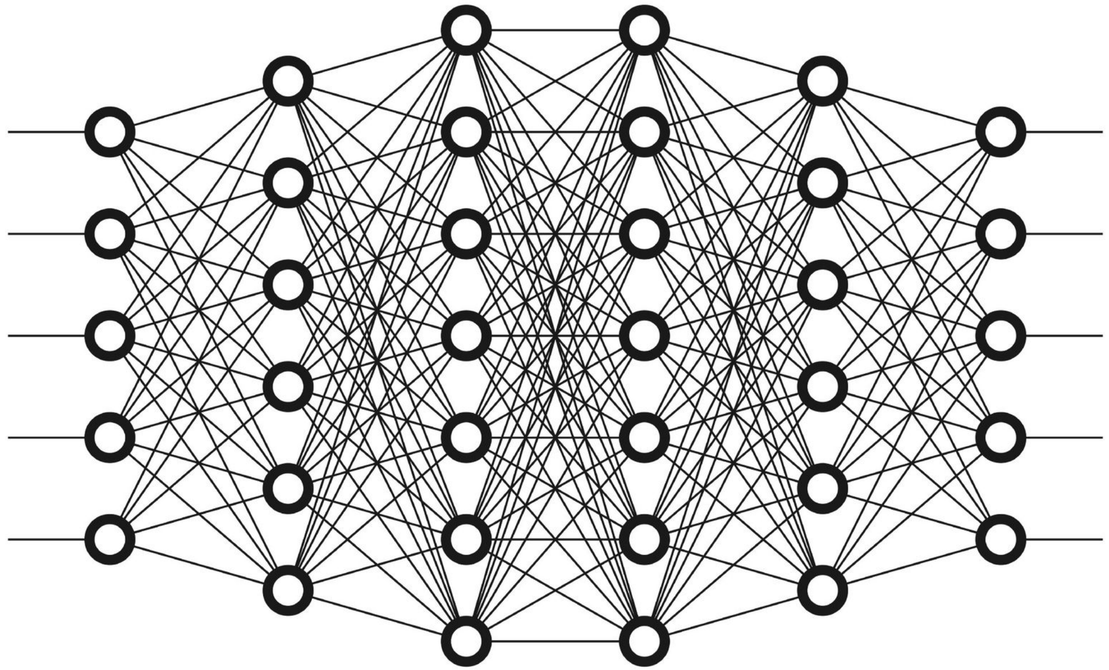
NATIONAL U

Computer Vision

CCDEPLRL: Deep Learning



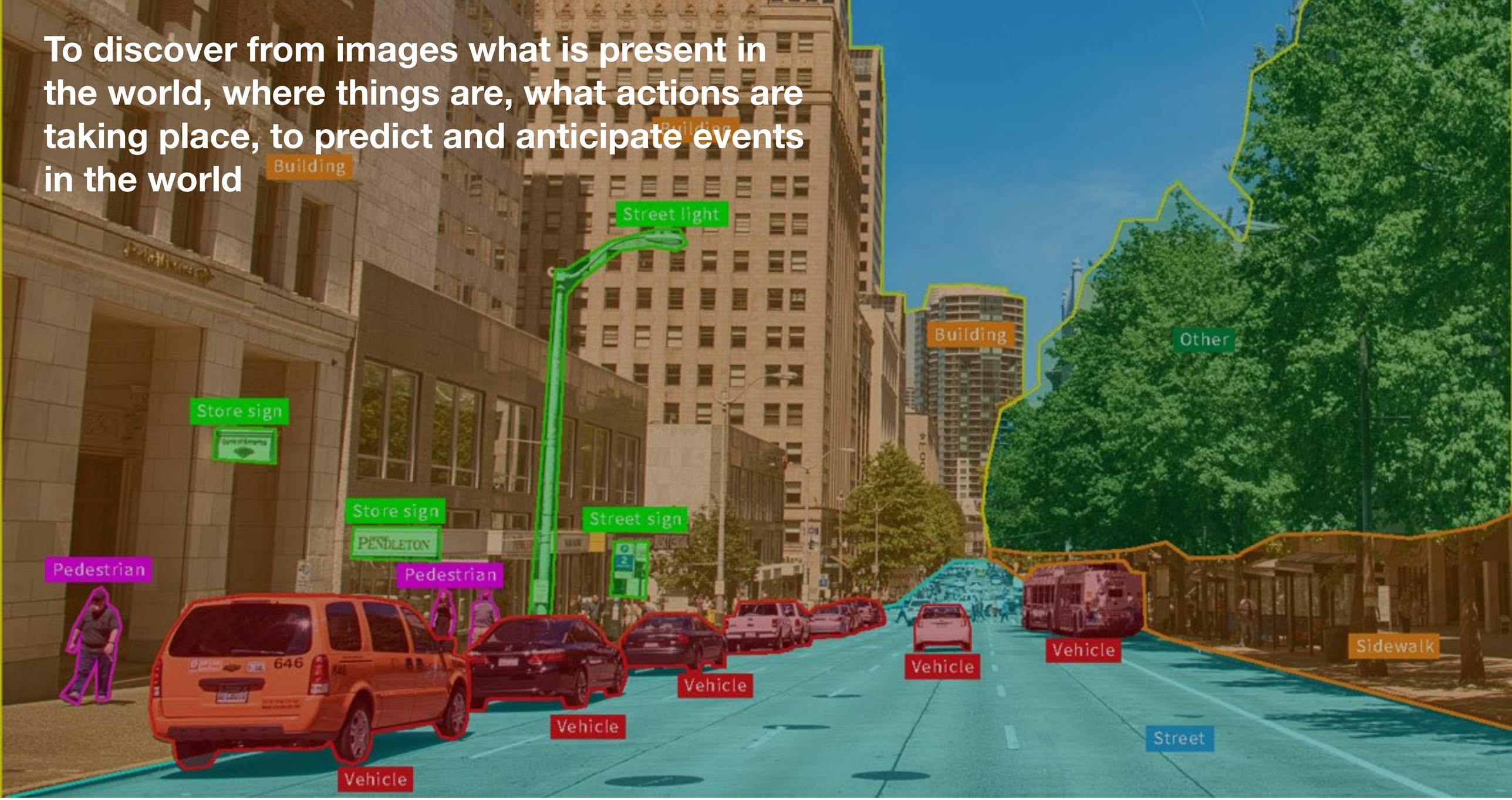
Joseph Jessie S. Oñate, MSc.

Scene Understanding

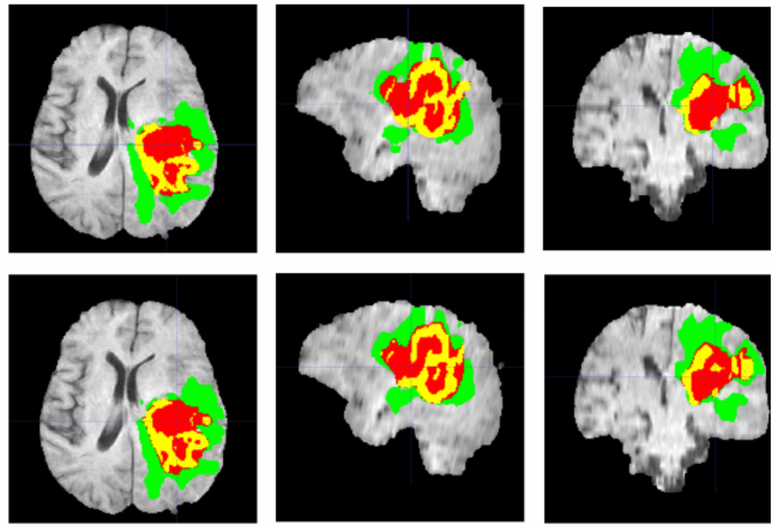
What do you see, and how?
Can we teach machines to see?

What do *you* see?

NATIONAL U


To discover from images what is present in the world, where things are, what actions are taking place, to predict and anticipate events in the world
Building
Street light
Building
Other
Store sign
Store sign
Street sign
Pedestrian
Pedestrian
Vehicle
Vehicle
Vehicle
Vehicle
Vehicle
Sidewalk
Street

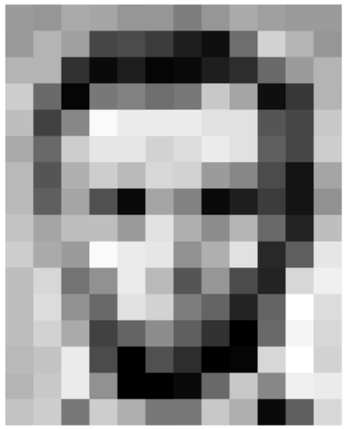
The rise and impact of Computer Vision



What computers ‘see’: Images as Numbers

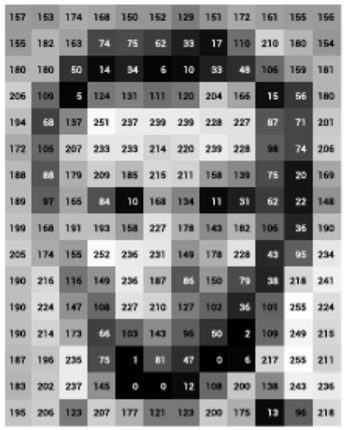
An image is just a matrix of numbers [0,255]. i.e., 1080x1080x3 for an RGB image.

