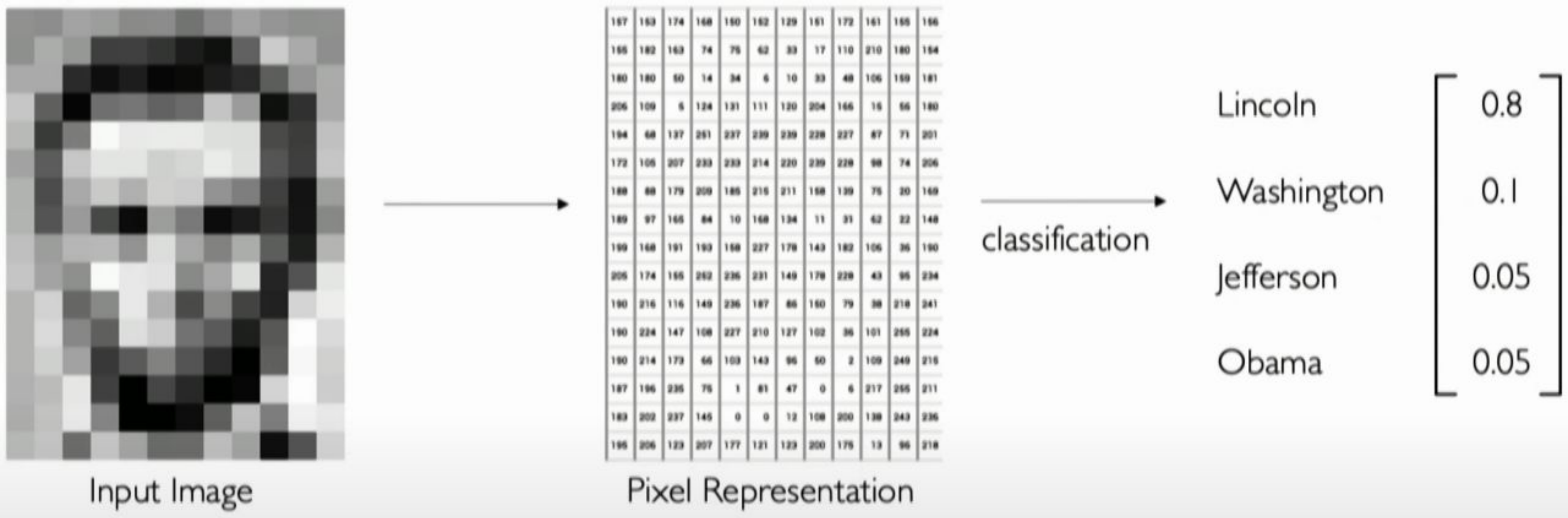
Introduction to Deep Learning for Computer Vision

Adhyayan '23 - ACA Summer School
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

Lecture 3

Tasks in Computer Vision



- ➔ **Regression**: Output variable takes continuous values.
- ➔ **Classification**: Output variable takes class label. Can produce probability of belonging to a particular class.

