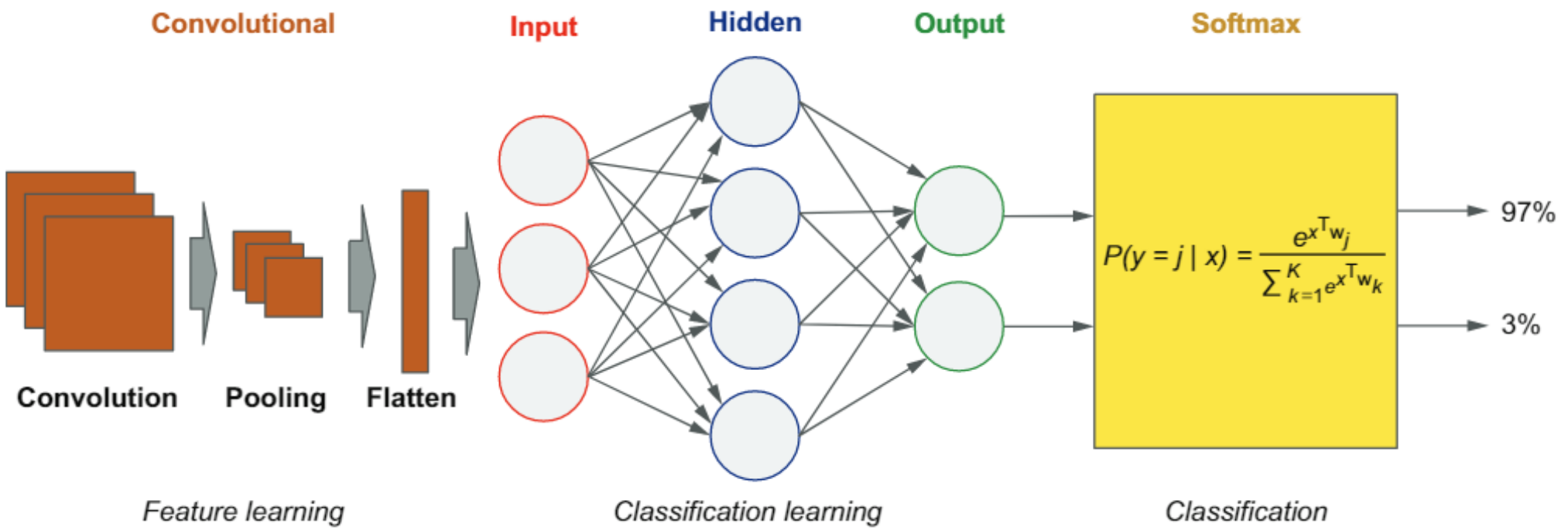
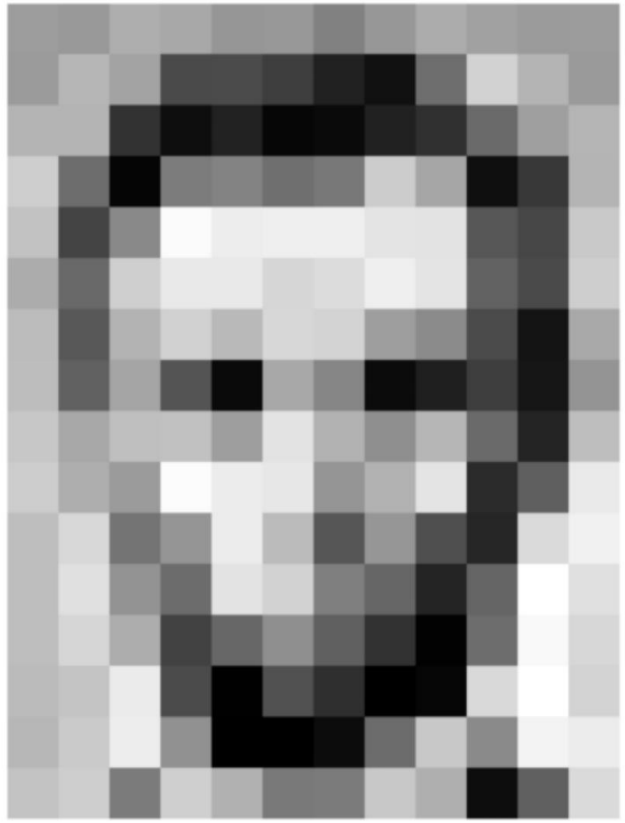
CSE 465
Lecture 5 & 6

CNN – Convolutional Neural Network

CNN in a nutshell



Images/videos are just matrix with numbers

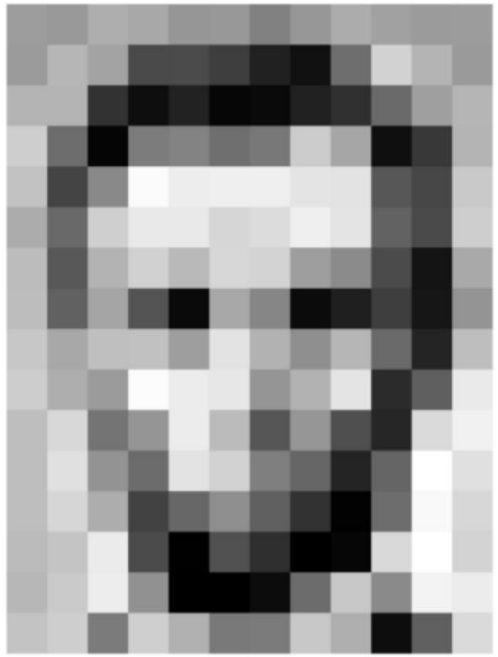


157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

What the computer sees

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

What do we want to do?



Input Image

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
196	206	123	207	177	121	123	200	175	13	96	218

Pixel Representation

classification

Lincoln	0.8
Washington	0.1
Jefferson	0.05
Obama	0.05

How to categorize images?

- Use features

- Rectangular shape
- Has bevels
- There is a logo of apple

- Long thick lines
- Hanging ropes
- Structure hanging over poles

- Has nose
- Has long ears
- Has eyes
- White color

Manual feature detection

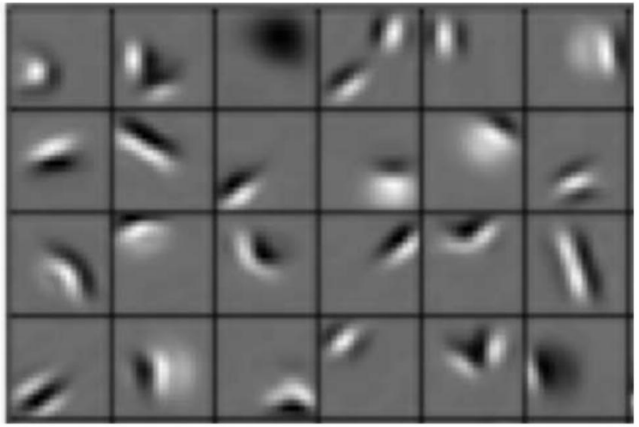
- Must have domain knowledge
  - Have previous experiences with best practices
- Define features
  - Define previously known features
  - Generate new features
- Use the features to classify
  - The classifier will use the features

Manual feature detection: Problems

- Occlusion
  - Blocking partially
- Different illumination
  - Amount of light/brightness change
- Scale variation/deformation
- Viewpoint variation

How can we use machine to learn features?

Low level features



Edges, dark spots

Mid level features

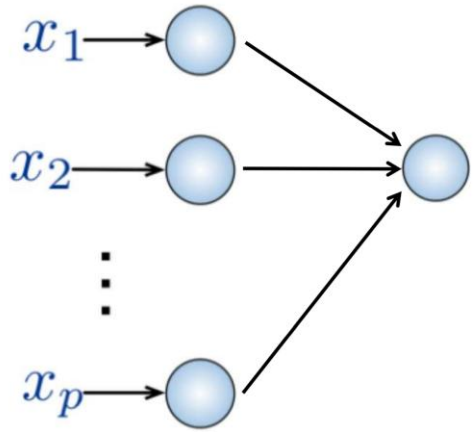
Eyes, ears, nose

High level features

Facial structure

Implementation so far

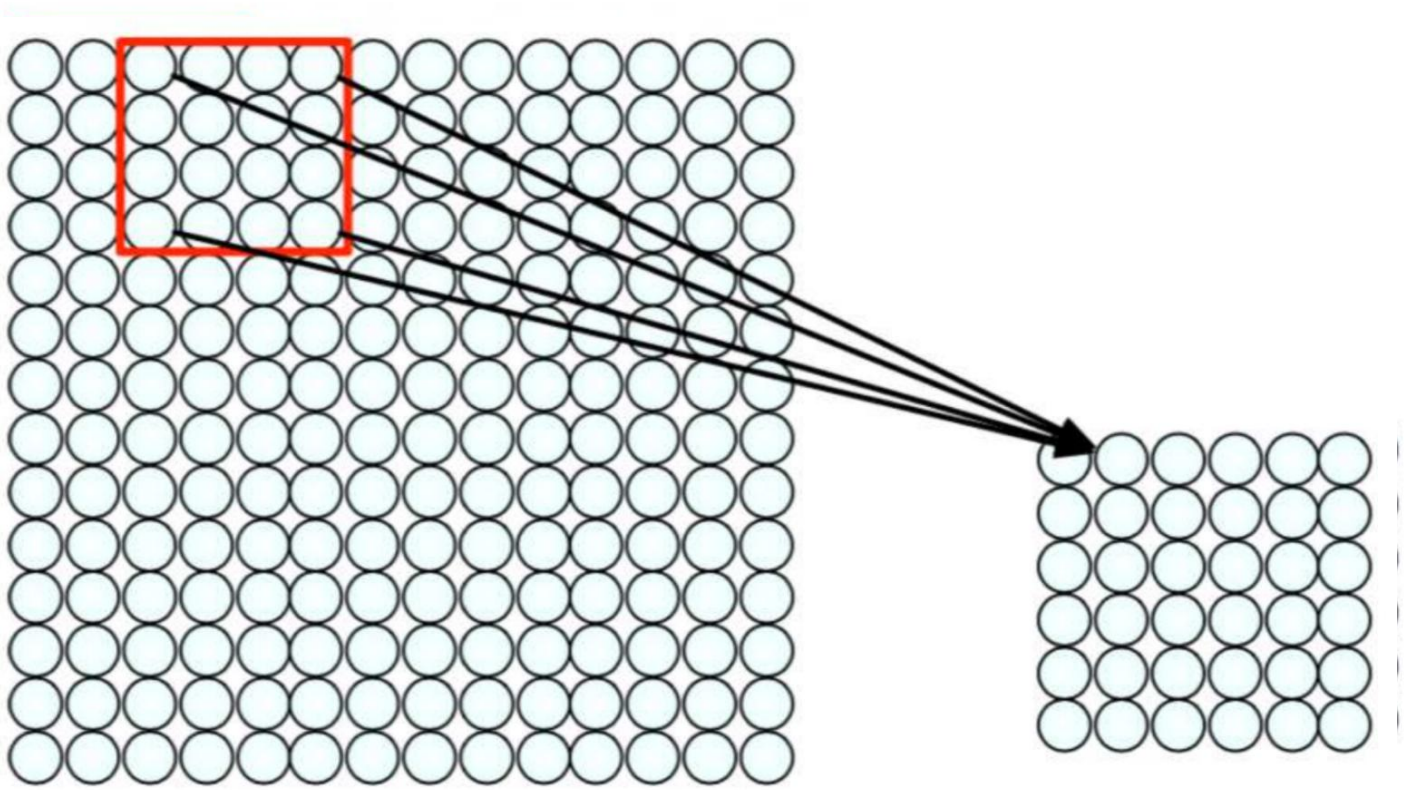
- Use fully connected feed-forward neural network with many layers
  - Hopefully each layer will learn some important features
  - And, finally we will be able to represent the correct function
  - However, no spatial information
  - Many, Many features
- Input
  - Flattened 1-D vector of image representing numbers
- However, important spatial information gone!!!



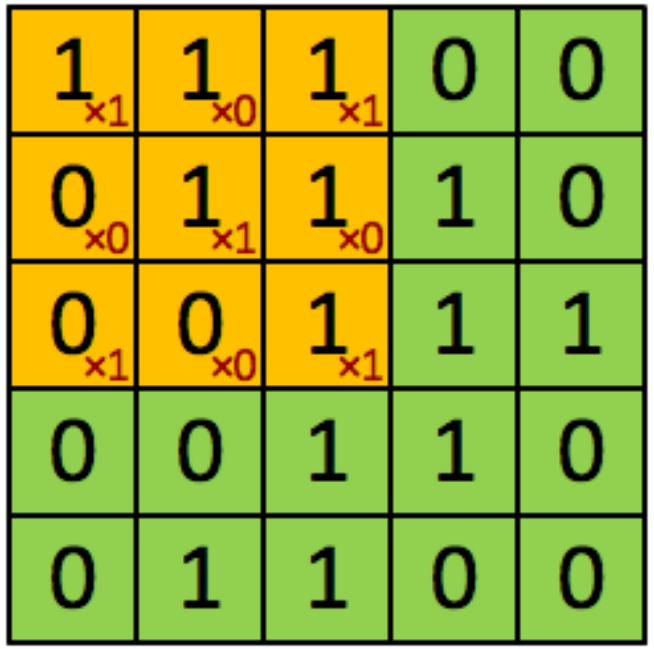
What can we do?

