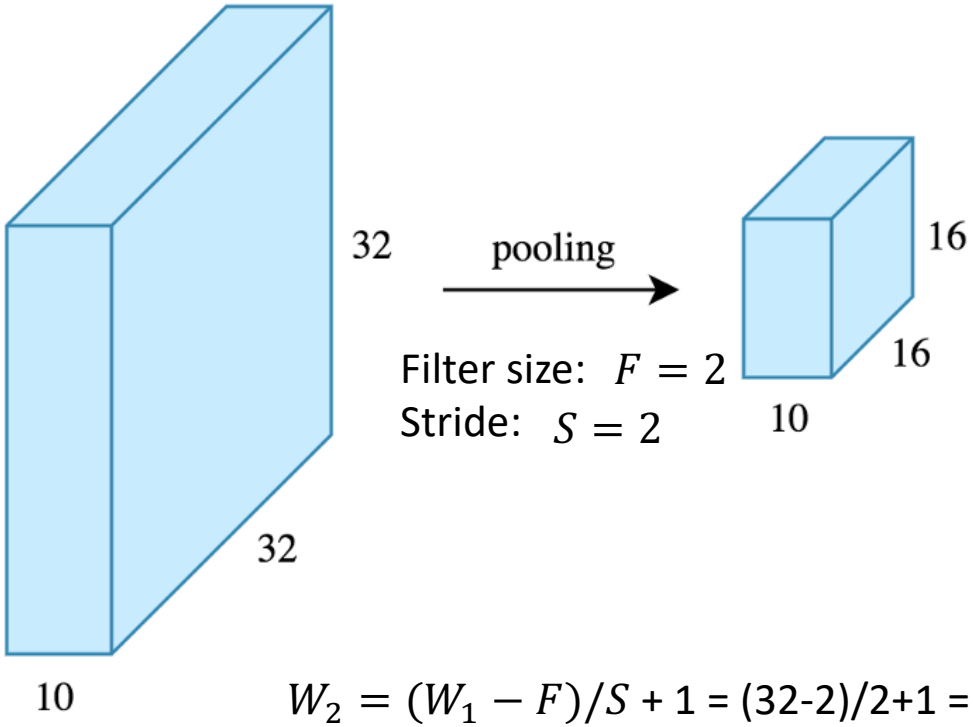

maximum
Filter size: $F = 2$
Stride: $S = 2$
0.77 -0.11 0.11 0.33 0.55 -0.11
-0.11 1.00 -0.11 0.33 -0.11 0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11
0.33 0.33 -0.33 0.55 -0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00
1.00 0.33 0.55
0.33