High Level Feature Detection

Let’s identify key-features in each image category

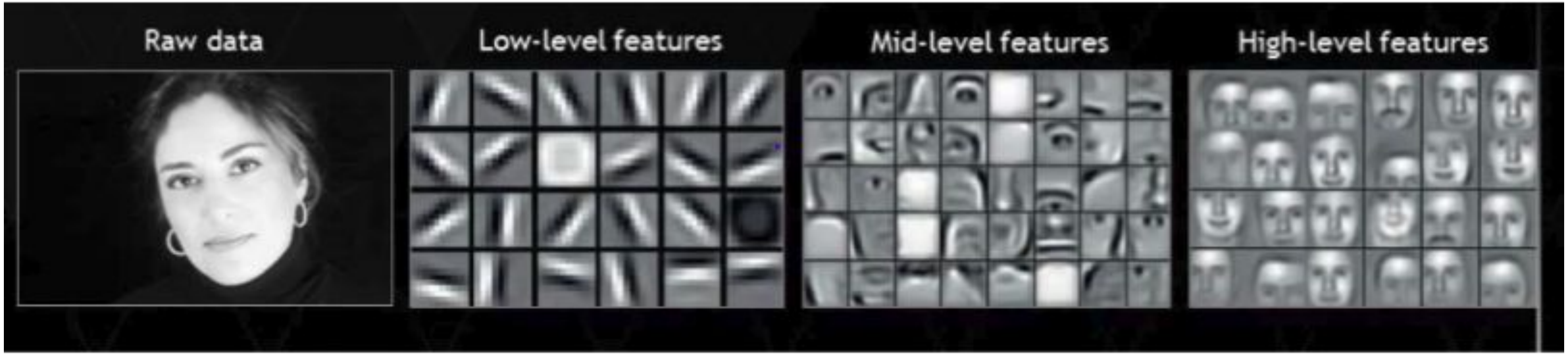
Nose,
Eyes,
Mouth

Wheels,
License Plate,
Headlights

Doors,
Windows,
Steps

Learning Feature Representations

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



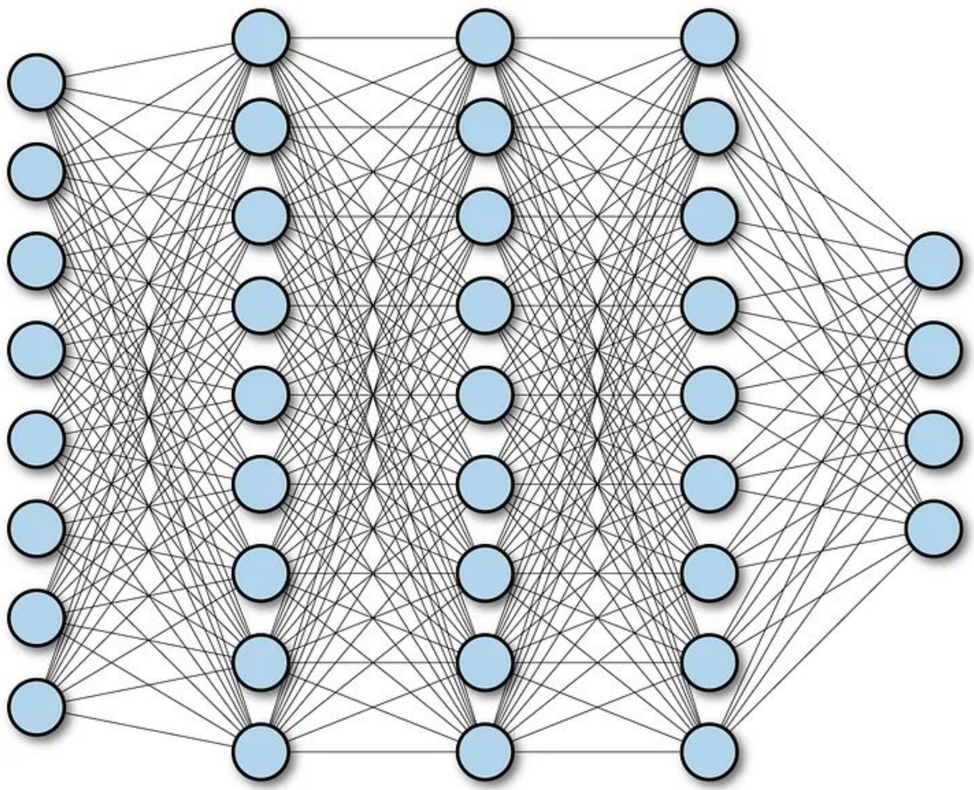
Edges, dark spots

Eyes, ears, nose

Facial Structure

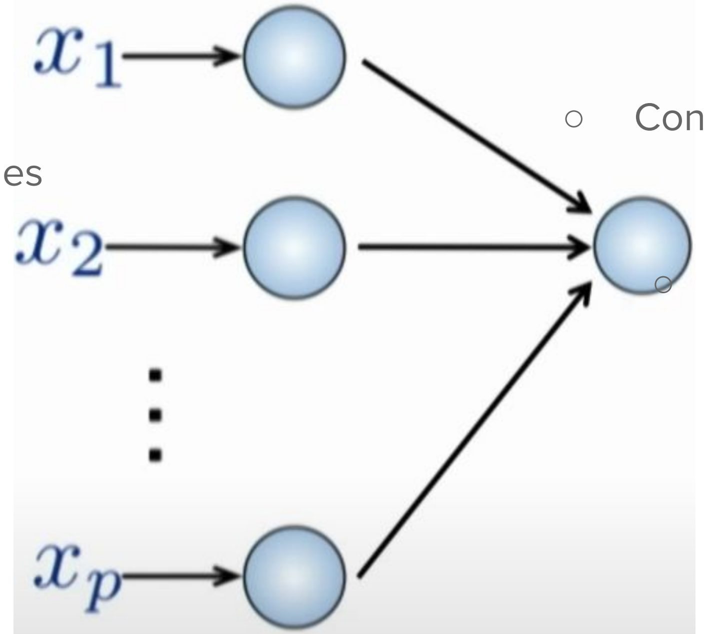
Learning Visual Features

Fully Connected Neural Network



Fully Connected Neural Network

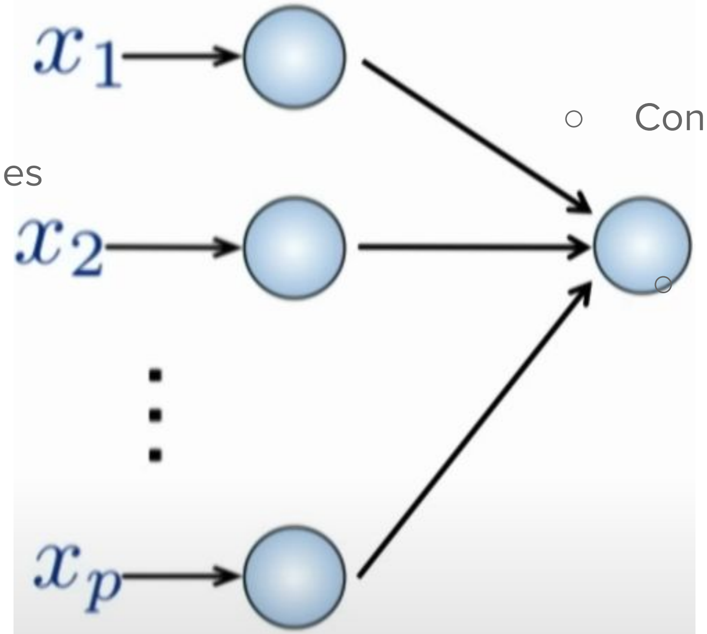
- Input:
  - 2D Image
  - Vector of pixel values



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

Fully Connected Neural Network

- Input:
  - 2D Image
  - Vector of pixel values

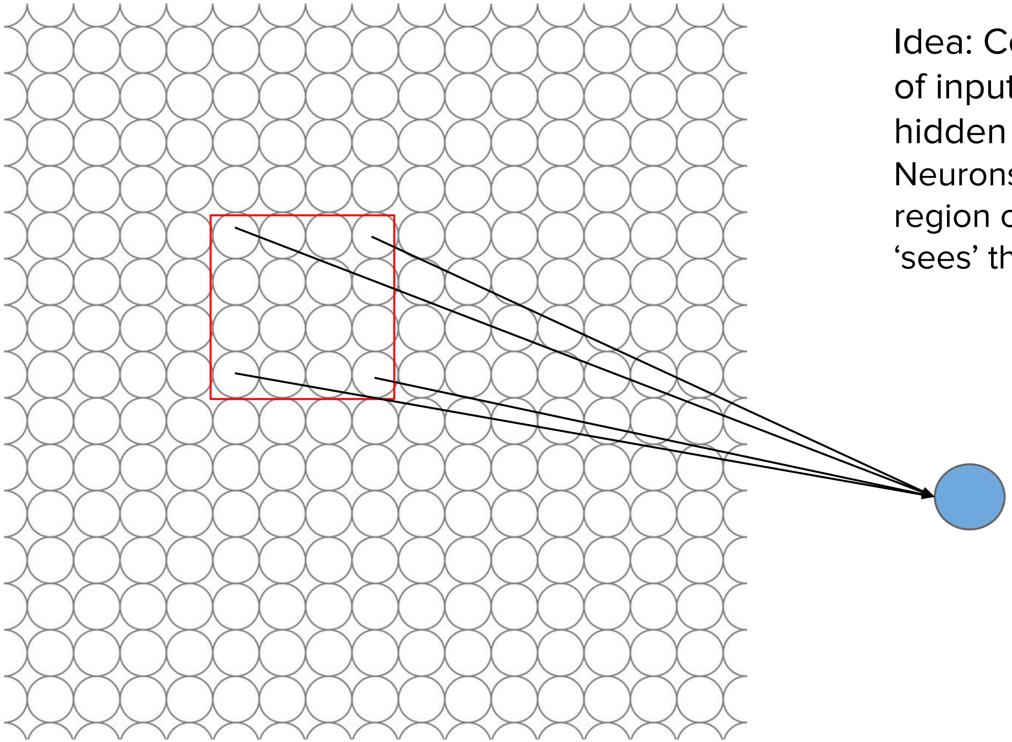


- Fully Connected:
  - Connect neuron in hidden layer to all neurons in output layer
  - No spatial information!
  - And many, many parameters.