What you see



Input Image

What you both see

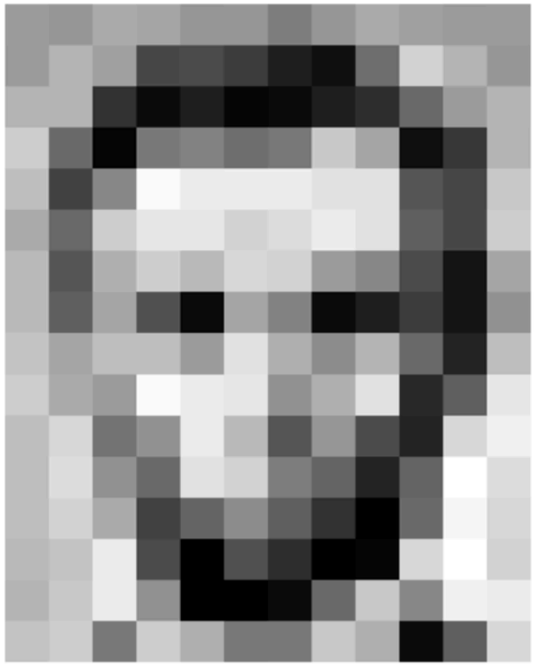


Input Image + values

What the computer "sees"

Pixel intensity values (“pix-el”=picture-element)

Computer Vision Task



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 intensity values

Lincoln	0.8
Washington	0.1
Jefferson	0.05
Obama	0.05

High Level Feature Detection

Let's identify key features in each image category

Nose, Eyes,
Mouth

Wheels,
License Plate,
Headlights

Door,
Windows,
Steps

Manual Feature Extraction

Domain knowledge

Define features

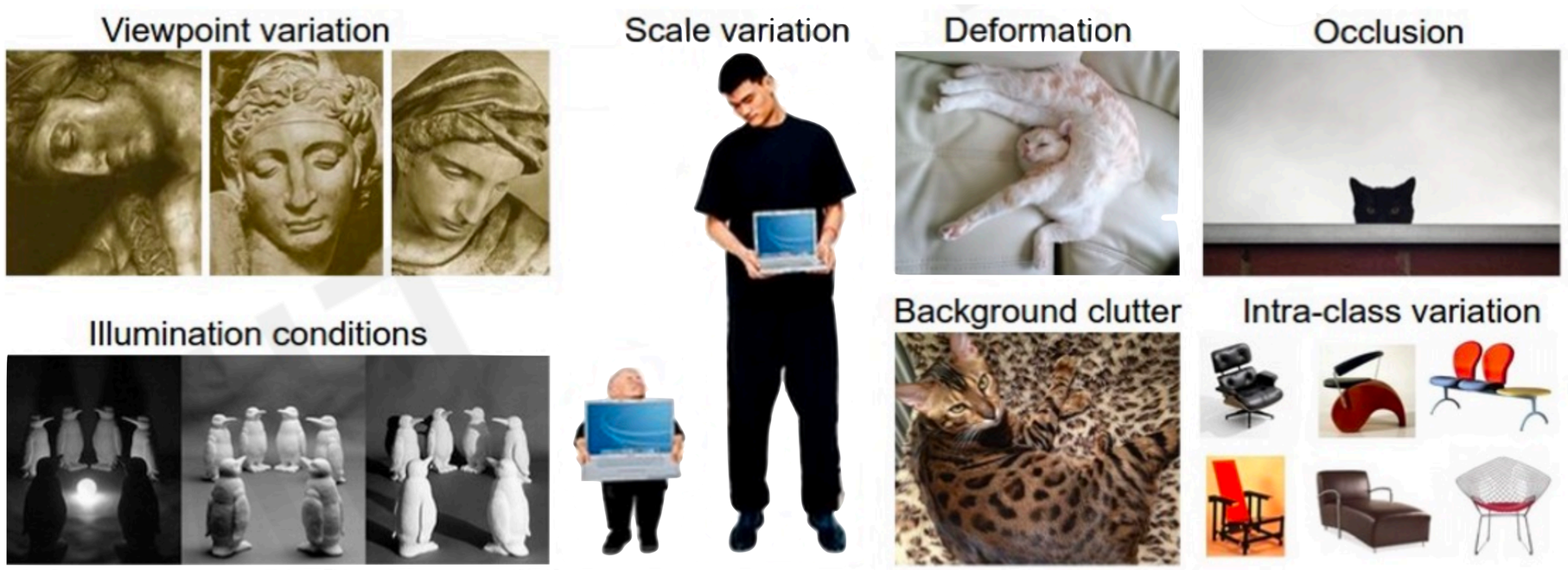
Detect features to classify

Problems?

Manual Feature Extraction

Problems?

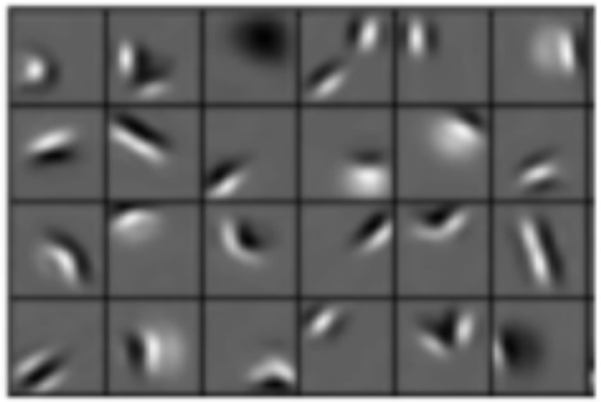
Domain knowledge | **Define features** | **Detect features to classify**



Learning Feature Representations

Can we learn a hierarchy of features directly from the data instead of hand engineering?

