Natural Language Processing
NLP_CLT_1c_June_8th_2025

Eng. Maytham Ghanoum
Artificial Intelligence & Deep Learning Specialist
MTN Syria – SCS – SVU CLT
+963947222064 - +963982018359
https://www.linkedin.com/in/maytham-ghanoum-697aa5207/
https://www.facebook.com/maytham.ghanoum

CNN

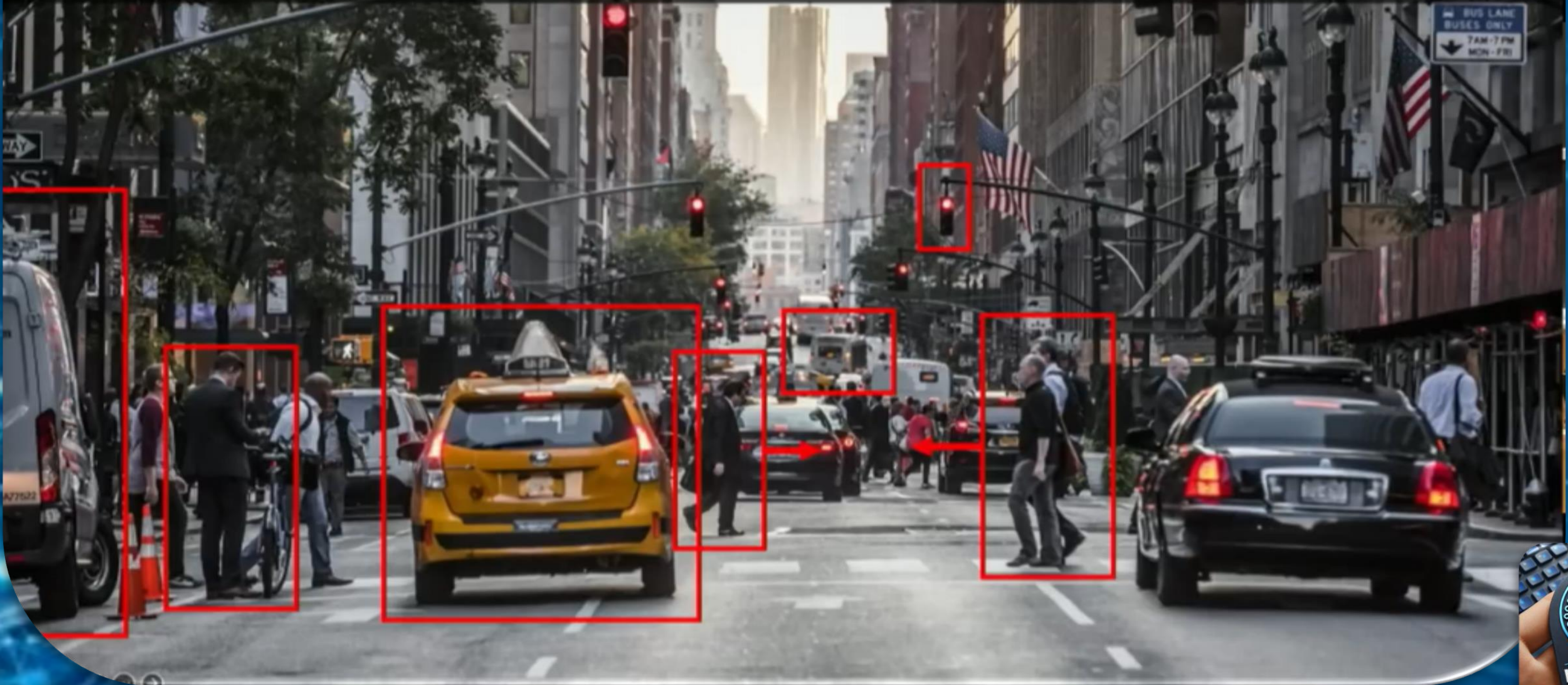
What is CNN?

Convolutional Neural Networks (CNNs) are a class of deep neural networks commonly used for analyzing visual data. They are particularly effective in image classification, object detection, and similar tasks

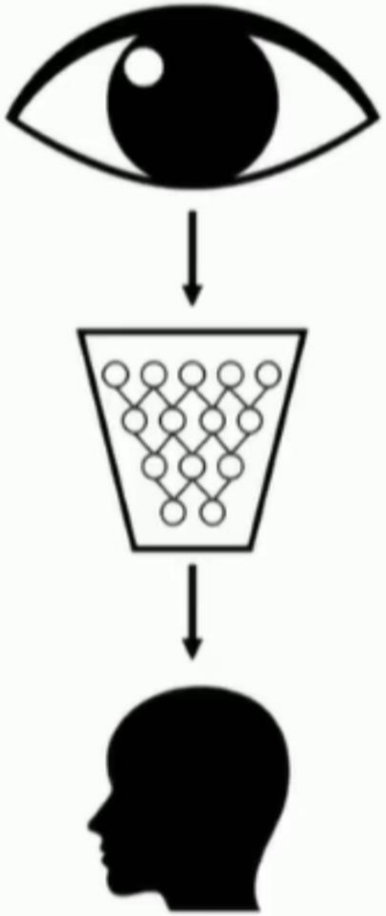
Key Concepts:

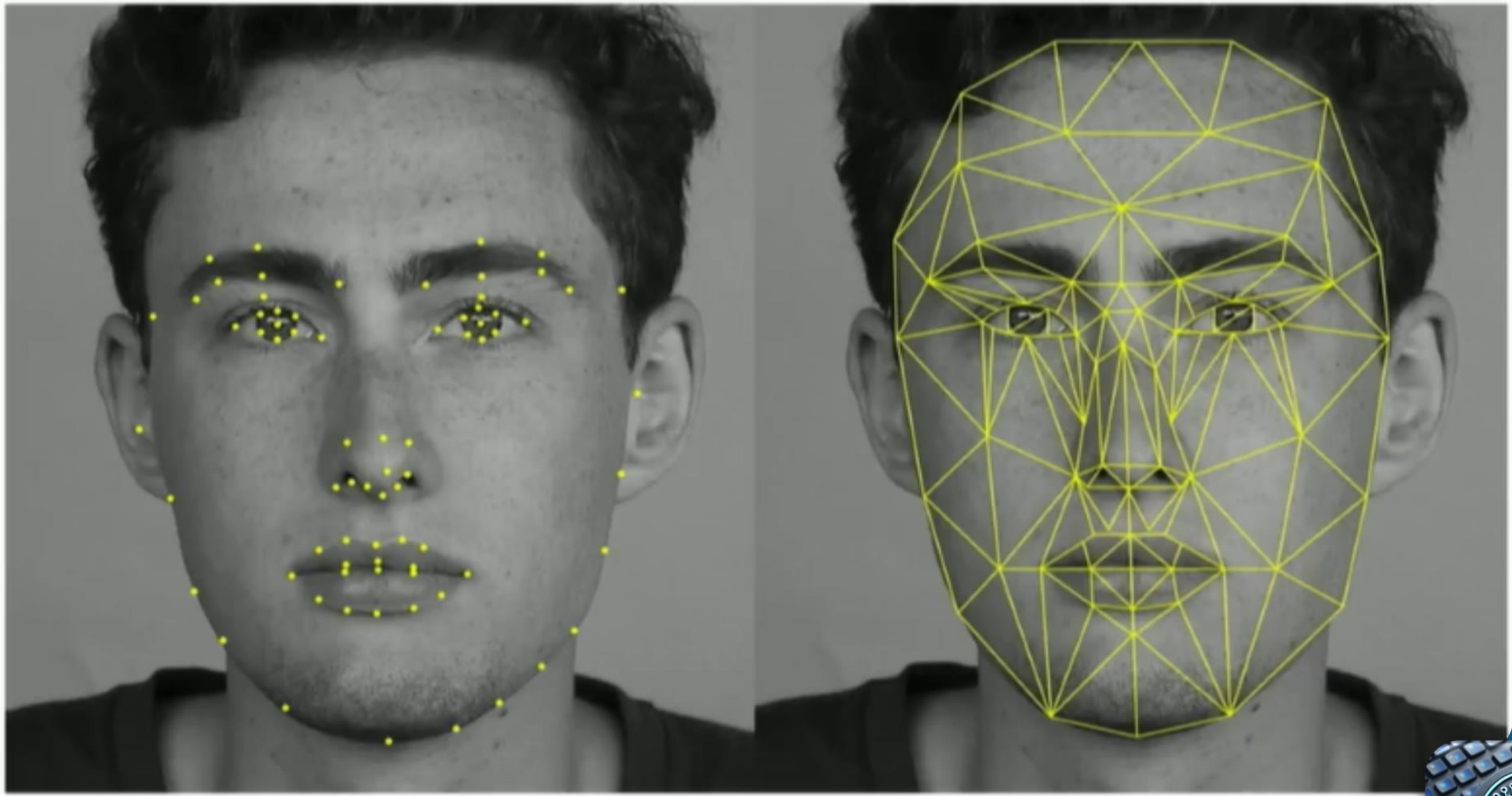
- **Convolution Layer:** The core building block, where a filter (or kernel) slides over the input image to detect features like edges, textures, etc.
- **Pooling Layer:** Reduces the spatial dimensions (width and height) of the feature maps, which helps in reducing computational complexity.
- **Fully Connected Layer:** Acts like a standard neural network layer where every node is connected to every other node in the previous layer.
- **Activation Functions:** Functions like ReLU (Rectified Linear Unit) are used to introduce non-linearity in the model.
- **Dropout:** A regularization technique to prevent overfitting by randomly setting some layer outputs to zero during training.

