

Machine learning

Introduction to Computer Vision

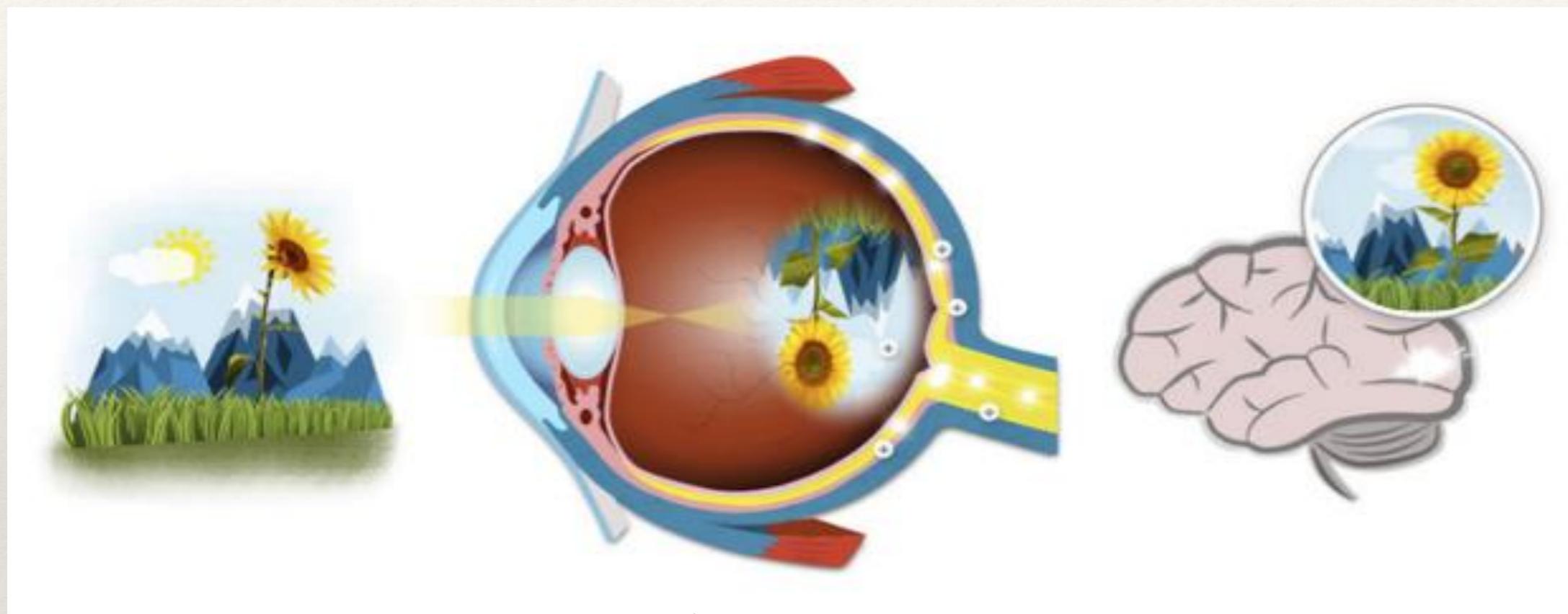
Lecture XI

פיתוח:
להב יפתח

Based on Alexander Amini @ MIT slides

© Alexander Amini and Ava Soleimany
MIT 6.S191: Introduction to Deep Learning
IntroToDeepLearning.com

Human Vision

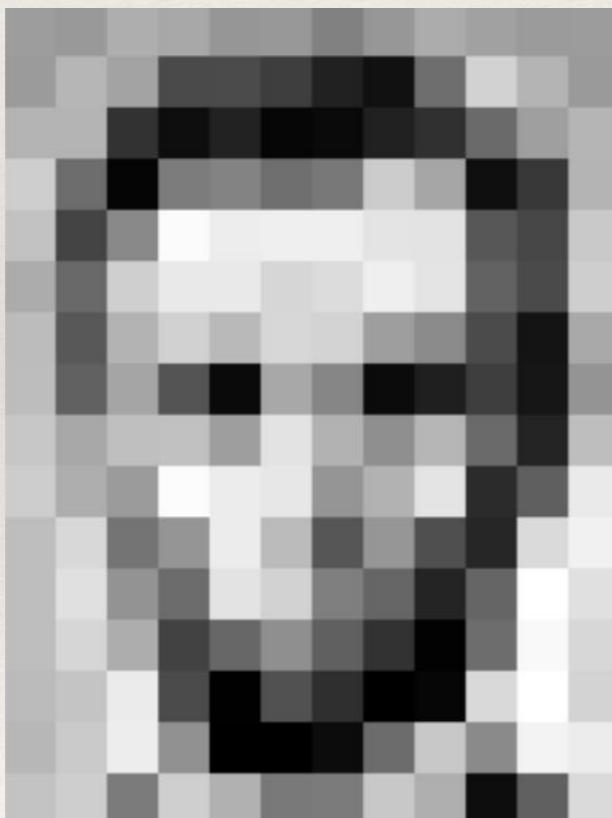


The eye Translates visual information into electric information

The brain's visual cortex interprets the electrical form of the image

Computer Vision

What computers “see”?



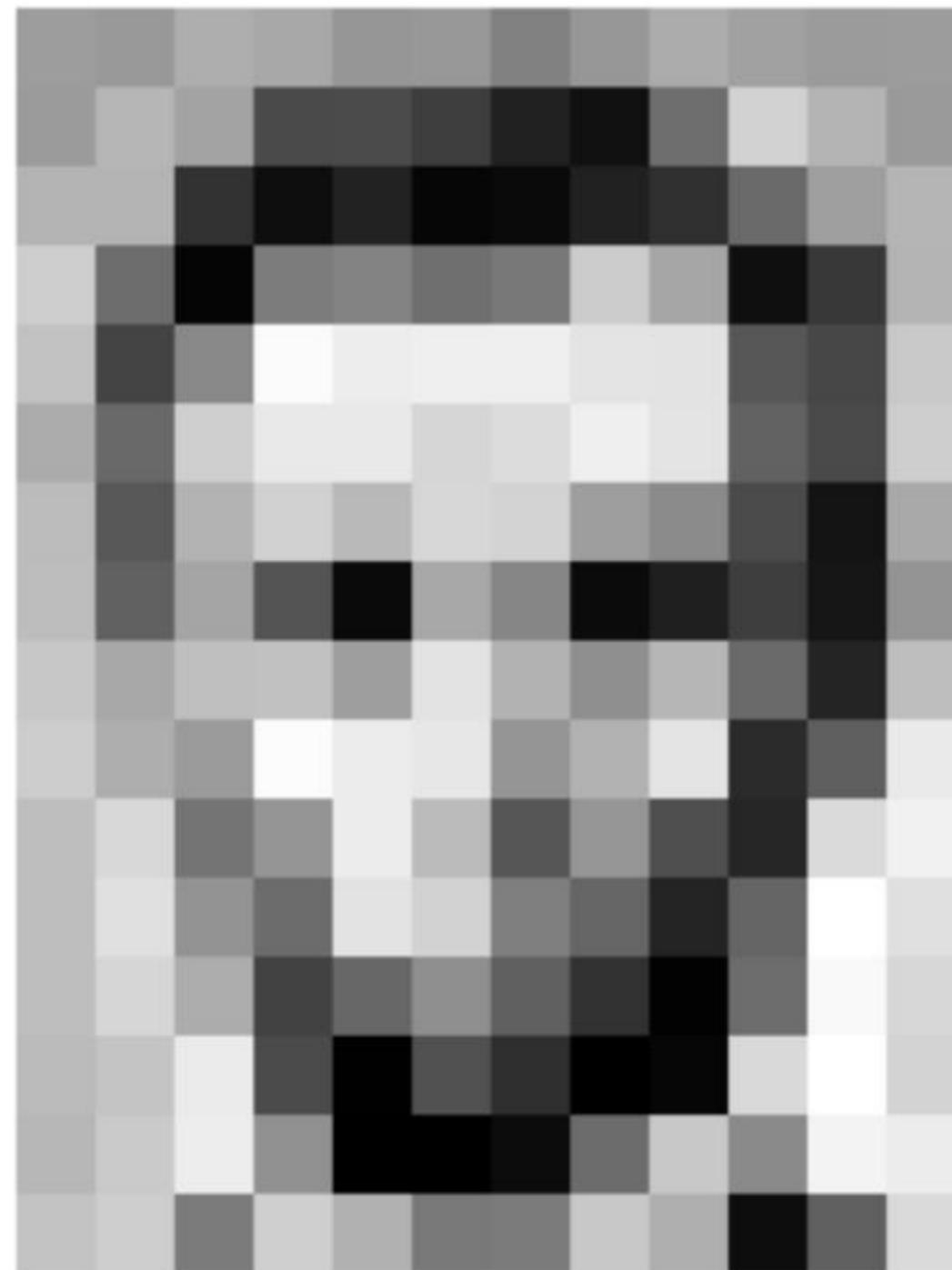
Images are numbers

Images are 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	191	111	120	204	166	15	56	180
194	68	197	251	297	299	299	228	227	87	71	201
172	106	207	253	253	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	35	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	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

Images are 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	169	5	124	191	111	120	204	166	15	56	180
194	68	197	251	297	299	299	228	227	87	71	201
172	106	207	253	253	214	220	239	228	98	74	206
188	68	179	209	185	215	211	158	139	75	20	169
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	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

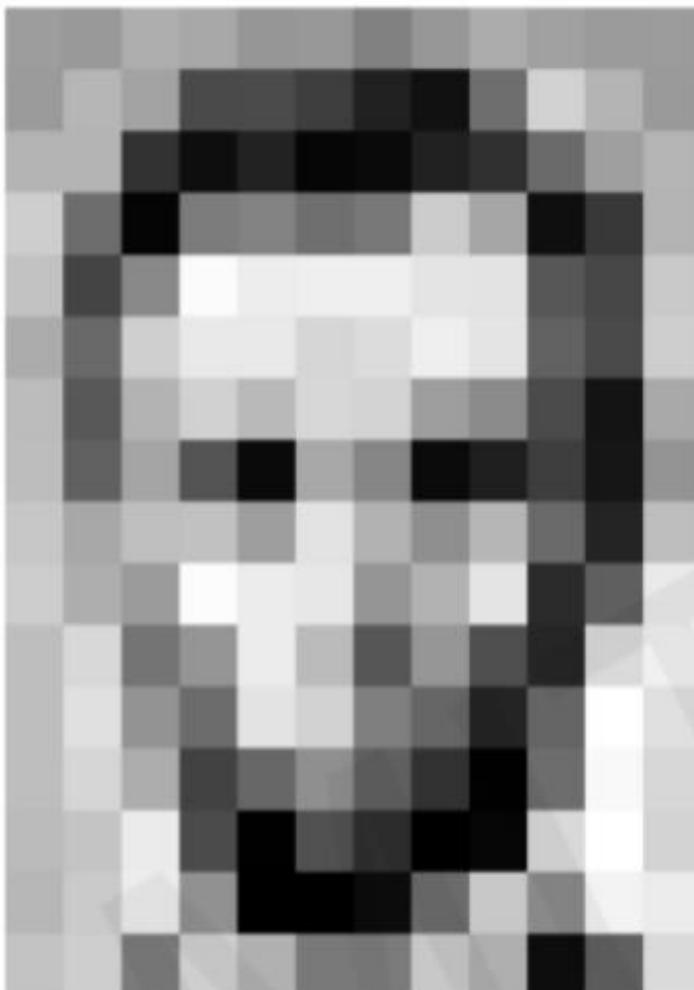
Low numbers are darker

Images are 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	6	124	101	111	120	204	166	15	56	180	
194	68	137	251	237	239	239	228	227	87	71	201	
172	105	207	253	253	214	220	239	228	98	74	206	
1	high numbers are brighter										20	169
189	97	165	84	10	168	134	11	31	62	22	148	
199	168	191	193	158	227	178	143	182	105	35	190	
205	174	155	252	236	231	149	178	228	43	95	234	
190	216	116	149	236	187	85	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	