- We need to use the spatial features
  - How
    - We can use filters to detect visual features like lines/segments etc.
- Do not use fully connected layers for the entire input
  - Image size could be more than 256X256 nowadays
  - If we use 1000 neurons for the first hidden layer
    - We need to learn around 200 million parameters for the first layer
    - Any bigger than that --- impossible to fit inside a single computer memory
- Use an architecture which reduces the images into features
  - Learn the features first
  - Then use the features to classify/recognize

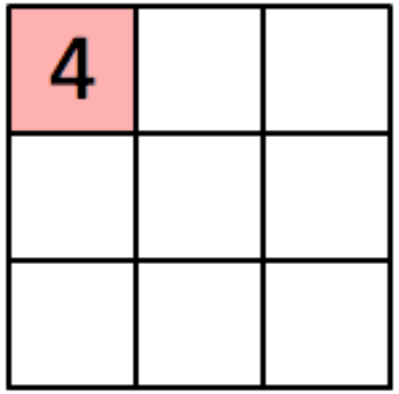
First: Use spatial feature



And a match made in heaven - convolution



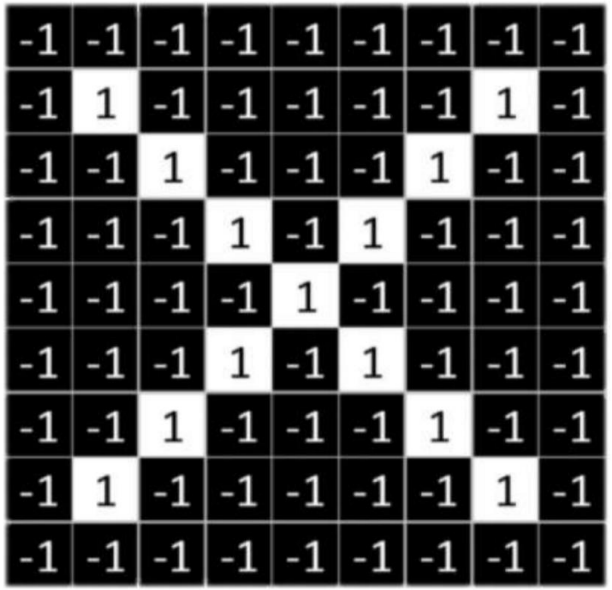
Image

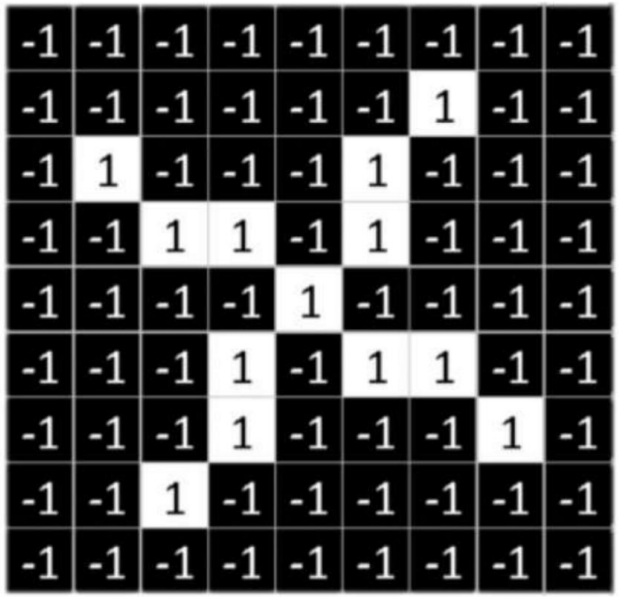


Convolved Feature

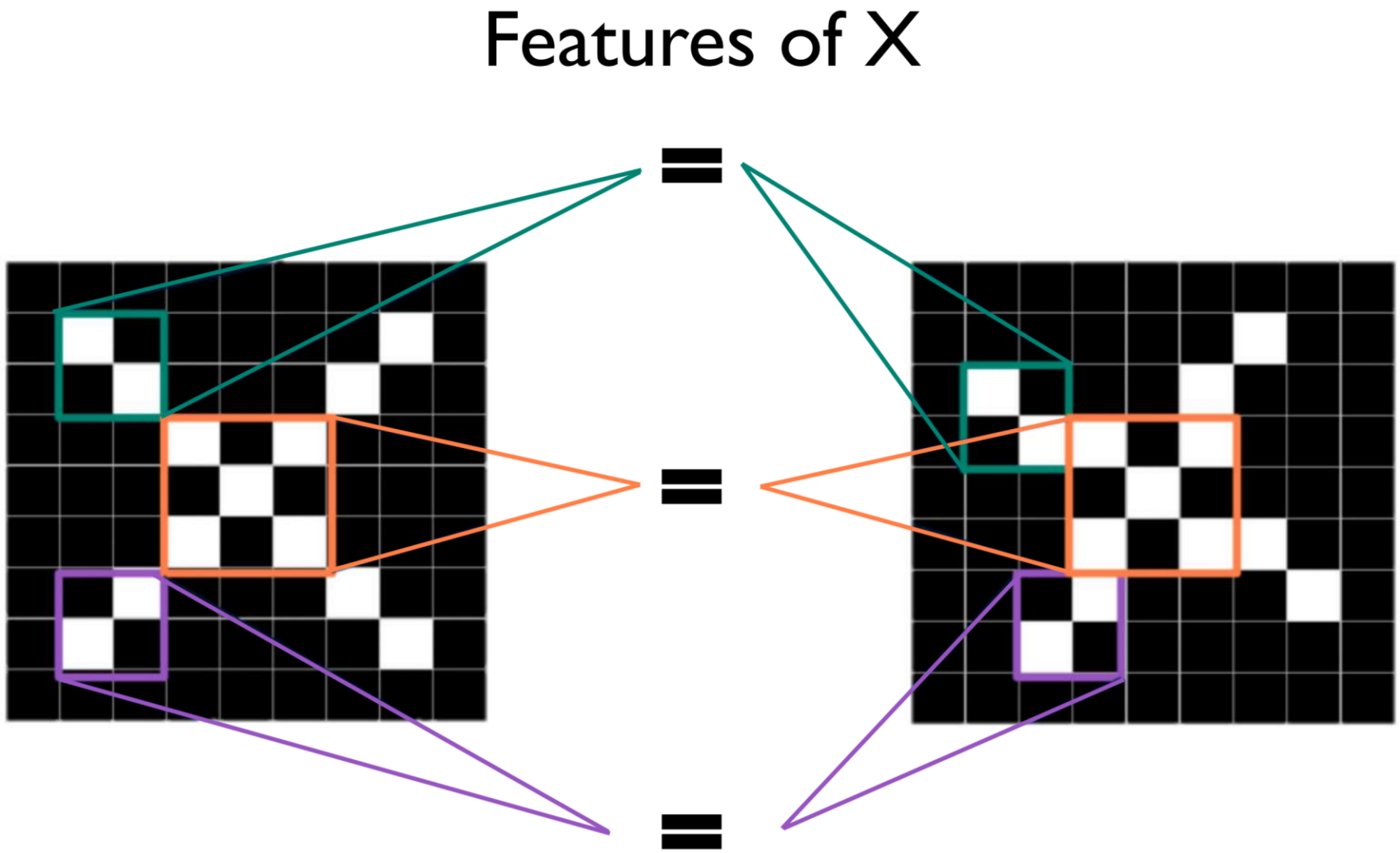
What does it do?

X or X?





What does it do?



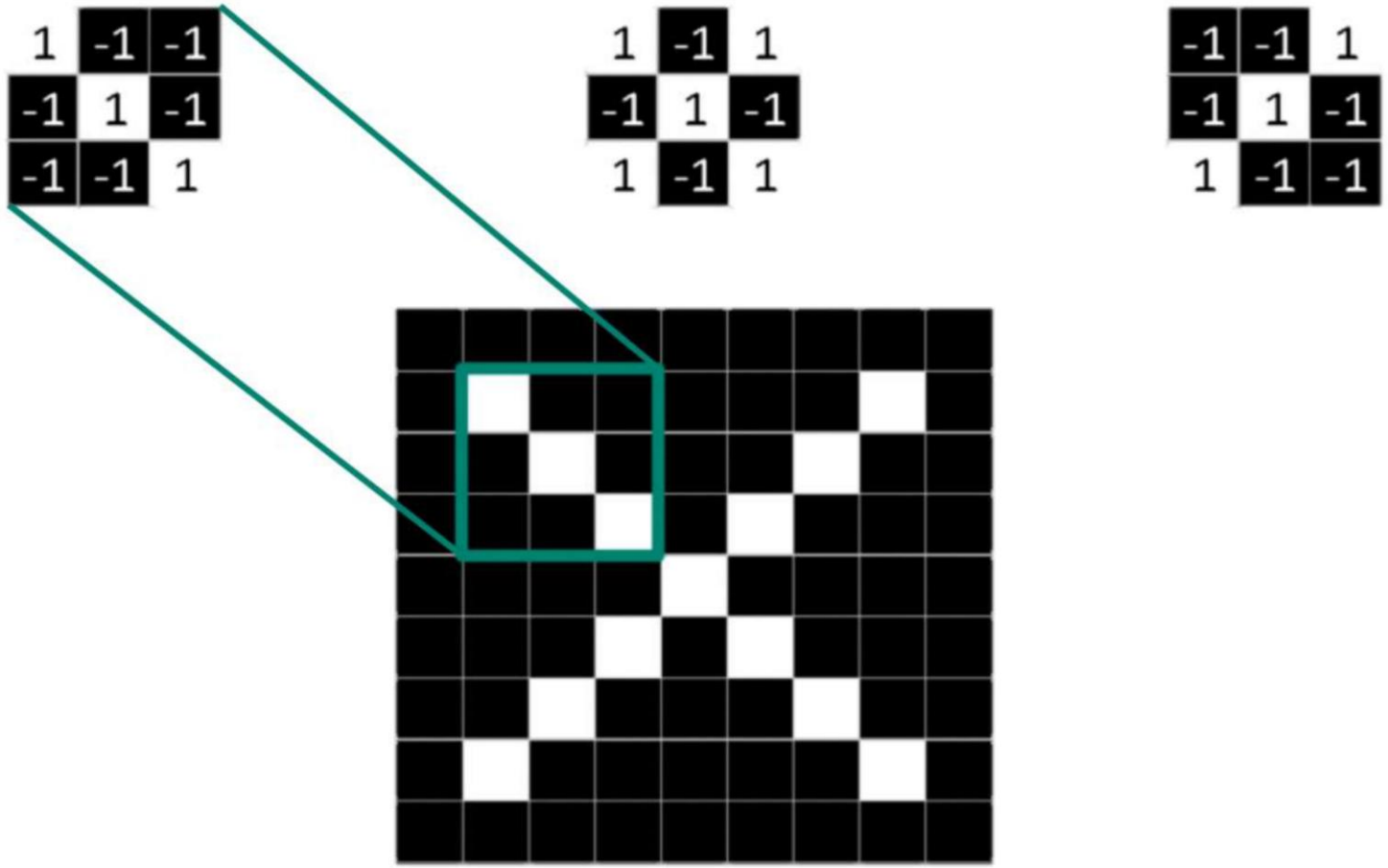
In practice: same filter sliding window algorithm

w_{11}	w_{12}	w_{13}	w_{14}
w_{21}	w_{22}	w_{23}	w_{24}
w_{31}	w_{32}	w_{33}	w_{34}
w_{41}	w_{42}	w_{43}	w_{44}

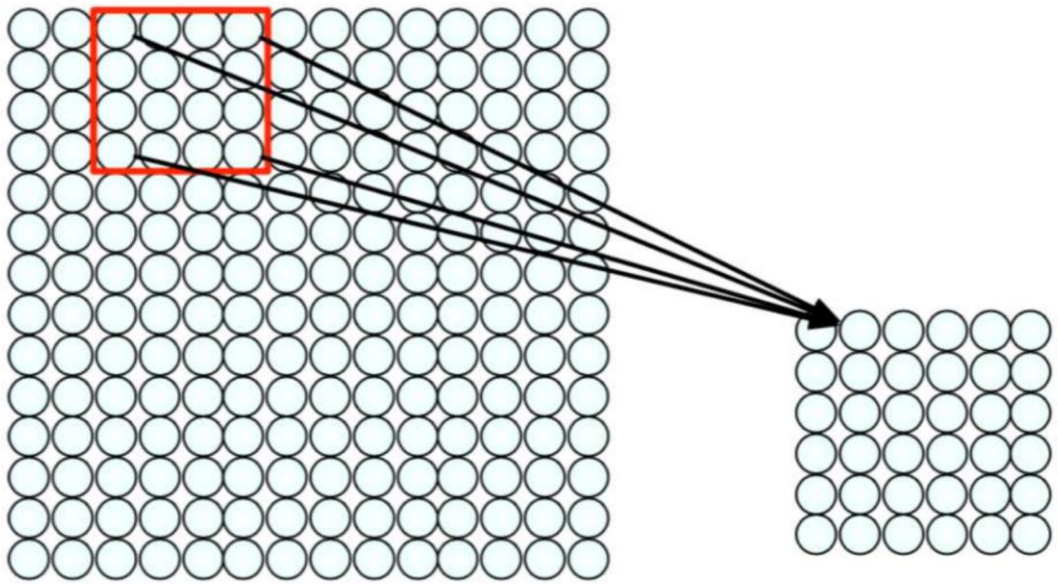
What does it do?