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

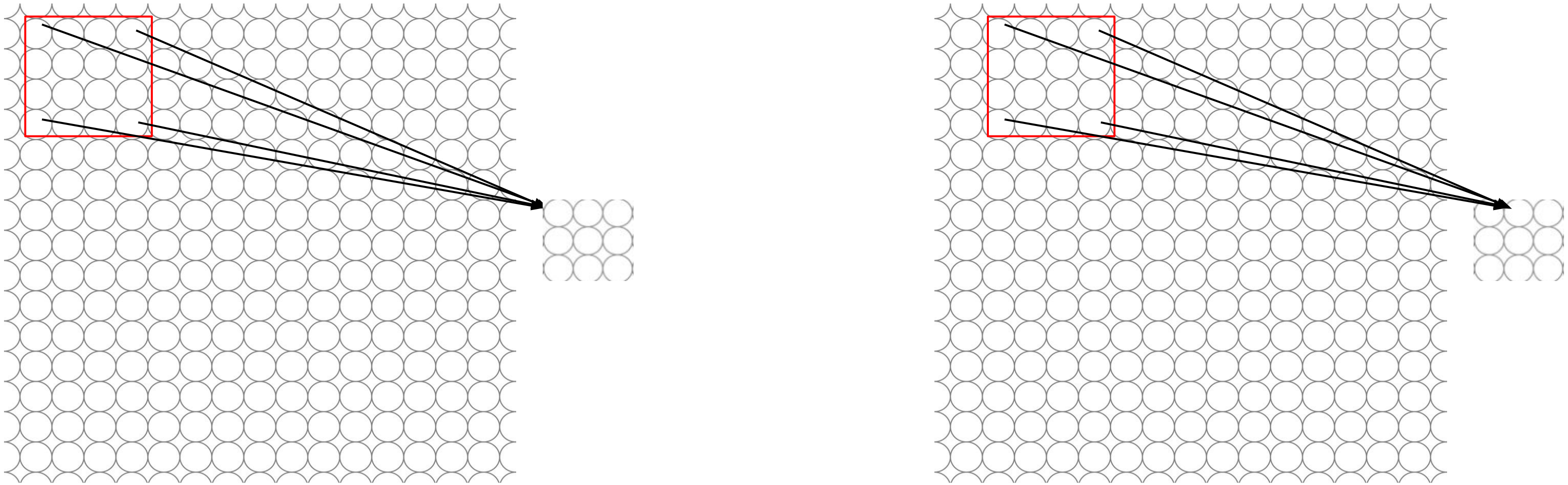
Using Spatial Structure

- Input:
  - 2D Image
  - Array of pixel values



Idea: Connect patches of input to neurons in hidden layer.
Neurons connected to region of input, only 'sees' these values.

Using Spatial Structure

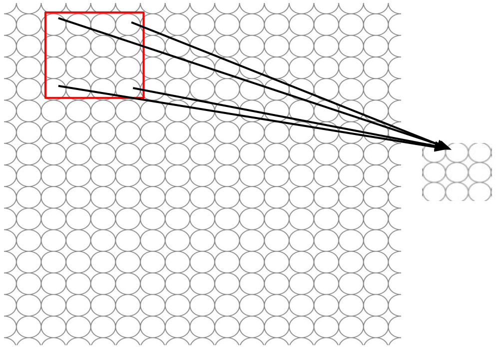


Connect patch in input layer to single neuron in subsequent layer.
Using a sliding window to define connections.
*How can we **weight** the patch to detect particular features?*

Applying Filters to Extract Features

1. Apply a set of weights - a filter - to 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)

Feature extraction with Convolution



- ➔ Filter of size 4 x 4: 16 different weights
- ➔ Apply the same filter to 4 x 4 patches in input
- ➔ Shift by 2 pixels for next patch

This ‘patchy’ operation is called **convolution**

1. Apply a set of weights - a filter - to 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)

Feature Extraction and Convolution: A Case Study

X or X?