Pooling

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

1.00	0.33	0.55
0.33	1.00	0.33
0.55	0.33	1.00

The output volume

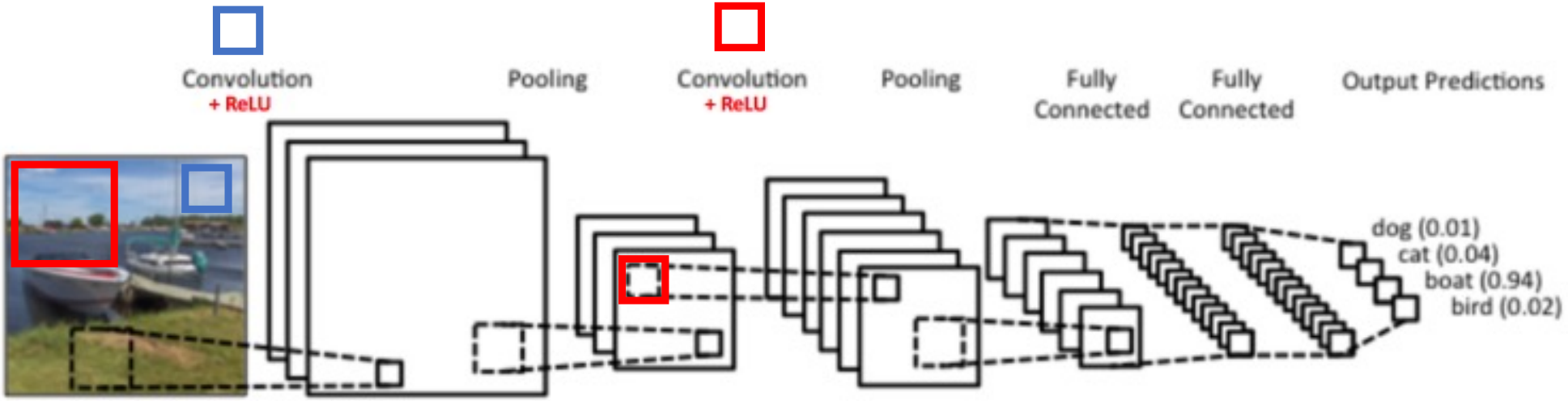


$W_2 = (W_1 - F)/S + 1 = (32\text{-}2)/2+1 = 16$

$H_2 = (H_1 - F)/S + 1 = (32\text{-}2)/2+1 = 16$

$D_2 = D_1 = 10$

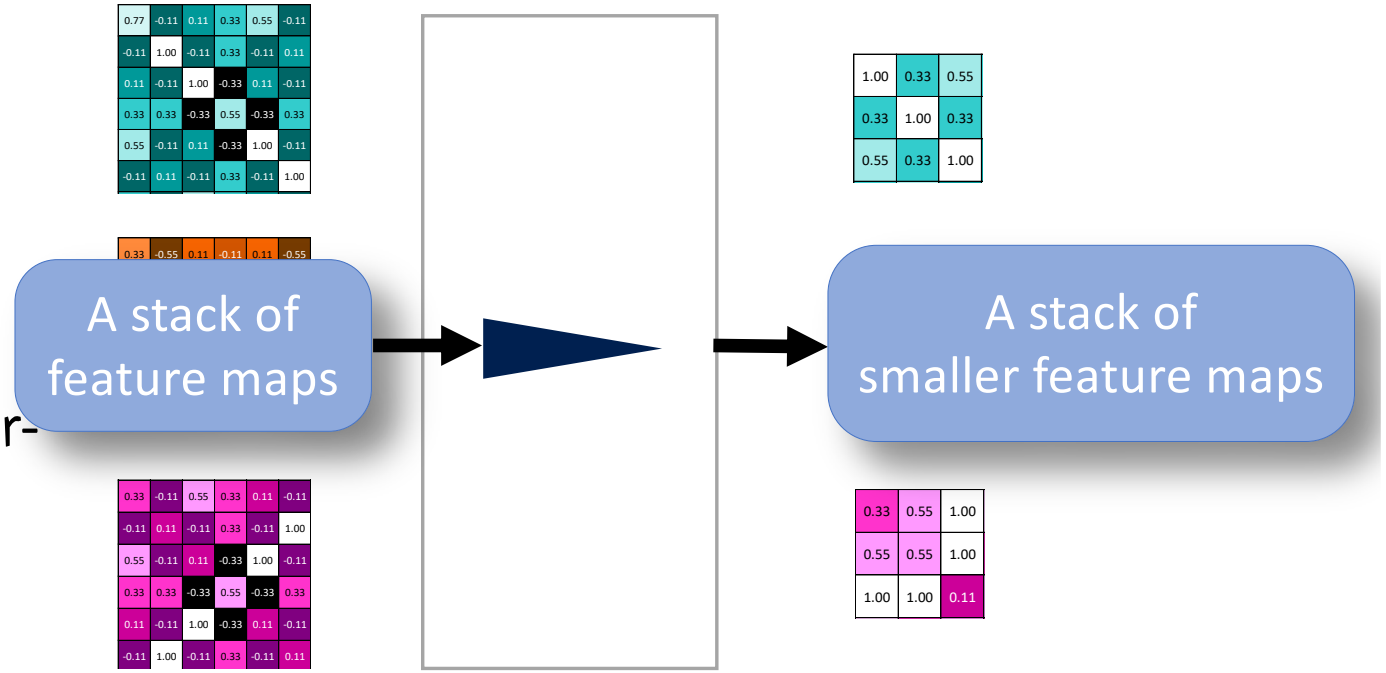
Q: If both the convolutional layers below use a same-sized filter, what is the difference of their receptive field size on the input image?



Because of Pooling, higher-level convolutional filters can cover a **larger region** of the image than equal-sized filters in the lower layers.

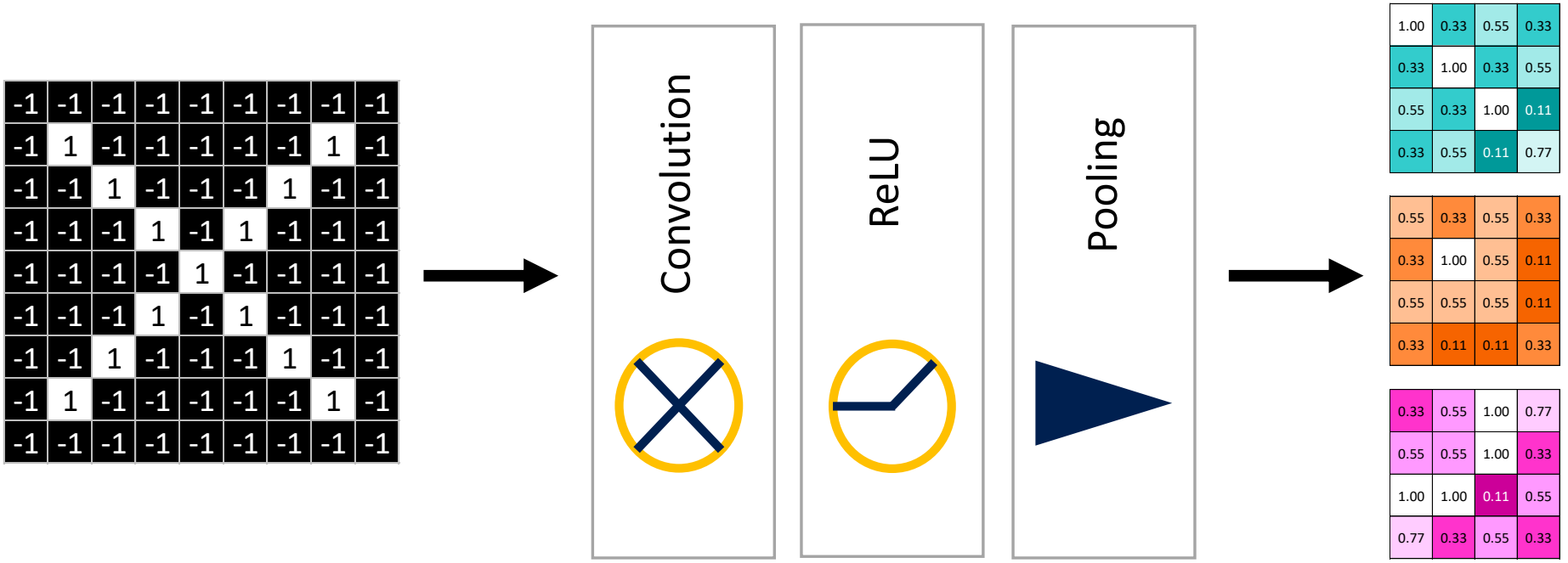
Summary of pooling

- Average pooling; L2-norm pooling; **Max pooling** (most common)
- Need no parameters
- Reduce dimensionality
- Some spatial invariance
- Larger coverage for higher-layer Conv layers, enable them represents higher-level features