Filters to Detect X Features

filters

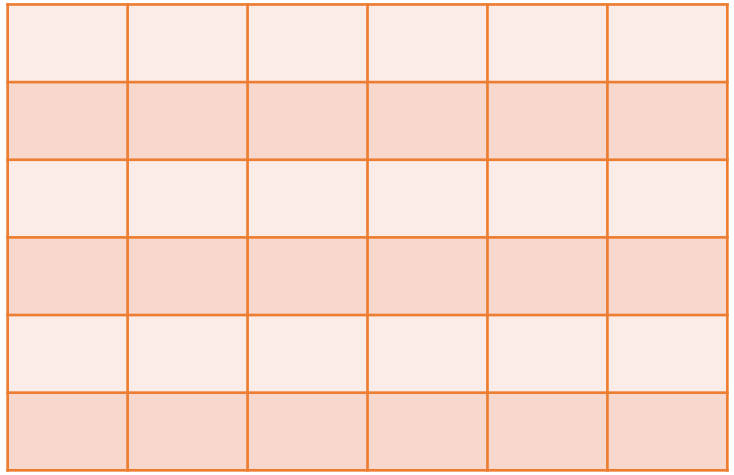


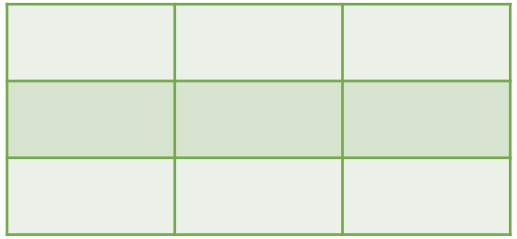
Feature extraction with convolution



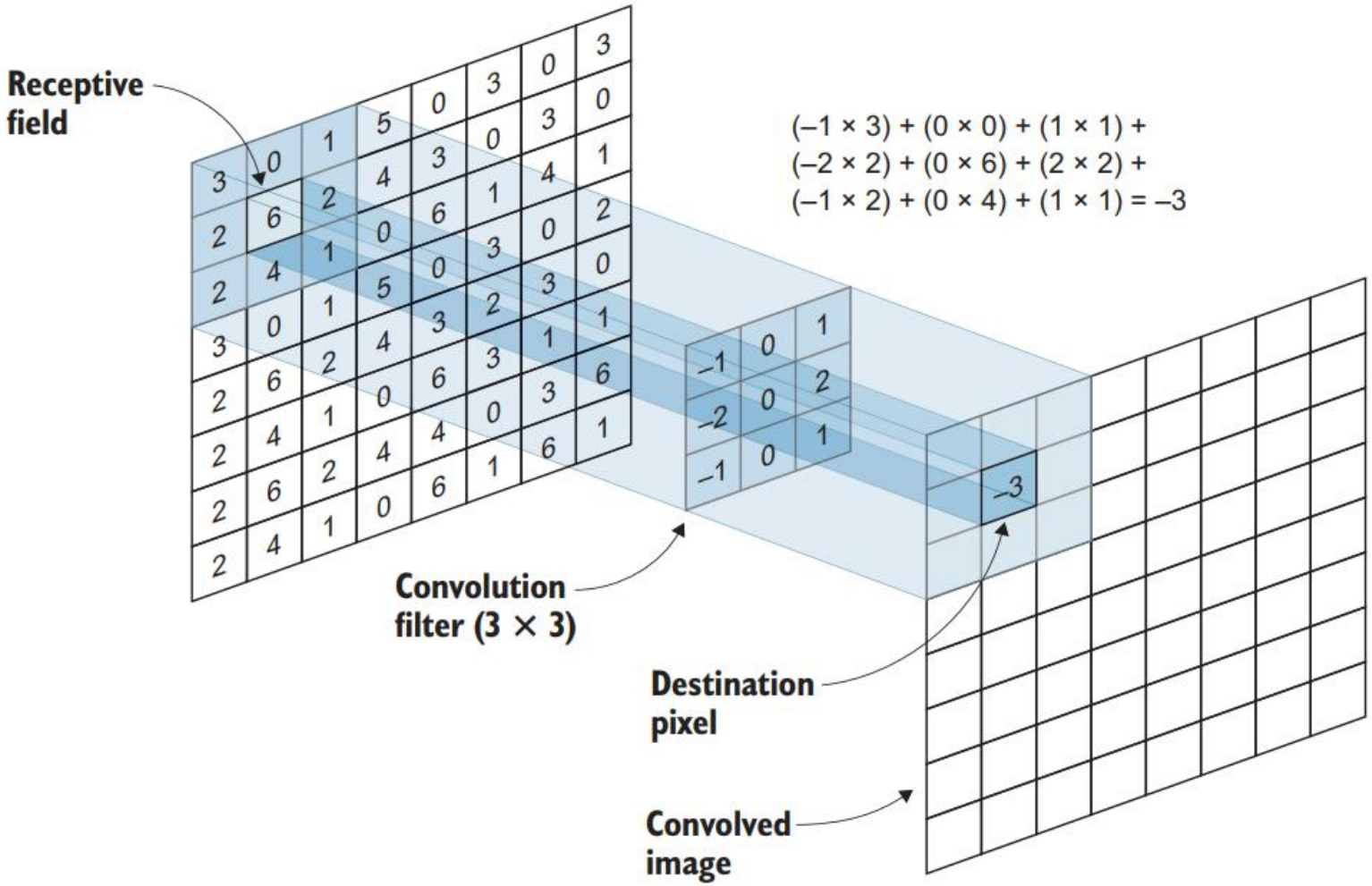
- Filter of size 4x4 : 16 different weights
- Apply this same filter to 4x4 patches in input
- Shift by 1/2 (stride) pixels for next patch

- Apply a set of weights – a filter – to extract local features
- Use multiple filters to extract different features
- Spatially share parameters of each filter

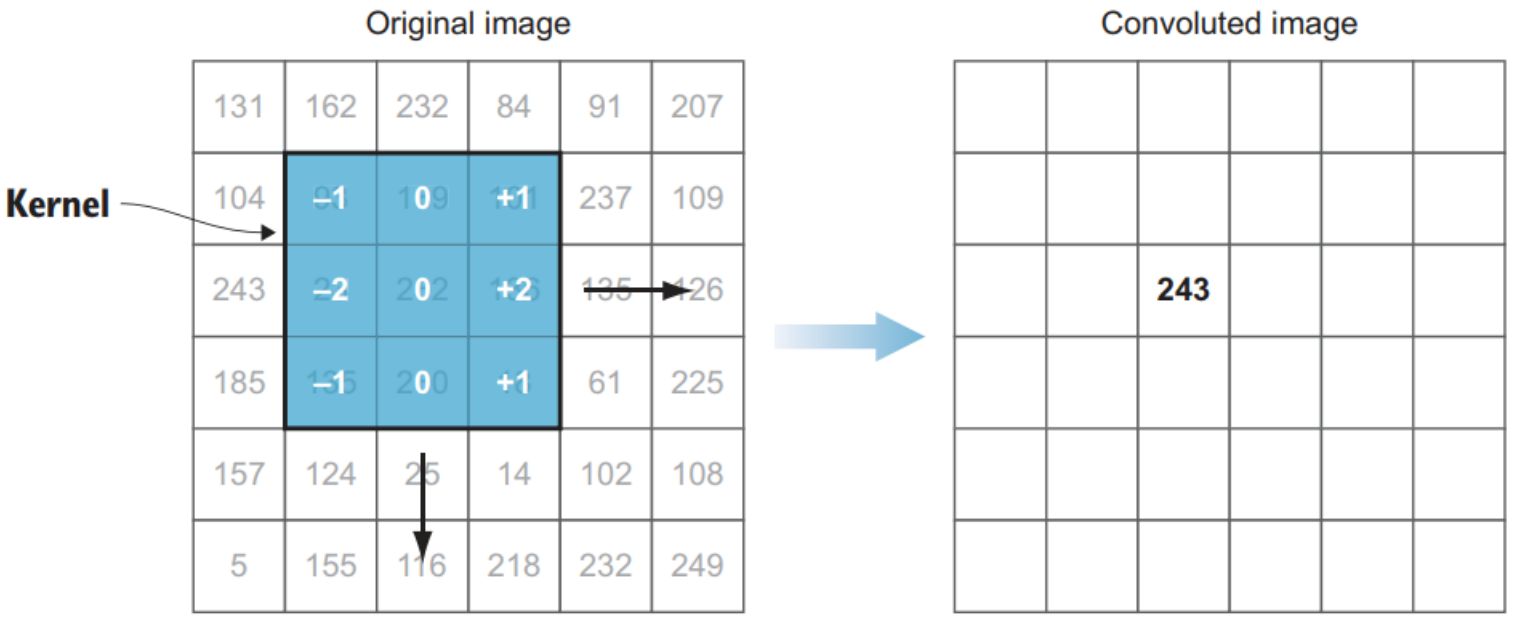




Convolution Operation

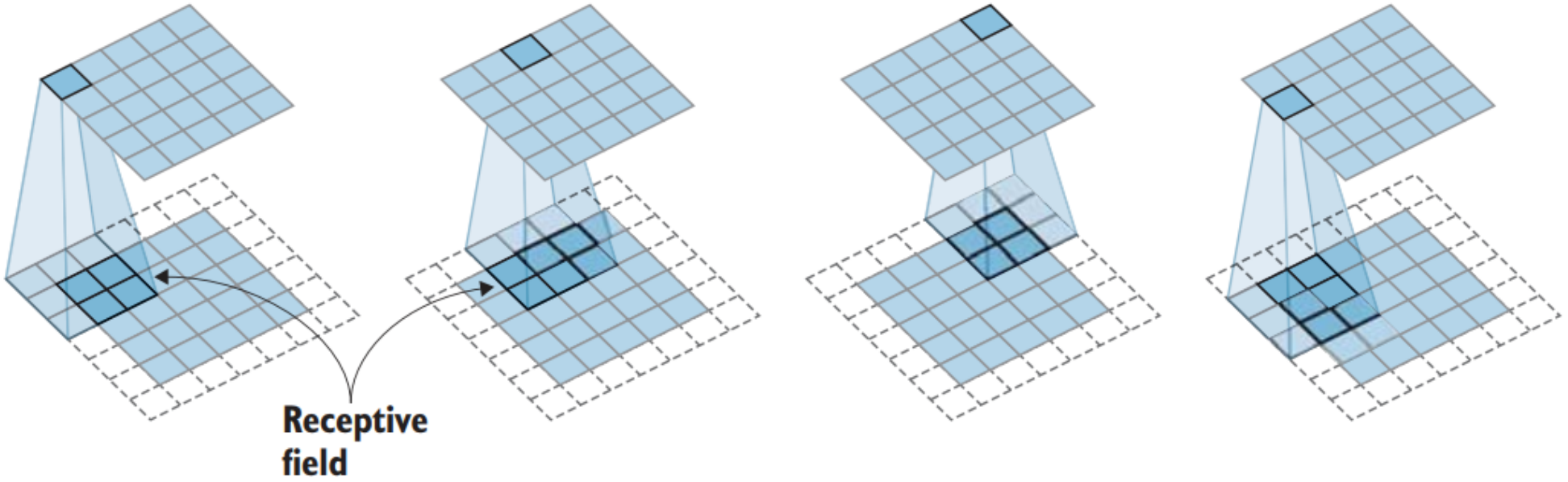


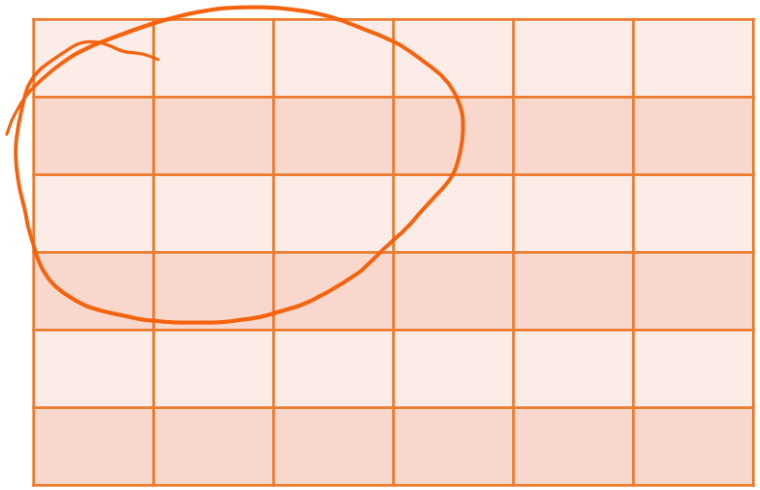
Convolution operation (2)