Images are numbers



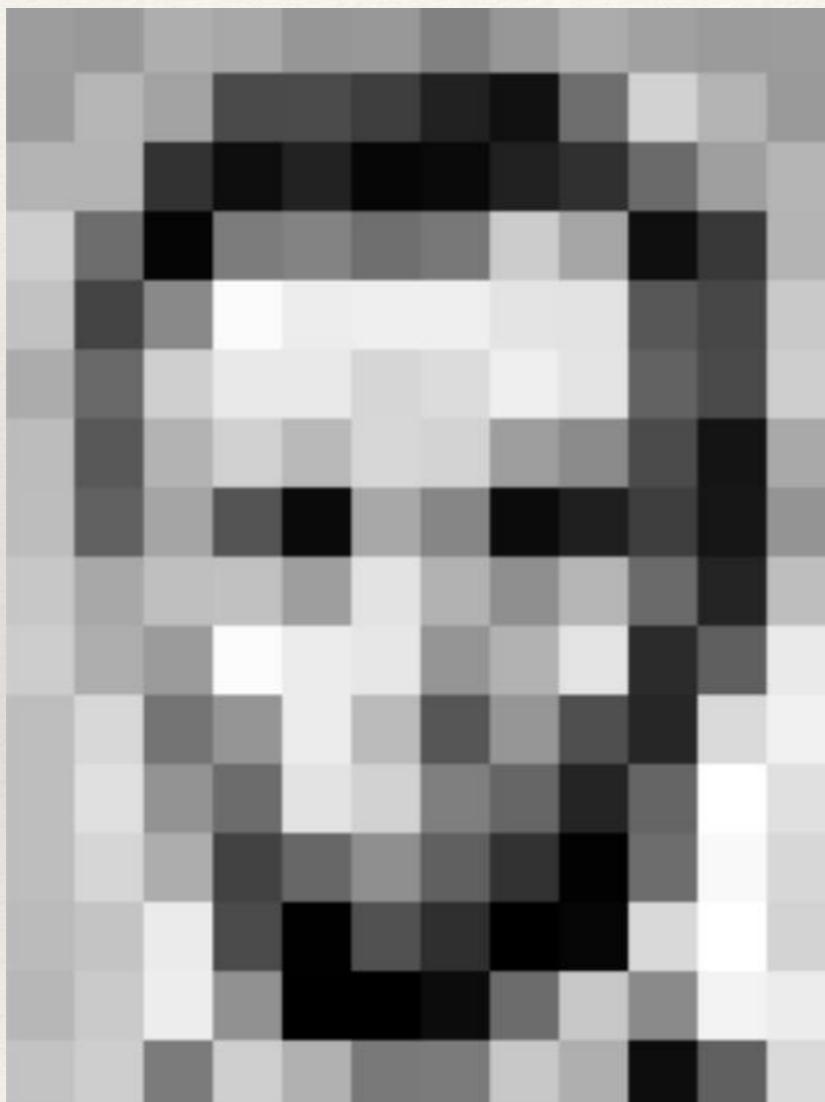
157	153	174	168	150	152	129	151	172	161	155	156
156	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	54	6	16	33	45	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	67	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	199	75	20	169
189	97	165	84	10	168	134	11	51	62	22	148
199	164	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	65	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
156	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	67	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	199	75	20	169
189	97	165	84	10	168	134	11	51	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	65	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

An image is just a matrix of numbers [0,255]!
i.e., 1080x1080x3 for an RGB image

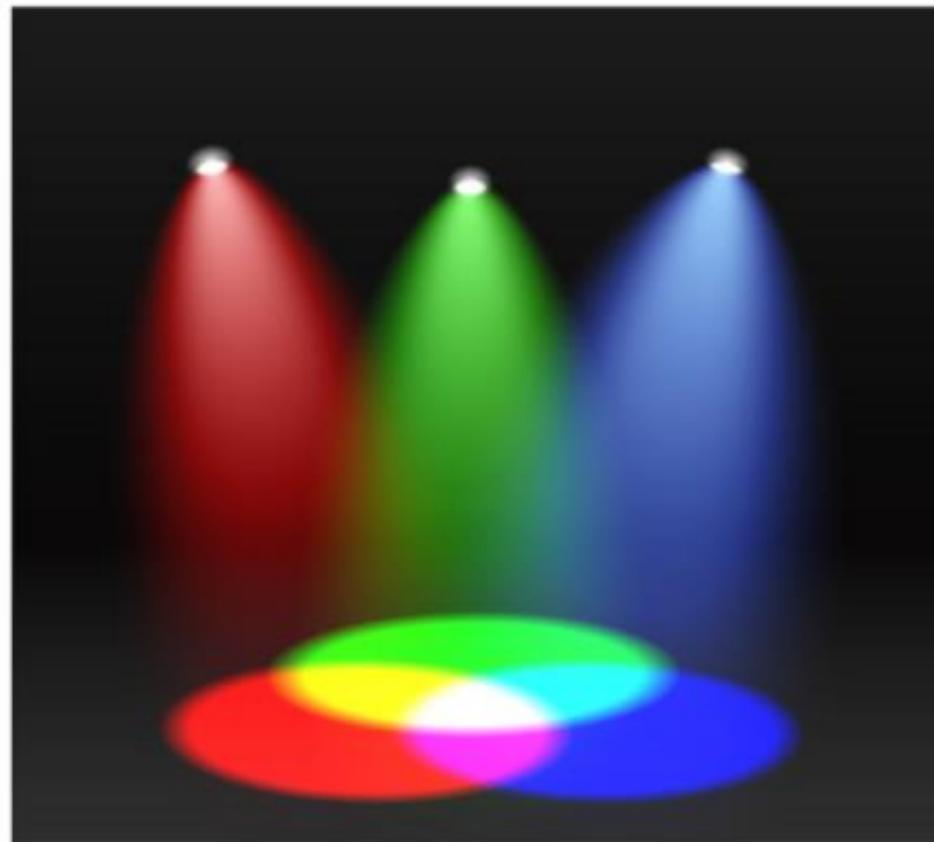
Grey level image



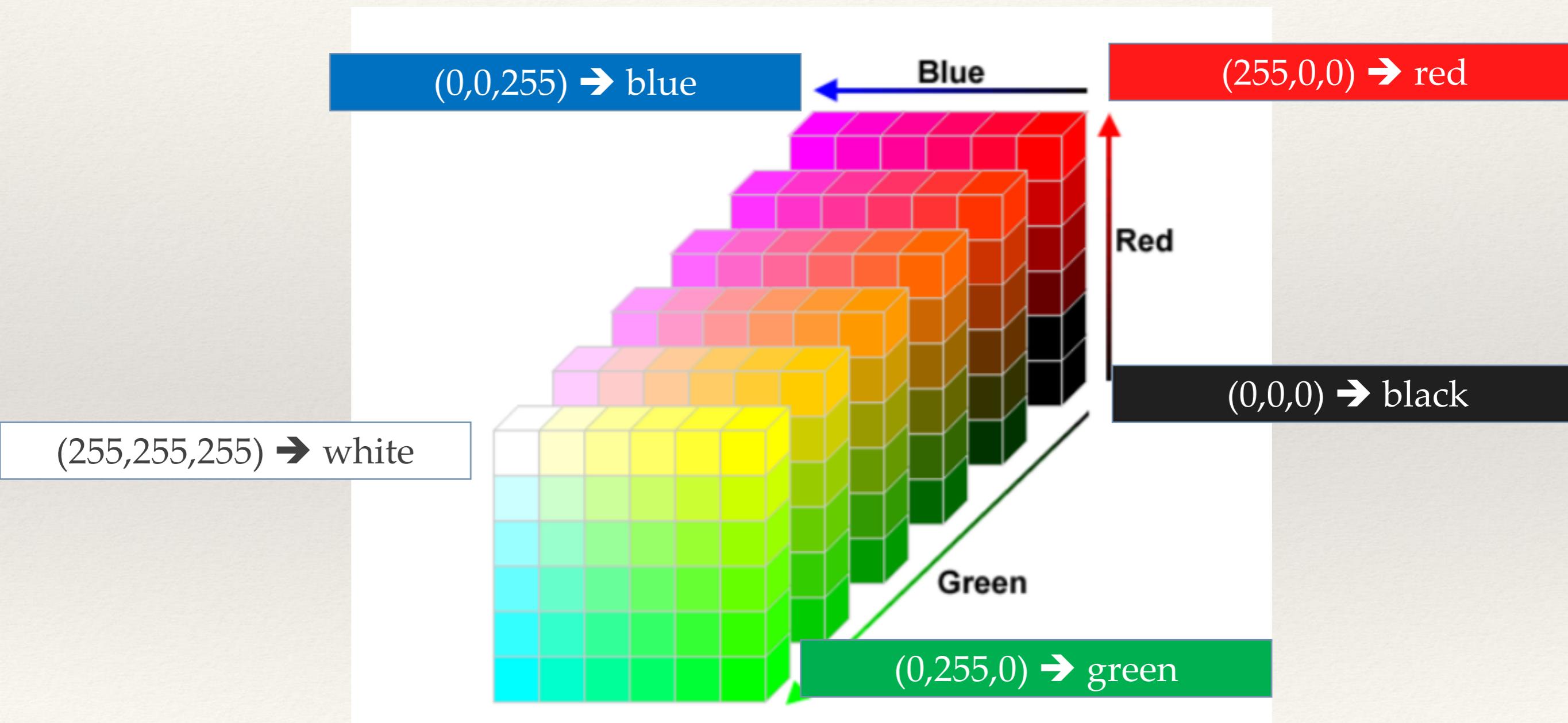
1 b/w band
0-255
0 = black
255 = white

Color image

תמונה צבעונית היא הרכבה של 3 תמונות



RGB Pallet



Tasks in computer vision – example 1

Classification



Dog	0.8
Cat	0.1
...	...
Horse	0.01

What is in the picture?

Tasks in computer vision – example 2

Regression #1



x, y, w, h

Where is the object?

Tasks in computer vision – example 3

Regression #2



X

How many cars in the parking lot?

Tasks in computer vision – example 4

Segmentation



Pixel (i,j) → segment

Tasks in computer vision

Many other tasks:

- Action Recognition
- Pose estimation
- Facial recognition
- 3d reconstruction
- ...