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

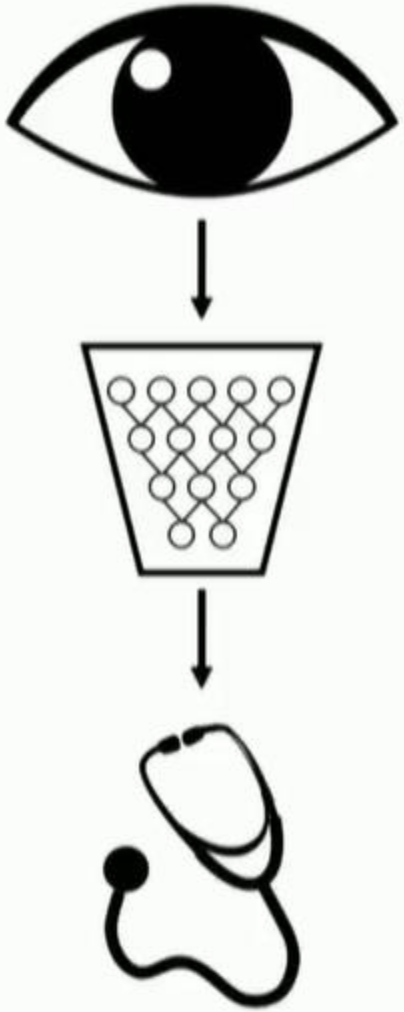


Impact: Facial Detection & Recognition





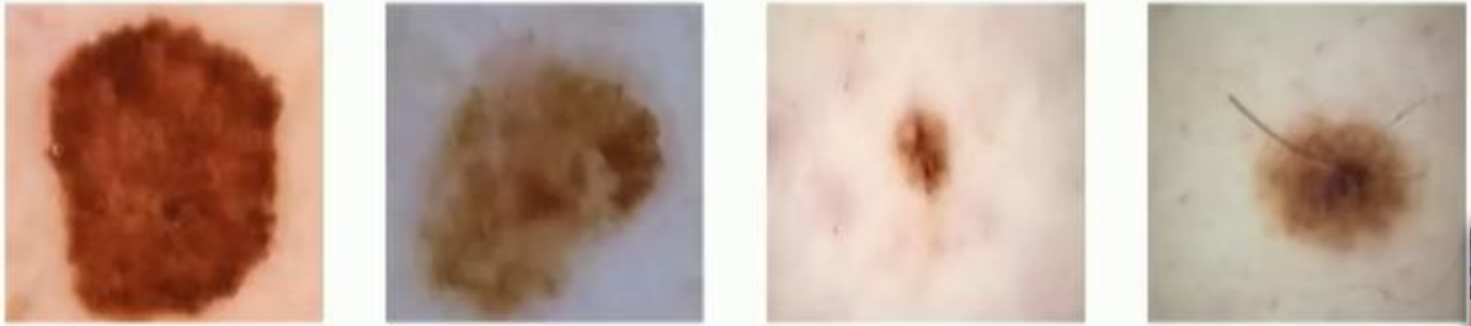
Impact: Medicine, Biology, Healthcare



Breast cancer

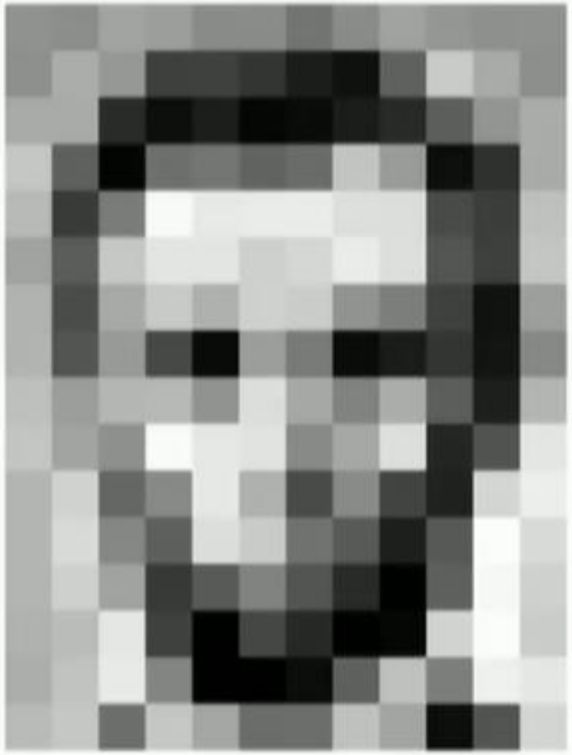
COVID-19

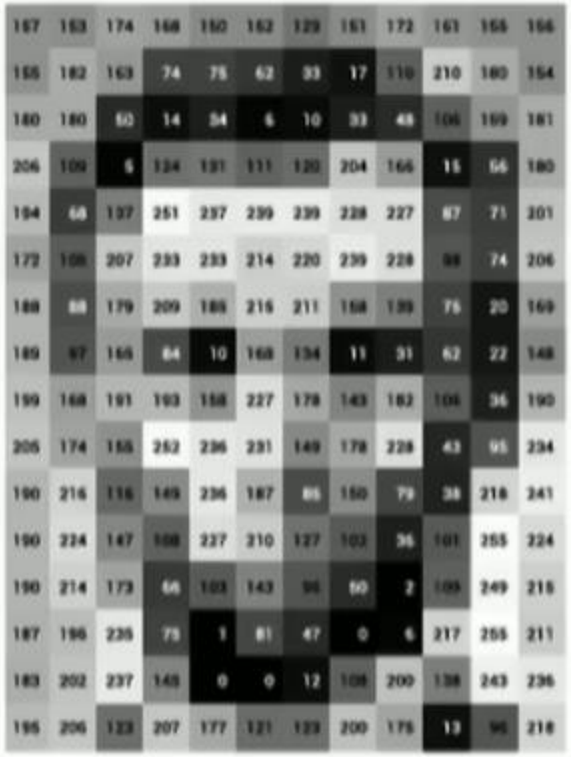
Skin cancer



What Computers “See”