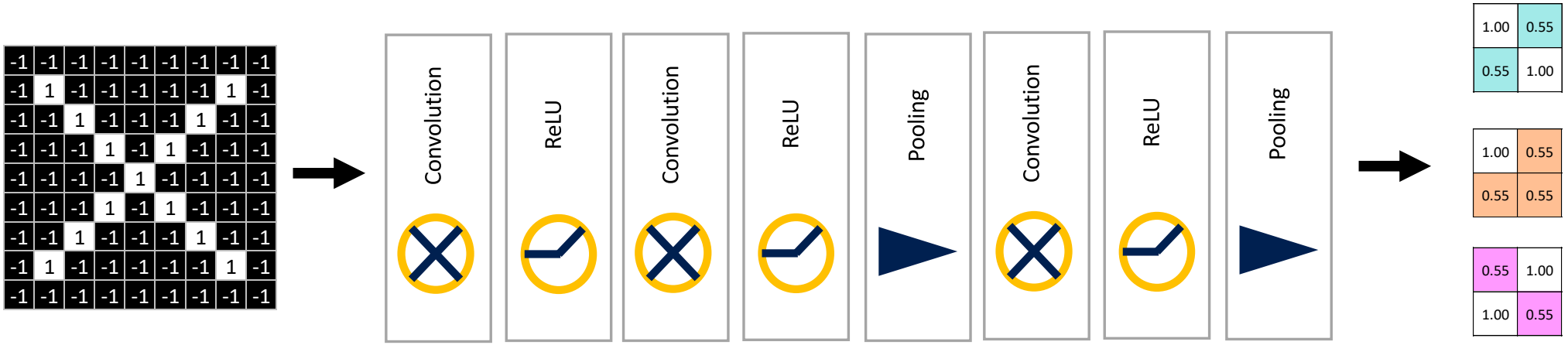
Layers get stacked

The output of one becomes the input of the next.



Deep stacking

Layers can be repeated several (or many) times.

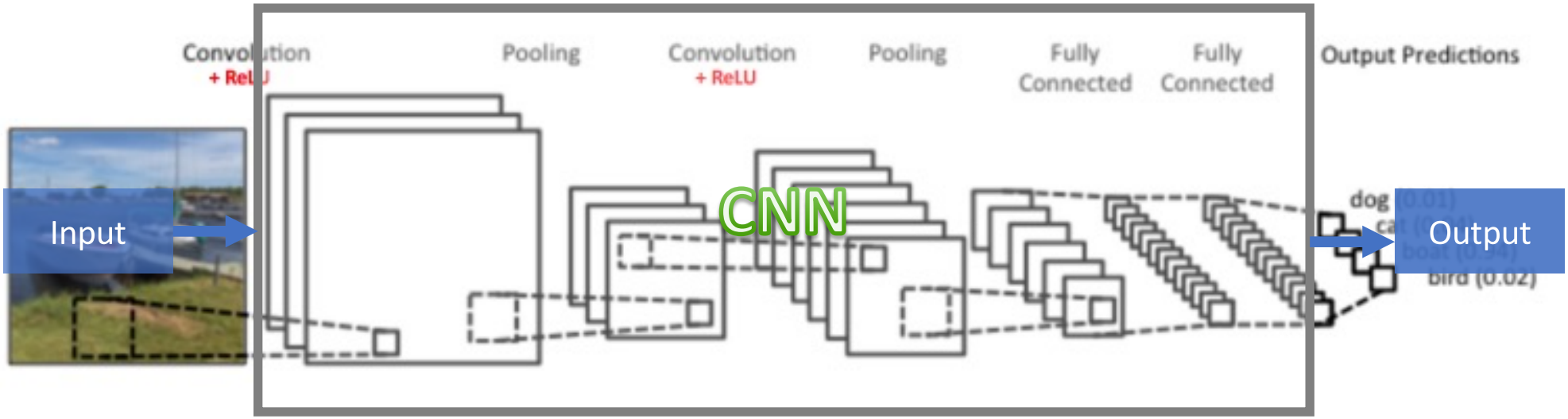


Reference for Topic 4

- Video lecture by Brandon Rohrer: How Convolutional Neural Networks work: https://www.youtube.com/watch?v=FmpDIaiMIeA
- Video lecture by Alexander Amini: MIT course on Convolutional Neural Networks: https://www.youtube.com/watch?v=iaSUYvmCekI
- Arden Dertat's Blog: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2
- Stanford course on CNNs: https://cs231n.github.io/convolutional-networks/

Topic 5:
CNN: Fully Connected (FC) Layer

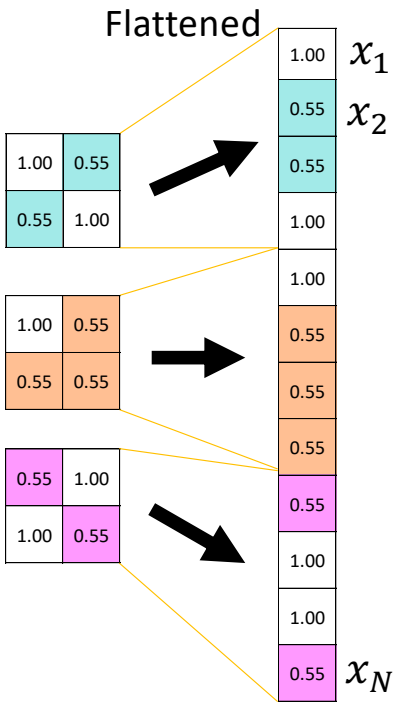
Image classification with CNN



- Convolutional Layer
- ReLU Layer
- Pooling Layer
- Fully Connected Layer

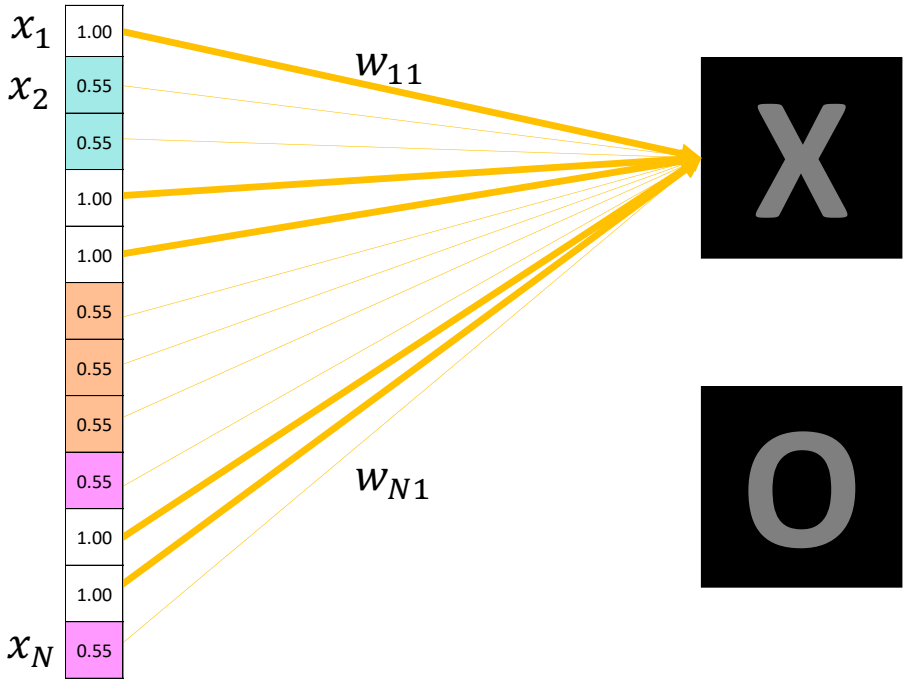
Image from: https://towardsdatascience.com/build-your-own-convolution-neural-network-in-5-mins-4217c2cf964f