Challenges in computer vision

Viewpoint variation



Scale variation



Illumination conditions



Challenges in computer vision

Deformation



Occlusion



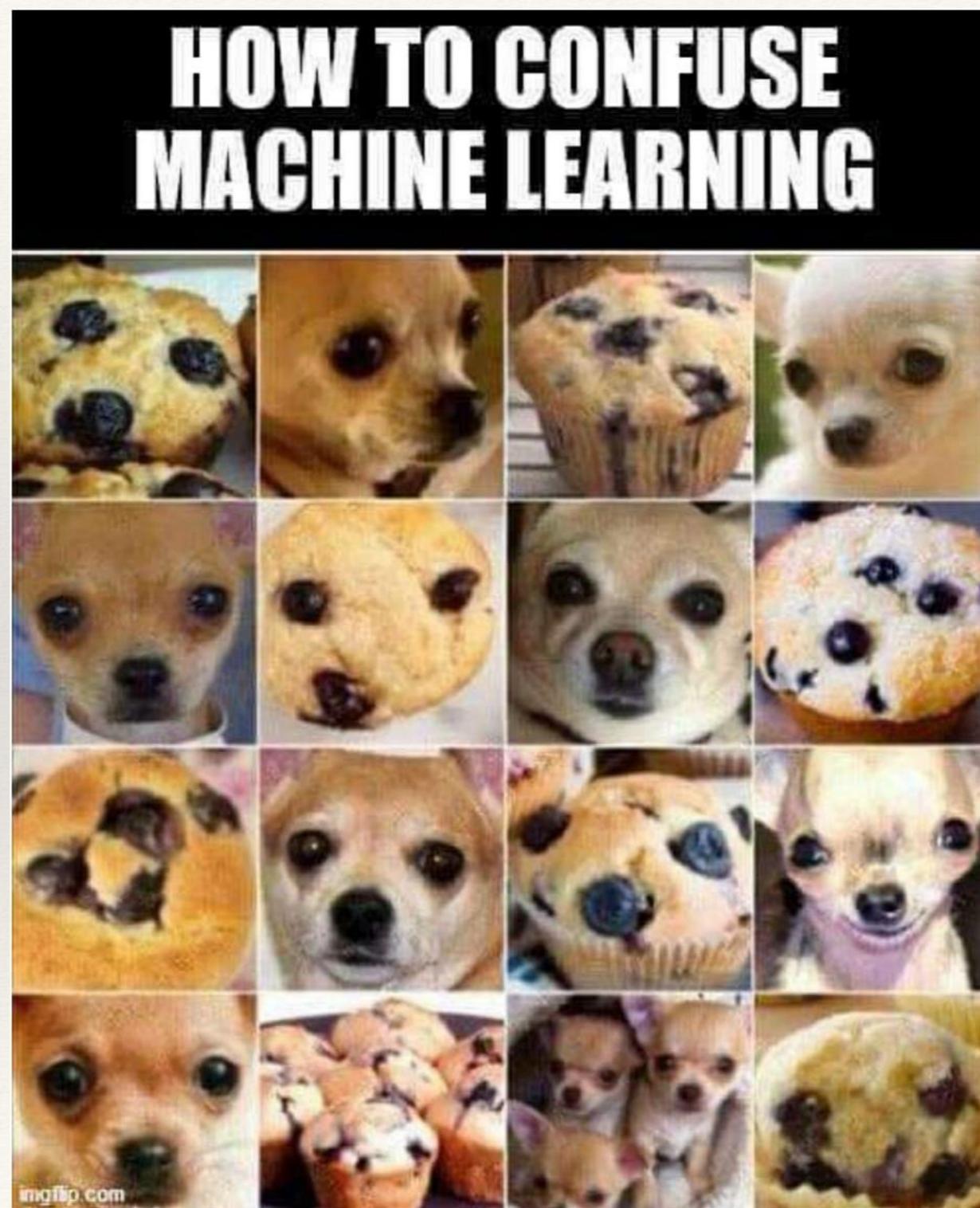
Background clutter



Intra-class variation

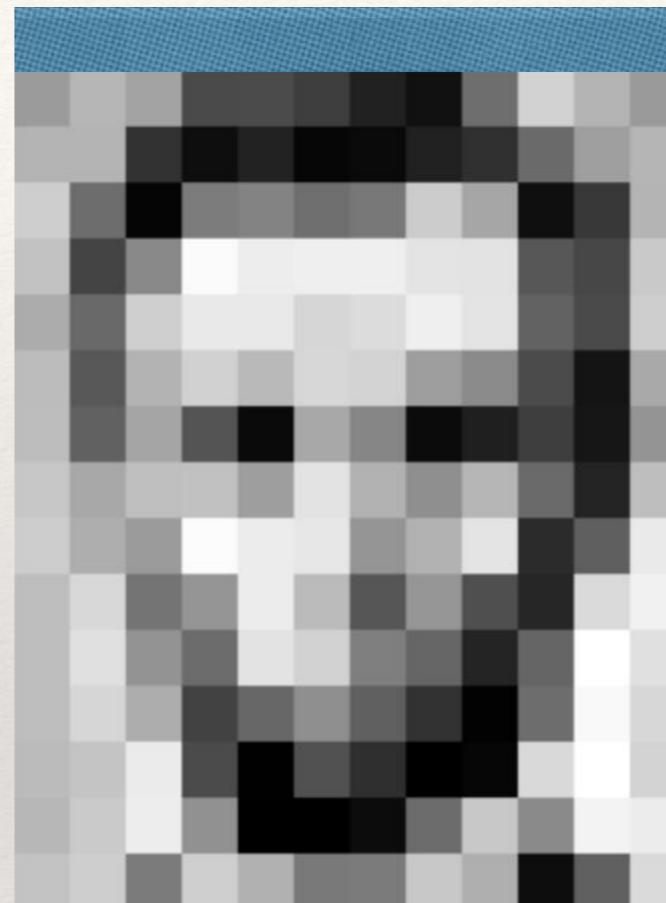


Challenges in computer vision



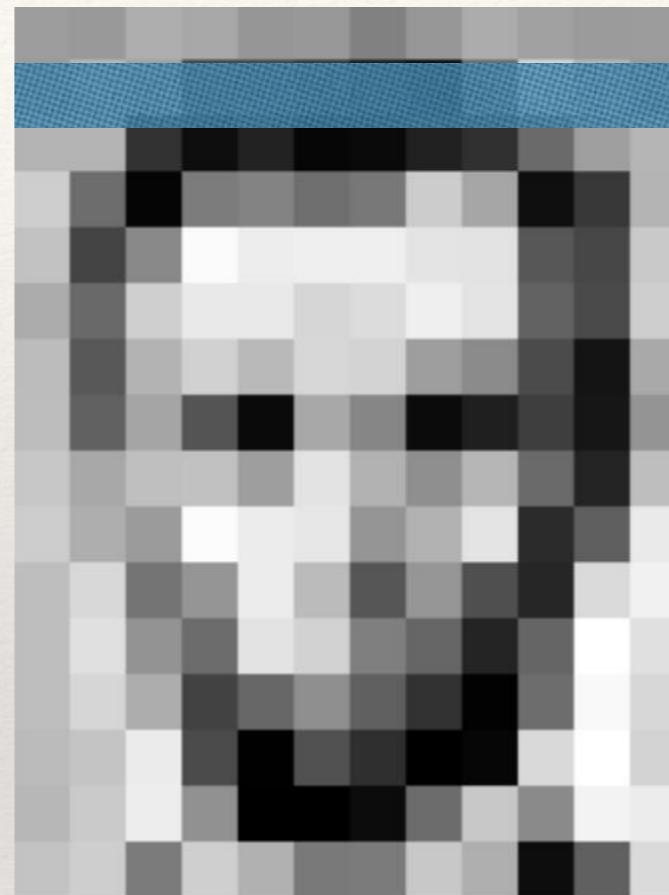
From image to features

Raw Pixels



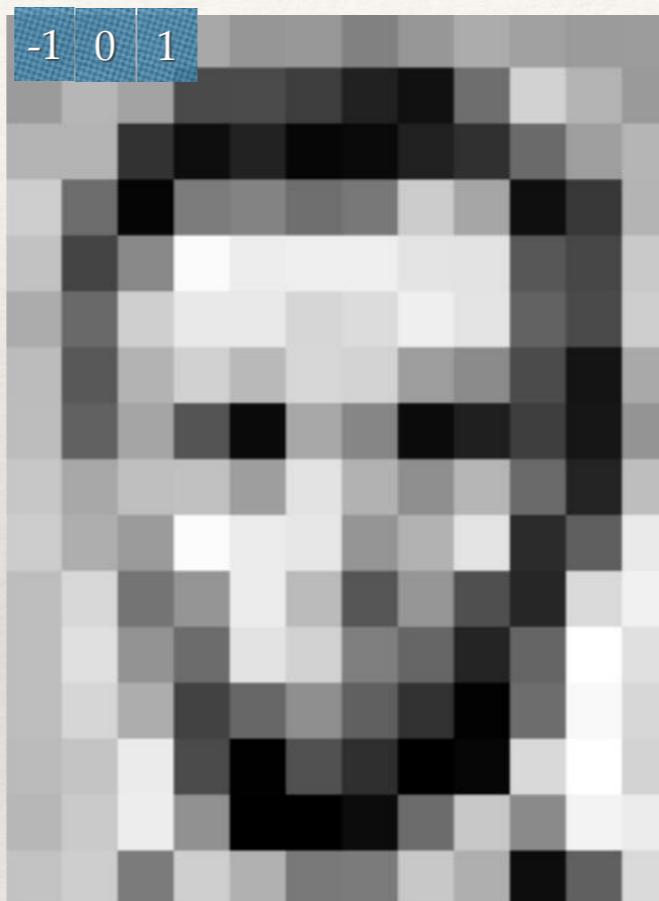
From image to features

Raw Pixels



...

From image to features



Horizontal Gradients

From image to features

Horizontal Gradients

233 | 214 | 220

-13

Weak gradient

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	94	6	10	33	49	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	106	207	233	-1	0	1	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	105	35	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	65	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	209	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

From image to features

Horizontal Gradients

10 | 168 | 134

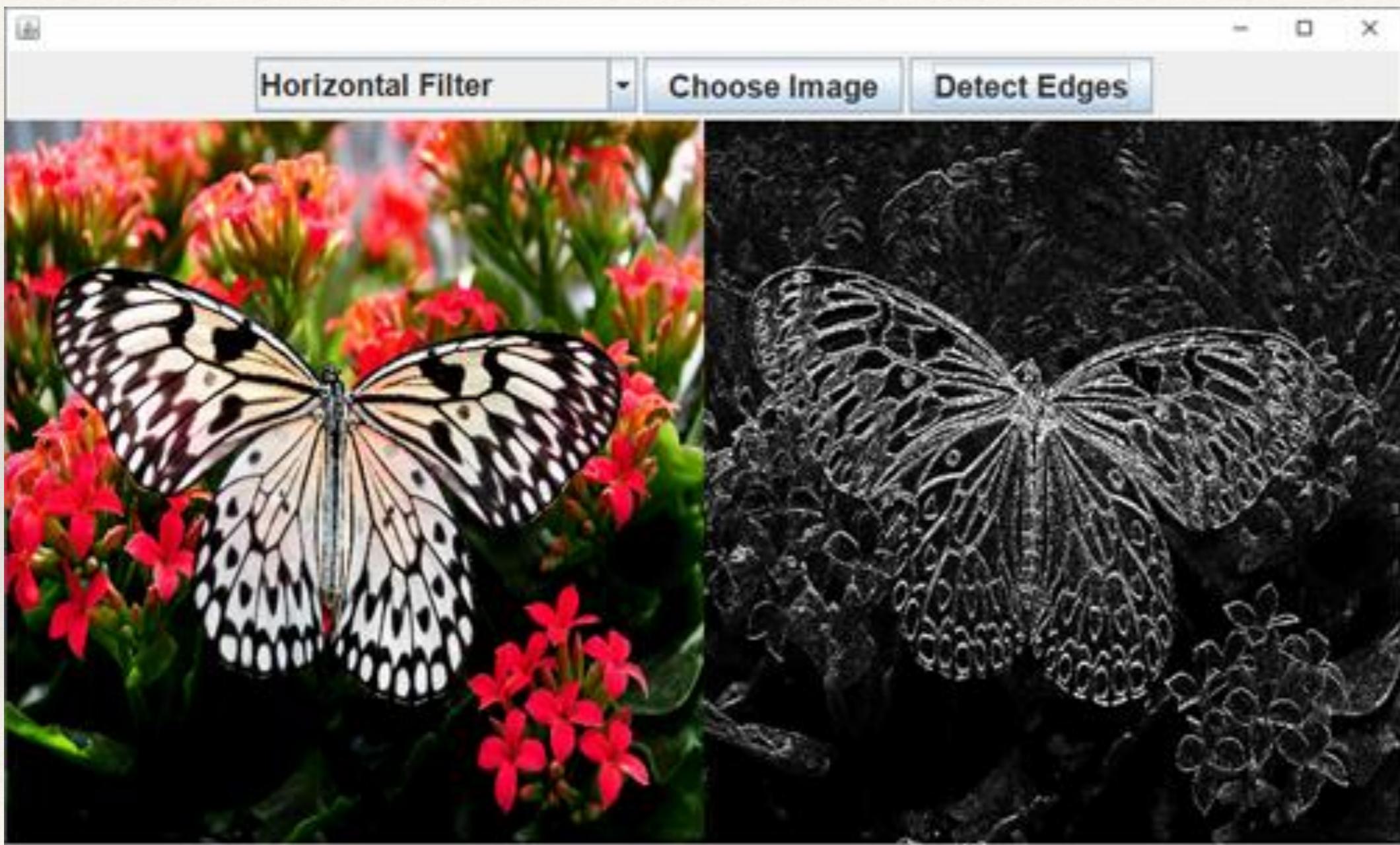
124

Strong gradient

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	94	6	10	33	49	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	106	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	-1	0	1	11	31	62	22	148
199	168	191	193	158	227	178	143	182	105	35	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	65	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	209	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