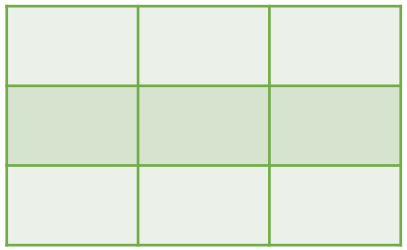


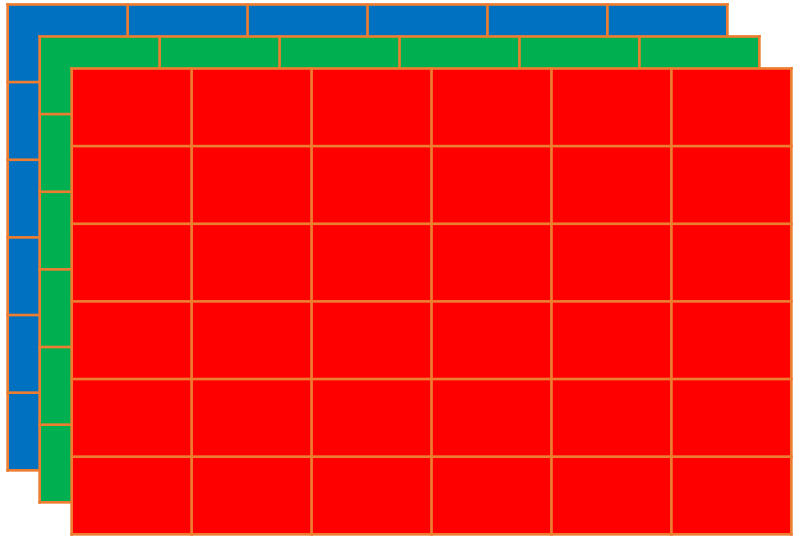
$$(93 \times -1) + (139 \times 0) + (101 \times 1) + (26 \times -2) + (252 \times 0) + (196 \times 2) + (135 \times -1) + (240 \times 0) + (48 \times 1) = 243$$

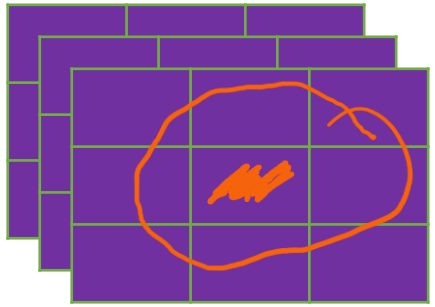
Filter sliding



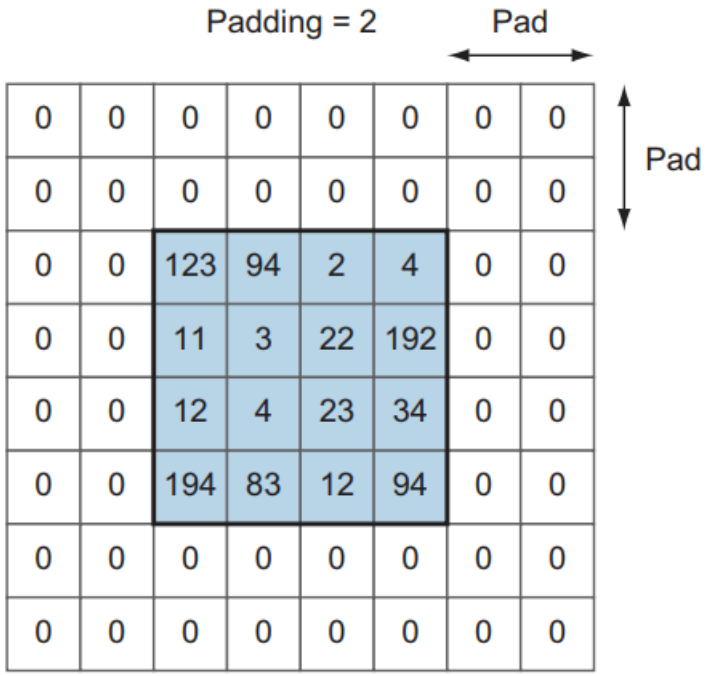








Padding

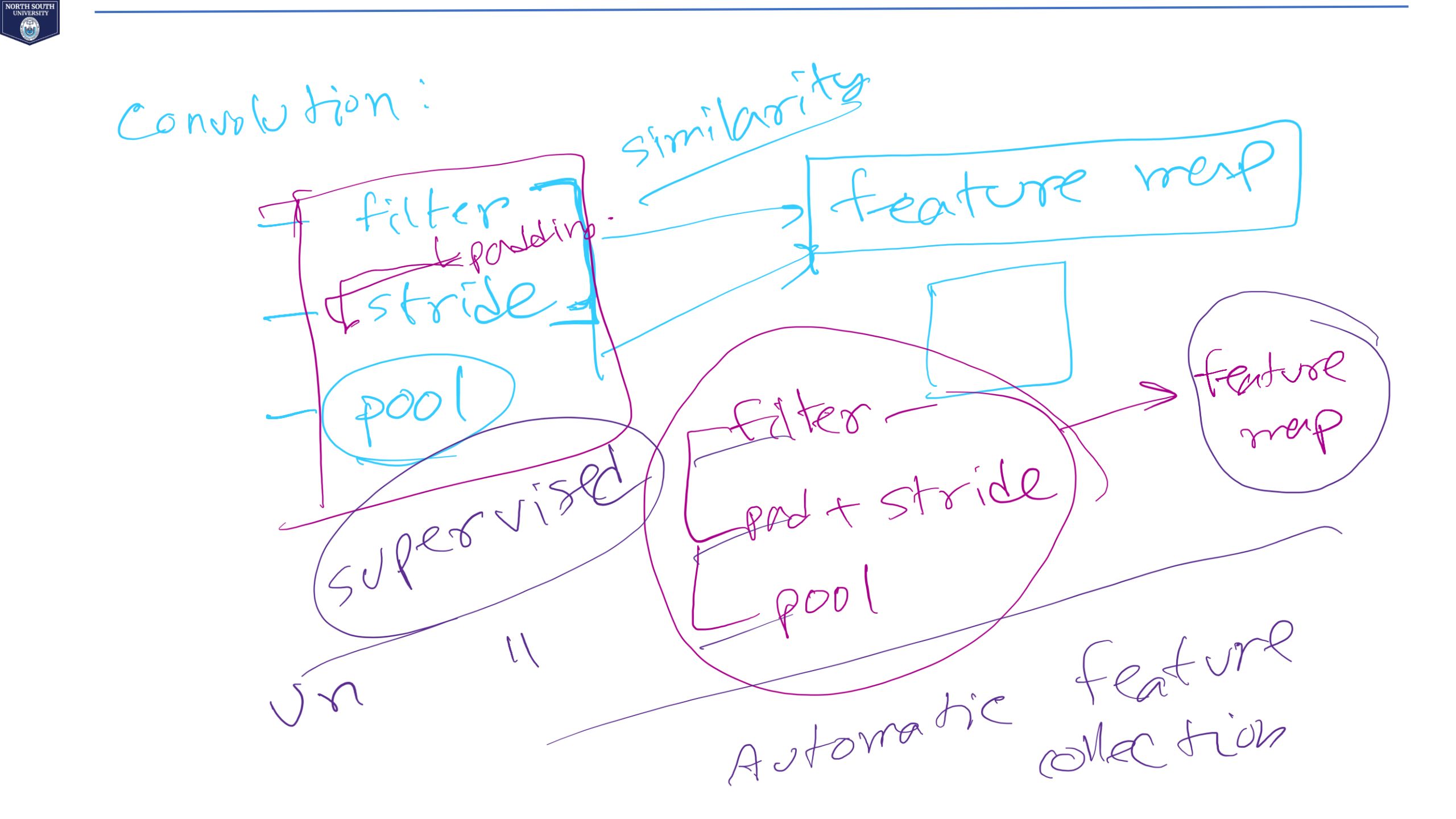


- Padding preserves the spatial size of the input image/volume
  - So the input and output width and height remain the same
- This is important for building deeper networks
  - Otherwise, the height/width would shrink as we go deeper layers

Stride

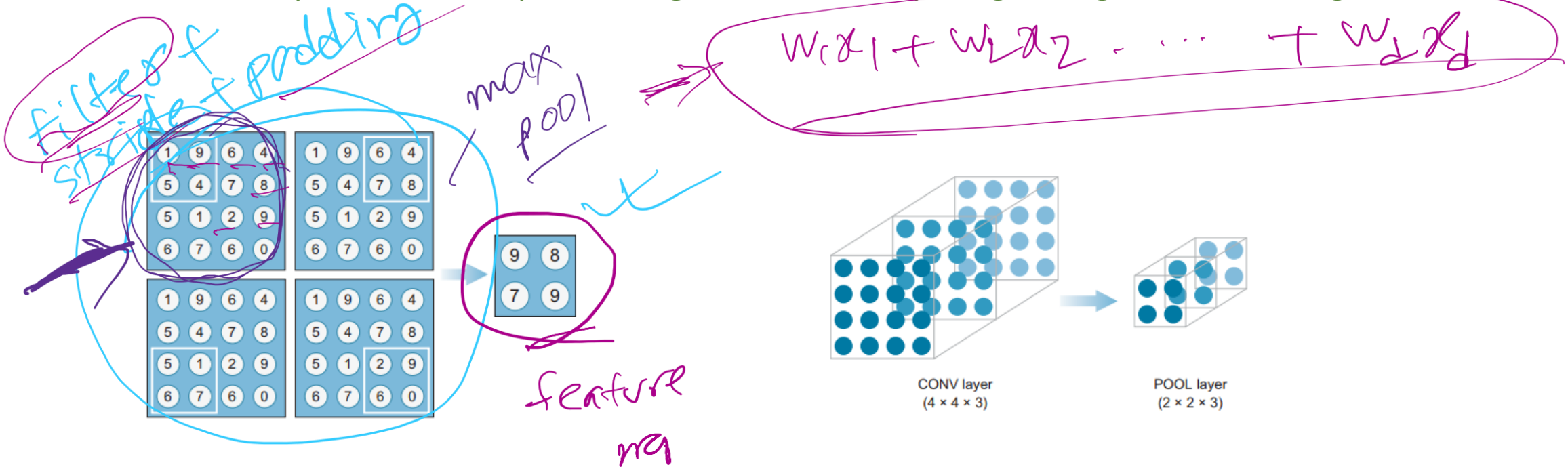
- The number of pixels the filter slides over the image is called stride
  - For example, to slide the convolution filter one pixel at a time, the strides value is 1
    - If we want to jump two pixels at a time, the strides value is 2
    - Strides of 3 or more are uncommon and rare in practice
- Jumping pixels produces smaller output volumes spatially
  - Strides of 1 will make the output image roughly the same height and width of the input image, while strides of 2 will make the output image roughly half of the input image size

$$O = \left\lfloor \frac{n - f + 2p}{s} + 1 \right\rfloor$$


Convolution:
similarity
filter
padding
stride
pool
feature map
supervised
filter
pad + stride
pool
feature map
Un
11
Automatic feature collection

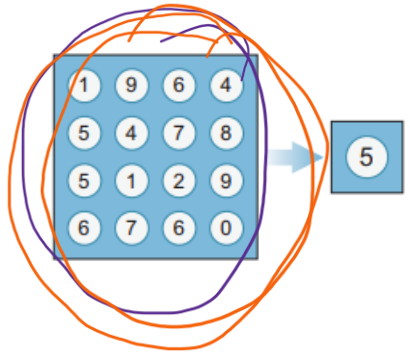
Pooling

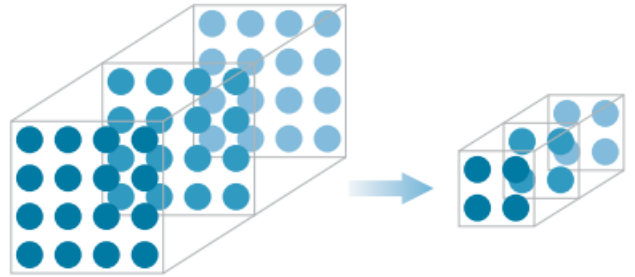
- The goal of the pooling layer is to down sample the feature maps produced by the convolutional layer into a smaller number of parameters, thus reducing computational complexity
- Pooling filters do not have weights or any values
  - All they do is slide over the feature map created by the previous convolutional layer and select the pixel value to pass along to the next layer, ignoring the ~~remaining values~~