Fully connected layer



Fully connected layer

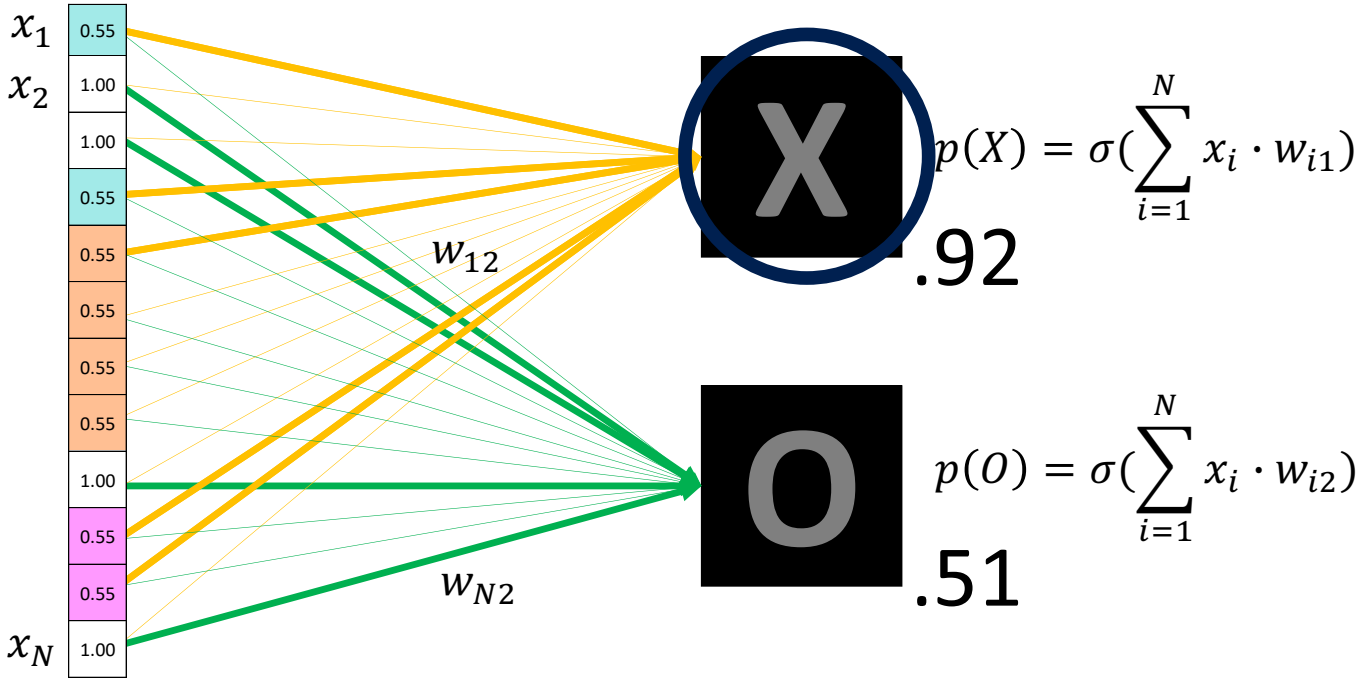
Every feature (x) gets a vote (w)