Low level features



Edges, dark spots

Mid level features

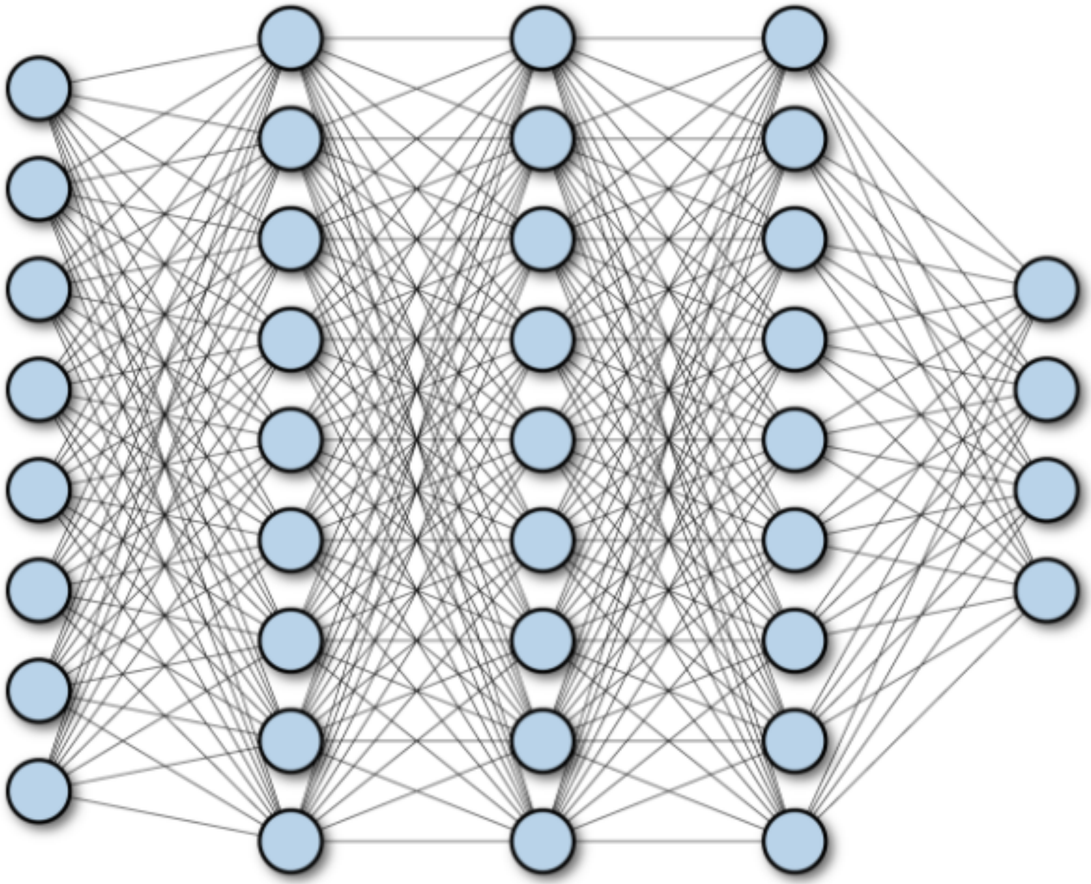
Eyes, ears, nose

High level features

Facial structure

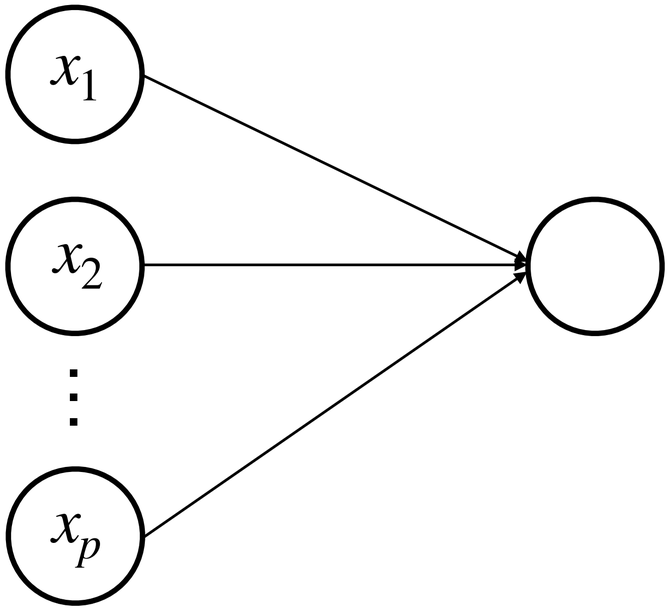
Learning
Visual Features

Fully Connected Layer



Fully Connected Layer

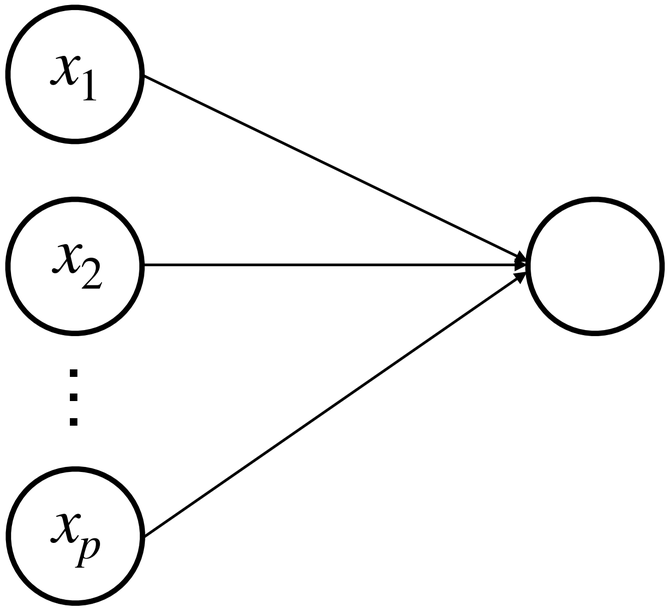
Input:
- 2D image
- Vector of pixel values



Fully Connected:
- Connect neuron in hidden layer to all neurons in input layer
- No spatial information!
- And many, many parameters.

Fully Connected Layer

Input:
- 2D image
- Vector of pixel values



Fully Connected:
- Connect neuron in hidden layer to all neurons in input layer
- No spatial information!
- And many, many parameters.

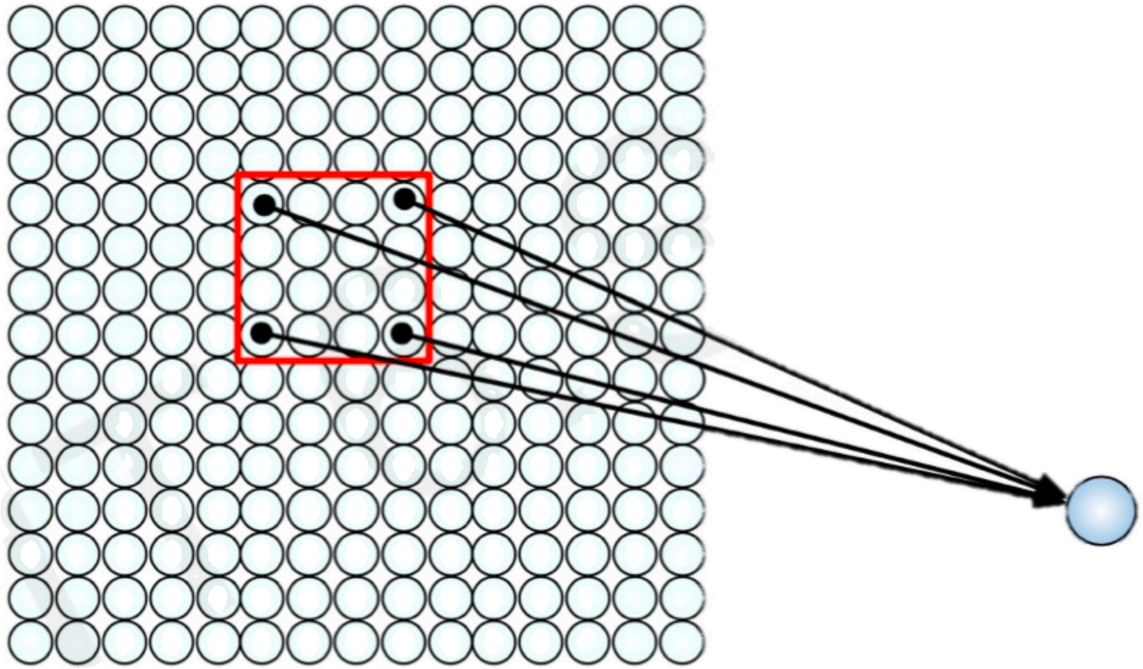
How can we use ***spatial structure*** in the input to inform the architecture of the network?

Using Spatial Structure

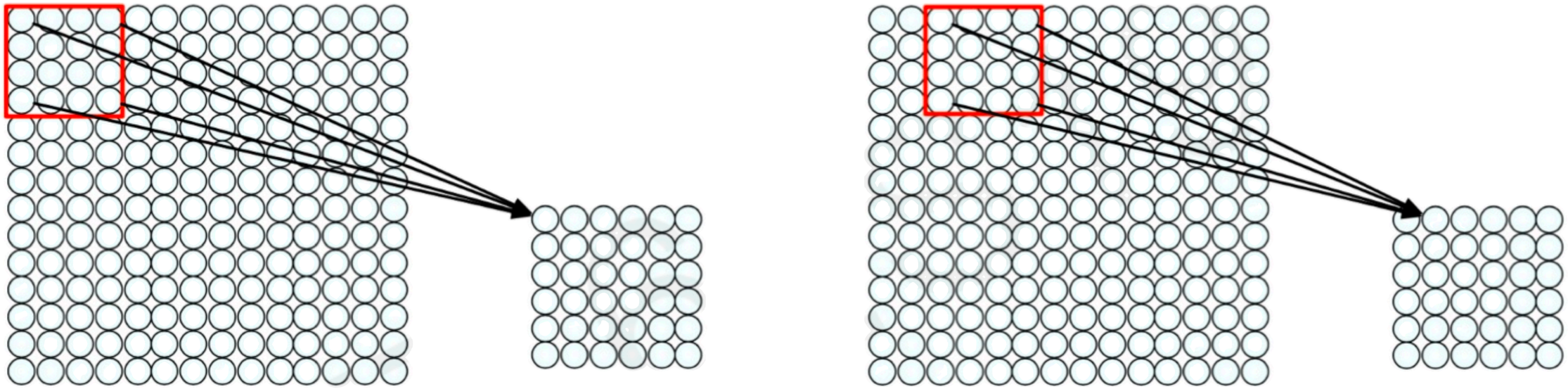
Input: 2D image. Array of pixel values

Idea: connect patches of input to neurons in hidden layer.

Neuron connected to region of input. Only “sees”these values.



Using Spatial Structure



Connect patch in input layer to a single neuron in subsequent layer.
Use a sliding window to define connections.

How can we **weight** the patch to detect particular features?

Applying Filters to Extract Features

1. Apply a set of weights — a filter - t o extract local features
2. Use multiple filters to extract different features
3. Spatially share parameters of each filter (features that matter in one part of the input should matter elsewhere)

X or X?