Images are Numbers



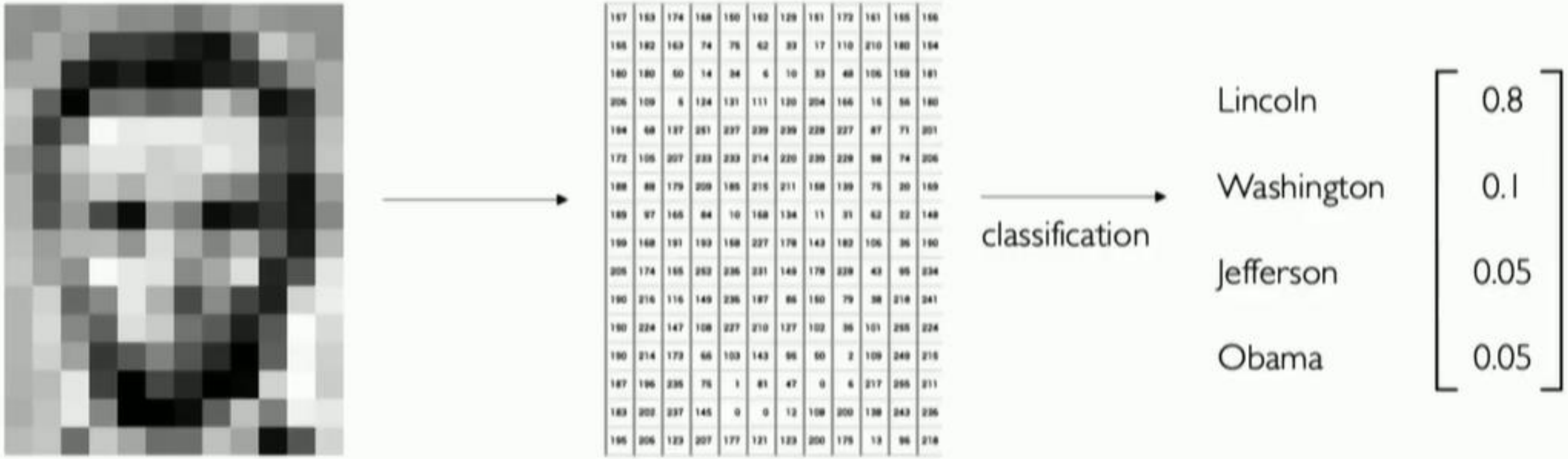


What the computer sees

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	95	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

Tasks in Computer Vision



Input Image

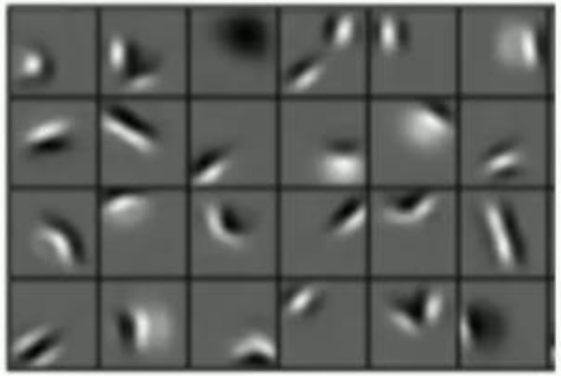
Pixel Representation

- **Regression**: output variable takes continuous value
- **Classification**: output variable takes class label. Can produce probability of belonging to a particular class