$$p(X) = \sigma(\sum_{i=1}^{N} x_i \cdot w_{i1})$$

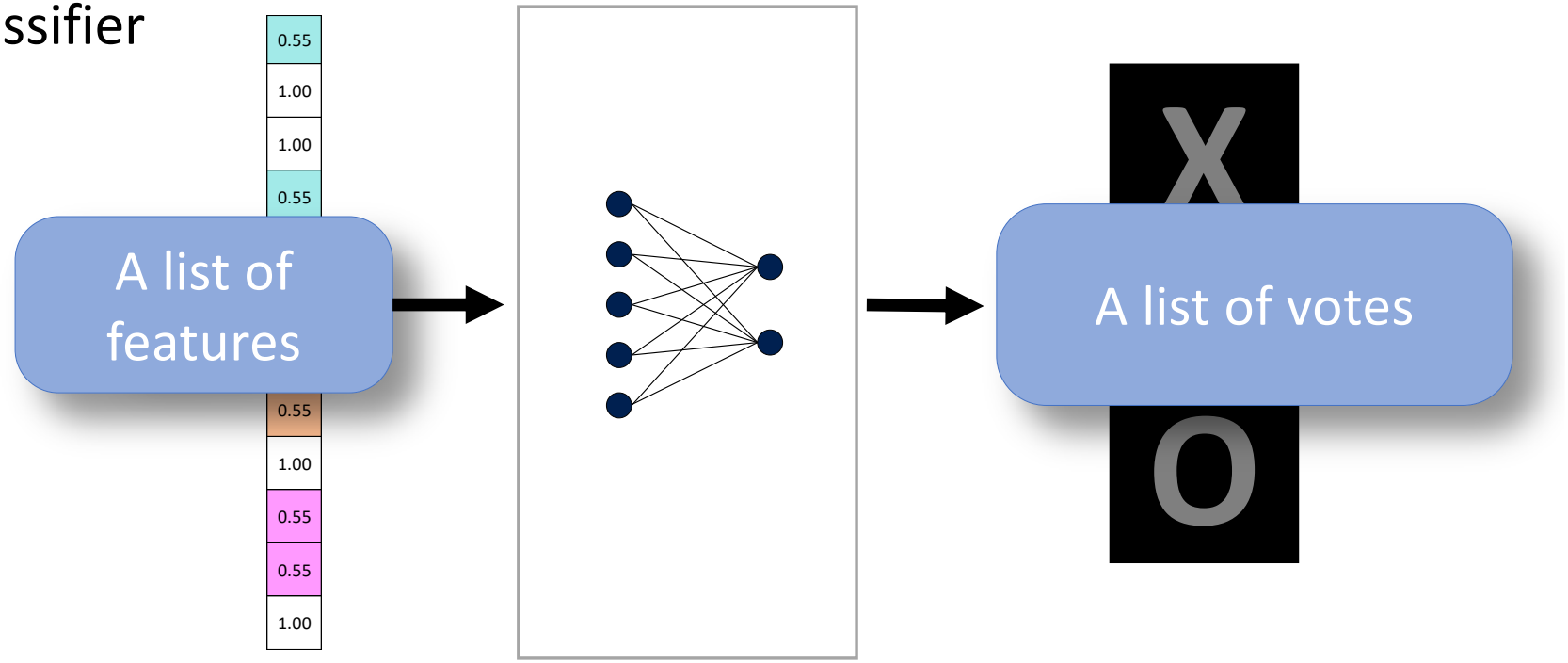
Fully connected layer

Every feature (x) gets a vote (w)



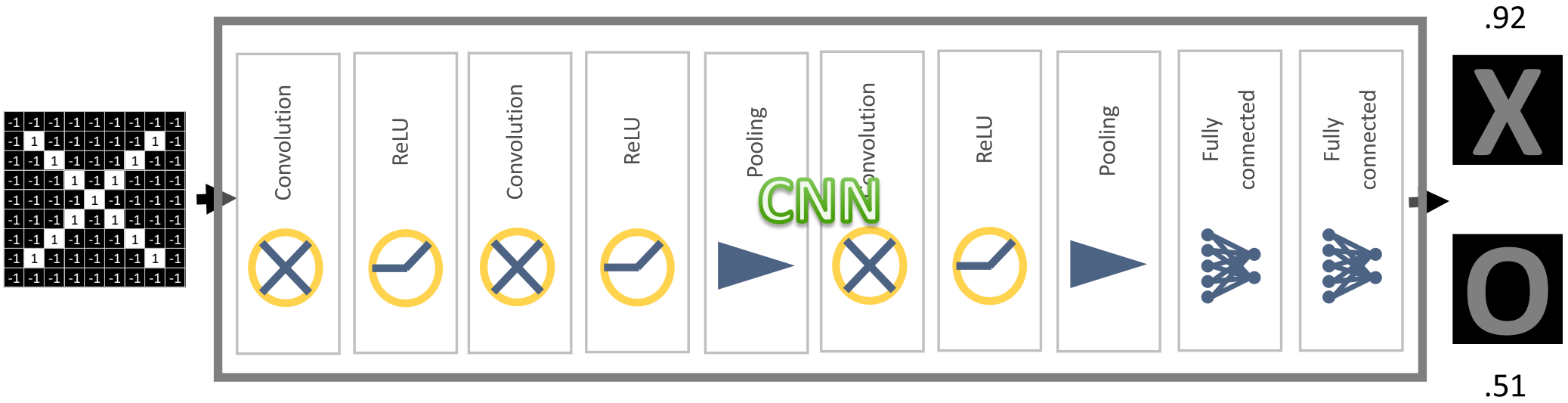
Summary of fully connected (FC) layer

- Usually at the end of the CNN
- Can stack one or more FC layers
- Served as a classifier



Putting it all together

A set of pixels becomes a set of votes.

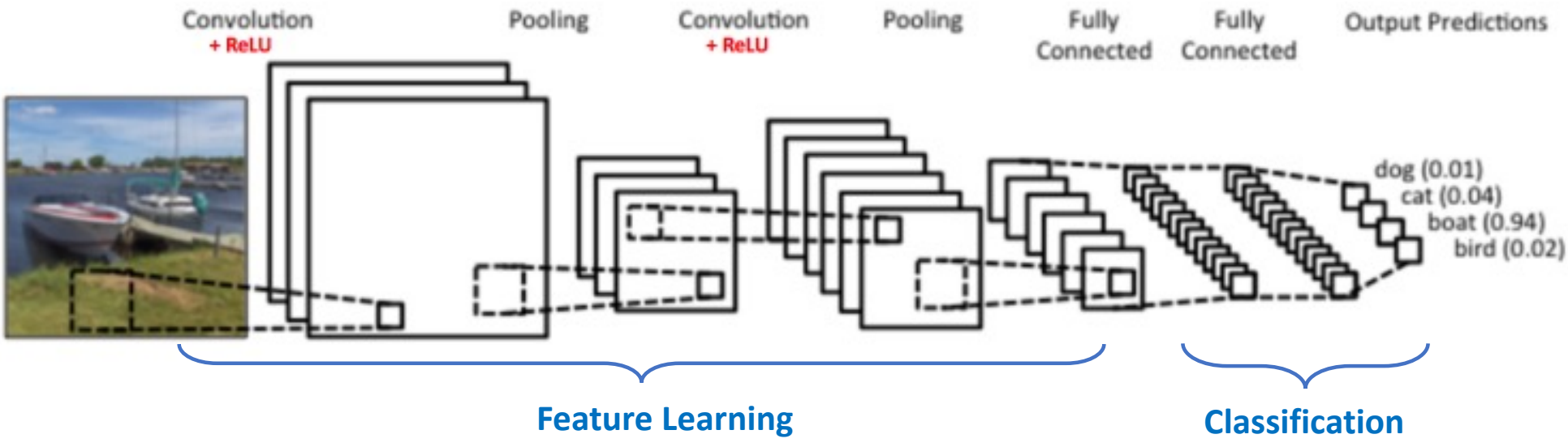


Reference for Topic 5

- Video lecture by Brandon Rohrer: How Convolutional Neural Networks work: https://www.youtube.com/watch?v=FmpDIaiMIeA
- Video lecture by Alexander Amini: MIT course on Convolutional Neural Networks: https://www.youtube.com/watch?v=iaSUYvmCekI
- Arden Dertat's Blog: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2
- Stanford course on CNNs: https://cs231n.github.io/convolutional-networks/