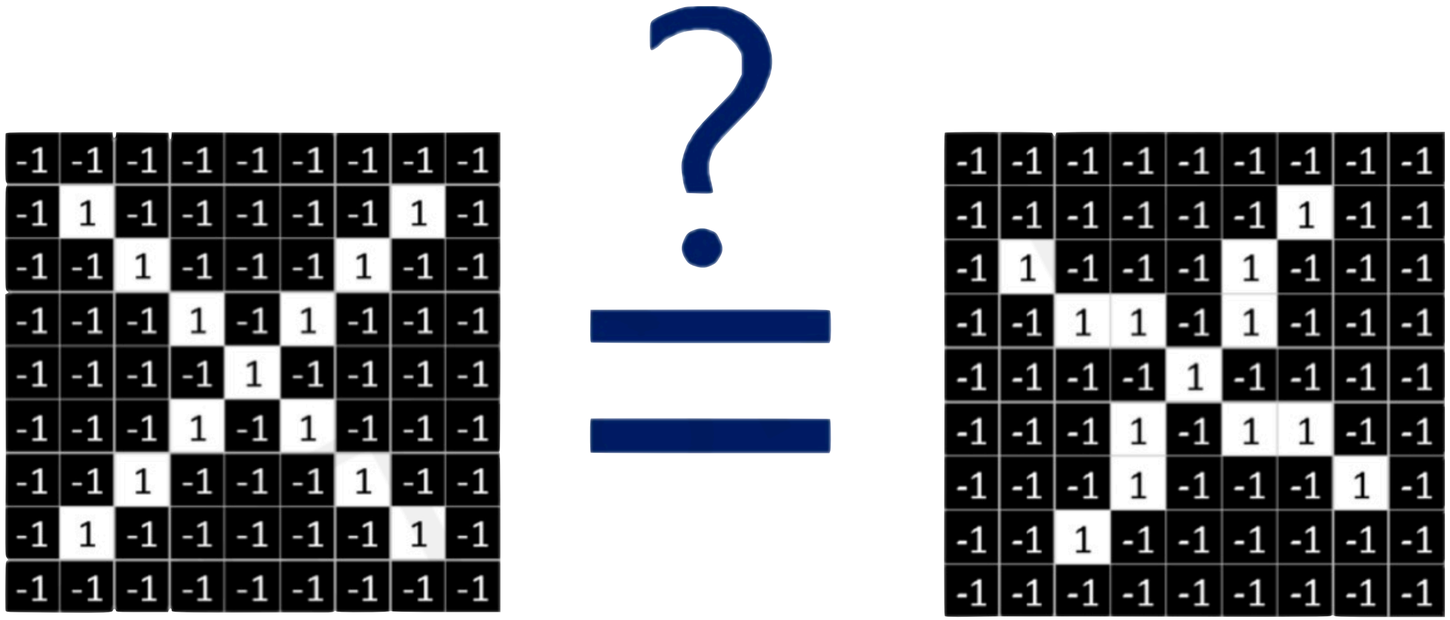
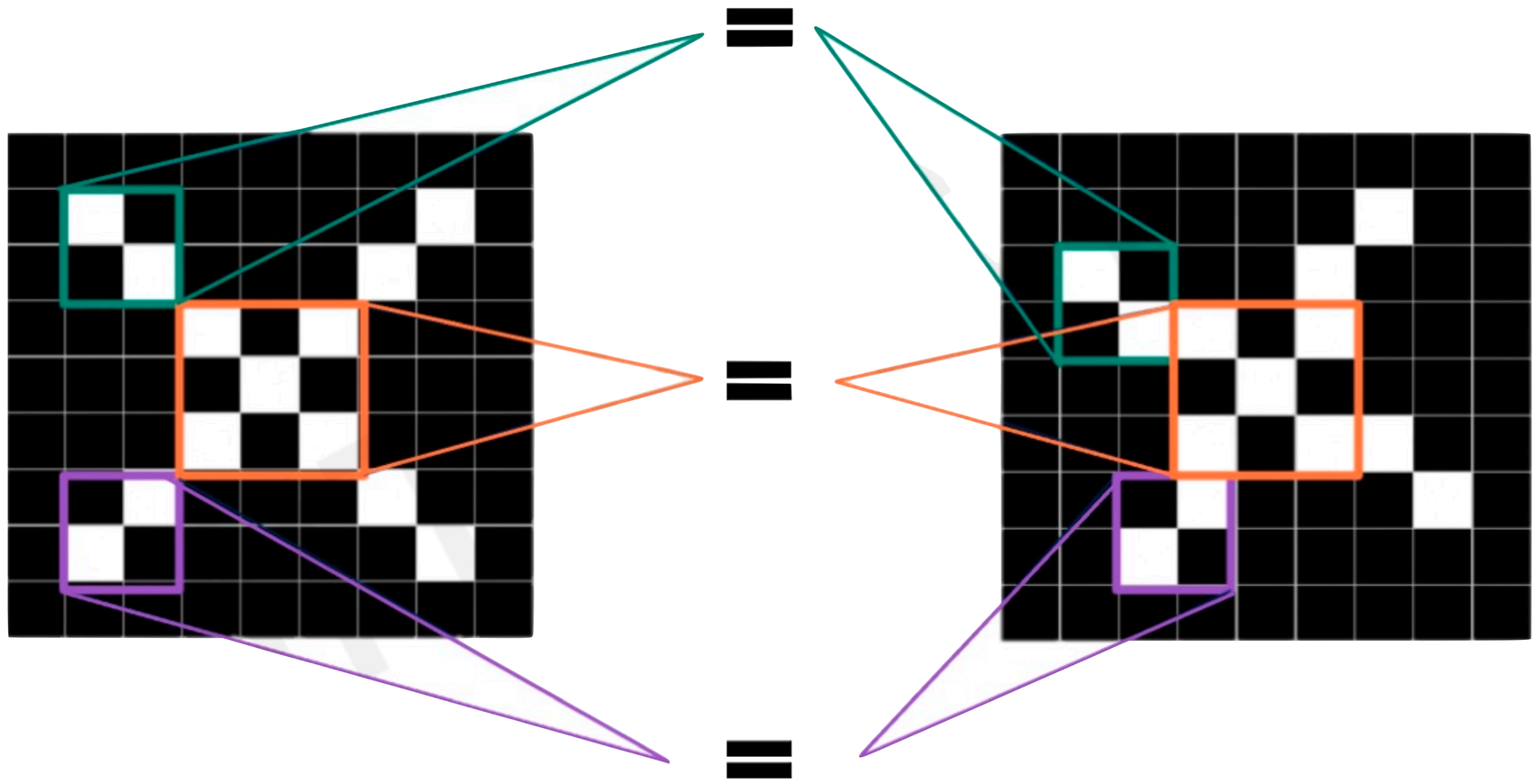


Image is represented as matrix of pixel values... and computers are literal!

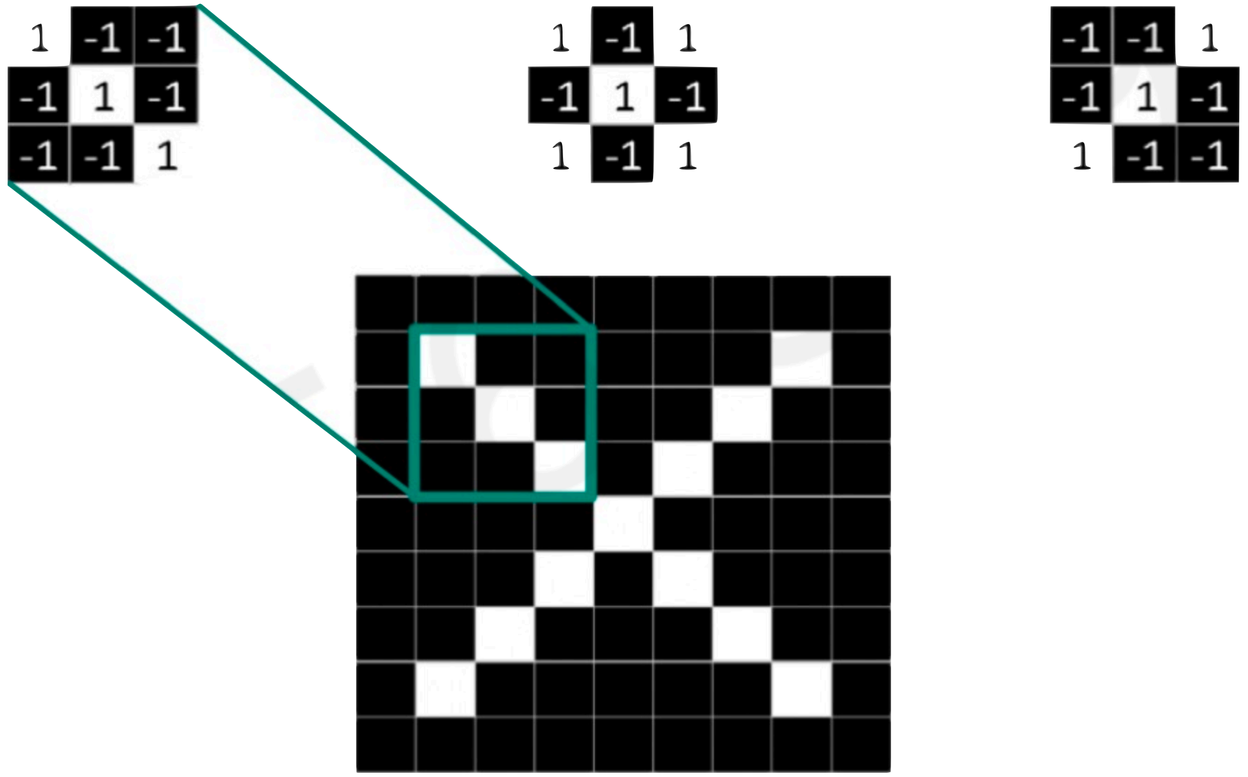
We want to b e able to classify an X a s an X even if it's shifted, shrunk, rotated, deformed.