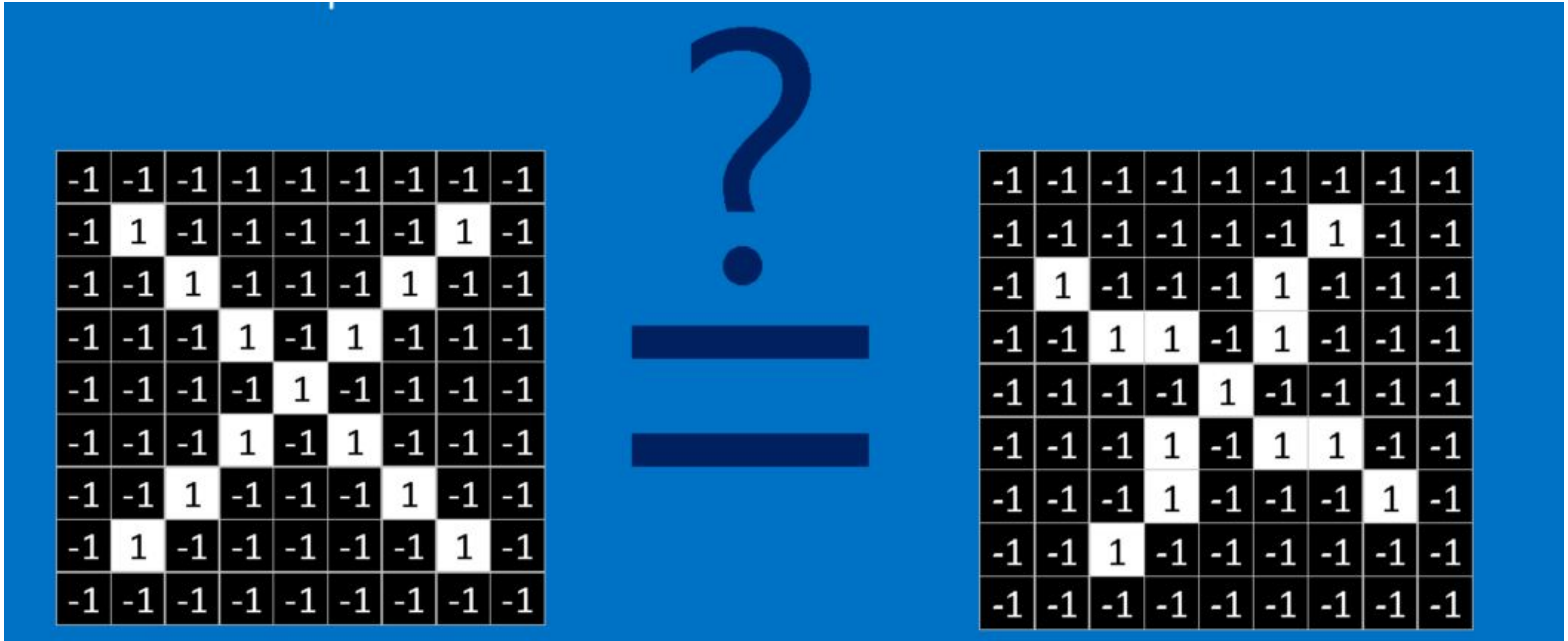
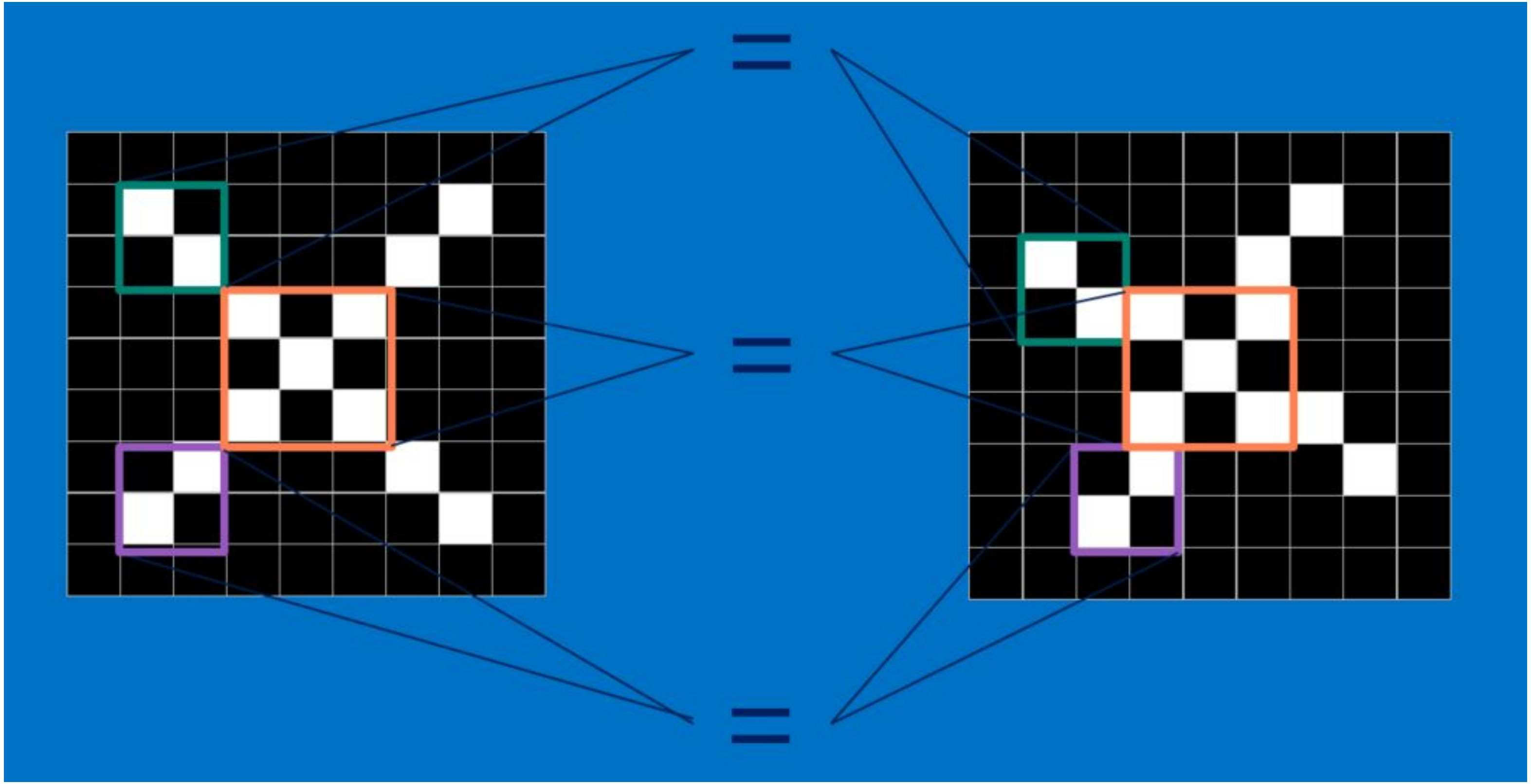
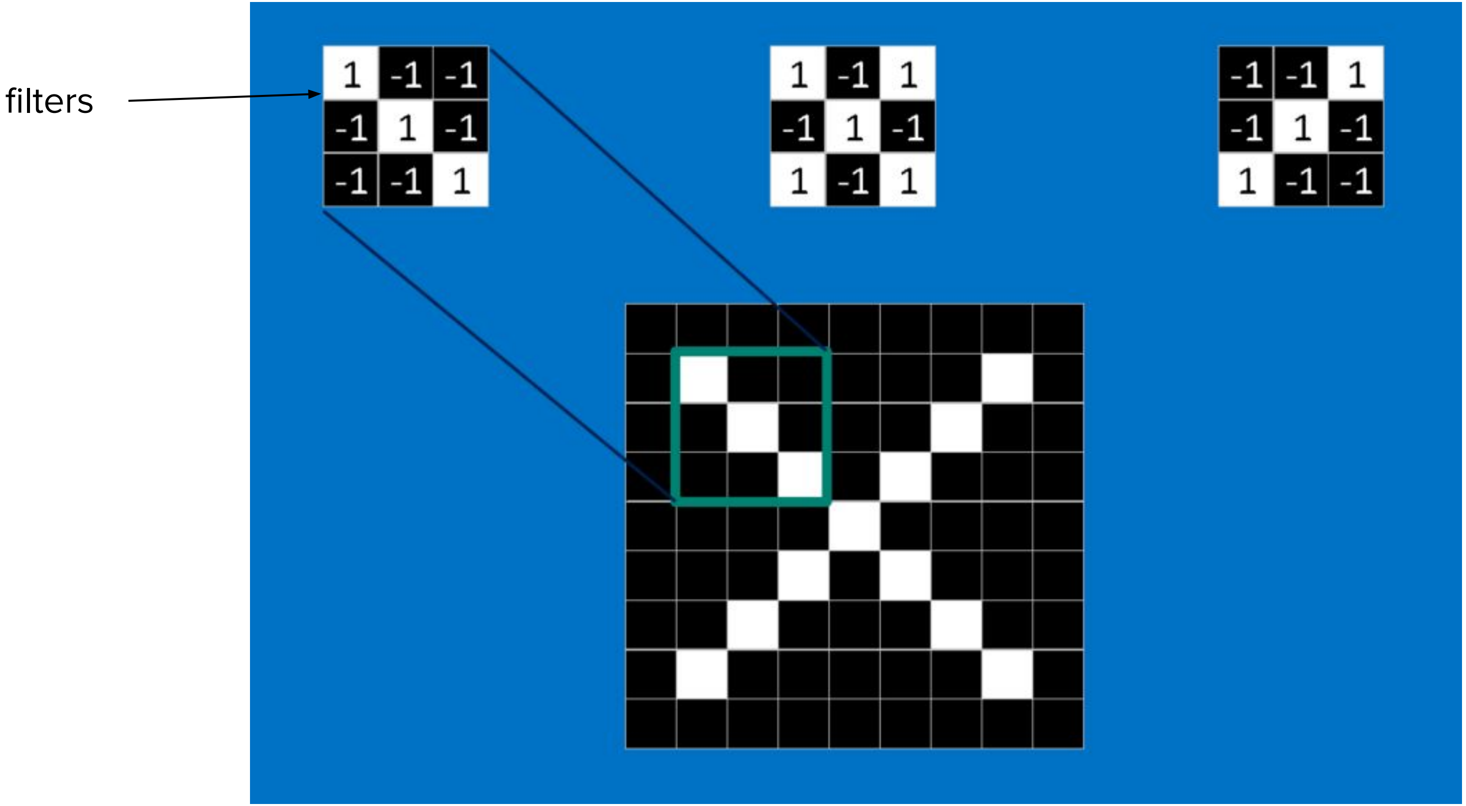


Image is represented as matrix of pixel values...and computers are literal!
We want X to be classified as X even if it is shifted, rotated, shrunken or deformed.