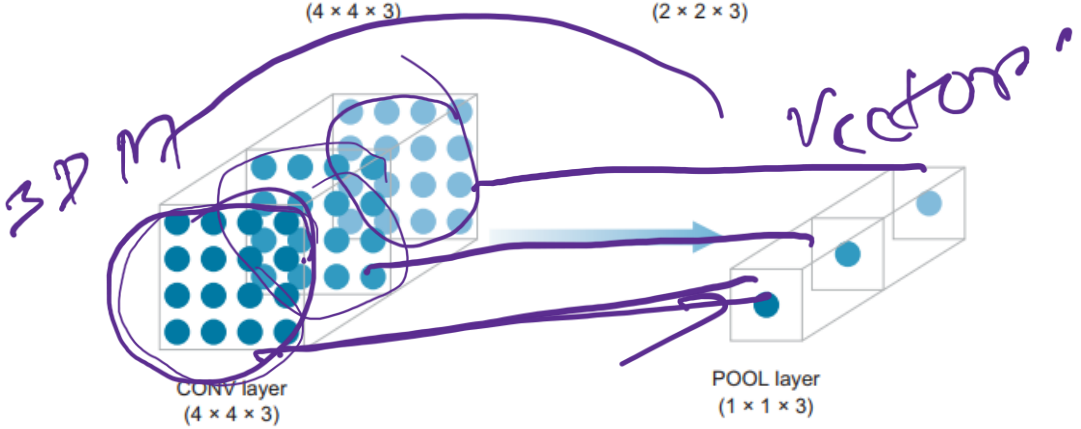


Pooling

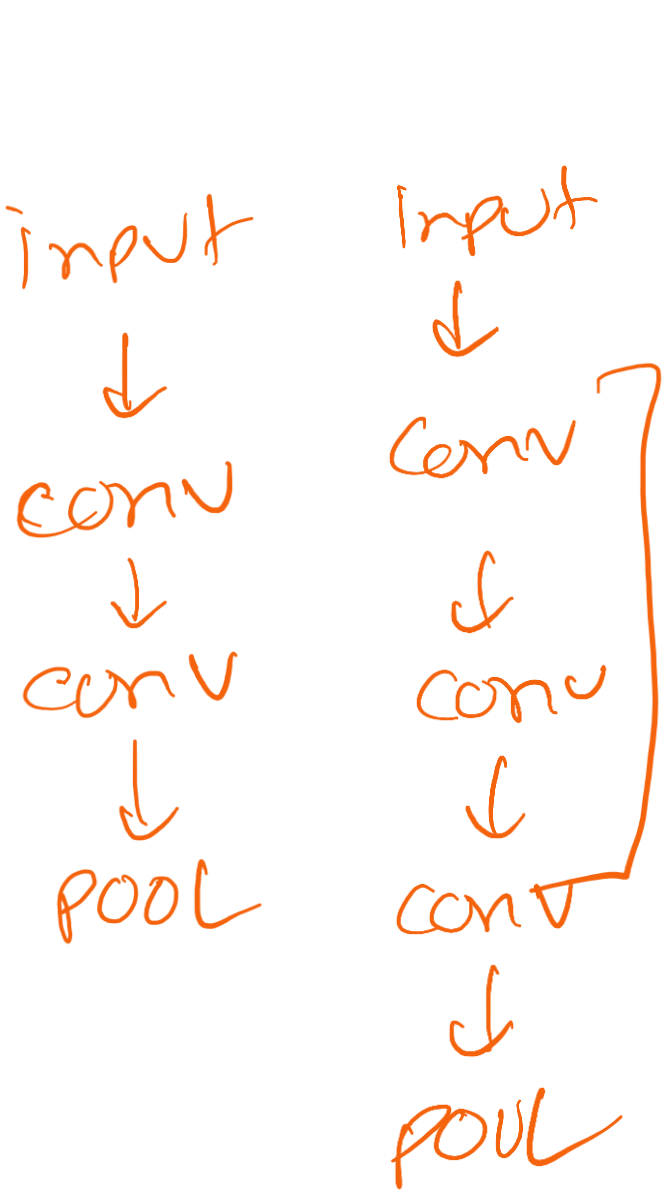
- Max pooling: Selects the max of the numbers on the window
- Average pooling: Selects the average of the numbers on the window
- Global average pooling: Selects the average of all the pixels in the feature map