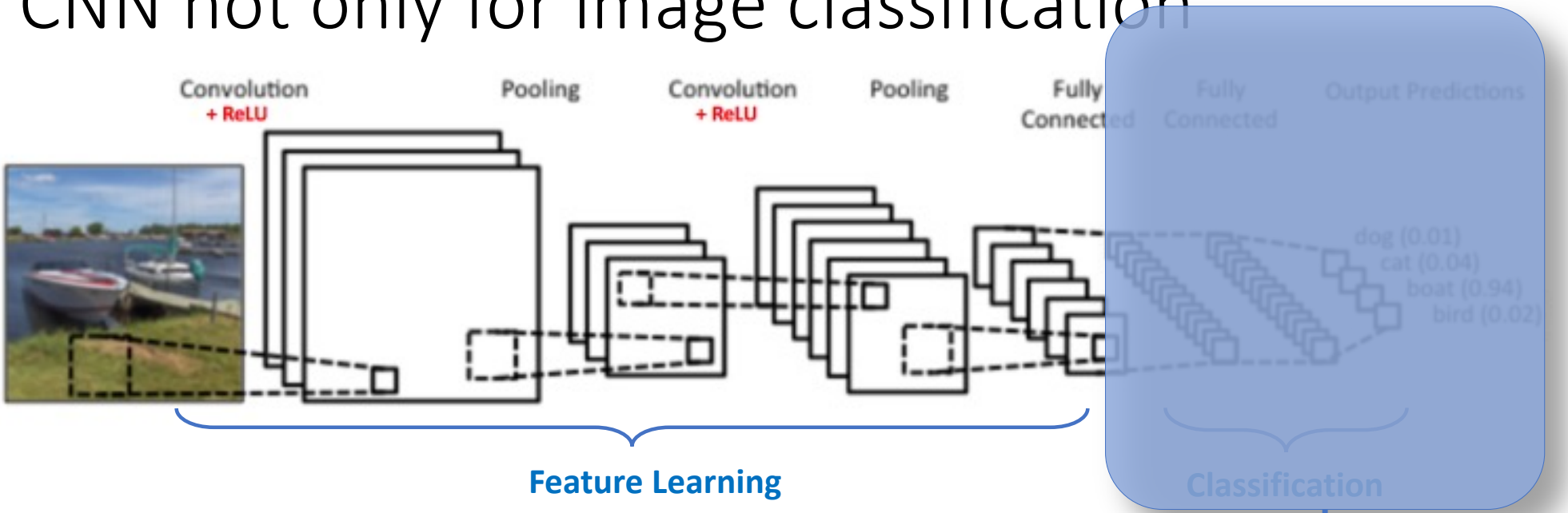
Topic 6:
Summary of CNNs

Structure of CNN



- Convolutional Layer ← Feature extraction (from lower level to higher level)
- ReLU Layer ← Introduce non-linearity
- Pooling Layer ← Reduce dimensionality and preserve spatial invariance
- Fully Connected Layer ← Classification with softmax activation

CNN not only for image classification



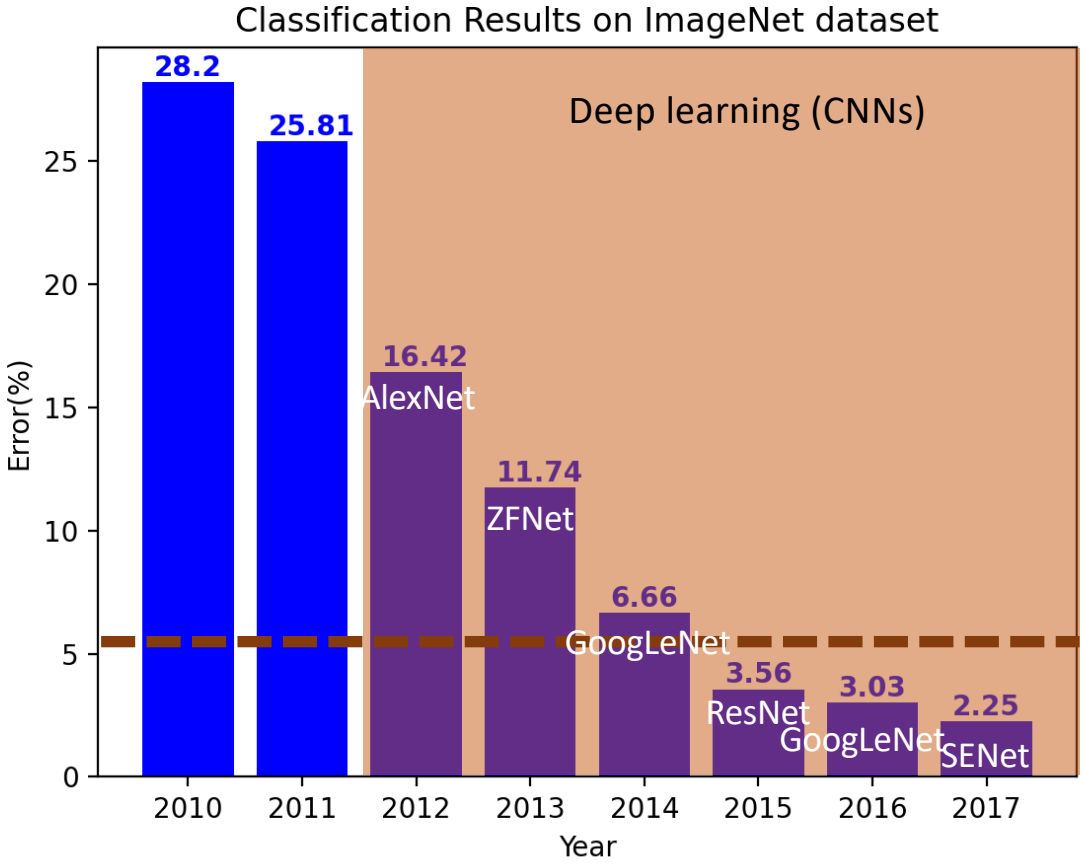
- Object detection
- Semantic segmentation
- Image captioning
- …