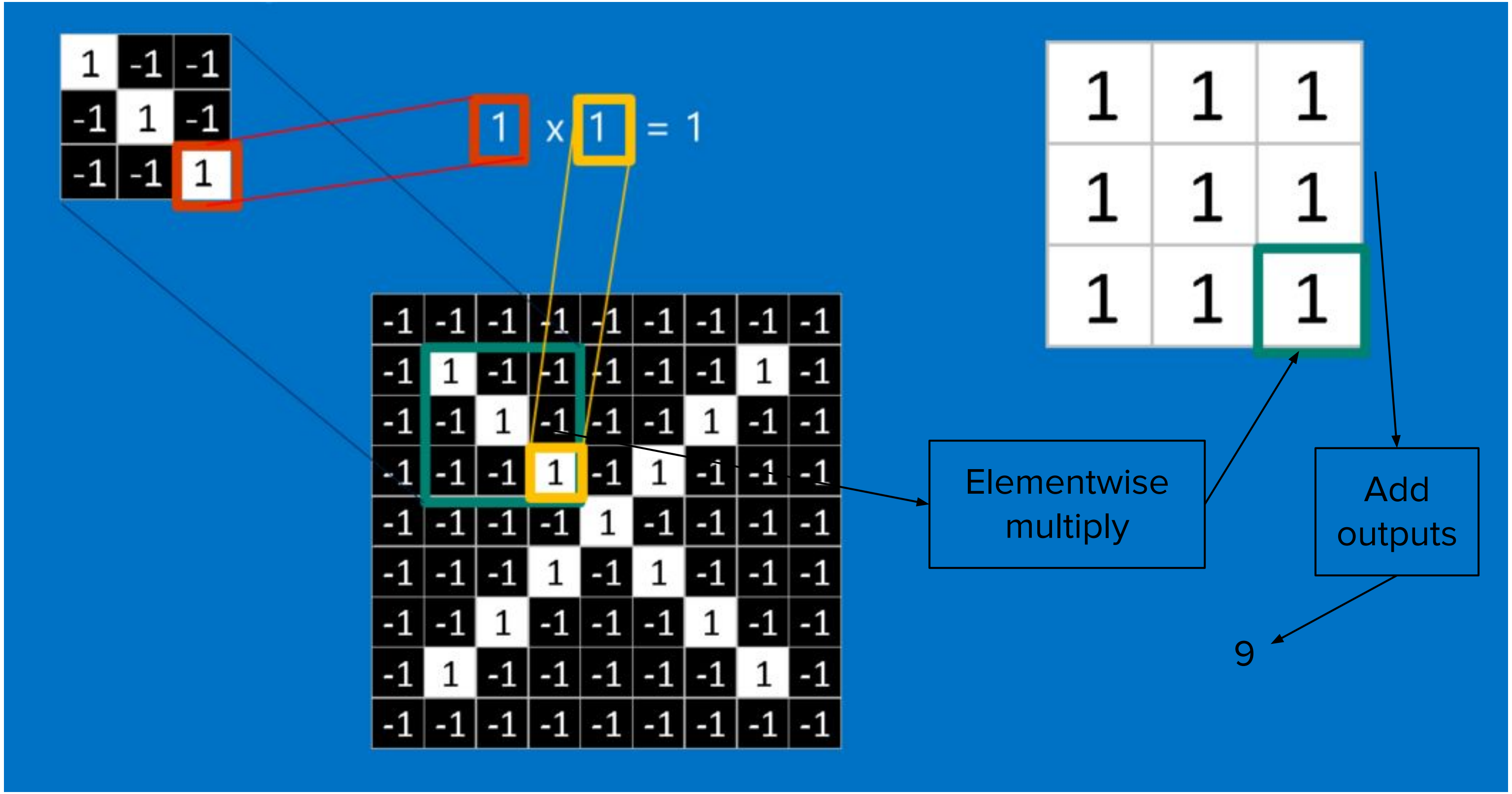
Features of X



Filters to Detect X Features



The Convolution Operation



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

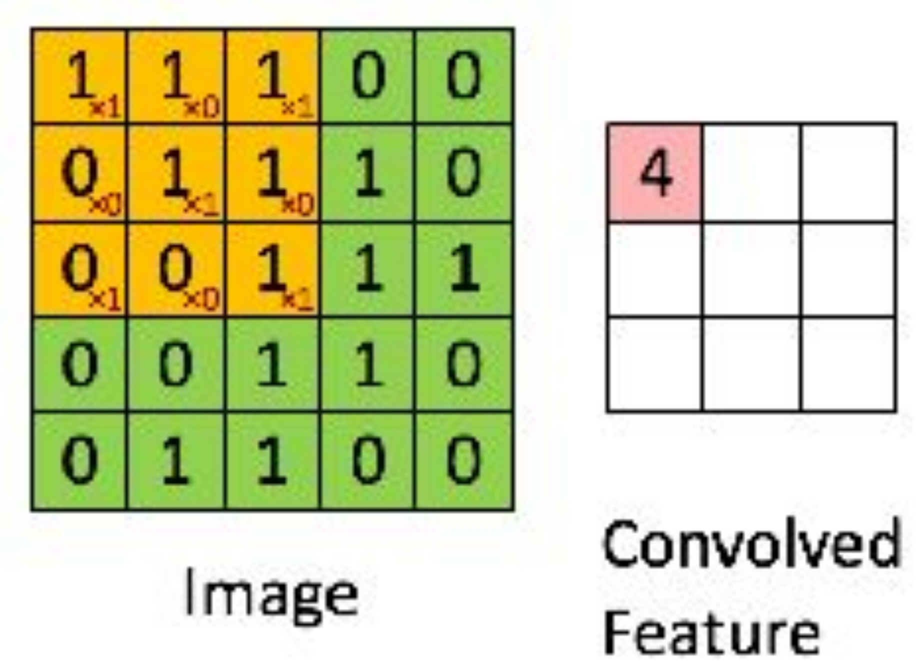
$\otimes$

1	0	1
0	1	0
1	0	1

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

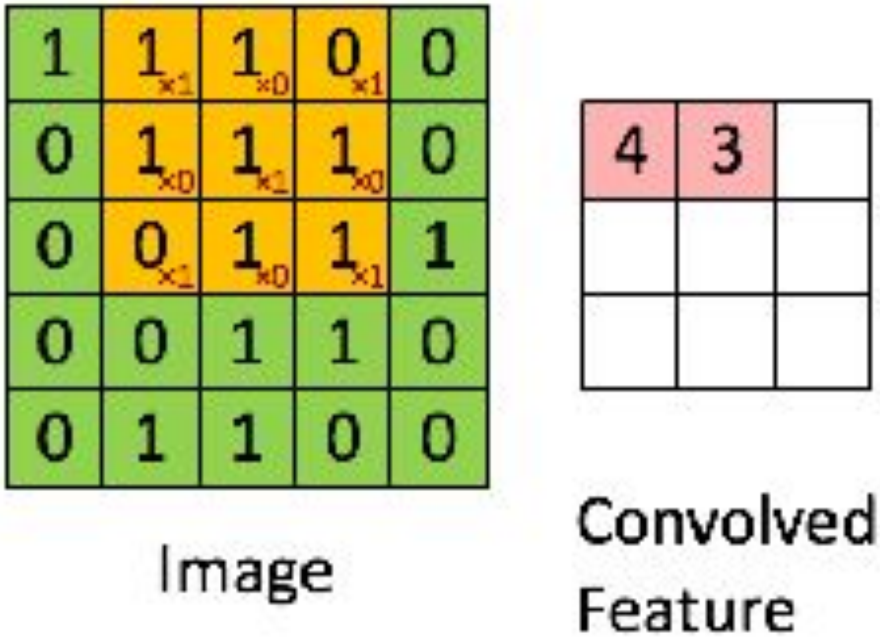
The Convolution Operation

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



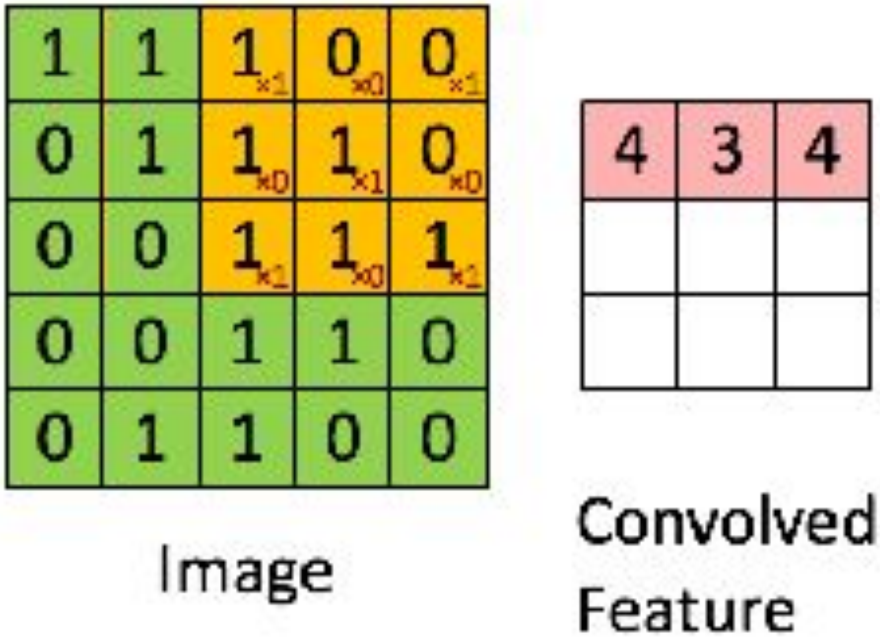
The Convolution Operation

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



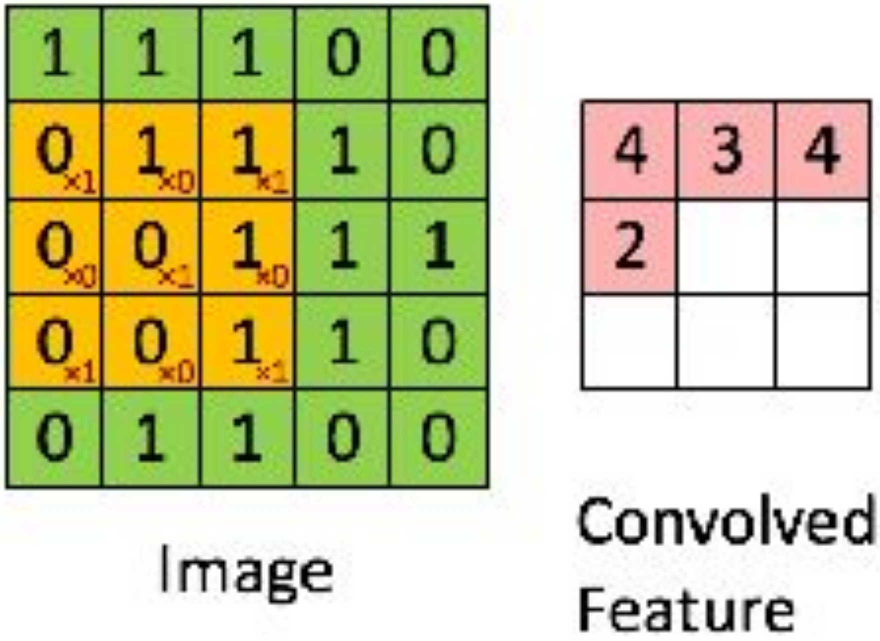
The Convolution Operation