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

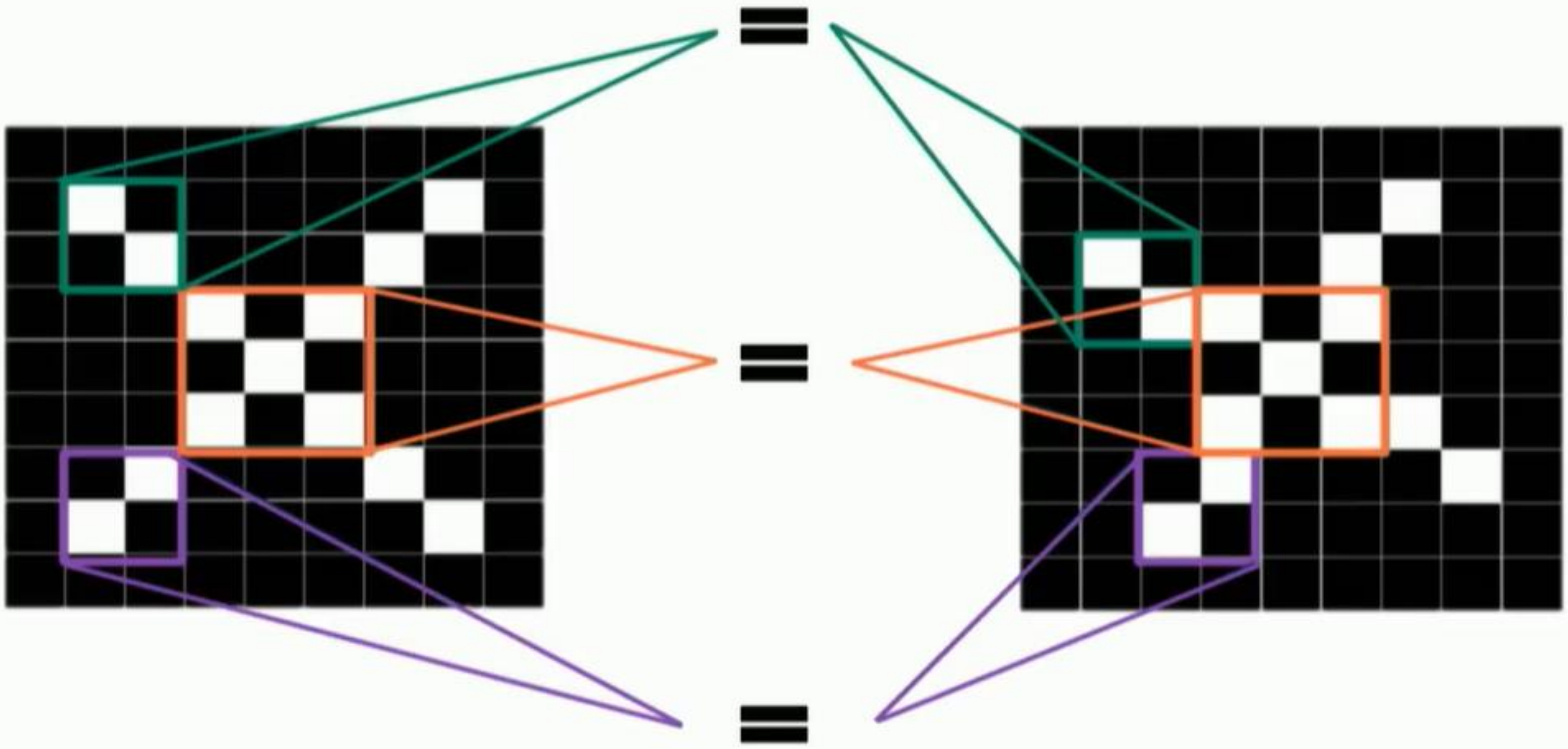
High level features

Facial structure

Feature Extraction and Convolution

A Case Study

Features of X



The Convolution Operation

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

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

image

$\otimes$

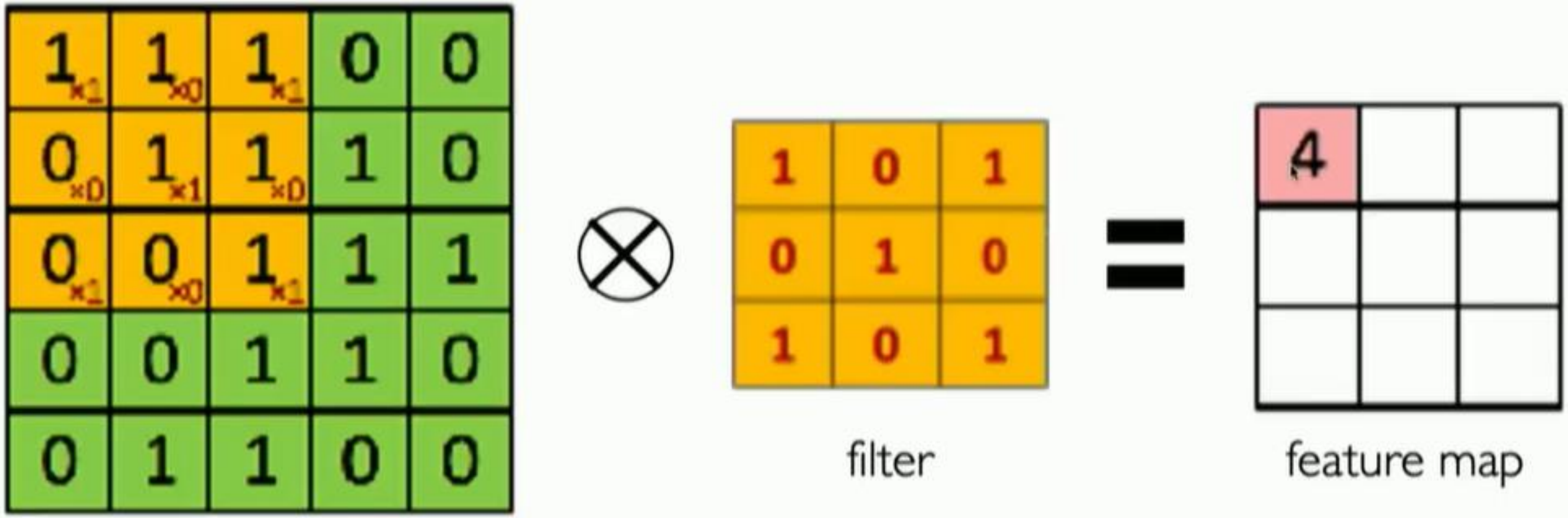
1	0	1
0	1	0
1	0	1

filter