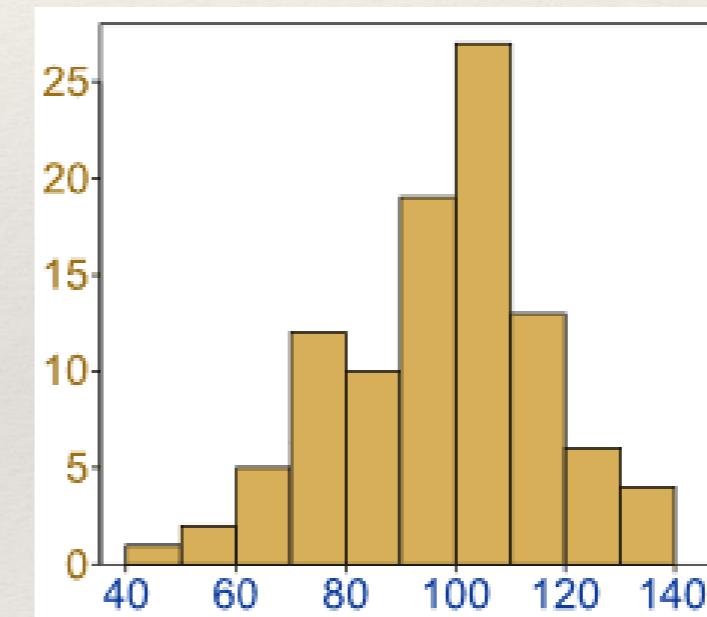
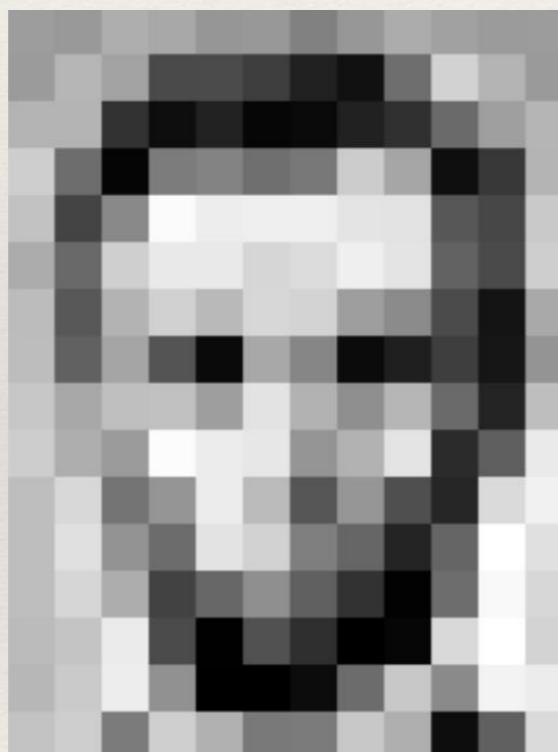
From image to features

Horizontal Gradients - example



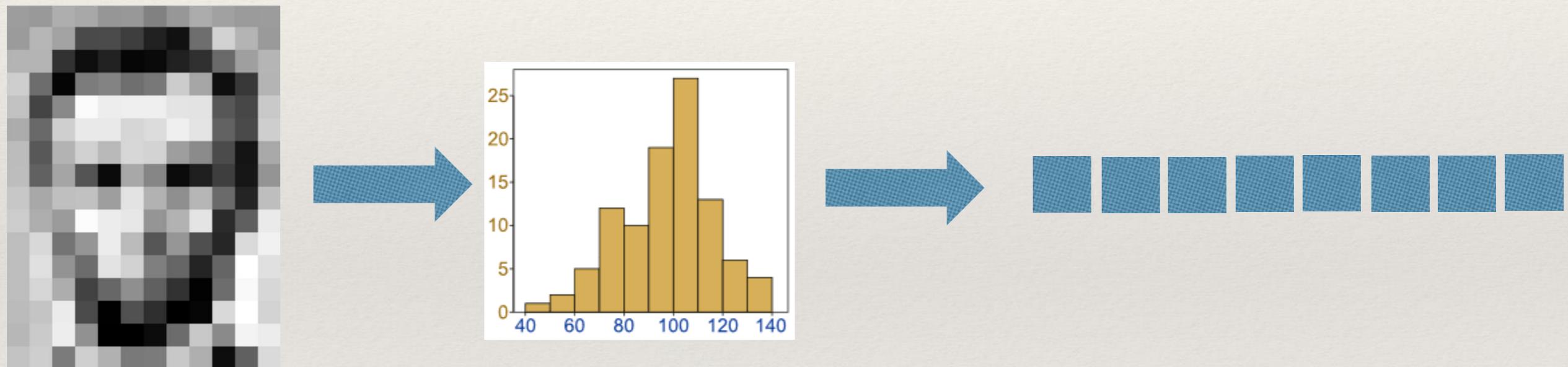
From image to features

Image Descriptors, e.g. Histograms



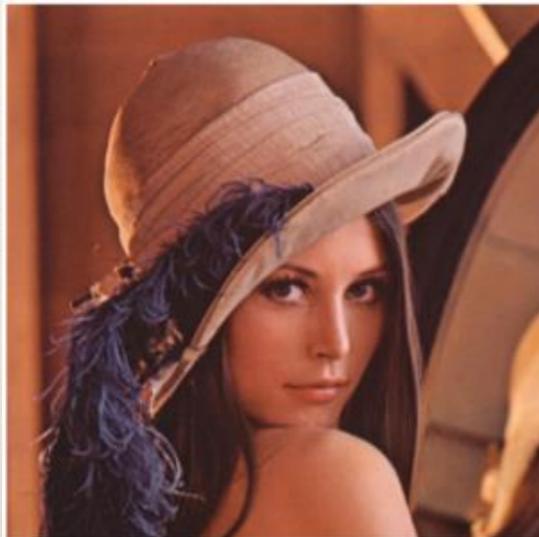
From image to features

Image Descriptors, e.g. Histograms



Traditional approach

Manually Identify key features in each category.



Nose
Eyes
Mouth



Wheels
License Plate
Headlights

Train ML classifier on these features.

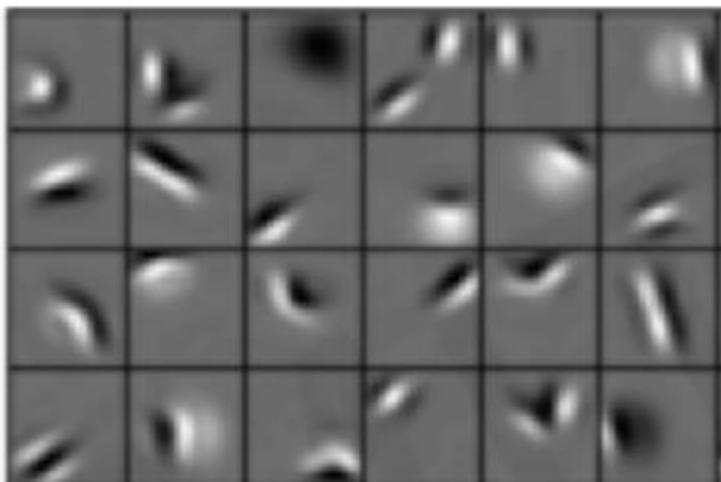
Traditional approach

- This was the traditional computer vision approach
- Engineers handcrafted new features & handed them as vectors to ML algorithms
- Better features / combinations of features improved performance

Automatic feature extraction

Can we learn hierarchy of features directly from the data?

Low level features



Edges, dark spots

Mid level features



Eyes, ears, nose

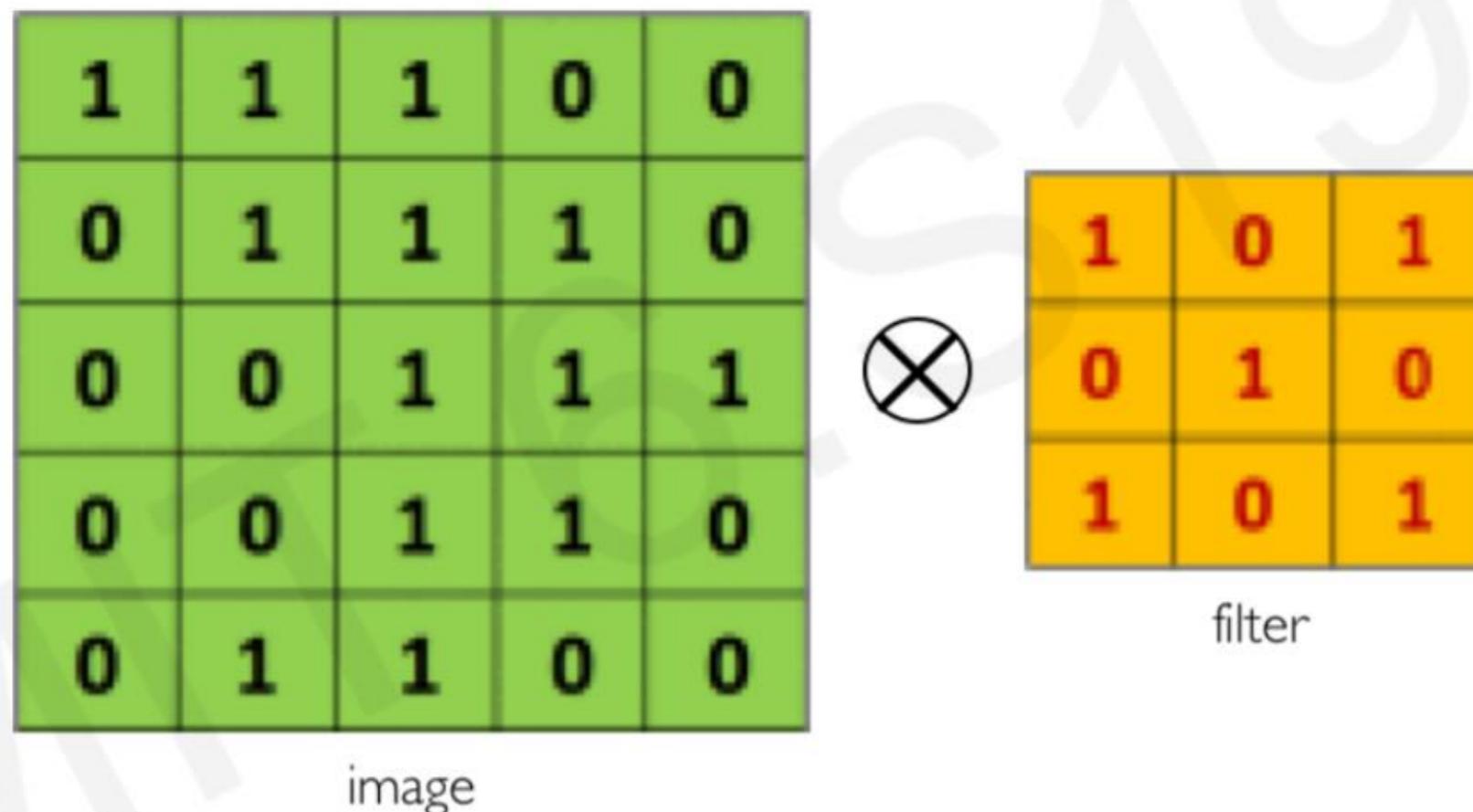
High level features



Facial structure

convolutions

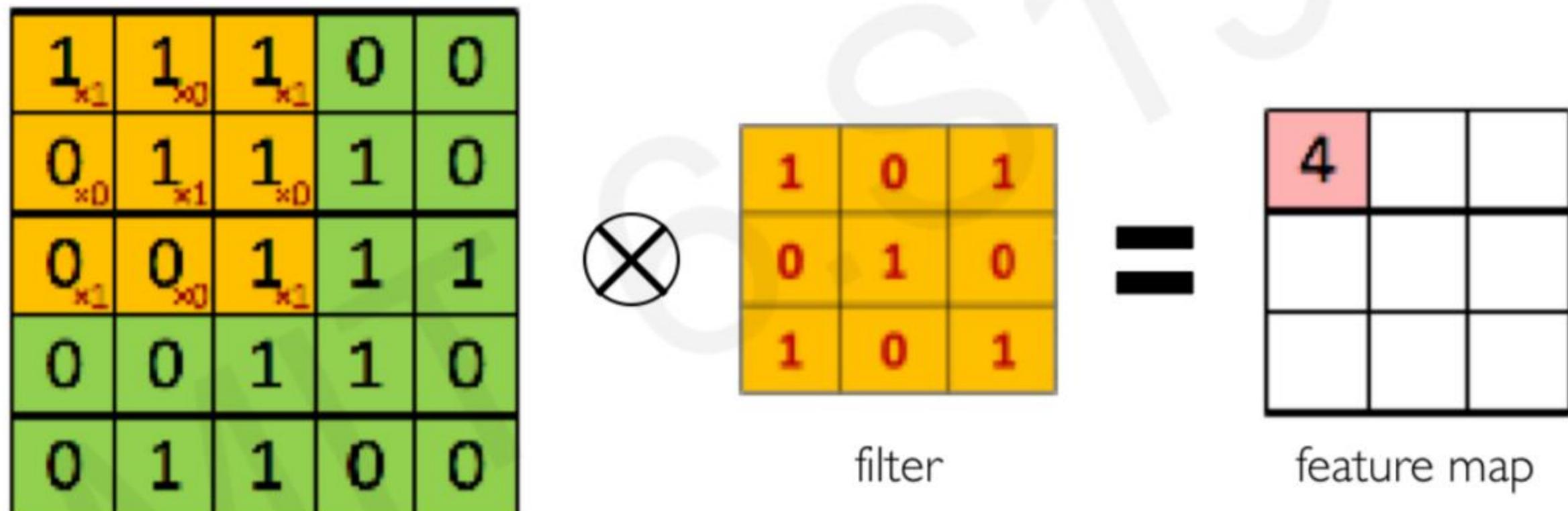
Suppose we want to compute the convolution of a 5x5 image and a 3x3 filter:



We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs..

convolutions

We slide the 3×3 filter over the input image, element-wise multiply, and add the output



convolutions

We slide the 3x3 filter over the input image, element-wise multiply, and add the output.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0



1	0	1
0	1	0
1	0	1

filter



4	3	

feature map

convolutions

We slide the 3×3 filter over the input image, element-wise multiply, and add the outputs:

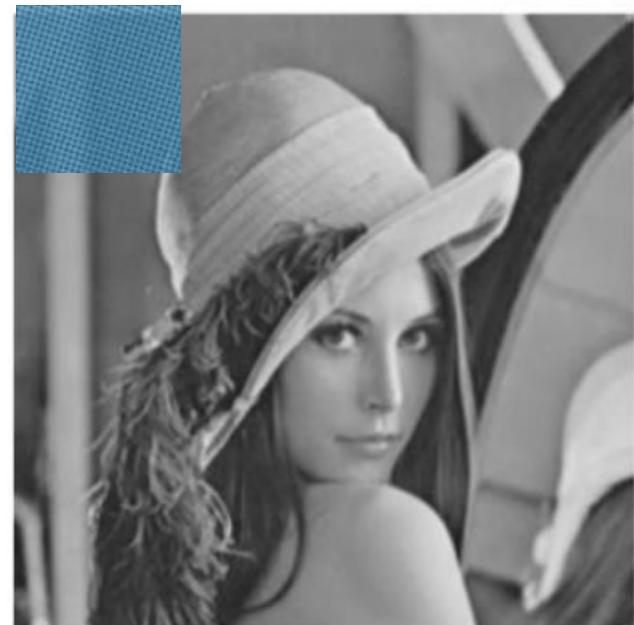
The diagram illustrates the convolution process. On the left is a green 5×5 input image with values: $\begin{matrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{matrix}$. A yellow 3×3 filter with values $\begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix}$ is shown in the center. An element-wise multiplication symbol (\otimes) is placed between the input and filter. To the right of the filter is an equals sign (=). Below the filter is the label "filter". To the right of the equals sign is a pink 3×3 feature map with values $\begin{matrix} 4 & 3 & 4 \\ 2 & 4 & 3 \\ 2 & 3 & 4 \end{matrix}$, labeled "feature map". Red annotations show the result of element-wise multiplication for the first step: $1 \times 1 = 1$, $1 \times 0 = 0$, $1 \times 1 = 1$, $0 \times 1 = 0$, $1 \times 0 = 0$, $1 \times 1 = 1$.

convolutions

We slide the 3×3 filter over the input image, element-wise multiply, and add the outputs:

The diagram illustrates the convolution process. On the left is a green 5×5 input image with values: $\begin{matrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{matrix}$. In the middle, a yellow 3×3 filter with values $\begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix}$ is shown with a multiplication symbol (\otimes). To the right is a pink 3×3 feature map with values $\begin{matrix} 4 & 3 & 4 \\ 2 & 4 & 3 \\ 2 & 3 & 4 \end{matrix}$. Below the filter is the label "filter", and below the feature map is the label "feature map".

Convolutions produce feature maps



Original



Sharpen



Edge Detect



"Strong" Edge
Detect

Convolutions large or small?

X or X?

?

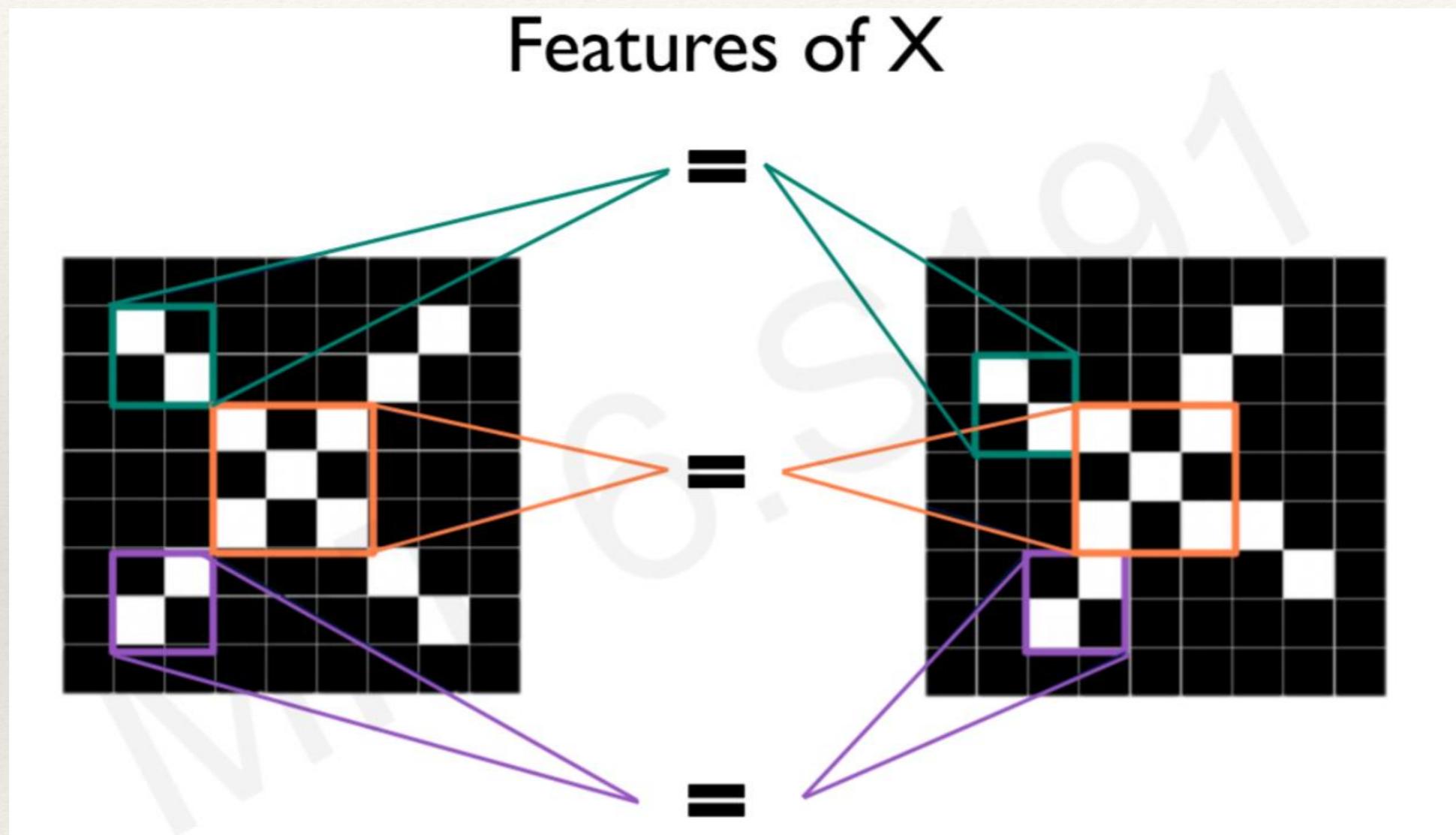
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

-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

Image is represented as matrix of pixel values... and computers are literal!
We want to be able to classify an X as an X even if it's shifted, shrunk, rotated, deformed.

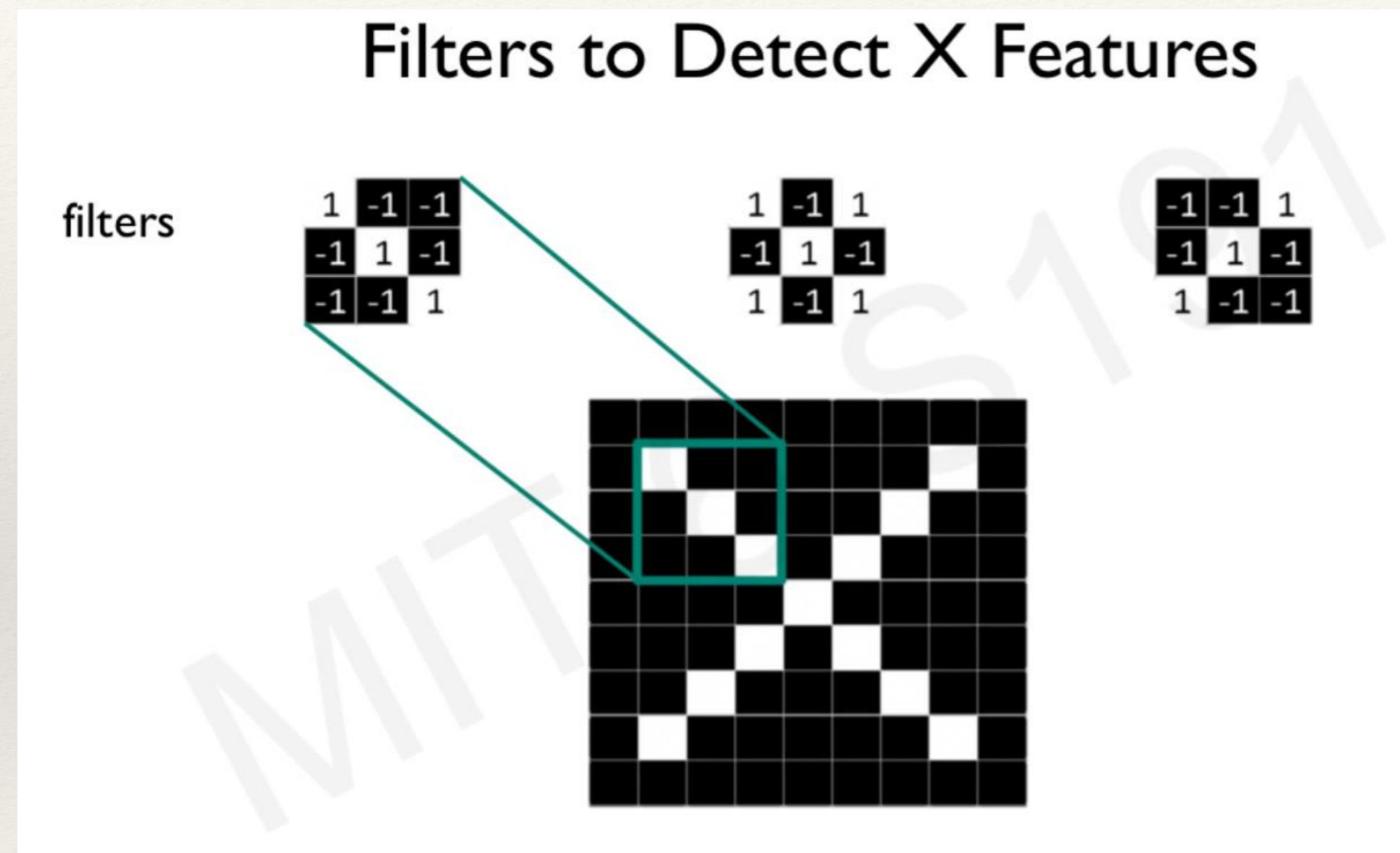
Large convolutions will often fail to capture these variations

Convolutions large or small?