Features of X

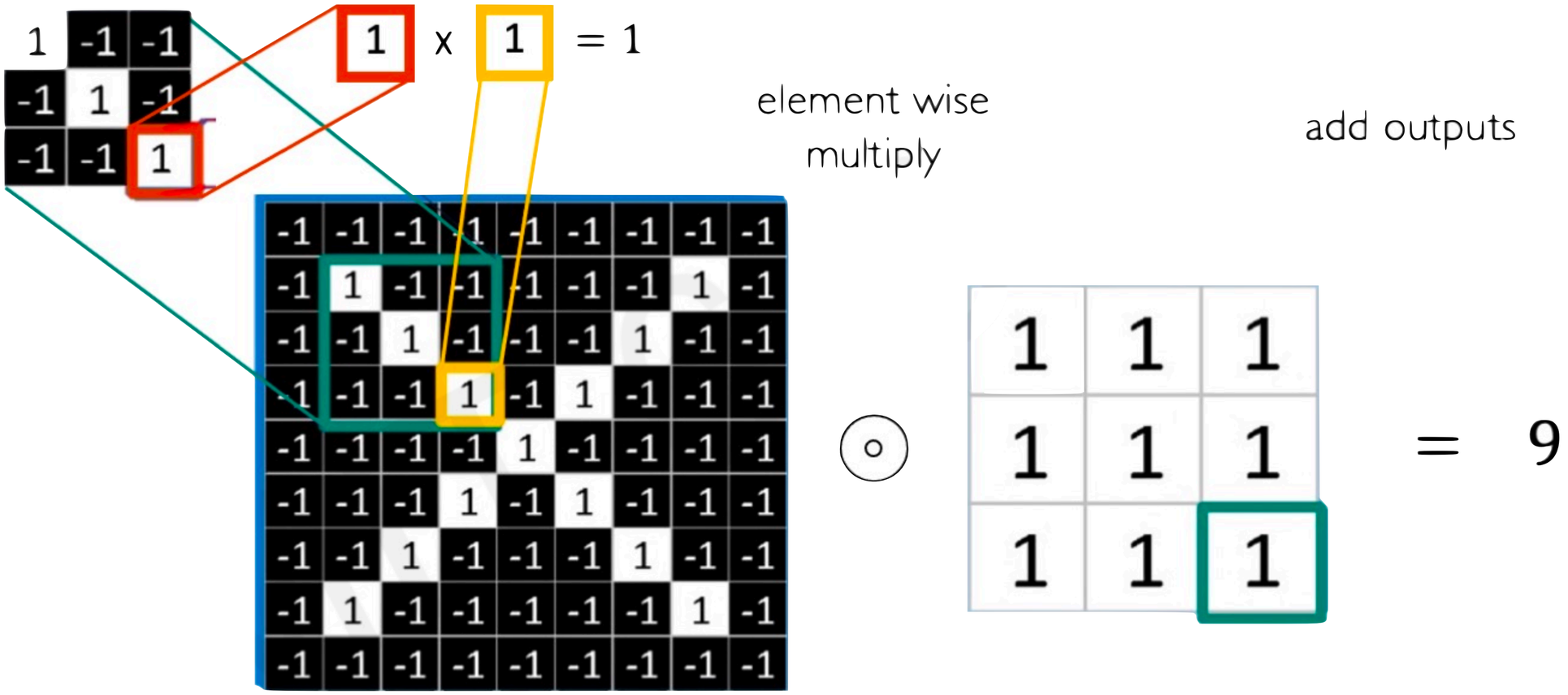


Filters to Detect X Features

filters

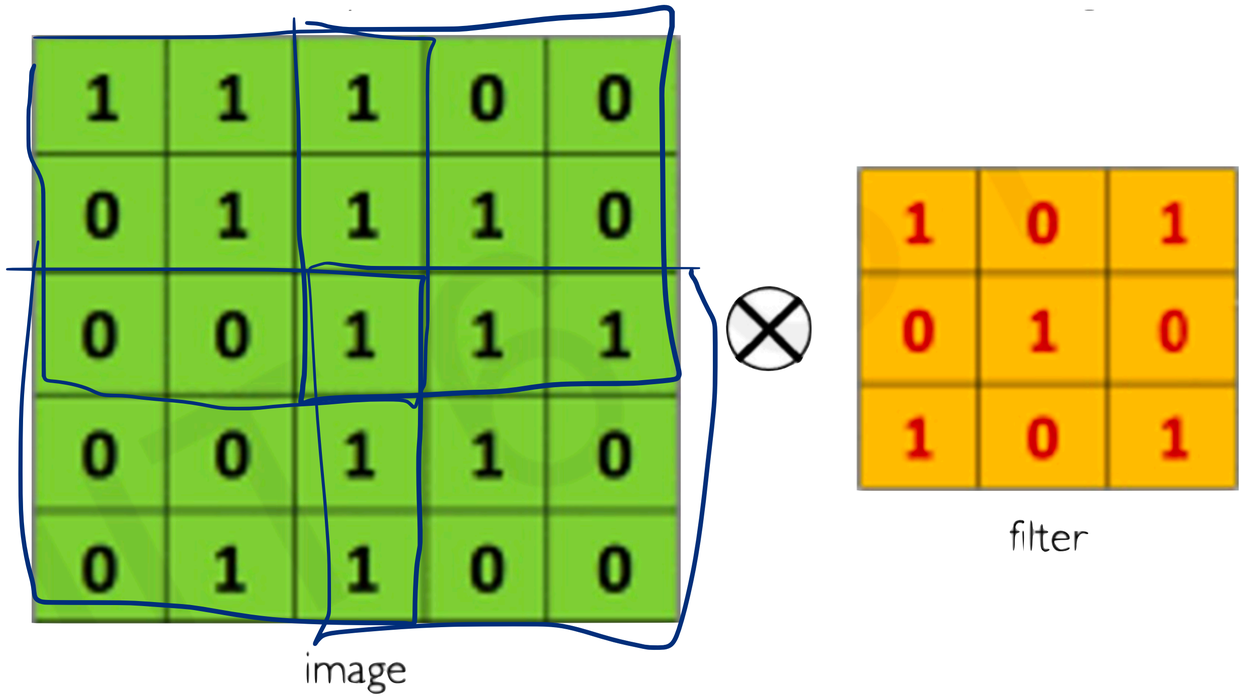


The Convolution Operation



The Convolution Operation

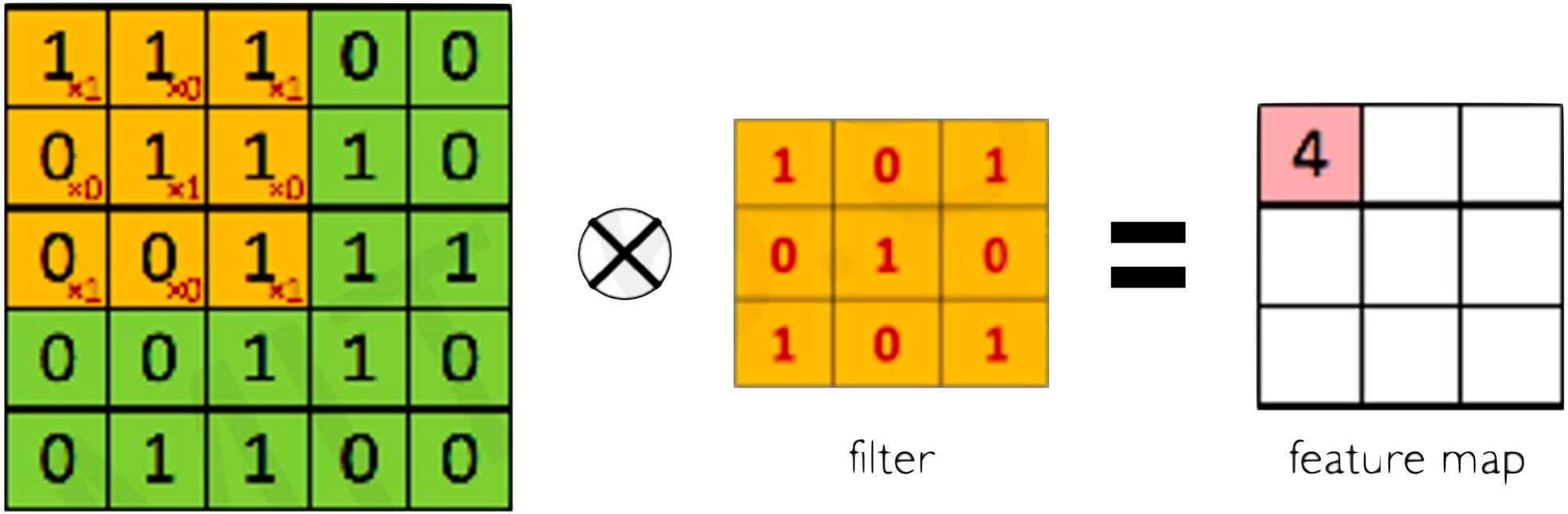
Suppose we want to compute the convolution of a 5x5 image and a 3x3 filter:



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

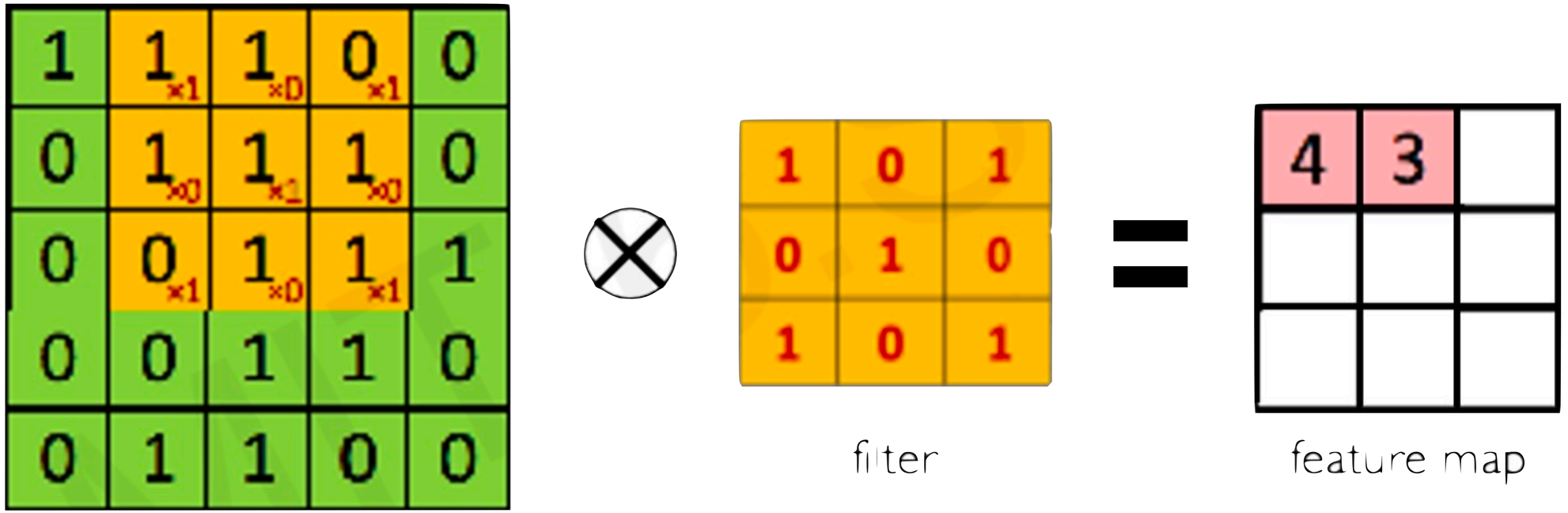
The Convolution Operation

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



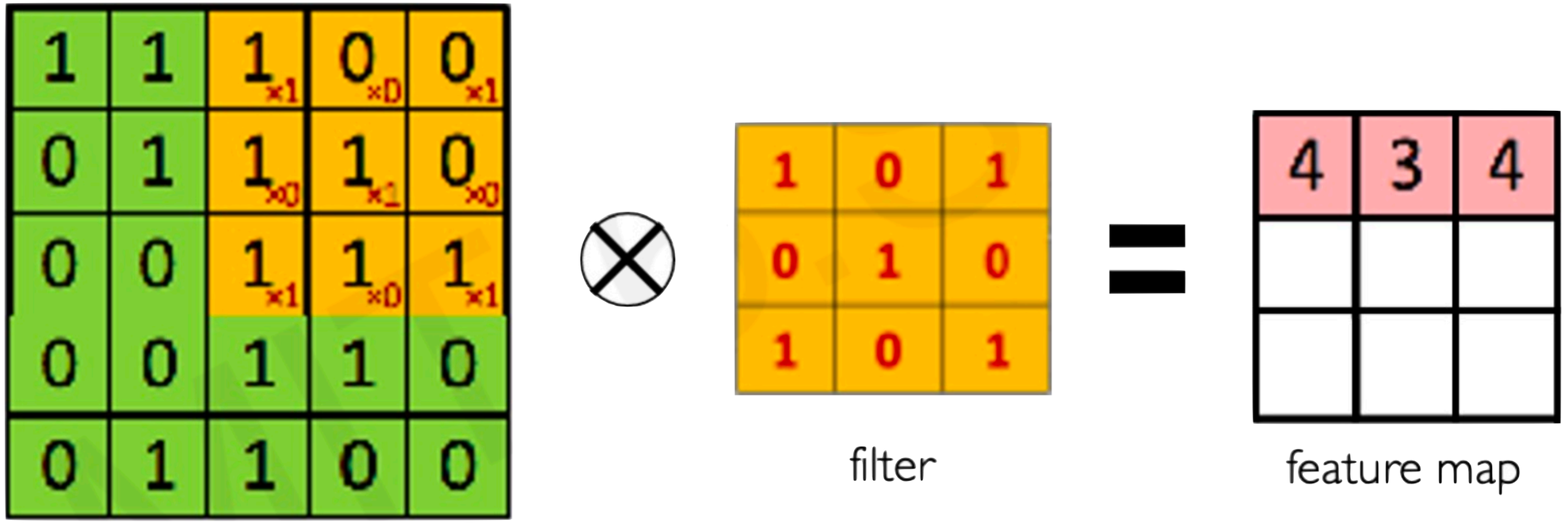
The Convolution Operation

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



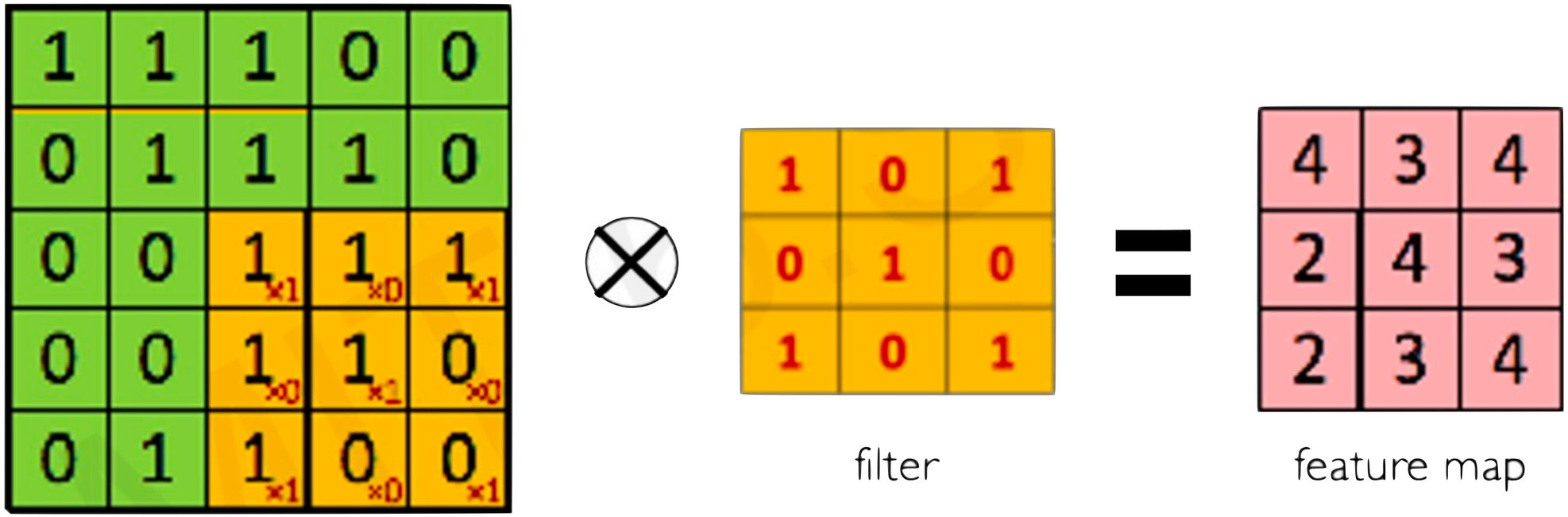
The Convolution Operation

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



The Convolution Operation

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



Producing Feature Maps

Original

Sharpen

Edge Detect

“Strong” Edge Detect

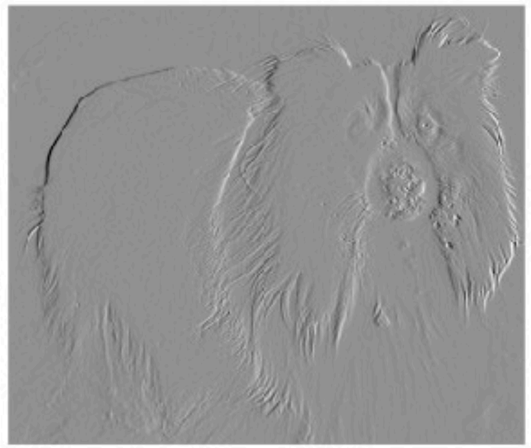
Producing Feature Maps

A simple pattern: Edges. How can we detect edges with a kernel?