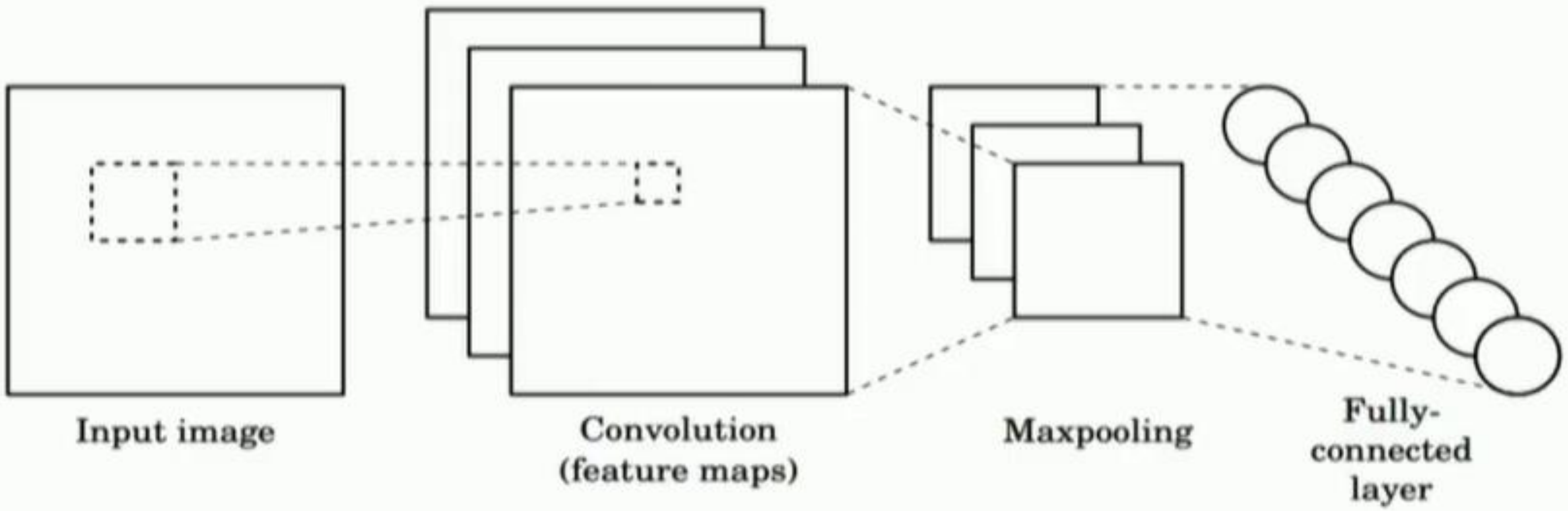
We slide the 3x3 filter over the input image, element-wise multiply, and add the outpu

The Convolution Operation

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



CNNs for Classification



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

```
tf.keras.layers.Conv2D
```

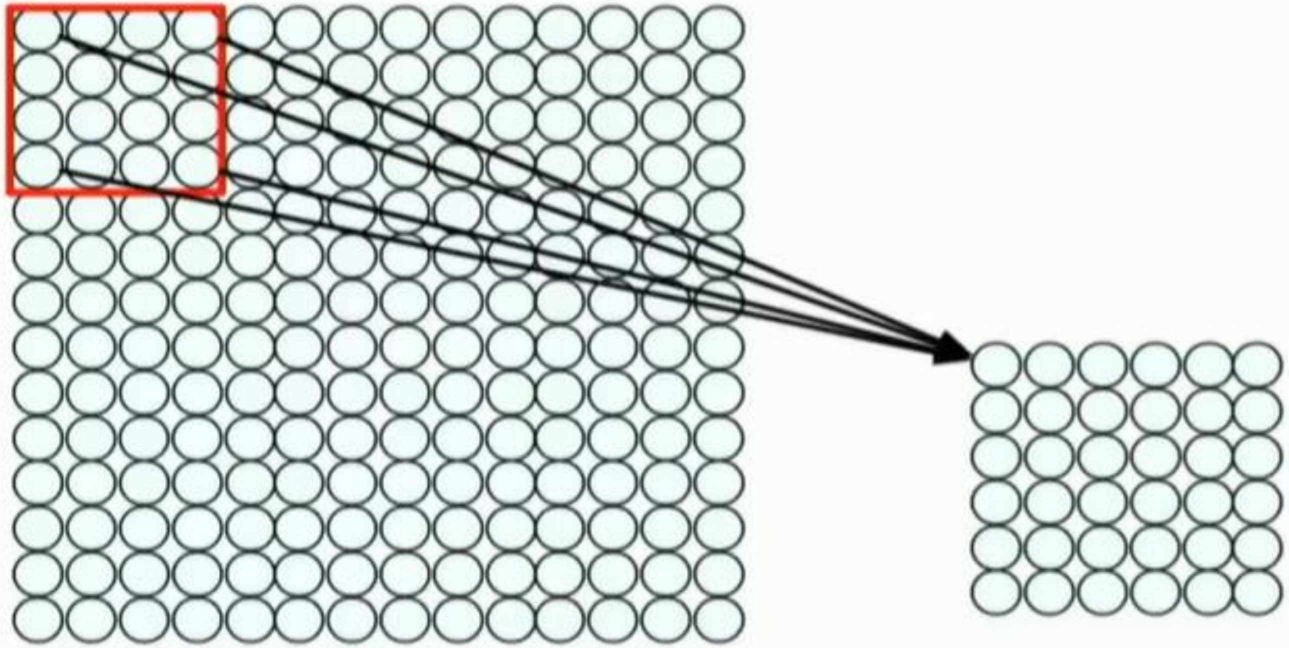
```
tf.keras.activations.*
```

```
tf.keras.layers.MaxPool2D
```