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



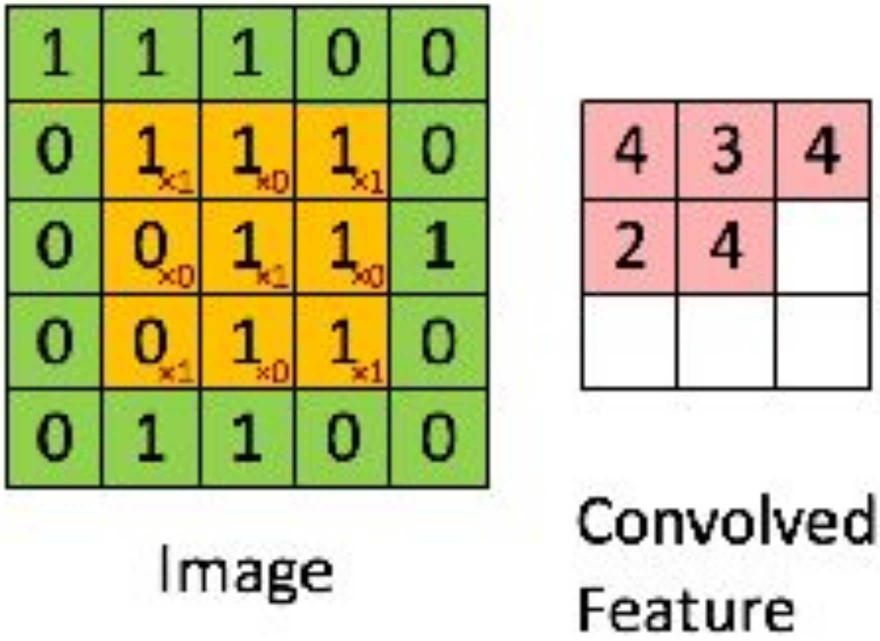
The Convolution Operation

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



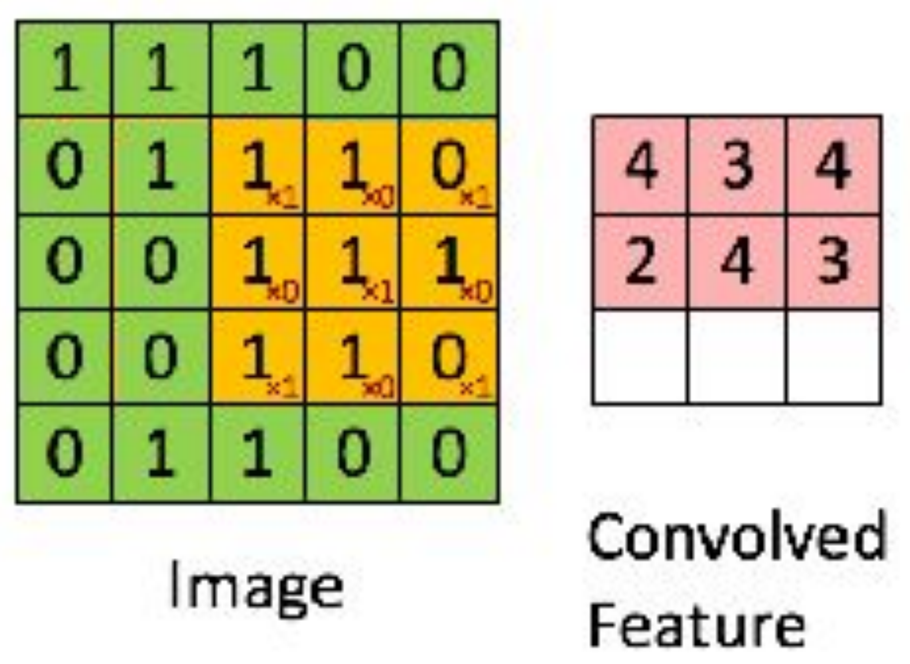
The Convolution Operation

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



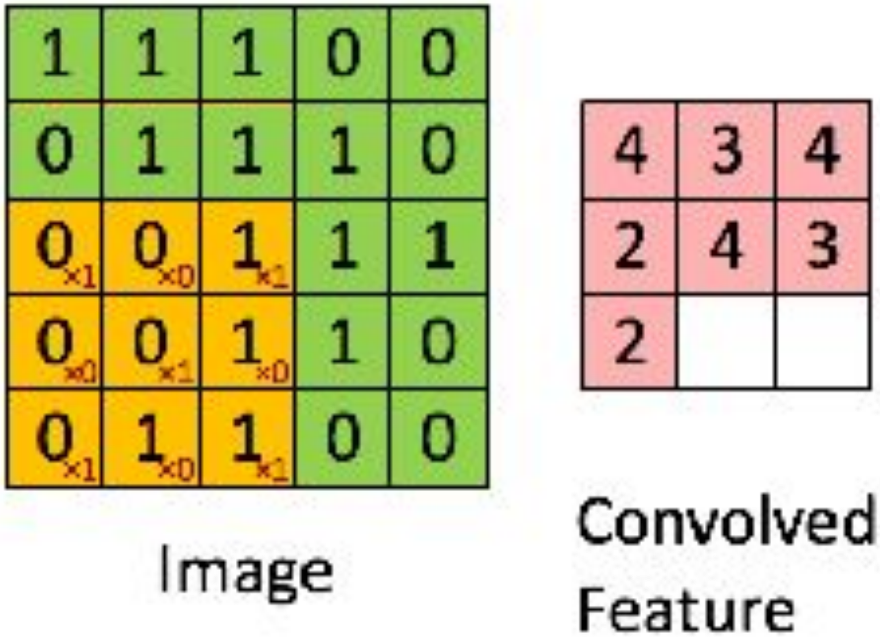
The Convolution Operation

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



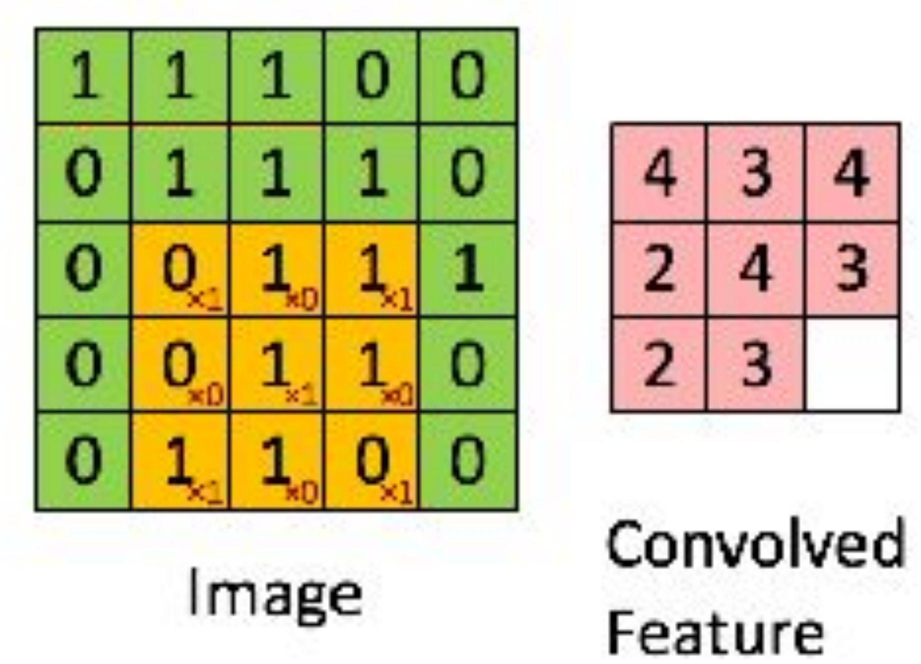
The Convolution Operation

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



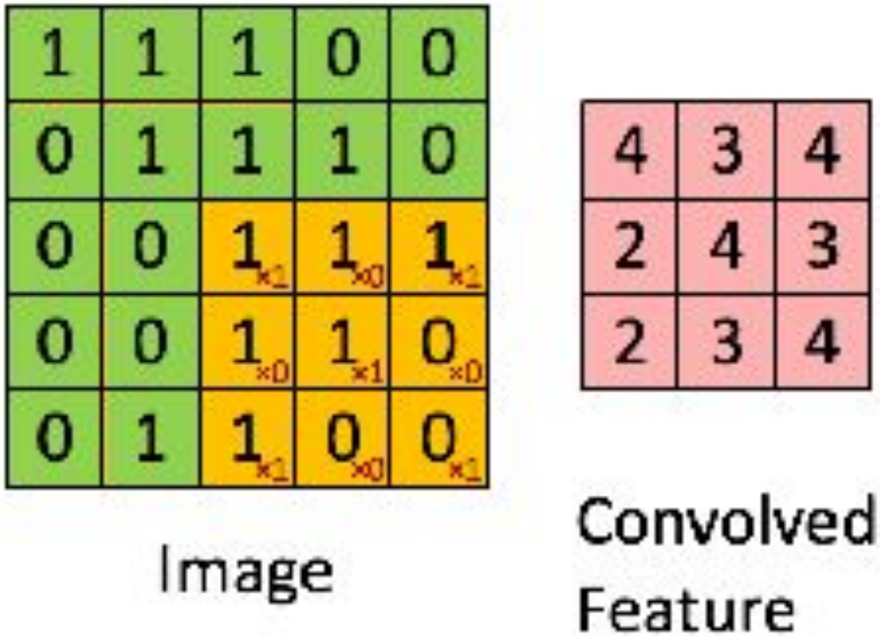