Input

1	-1

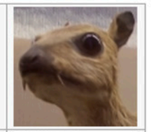
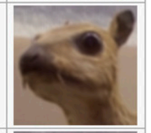
Filter



Output

Producing Feature Maps

Simple Kernels / Filters

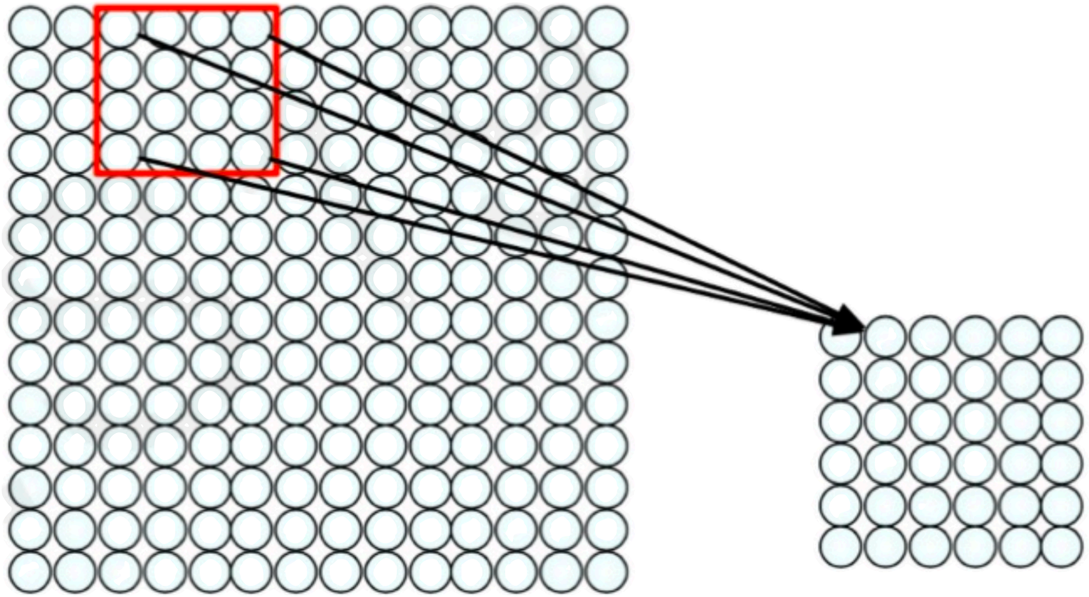
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Feature Extraction with Convolution

1. Apply a set of weights — a filter - to extract local features

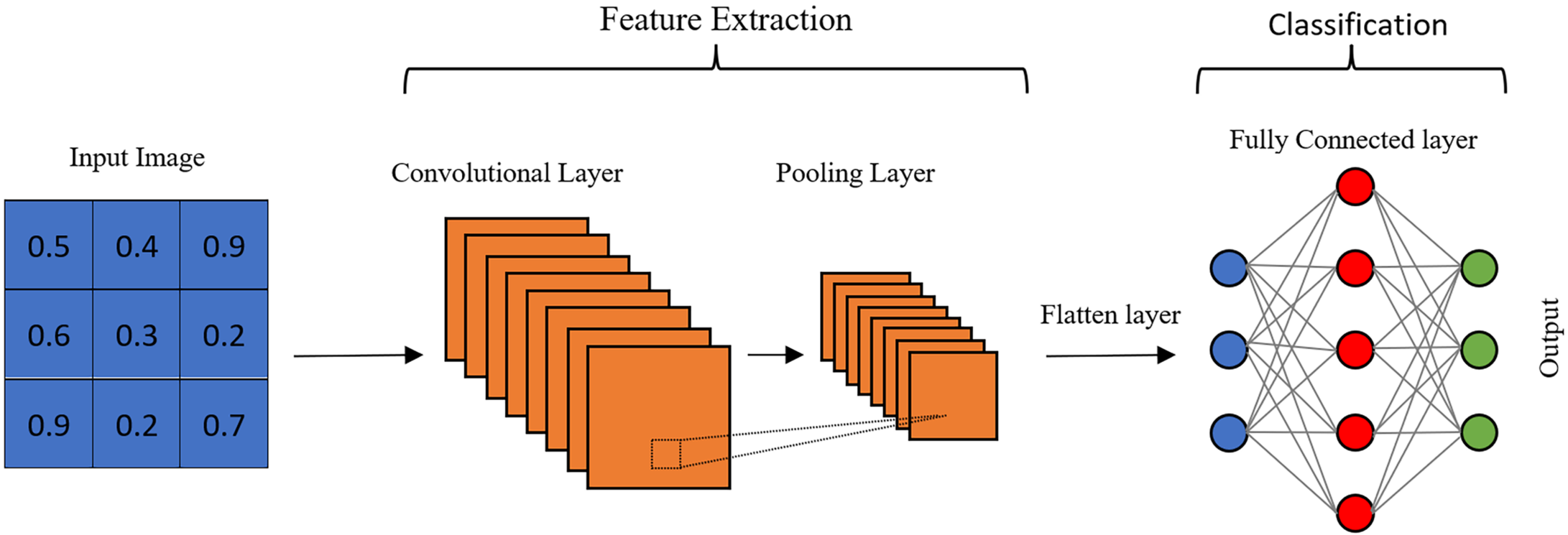
2. Use multiple filters t o extract different features

3. Spatially share parameters of each filter



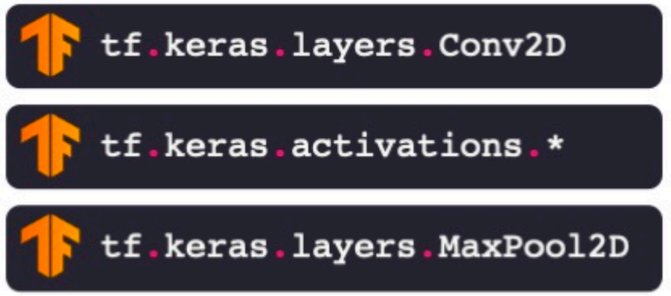
Convolutional Neural Networks

CNNs for Classification



CNNs for Classification

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

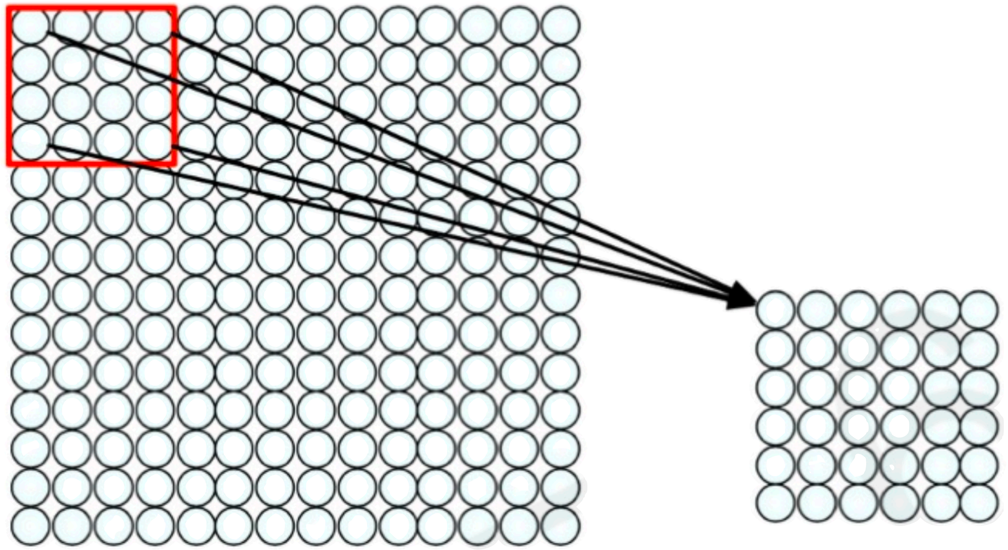


Train model with image data. Learn weights of filters in convolutional layers.

Convolutional Layers: Local Connectivity

For a neuron in hidden layer:

1. Take inputs from patch
2. Compute weighted sum
3. Apply bias

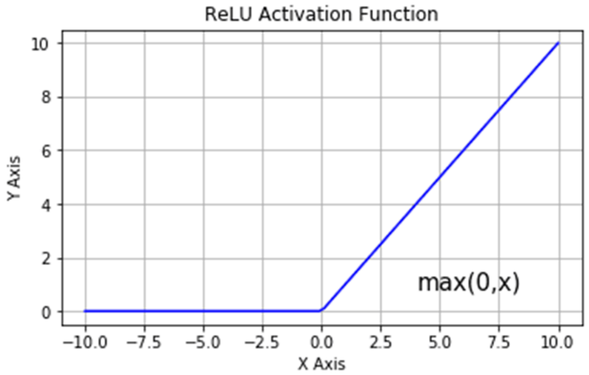


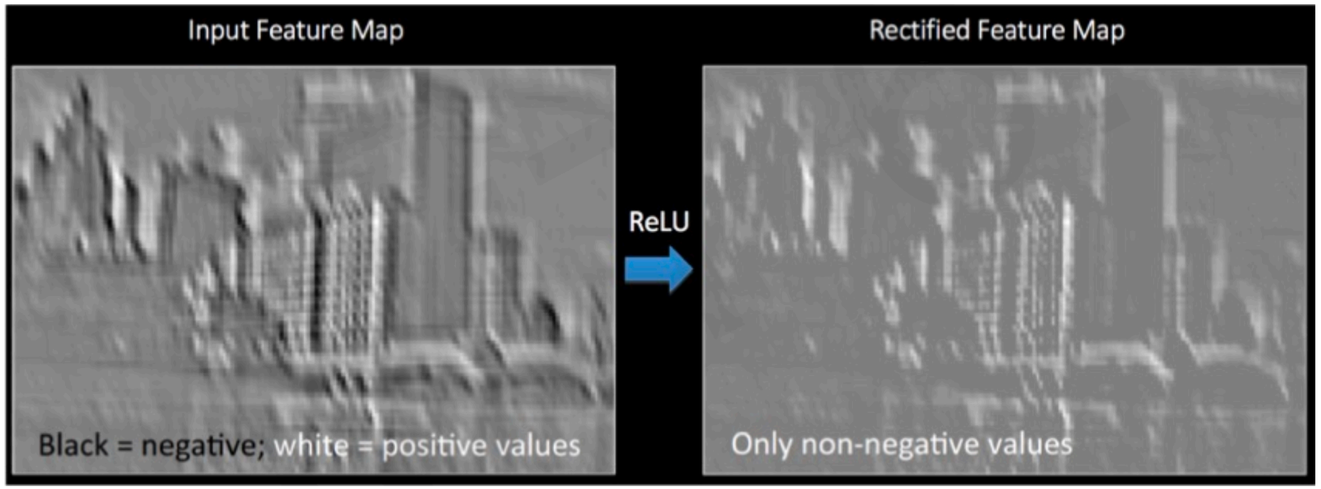
`tf.keras.layers.Conv2D`

Introducing Non-Linearity

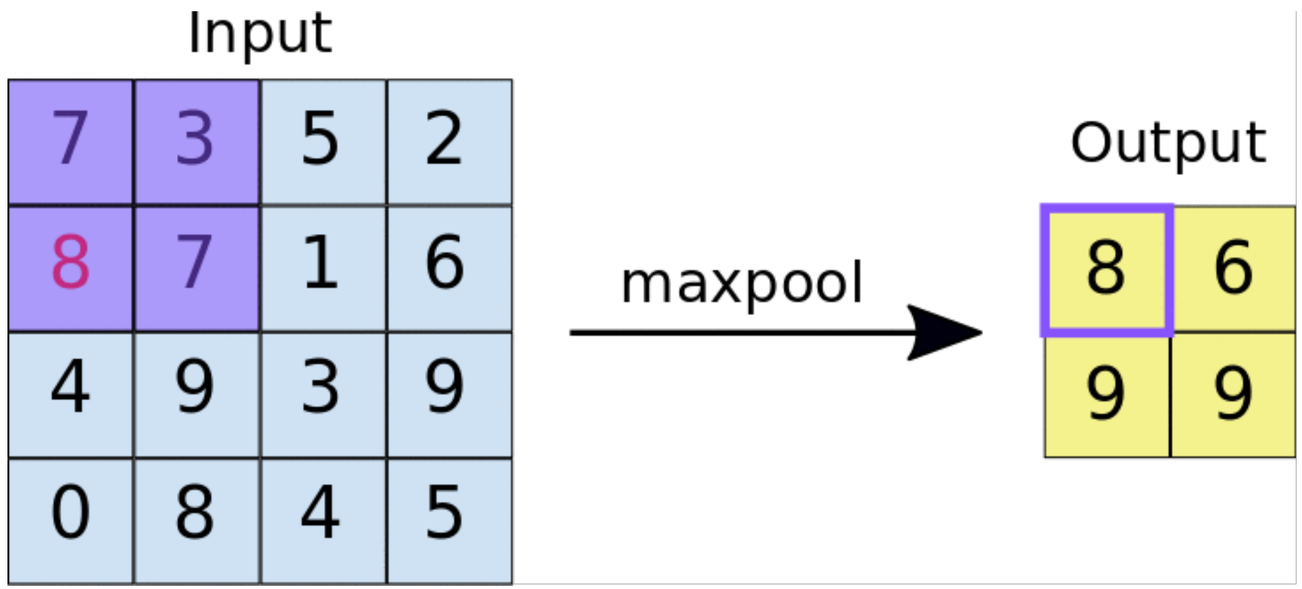
Apply after every convolution operation (i.e., after convolutional layers)

ReLU: pixel-by-pixel operation that replaces all negative values by zero. Non-linear operation!





Pooling



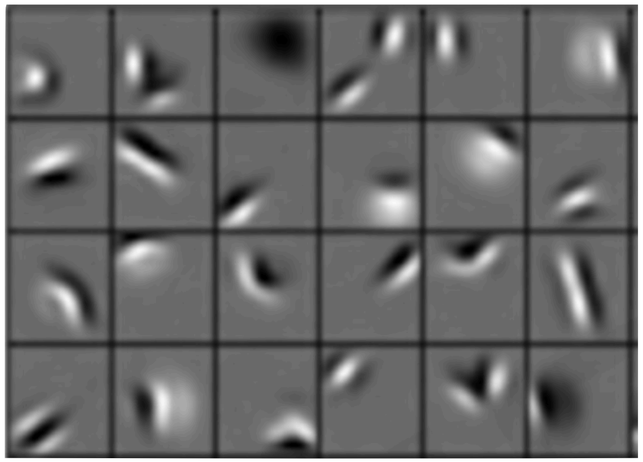
1. Reduced dimensionality
2. Spatial invariance

```
tf.keras.layers.MaxPool2D(
    pool_size=(2,2),
    strides=2
)
```

How else can we downsample and preserve spatial invariance?

Representation Learning in Deep CNNs

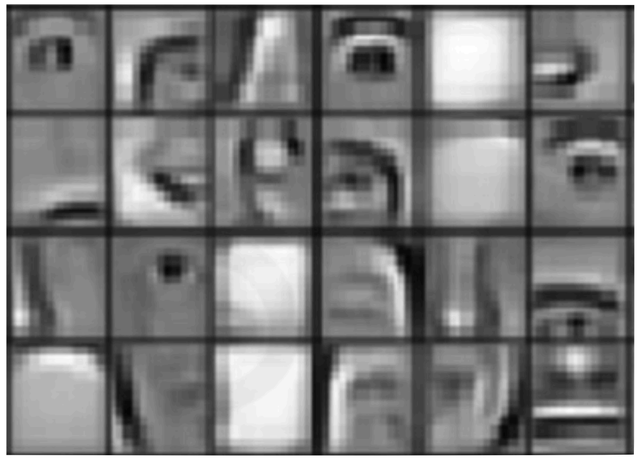
Low level features



Edges, dark spots

Conv Layer 1

Mid level features



Eyes, ears, nose

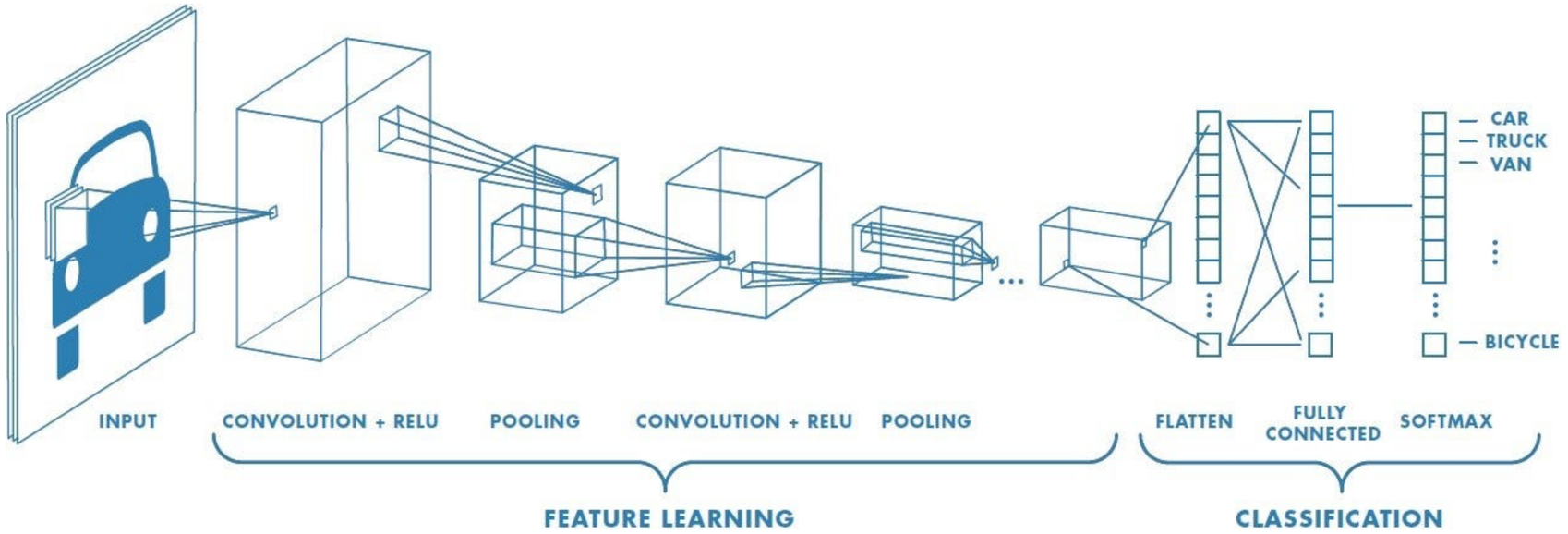
Conv Layer 2

High level features

Facial structure

Conv Layer 3

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 t o a particular class

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

Thank you!

CCDEPLRL: Deep Learning