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

Convolutional Layers: Local Connectivity

`tf.keras.layers.Conv2D`



4x4 filter: matrix of weights w_{ij}

$$\sum_{i=1}^{4}\sum_{j=1}^{4} w_{ij}\, x_{i+p,j+q} + b$$

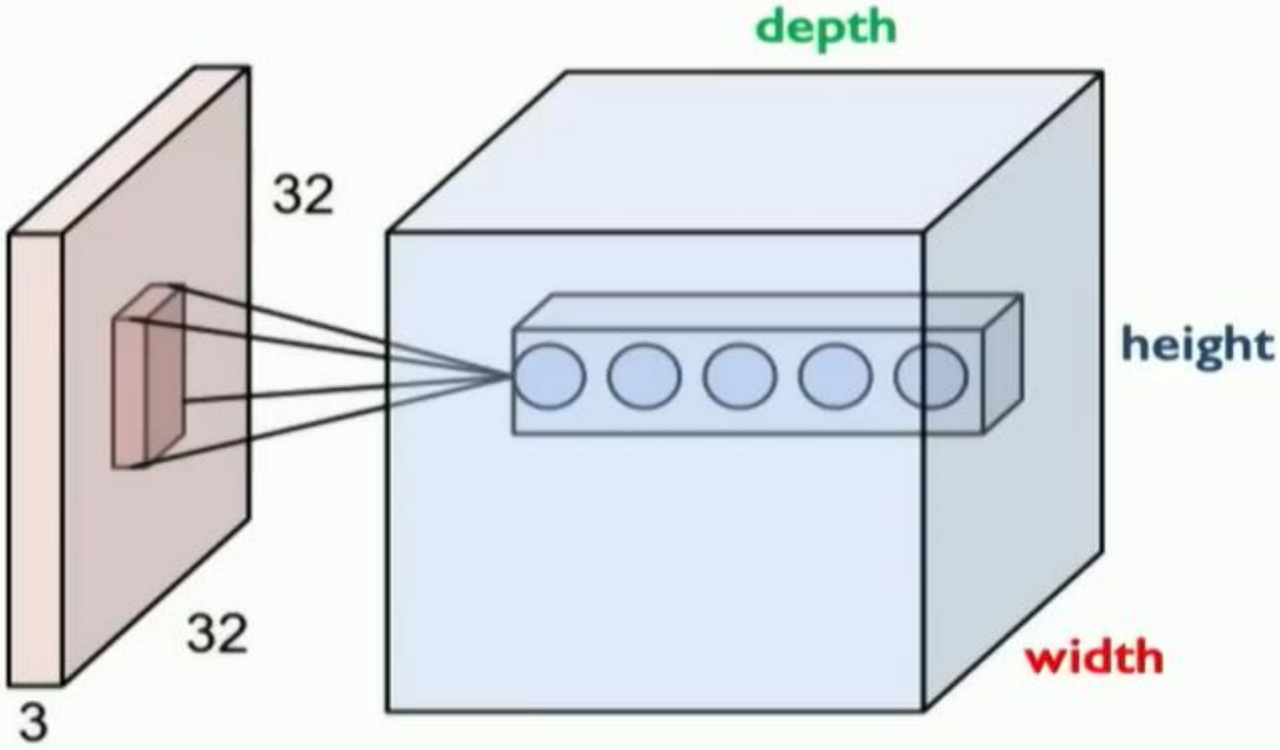
for neuron (p,q) in hidden layer

For a neuron in hidden layer:

- Take inputs from patch
- Compute weighted sum
- Apply bias

1) applying a window of weights
2) computing linear combinations
3) activating with non-linear function

CNNs: Spatial Arrangement of Output Volume



Layer Dimensions:

$h \times w \times d$

where h and w are spatial dimensions
d (depth) = number of filters

Stride:

Filter step size

Receptive Field:

Locations in input image that
a node is path connected to

```
tf.keras.layers.Conv2D( filters=d, kernel_size=(h,w), strides=s )
```

Pooling

x

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

y

max pool with 2x2 filters
and stride 2

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

6	8
3	4

1) Reduced dimensionality
2) Spatial invariance

How else can we downsample and preserve spatial invariance?