The Convolution Operation

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



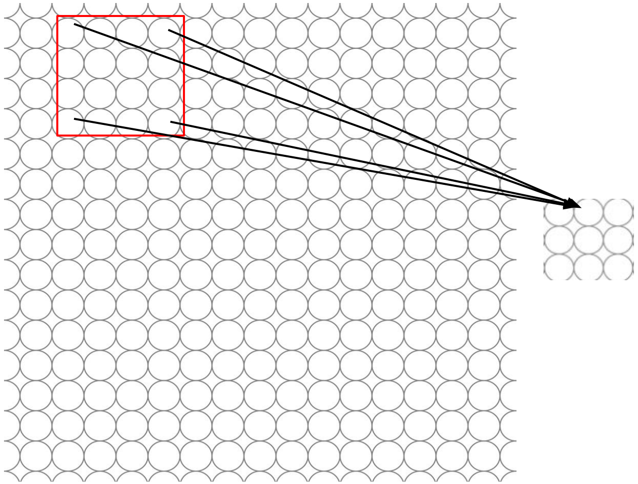
The Convolution Operation

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



Producing Feature Maps

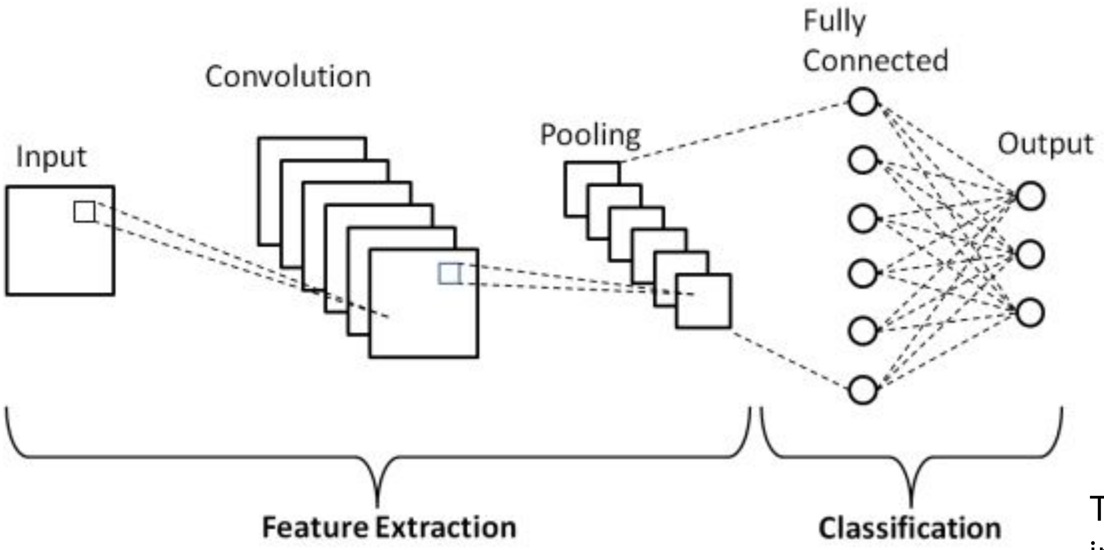
Feature Extraction with Convolution



1. Apply a set of weights - a filter - to 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)

Convolutional Neural Networks (CNNs)

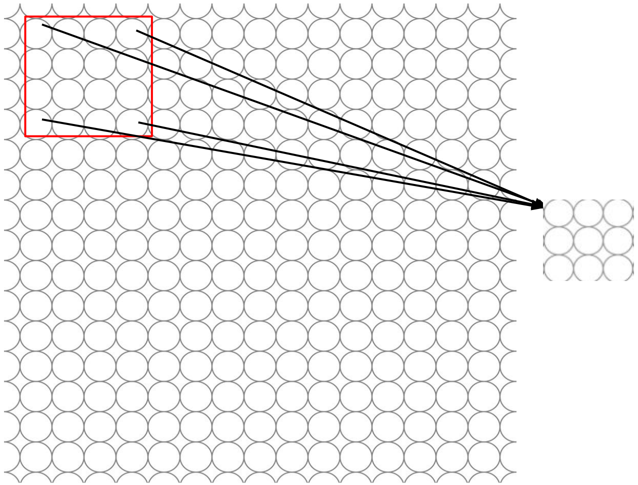
CNNs for Classification



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

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