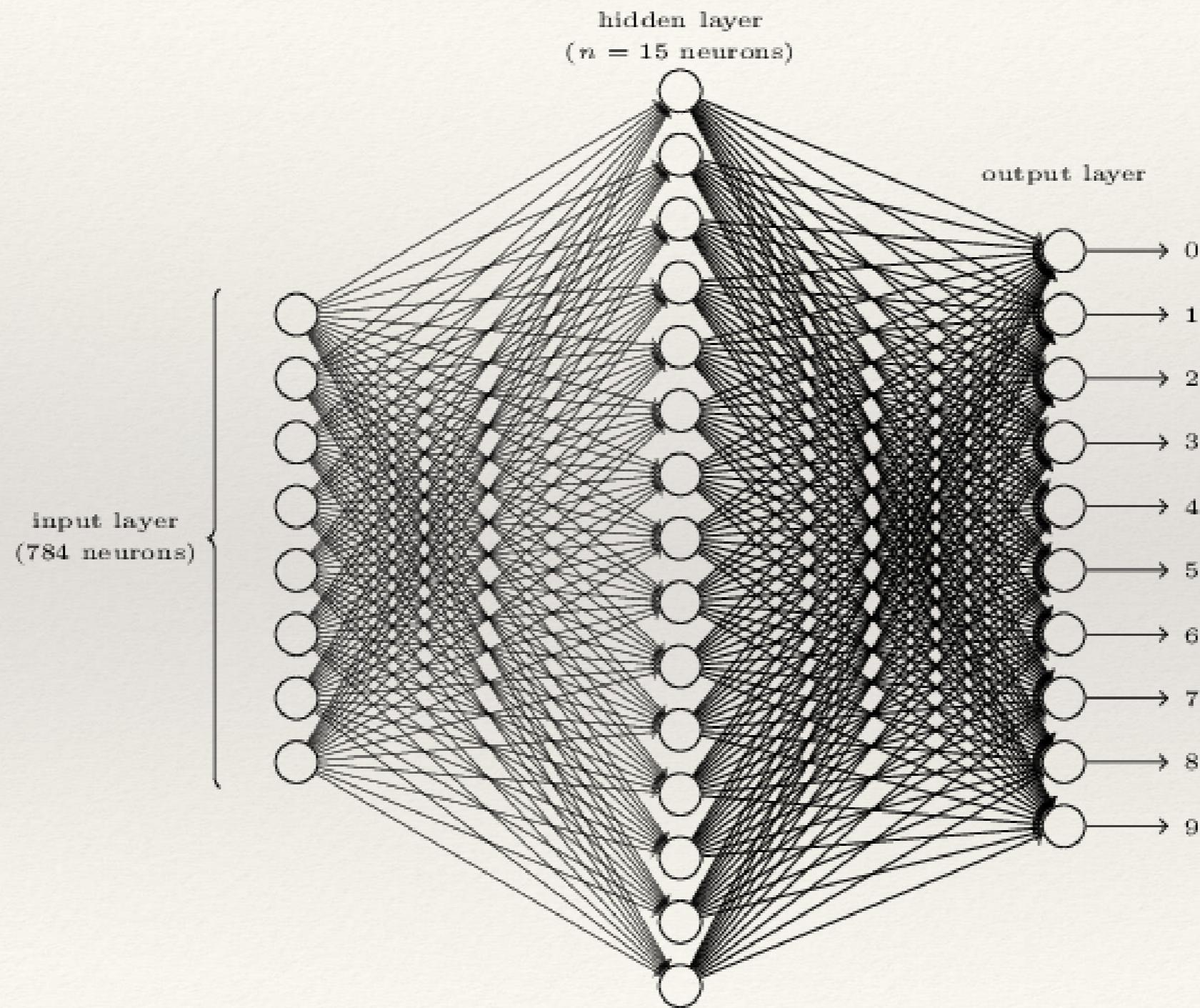
Small convolutions can identify low level features

Convolutions large or small?



Convolutions on the low-level feature map produce mid-level features and so on

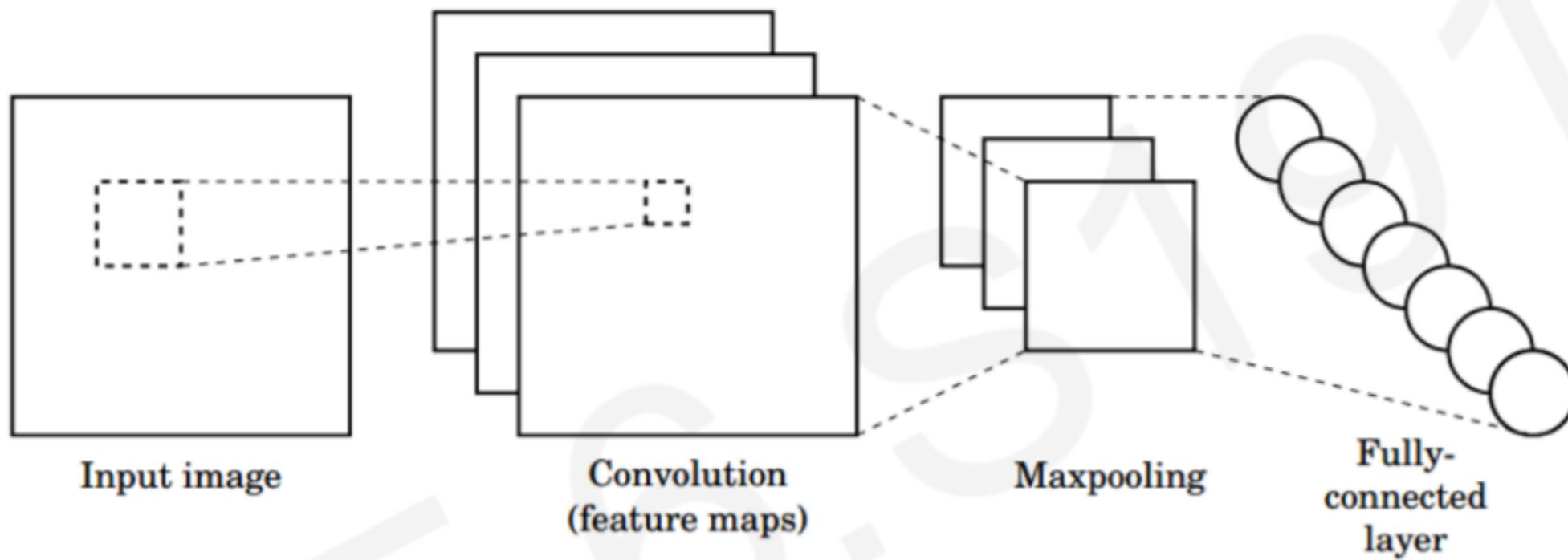
Fully connected NN - reminder



Convolutional Neural Networks

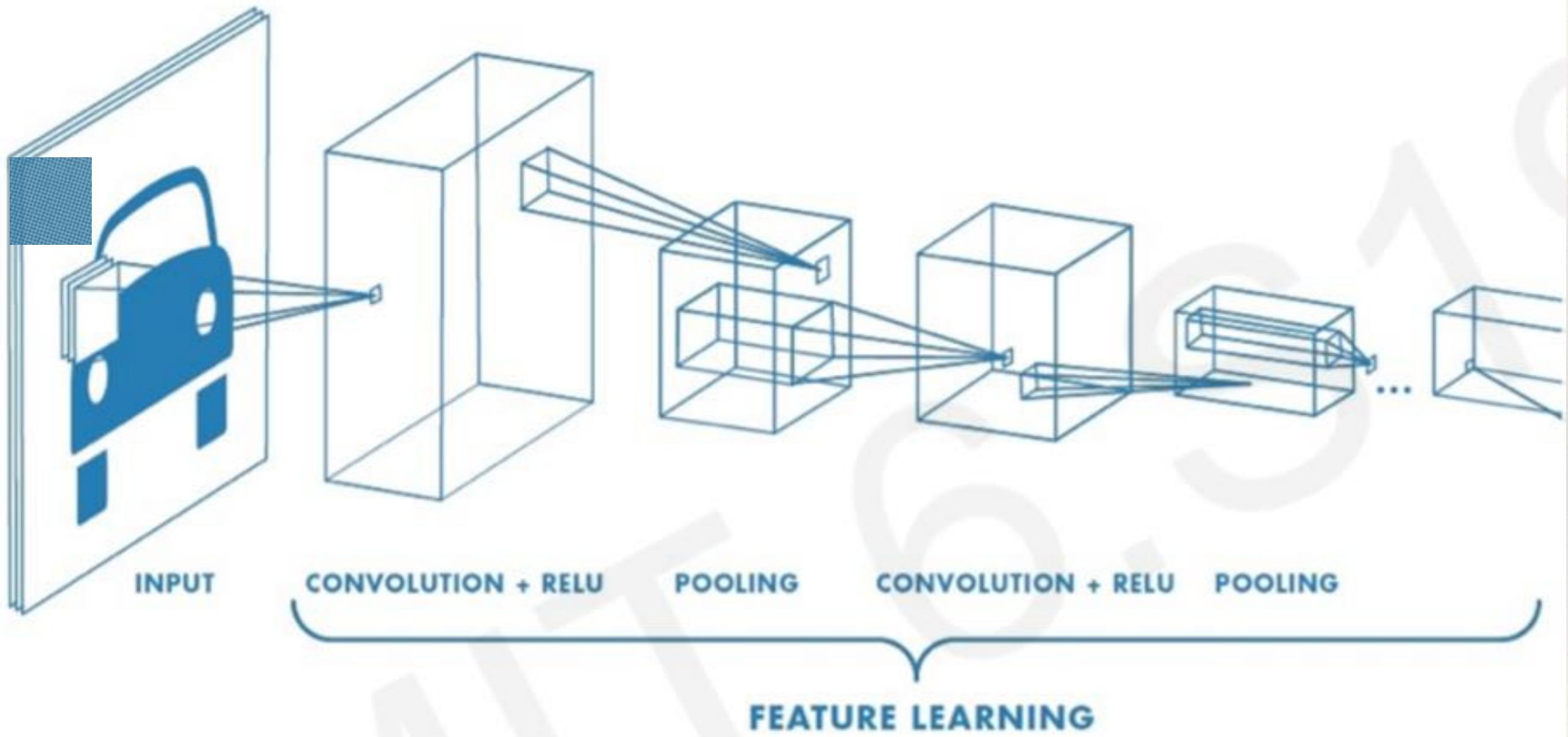
Deep Learning

CNNs for Classification



- 1. Convolution:** Apply filters to generate feature maps.
- 2. Non-linearity:** Often ReLU.
- 3. Pooling:** Downsampling operation on each feature map.