Implementation

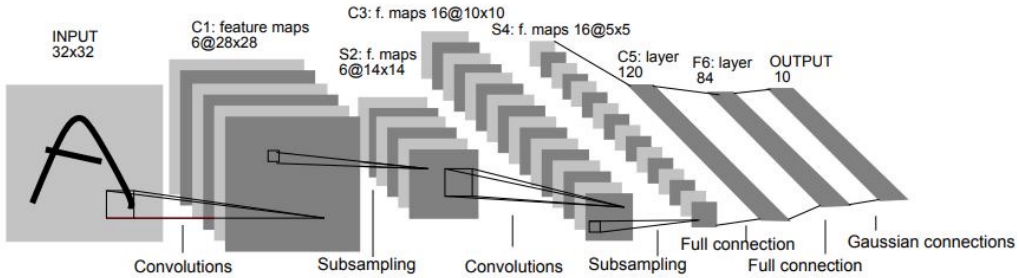
```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten

model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Common CNN architectures



- LeNet-5 (1998): one of the very first CNNs.



LeCun, Y. et al, "Gradient-based learning applied to document recognition".1998.

More details of common CNN architectures:
https://machinelearningknowledge.ai/popular-image-classification-models-in-imagenet-challenge-ilsvrc-competition-history/

Case study: VGGNet

- Proposed by Oxford's Visual Geometry Group (VGG) in 2014
- 7.3% error rate in ILSVRC2014
- 13 conv layers + 3 fully-connected layers – also called **VGG16**
- **Q**: All stacked 3x3 conv layers, *why*?

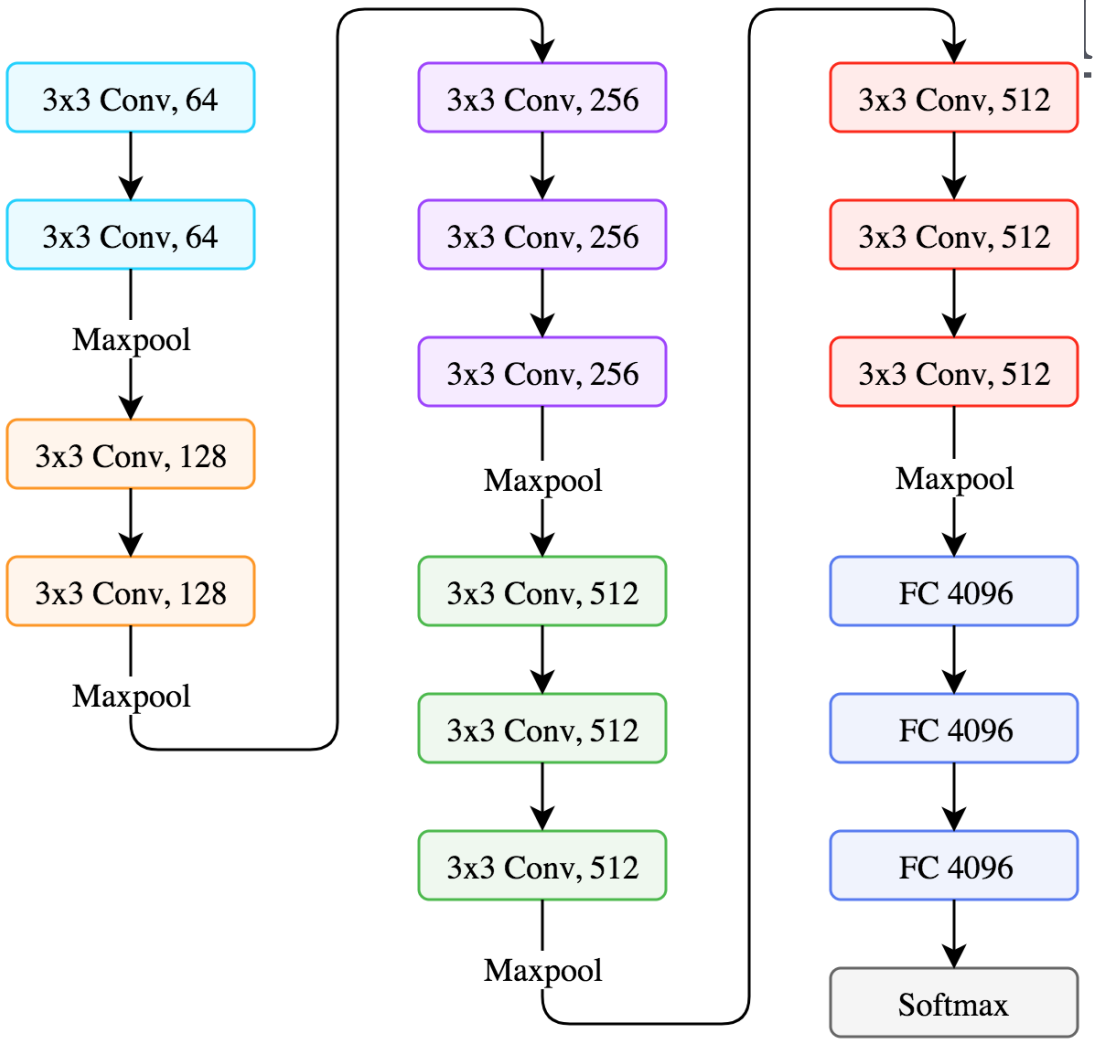
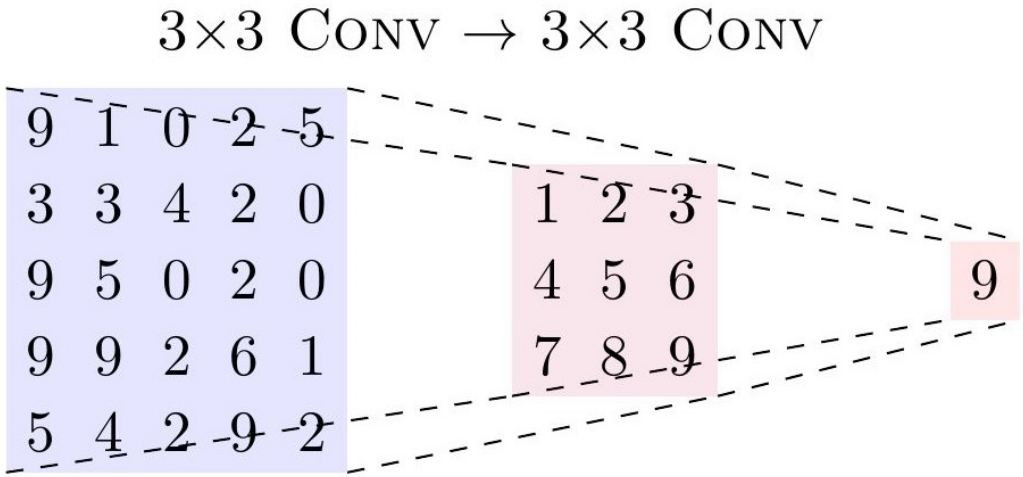


Image from: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Case study: VGGNet



- Two 3x3 conv layers has an effective receptive field of 5x5.
- Benefit for using two 3x3 conv layers:
  - a decrease in the number of parameters