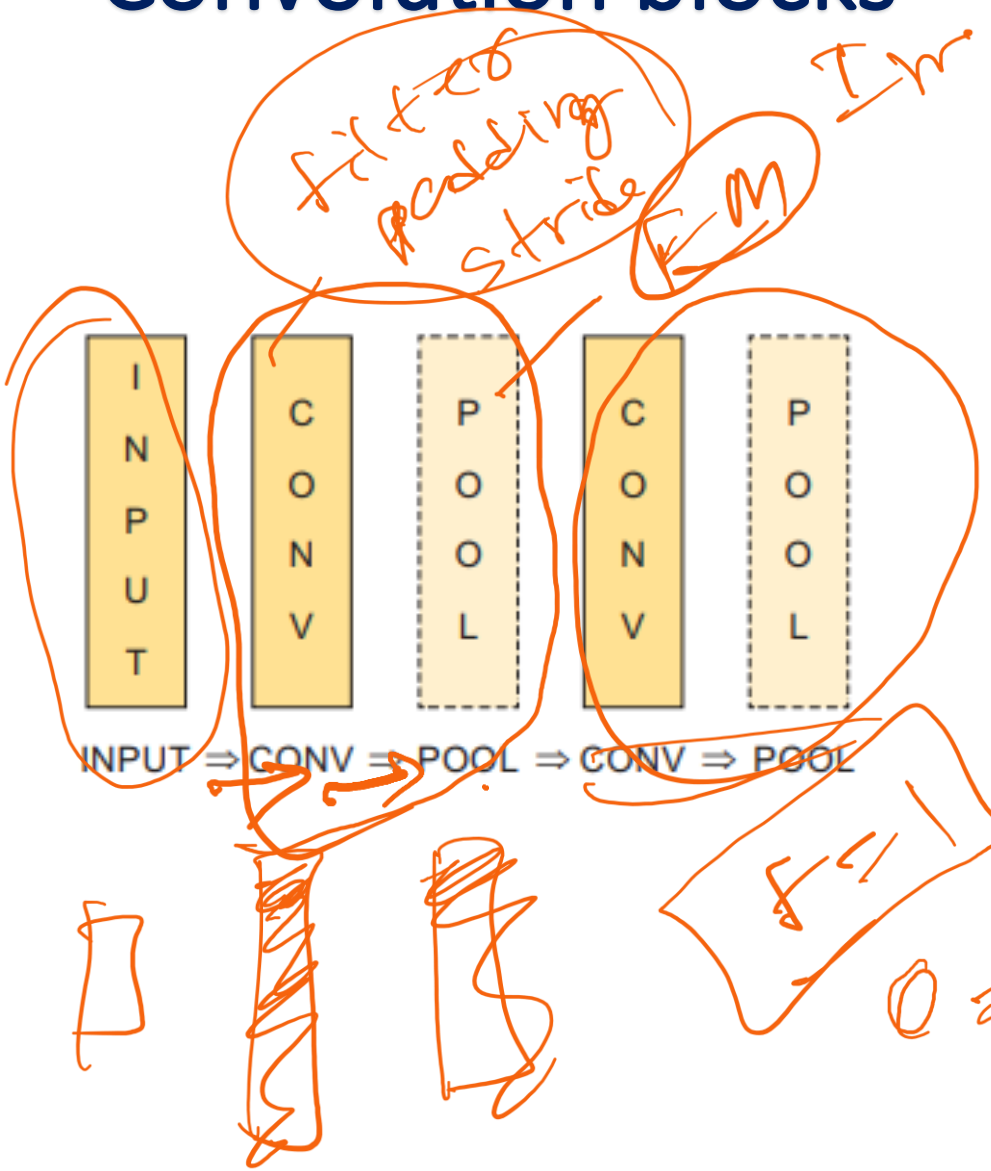


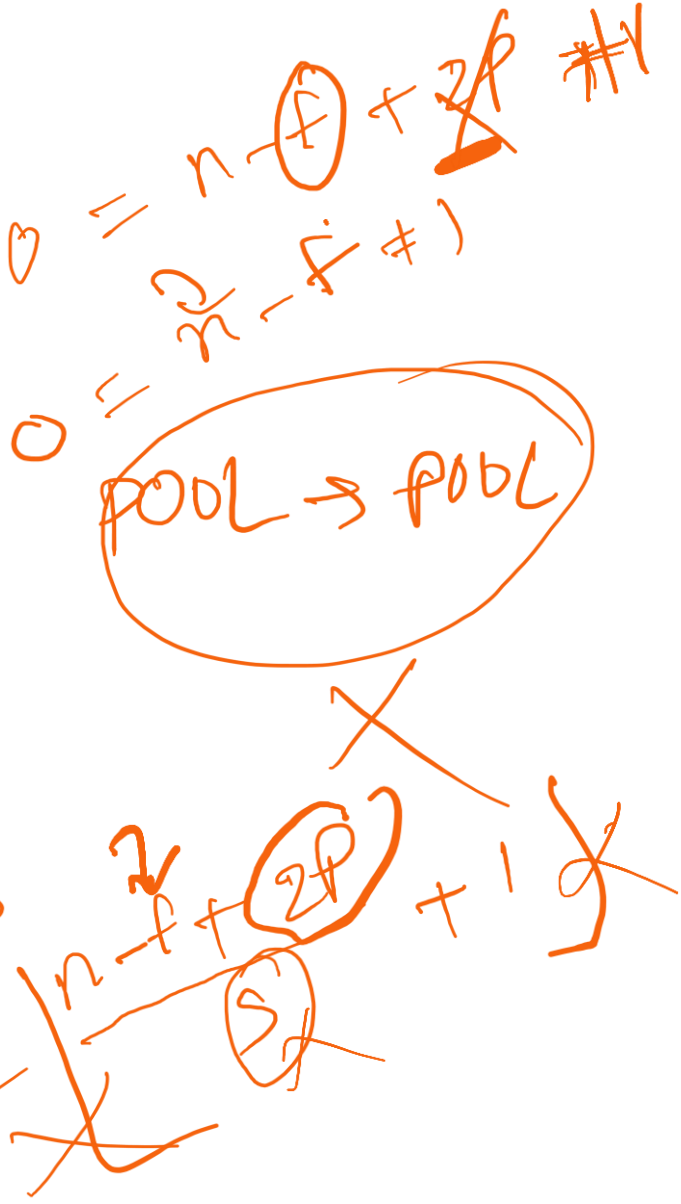




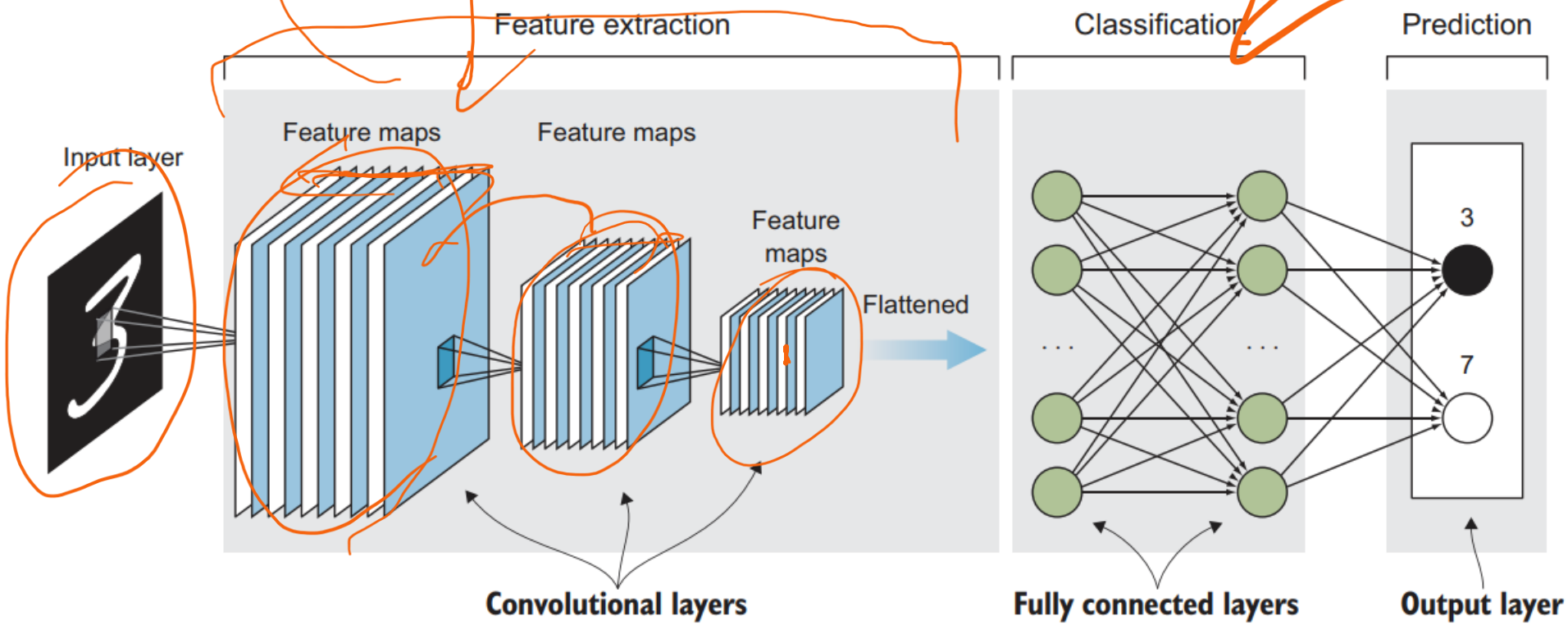
Convolution blocks



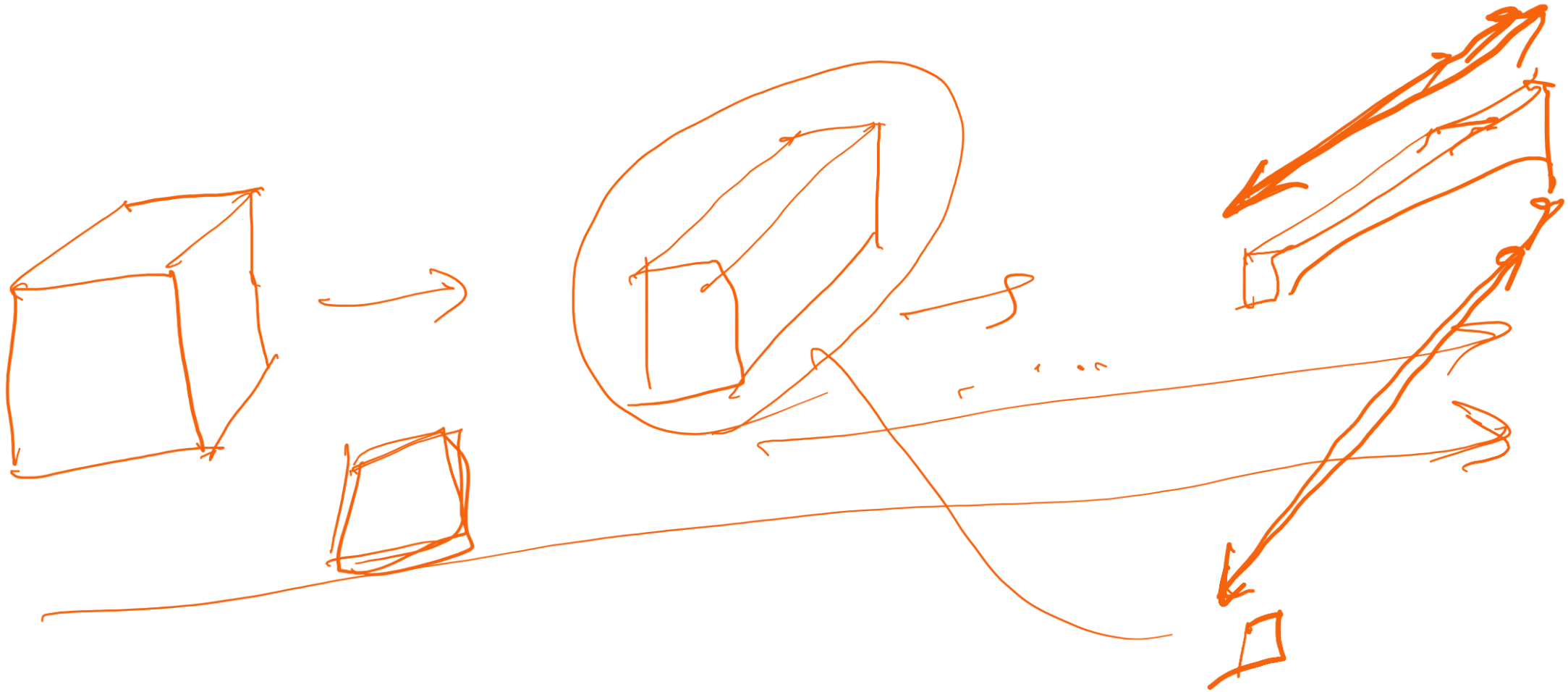




The complete network

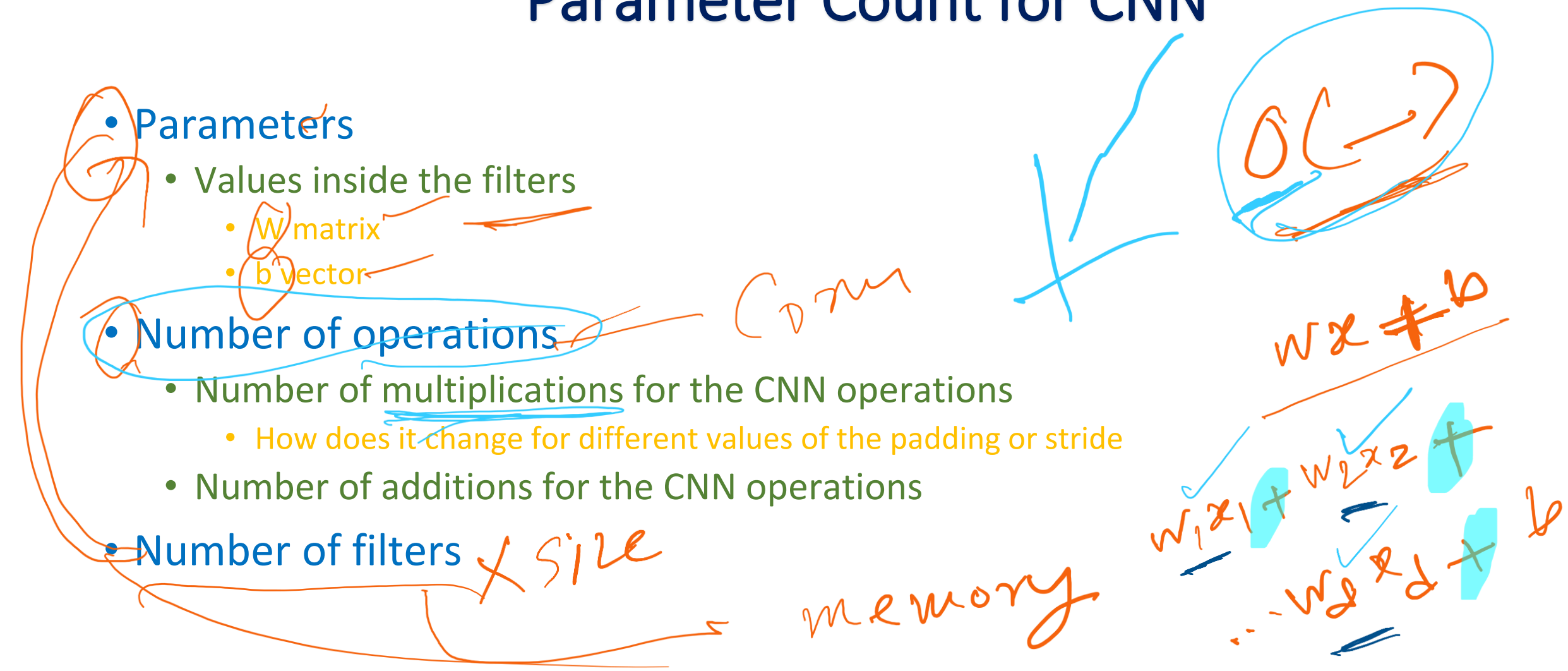


1. Input data	2. Preprocessing	3. Feature extraction	4. ML model
• Images • Videos (image frames)	Getting the data ready: • Standardize images • Color transformation • More...	• Find distinguishing information about the image	• Learn from the extracted features to predict and classify objects