Convolution

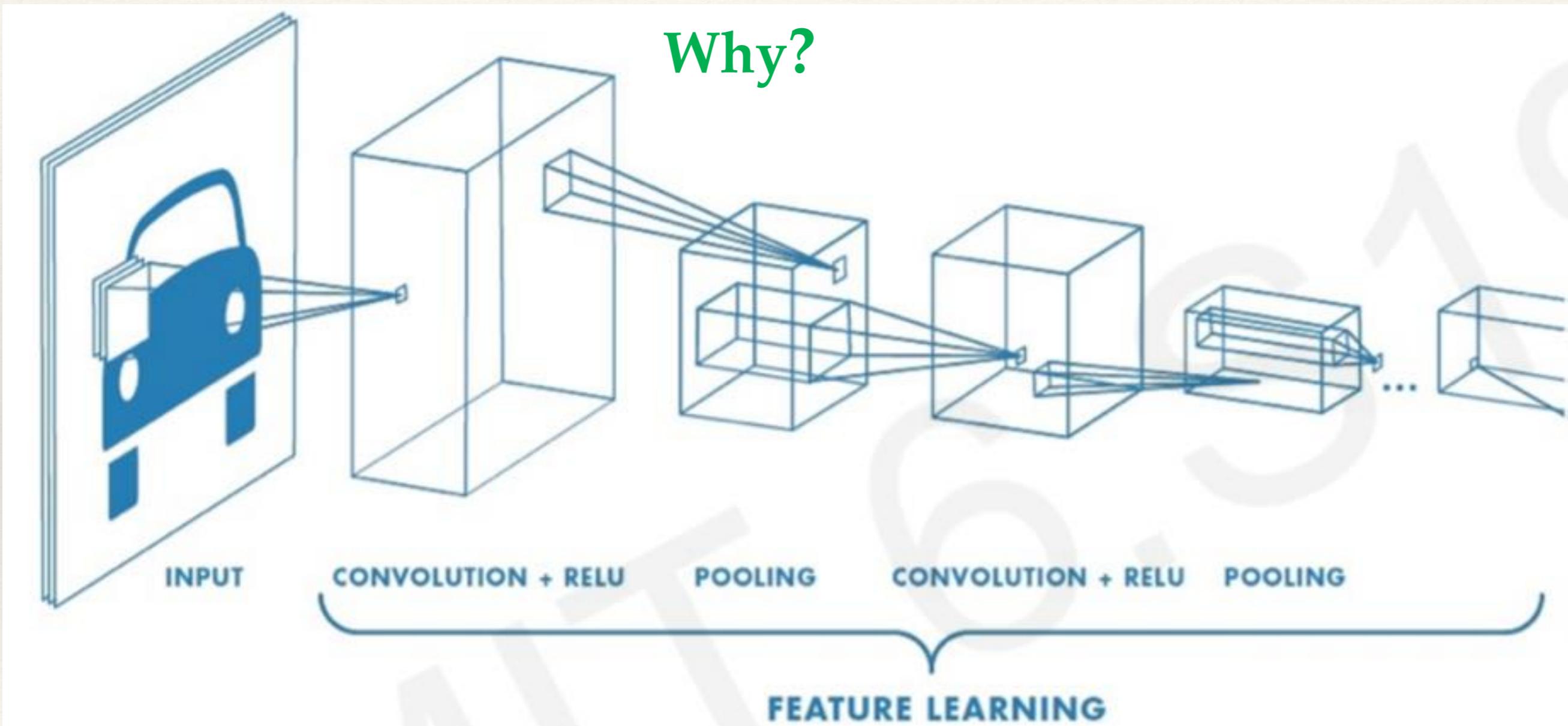


Convolution

Input: $w * h * 3$

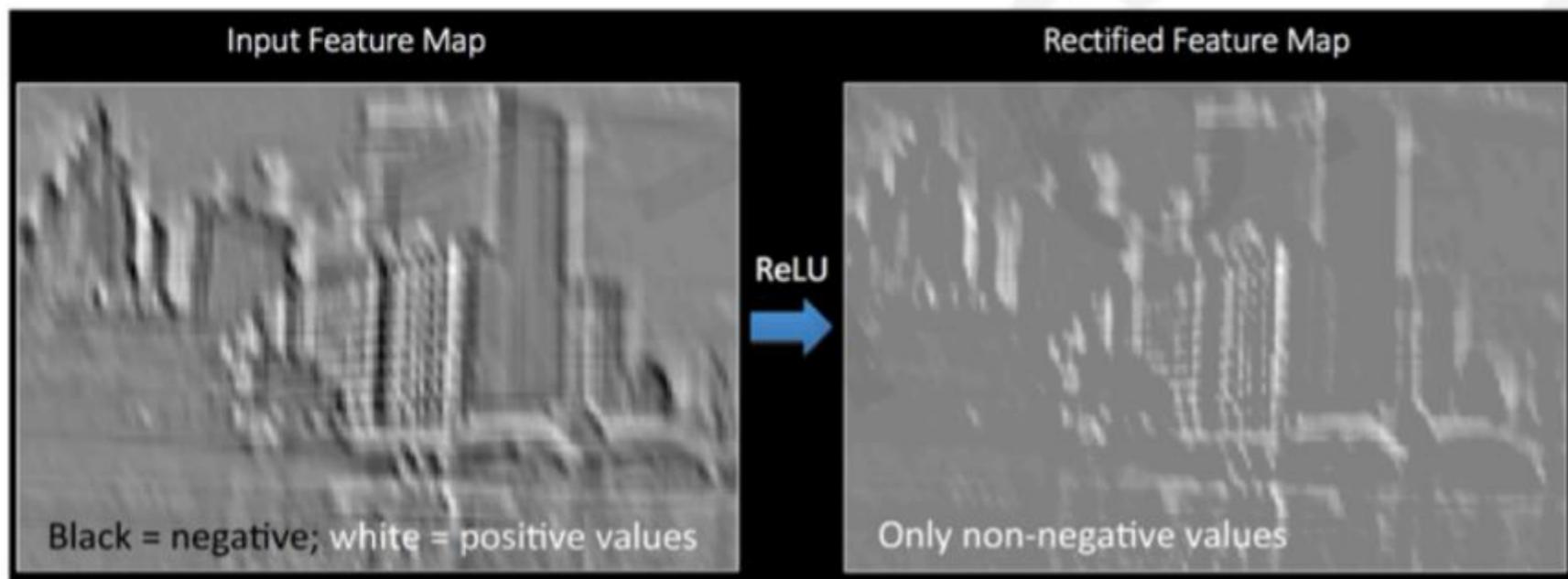
$w * h * k$

Why?

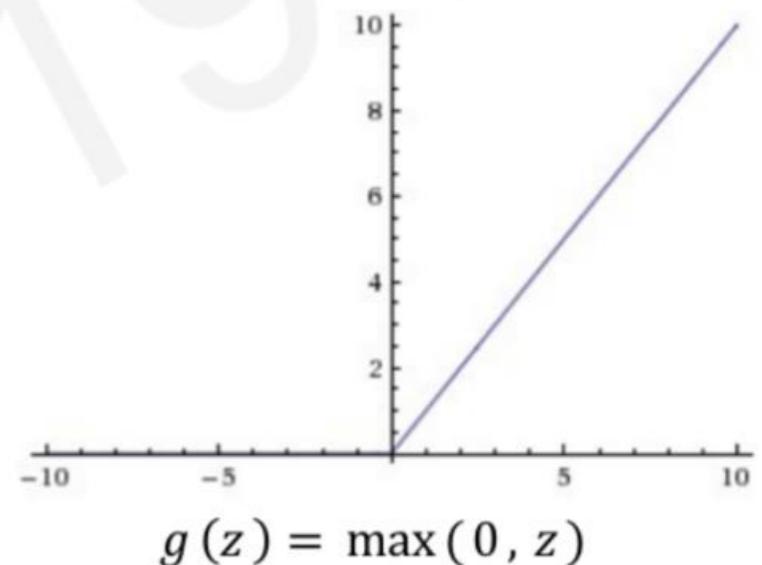


ReLU - activation

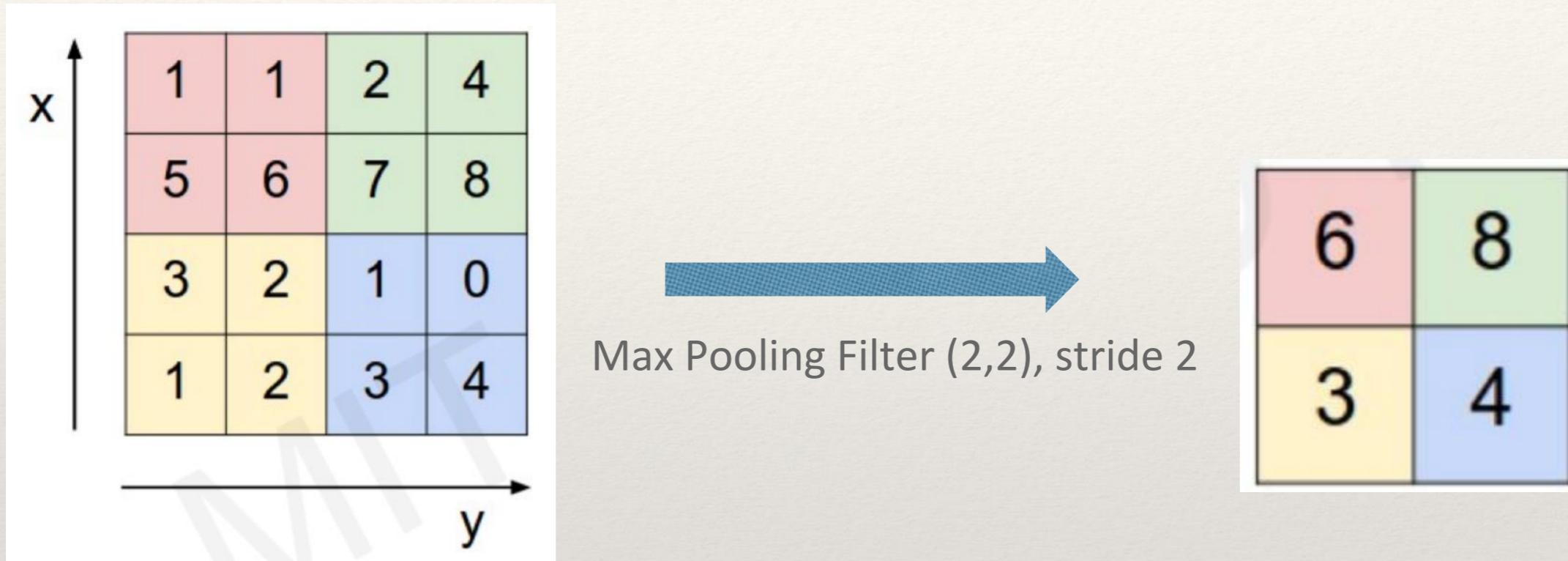
- Apply after every convolution operation (i.e., after convolutional layers)
- ReLU: pixel-by-pixel operation that replaces all negative values by zero. **Non-linear operation!**



Rectified Linear Unit (ReLU)



Pooling



Max pooling helps convolutions on the next layer to enlarge the spatial surrounding. Why?

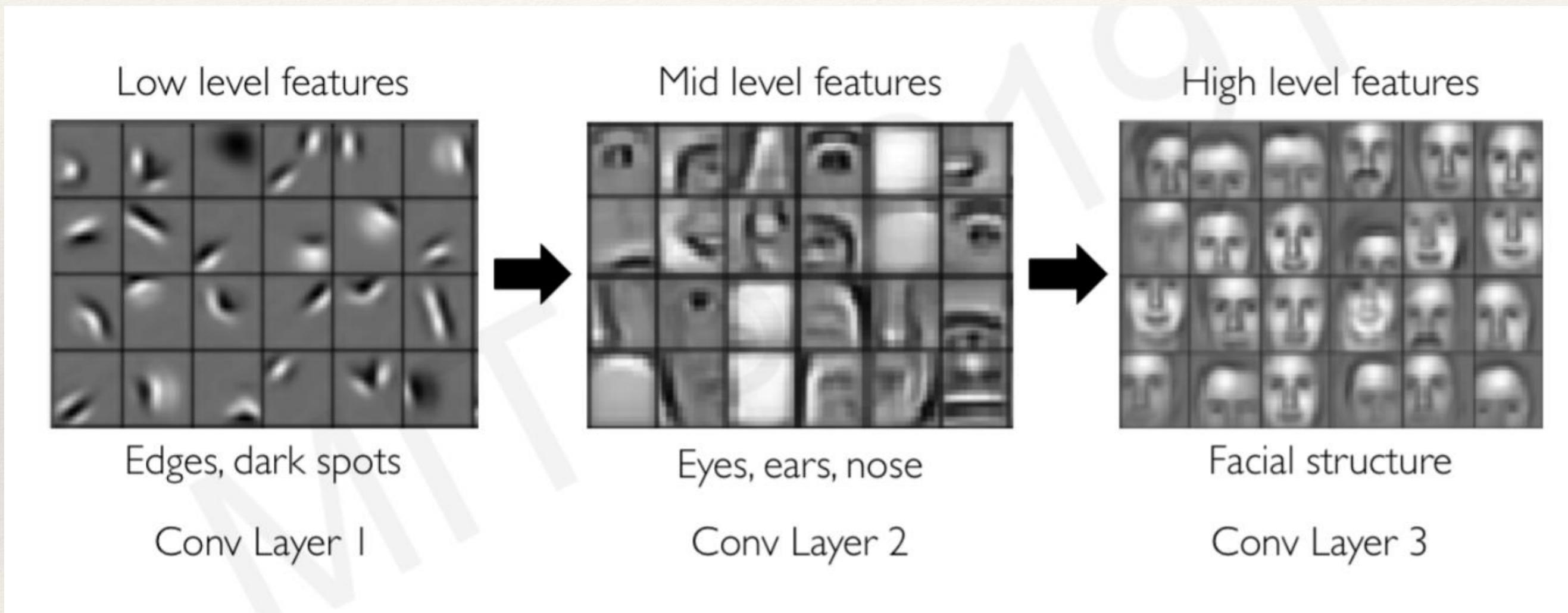
Pooling



Convolutions of constant size (typically 3x3)