  - two ReLU layers, and more non-linearity gives more power to the model

Case study: VGGNet

- Three 3x3 conv layers has an effective receptive field of 7x7, but less parameters, more non-linearity power.
- Prefer **a stack of small filter CONV** to one large receptive field CONV layer.

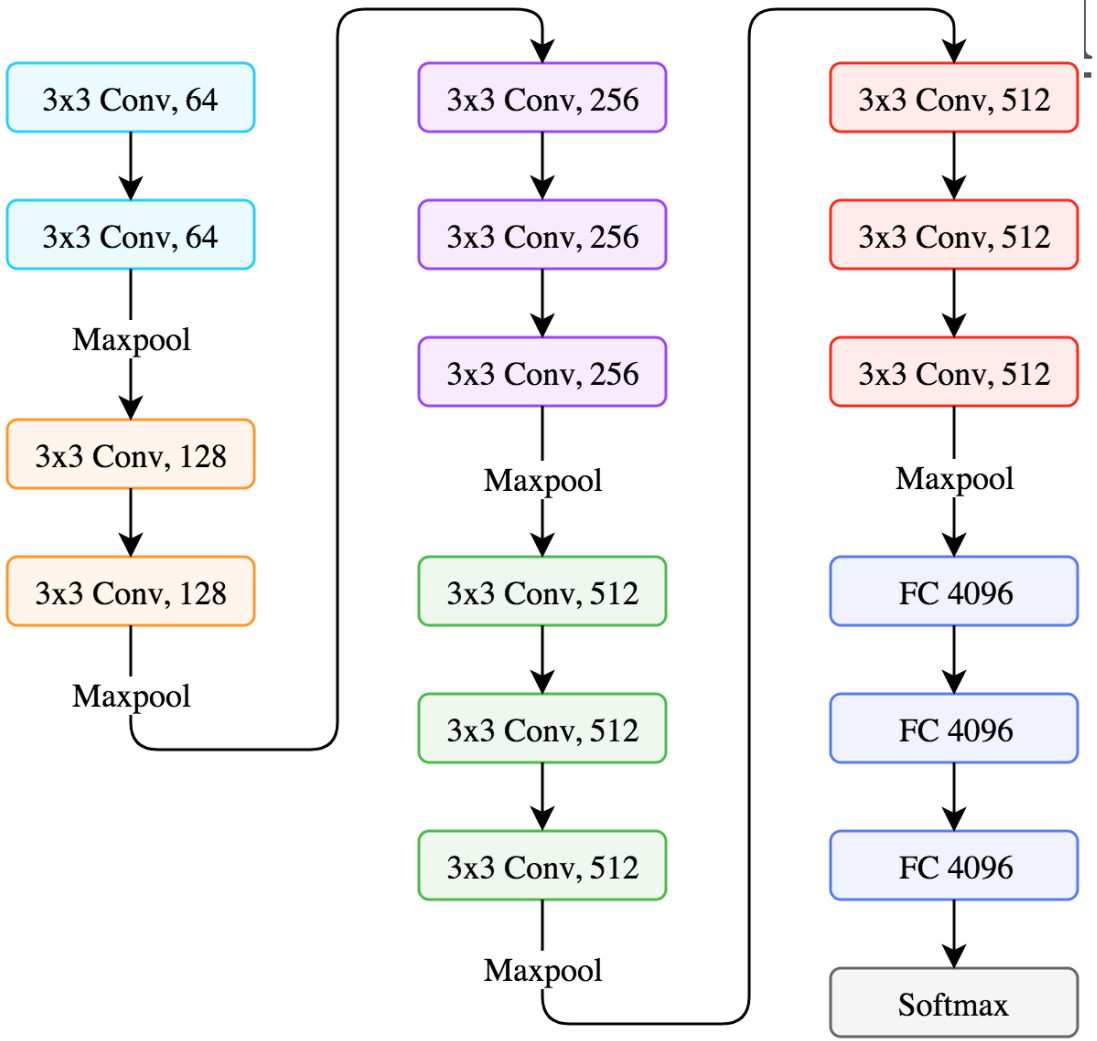


Image from: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Exercise

Search online the most popularly used CNN architectures. Choose at least 5 CNN models. For each model, give a brief introduction to it. What is the architecture? What are the main characteristics?

Reference for Topic 6

- Video lecture by Alexander Amini: MIT course on Convolutional Neural Networks: https://www.youtube.com/watch?v=iaSUYvmCekI
- Arden Dertat's Blog: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2
- Stanford course on CNNs: https://cs231n.github.io/convolutional-networks/
- https://machinelearningknowledge.ai/popular-image-classification-models-in-imagenet-challenge-ilsvrc-competition-history/