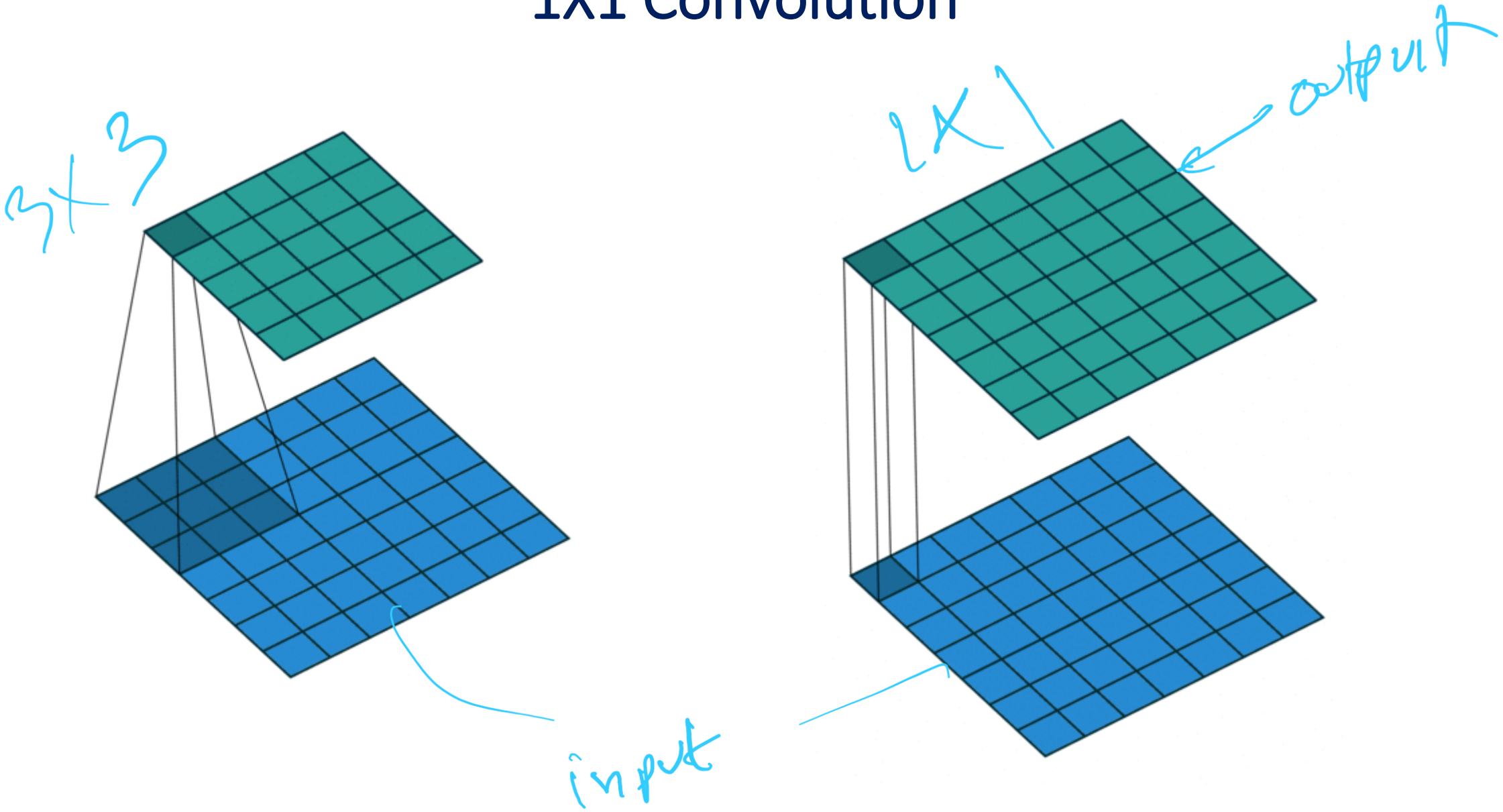


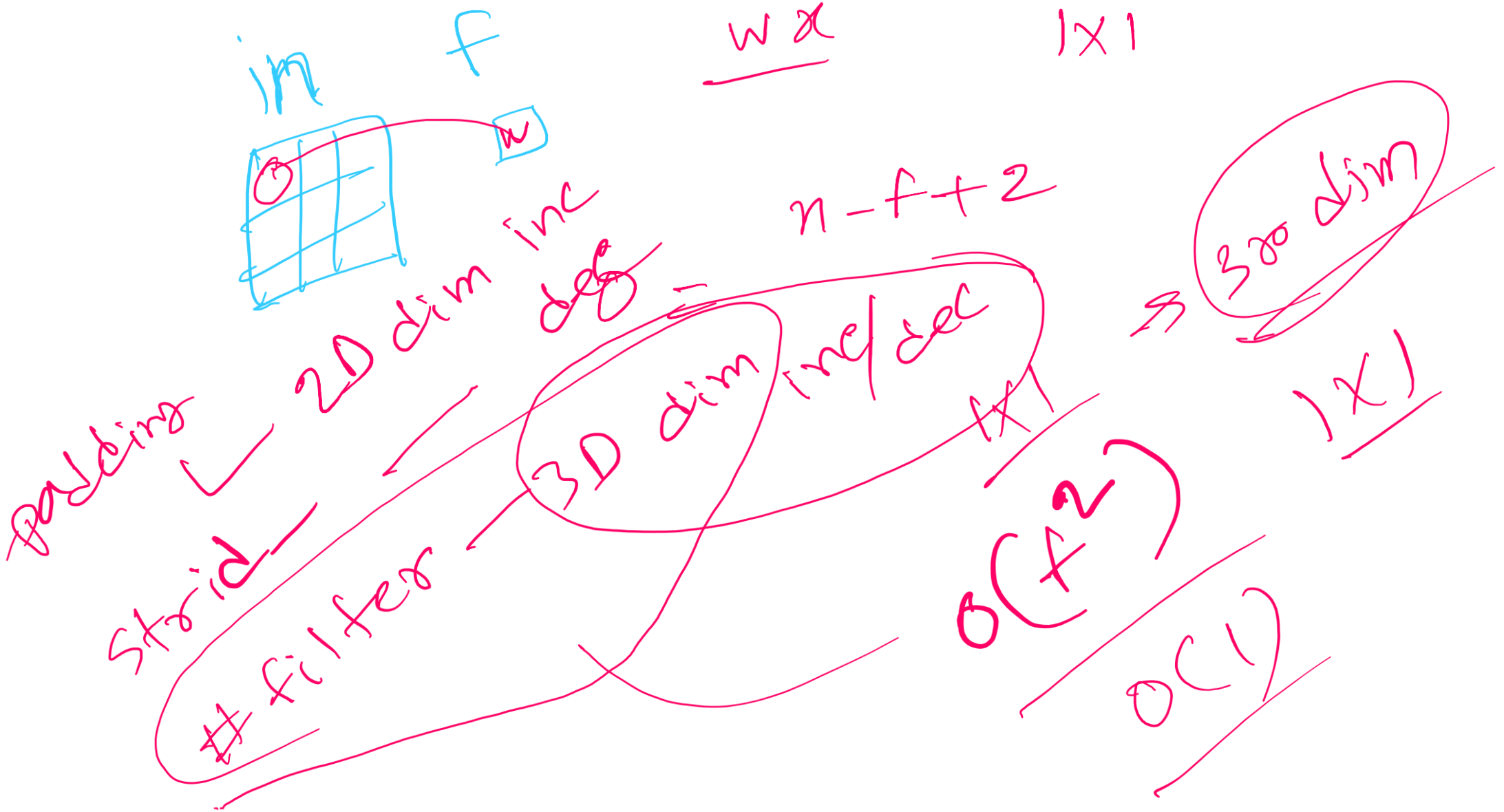
Parameter Count for CNN

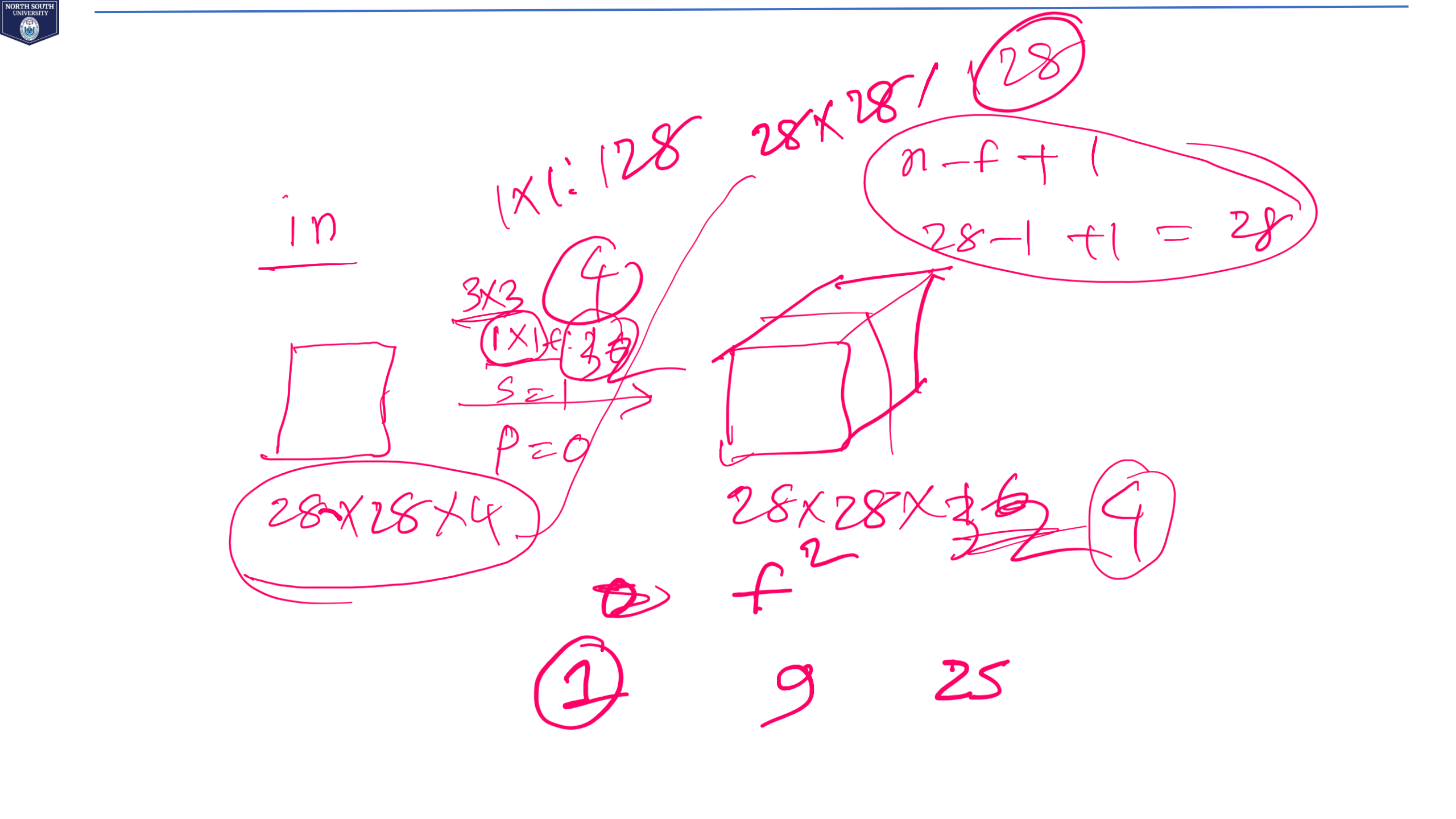
- Parameters
  - Values inside the filters
    - W matrix
    - b vector
- Number of operations
  - Number of multiplications for the CNN operations
    - How does it change for different values of the padding or stride
  - Number of additions for the CNN operations
- Number of filters



1X1 Convolution

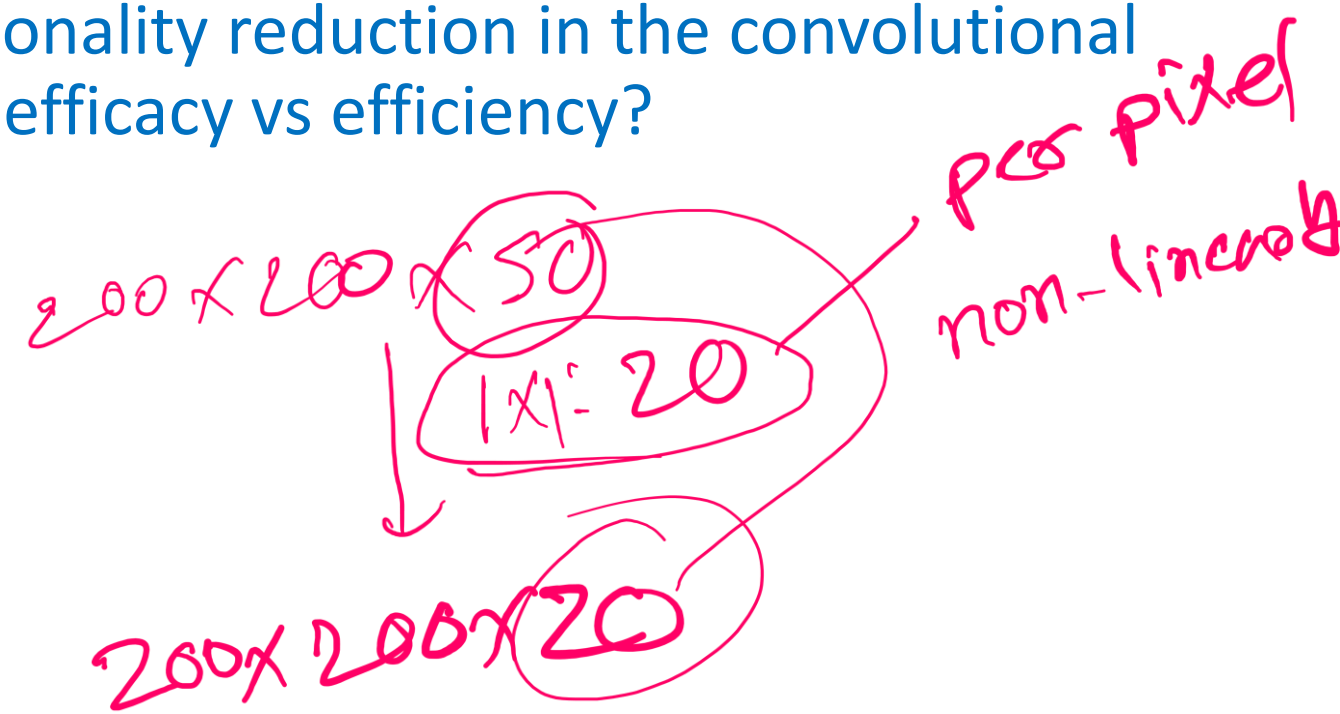



in
f
w x
1 x 1
w
padding
stride
2D dim inc
dec
n - f + 2
filters
3D dim
inc/dec
f x f
= 3D dim
1 x 1
O(f^2)
O(1)


in
28x28x4
1x1: 128
3x3
4
S=1
P=0
28x28x128
28
n-f+1
28-1+1 = 28
28x28x4
f²
1
9
25

Why 1X1 Convolution

- Most simplistic explanation would be that 1x1 convolution leads to dimension reduction
- For example, an image of 200 x 200 with 50 features on convolution with 20 filters of 1x1 would result in size of 200 x 200 x 20
- Is this the best way to do dimensionality reduction in the convolutional neural network? What about the efficacy vs efficiency?

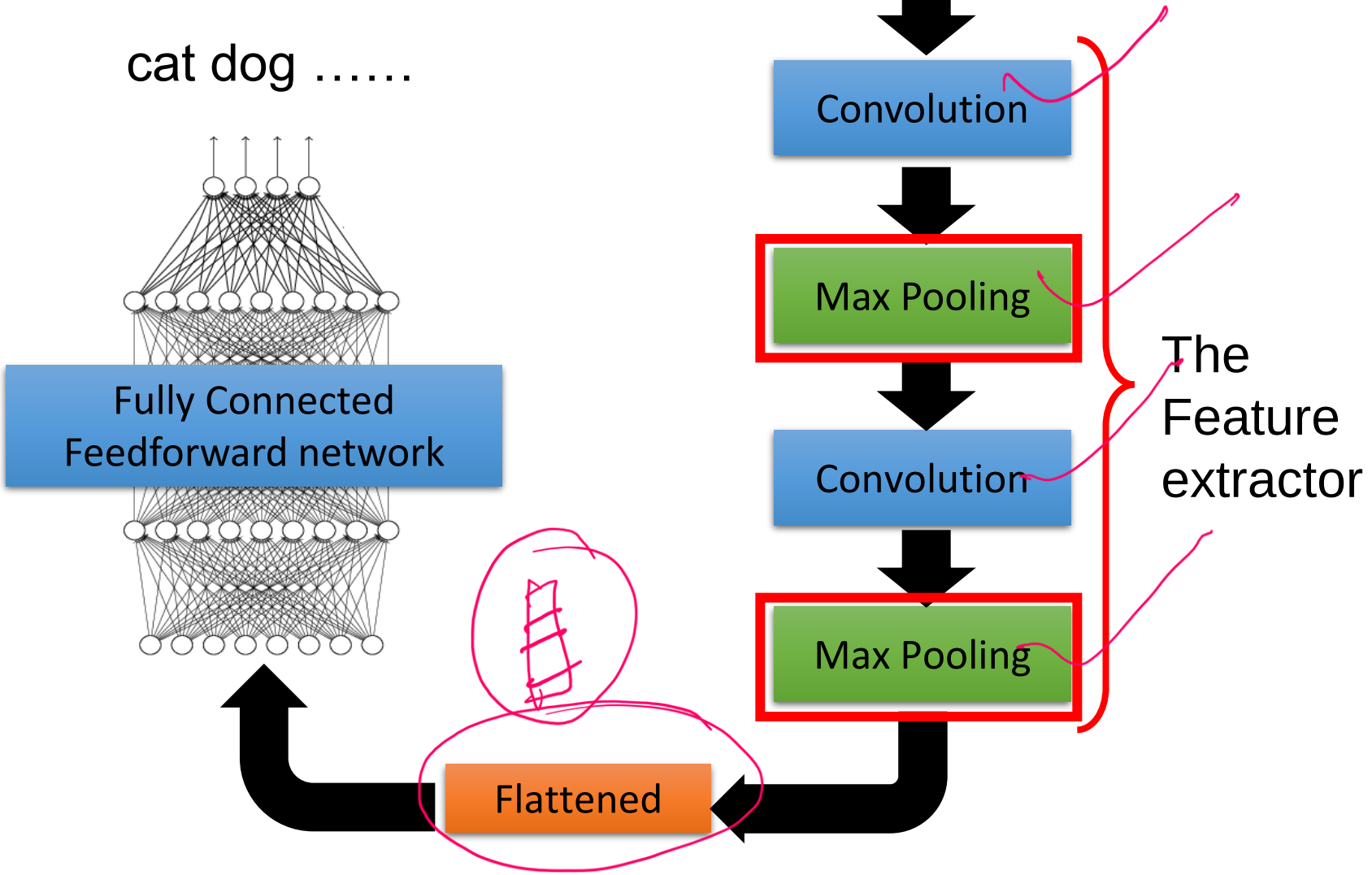


Why 1X1 Convolution

- Although 1x1 convolution is a ‘feature pooling’ technique, there is more to it than just sum pooling of features across various channels/feature-maps of a given layer
- 1x1 convolution acts like coordinate-dependent transformation in the filter space
- This transformation is strictly linear, but in most of application of 1x1 convolution is succeeded by a non-linear activation layer like ReLU
- This transformation is learned through the (stochastic) gradient descent
- An important distinction is that it suffers with less over-fitting due to smaller kernel size (1x1)