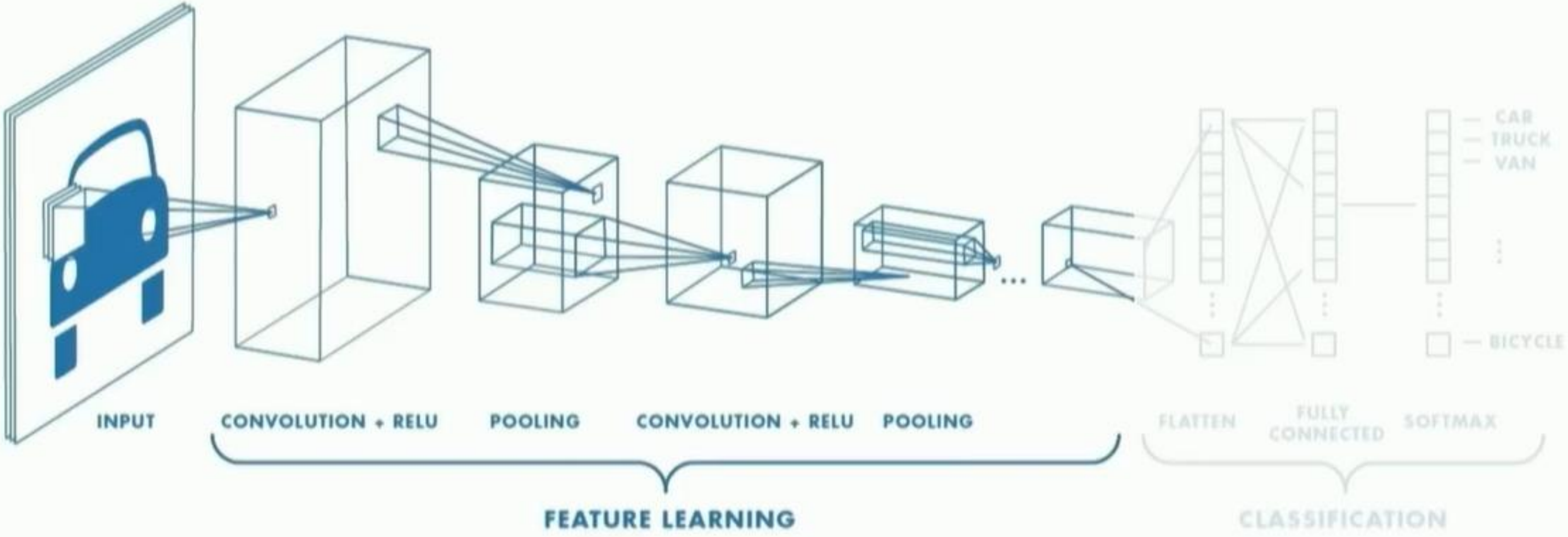
Representation Learning in Deep CNNs

Low level features | Mid level features | High level features

Edges, dark spots | Eyes, ears, nose | Facial structure

Conv Layer 1 | Conv Layer 2 | Conv Layer 3

CNNs for Classification: Feature Learning



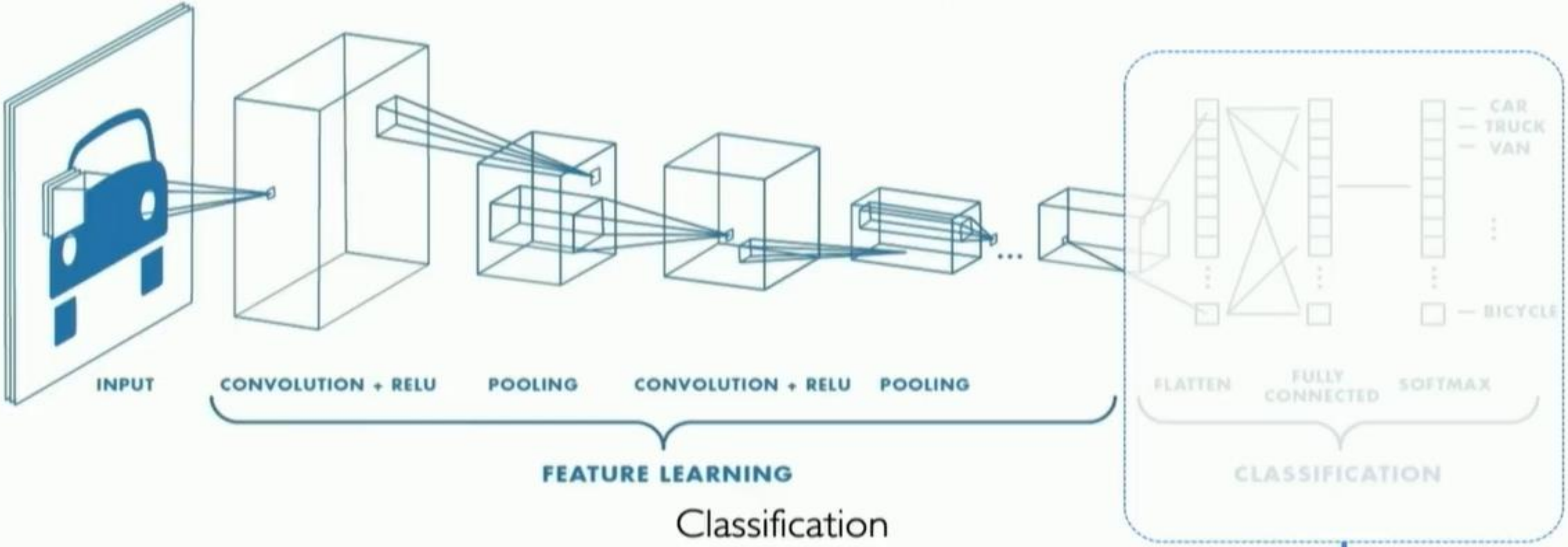
1. Learn features in input image through **convolution**
2. Introduce **non-linearity** through activation function (real-world data is non-linear)
3. Reduce dimensionality and preserve spatial invariance with **pooling**

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

$$\mathrm{softmax}(y_i) = \frac{e^{y_i}}{\sum}$$

An Architecture for Many Applications



Classification
Object detection
Segmentation
Probabilistic control