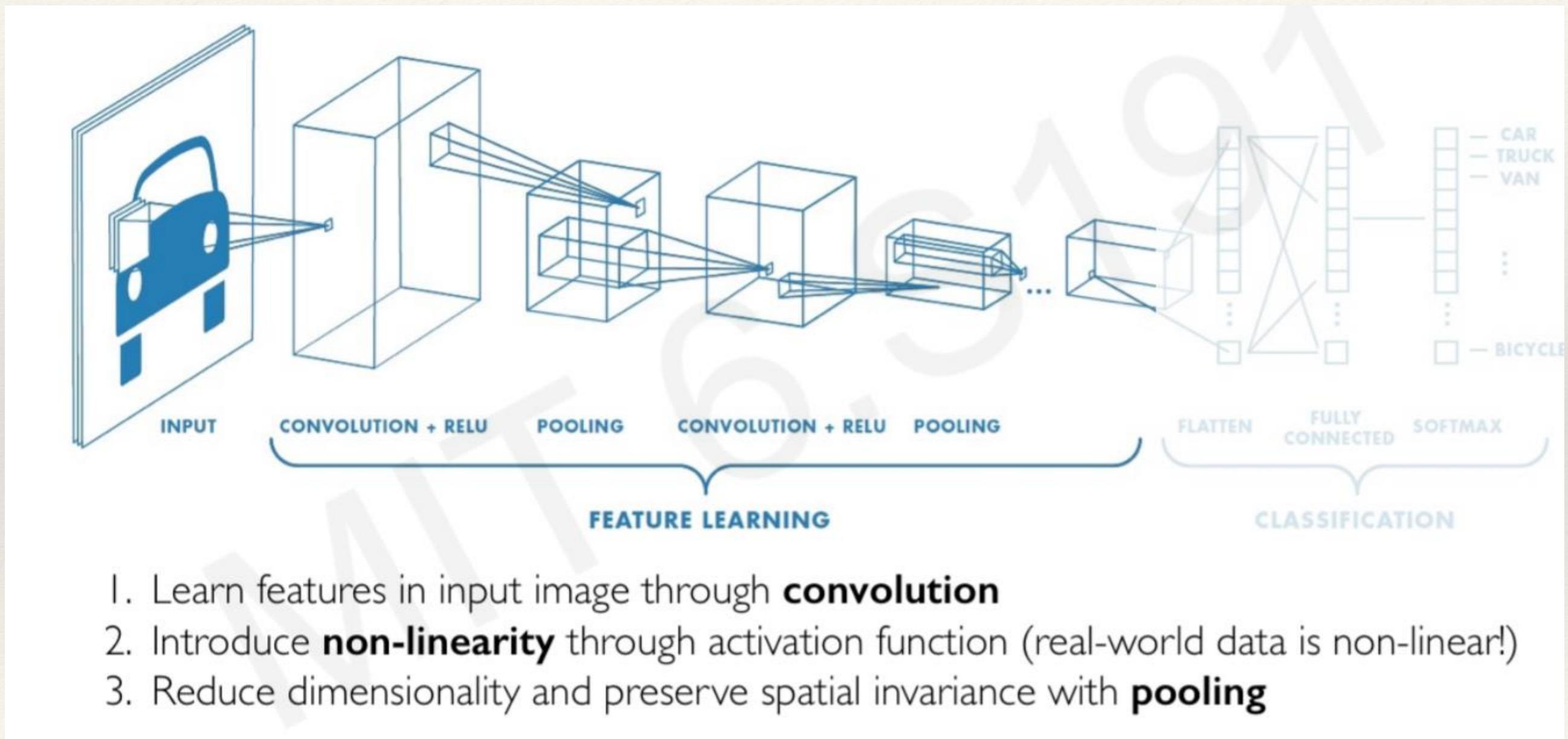
Feature map is scaled down

Representation learning with CNNs

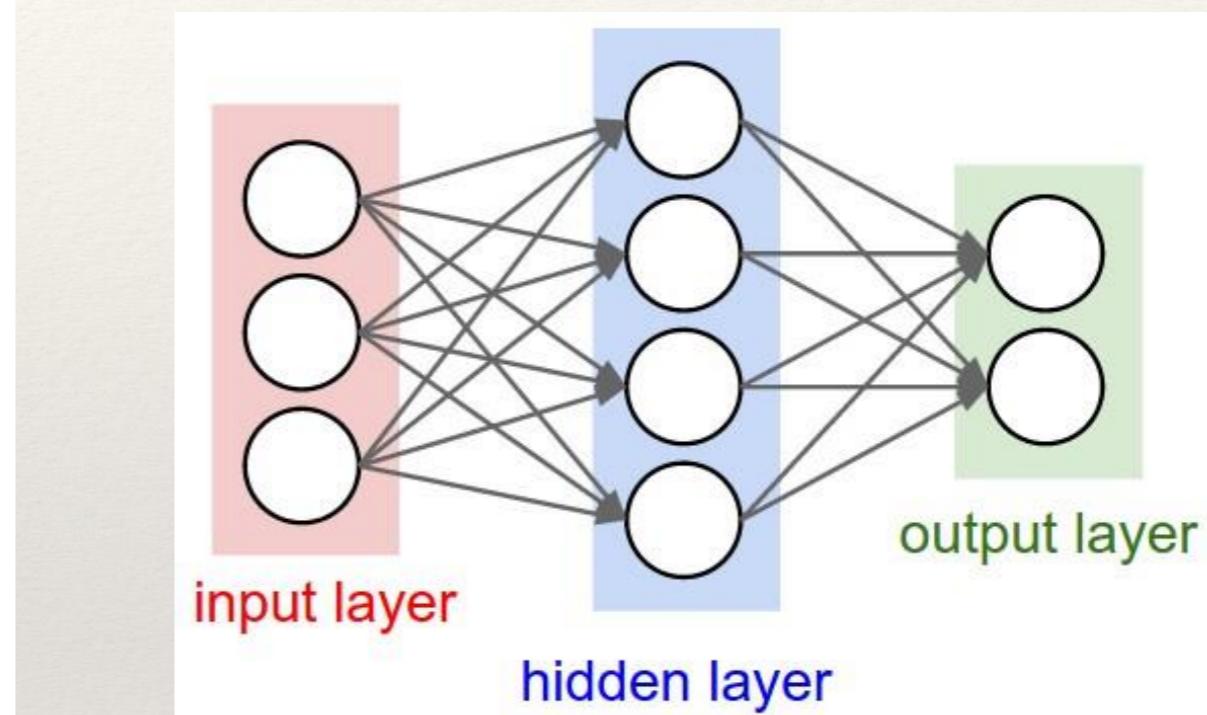
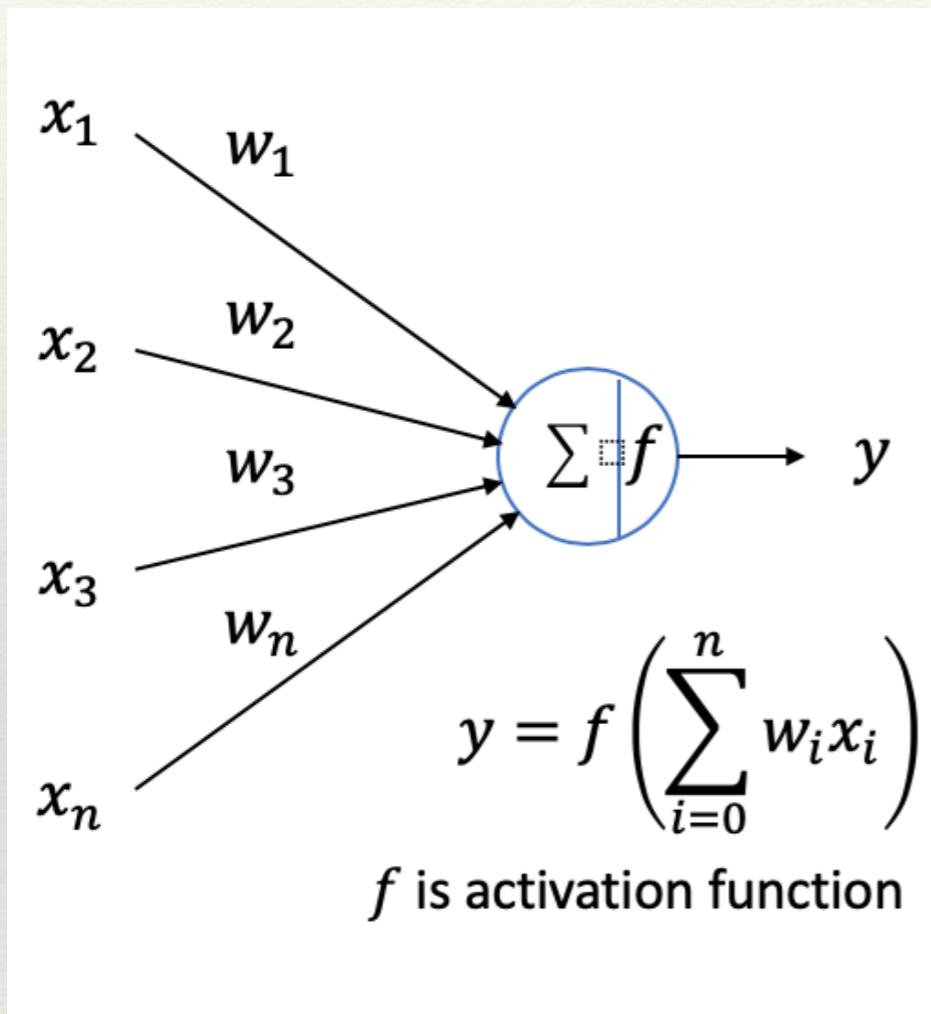


Each layer (Convolution + Relu + pooling) works on the previous layer's output
Produces higher level features

CNNs for classification – Features learning

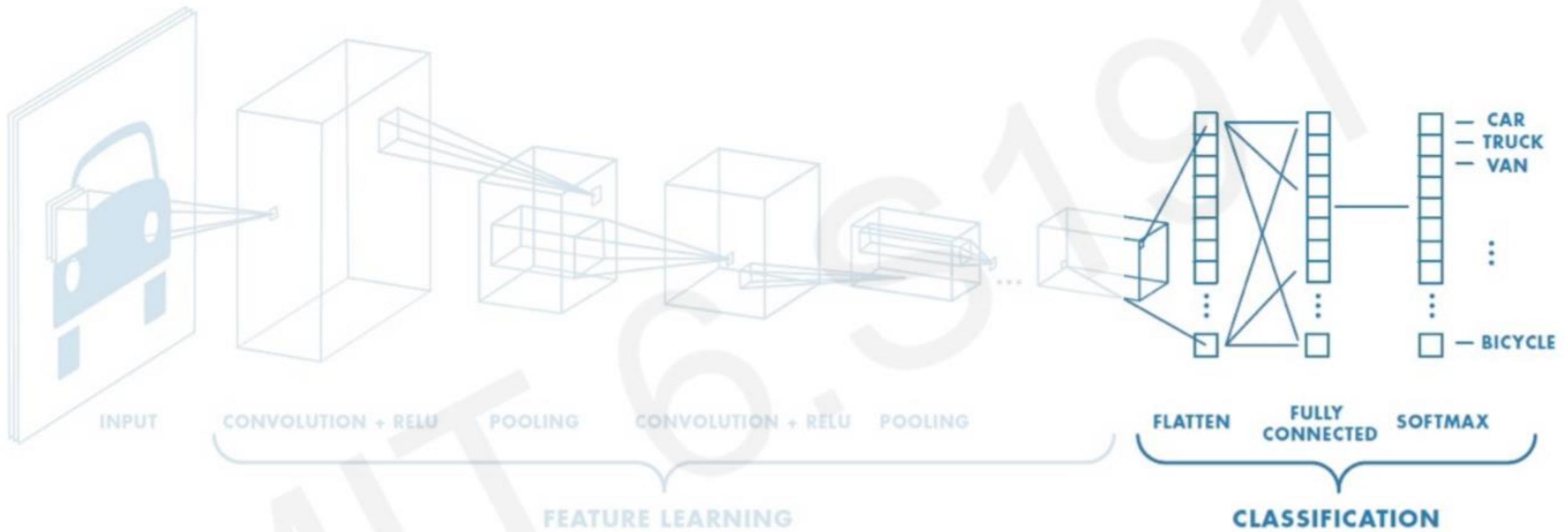


Fully connected layer



Use fully connected layer to predict classification output

CNNs for classification – Class probabilities



- CONV and POOL layers output high-level features of input
- Fully connected layer uses these features for classifying input image
- Express output as **probability** of image belonging to a particular class

$$\text{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

An architecture for many applications



Convolutional layer produces strong features automatically. connect the fully connected layer, and learn the appropriate weights

Summary

- Computers “see” image as a matrix of numbers
- Traditionally, ML specialists generated handcrafted features for each problem, and then applied machine learning techniques to solve classification \ regression problems.
- Nowadays, feature representation is learned automatically using CNNs
- CNNs generates powerful features, which can represent image data.
- The feature layers are then connected to a fully connected layer, to make a classification \ regression decision.