1X1 Convolution

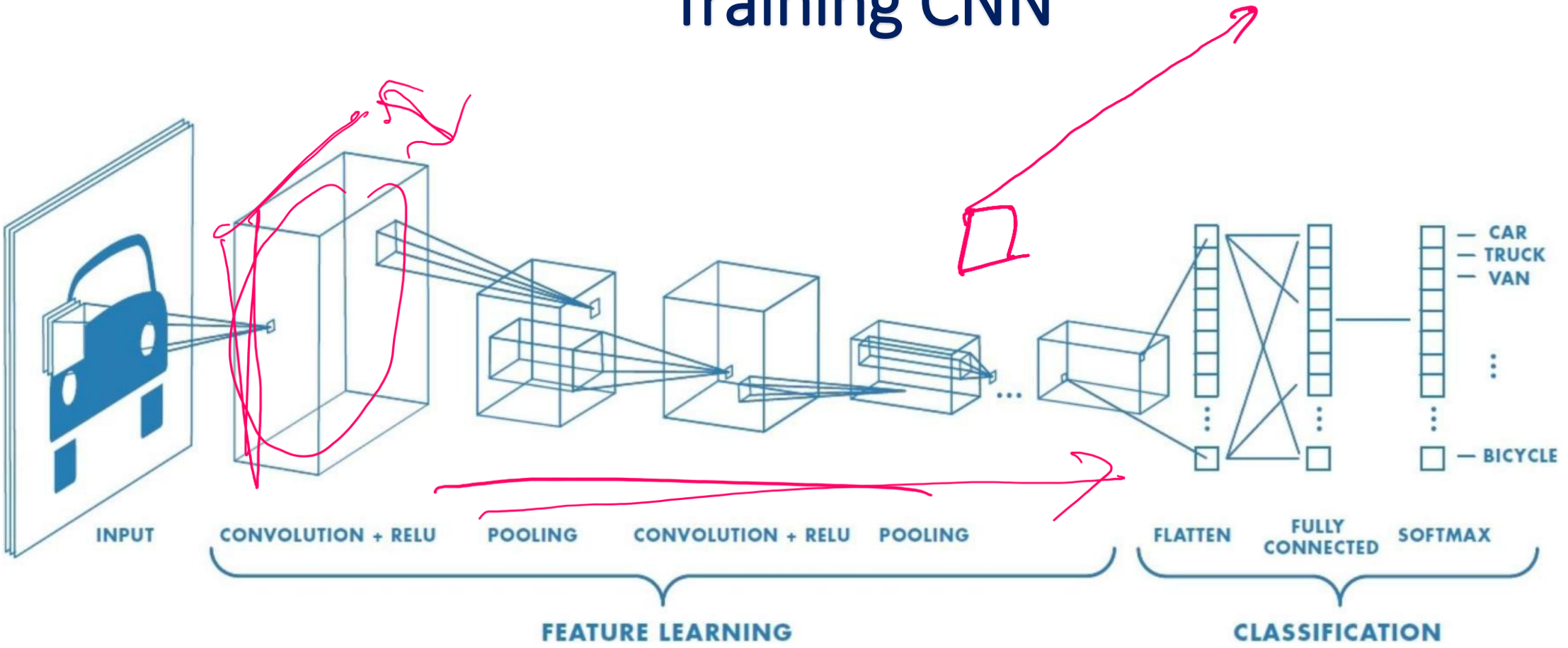
The feature extractor



A CNN compresses a fully connected network

- Reduces number of connections
- Shared weights on the edges
- Max pooling further reduces the complexity

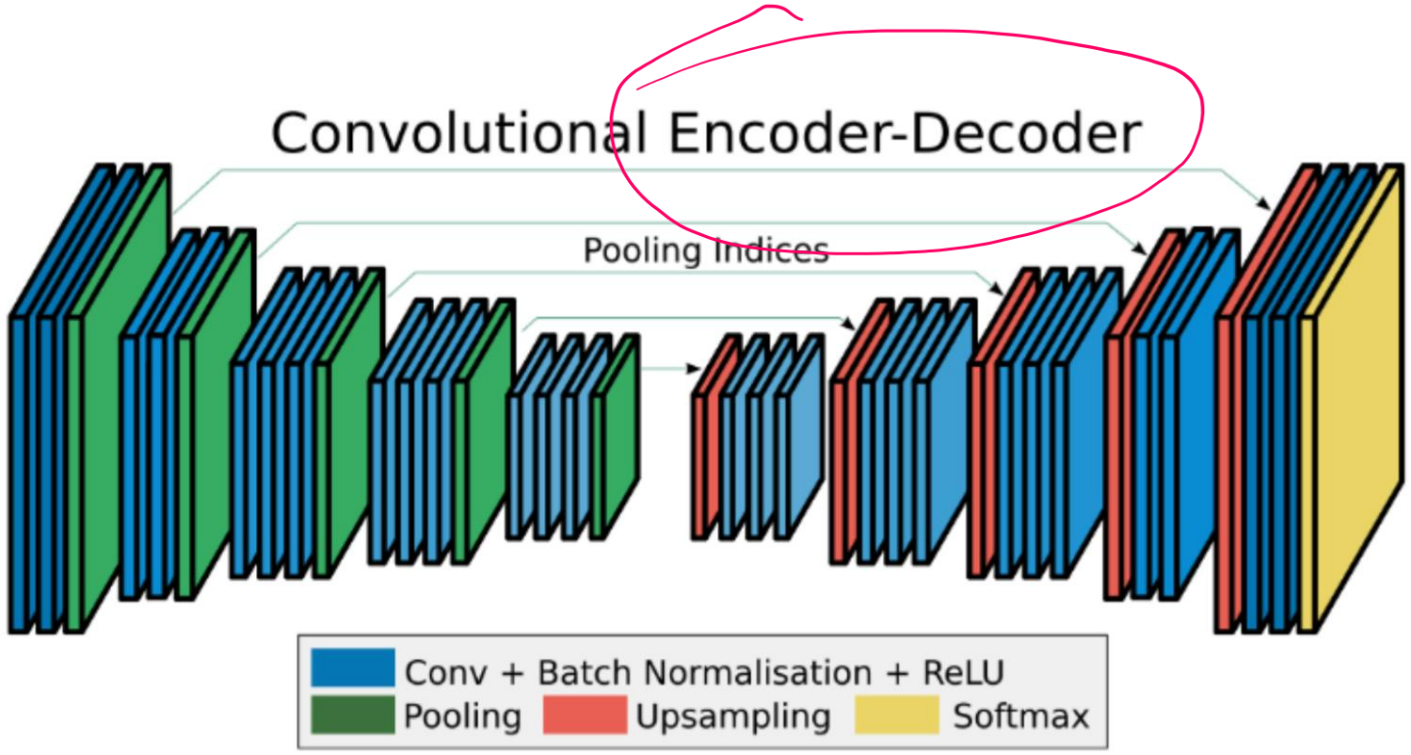
Training CNN



- Learn weights for convolutional filters and fully connected layers using backpropagation and the log loss (cross-entropy loss) function

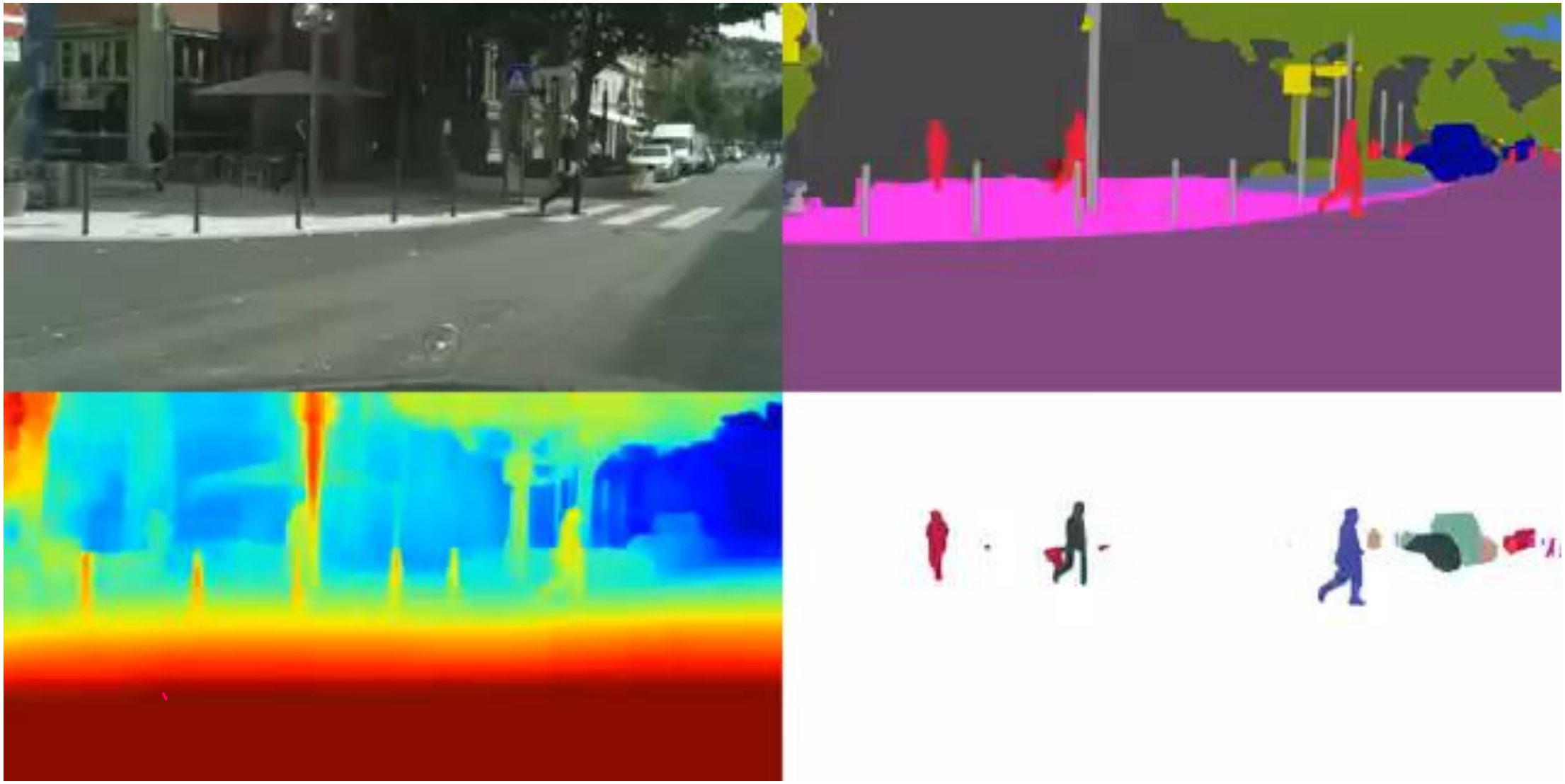
CNN Application: Self driving cars/drones

- Convolution then de-convolution
- Get a pixel level map



(Badrinarayanan, Kendall, & Cipolla, 2015)

Self driving cars/Drones



Generate image caption/description

- Use CNN to detect the image features
- Then instead of using the FCN
- Use RCN to generate the description