Convolutional Layers: Local Connectivity



For a neuron in hidden layer:

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

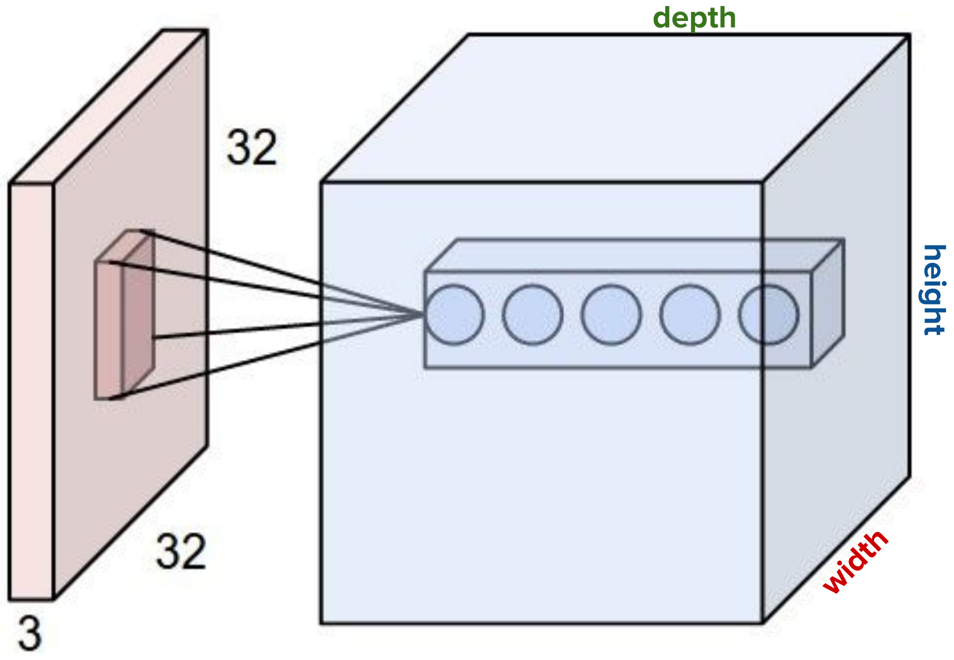
4x4 filter: matrix of weights w_{ij}

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

For neuron (p,q) in hidden layer

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

CNNs: Spatial Arrangement of Output Volume



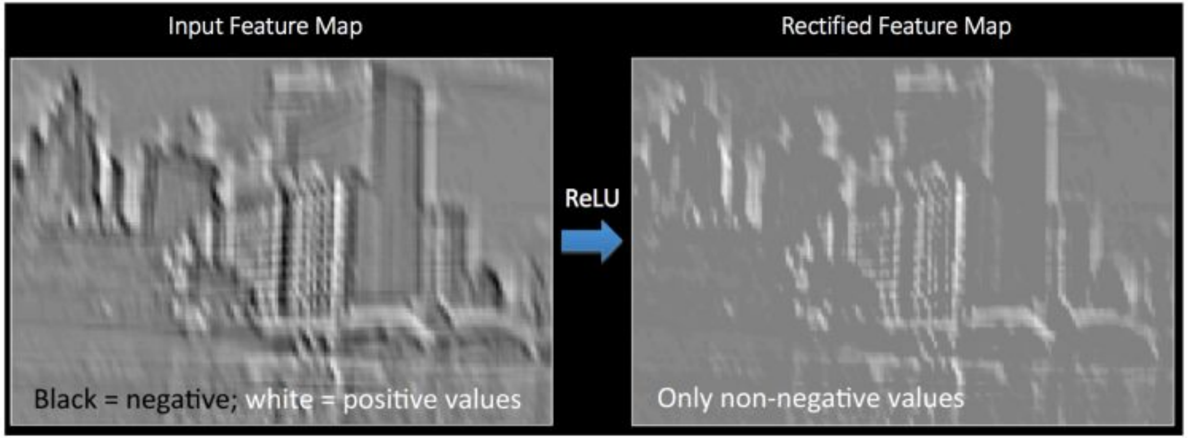
Layer Dimensions:
h x w x d

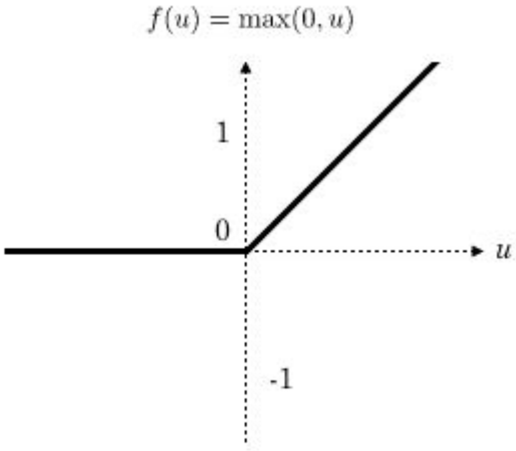
Stride:
Filter Step Size

Receptive Field:
Locations in input image that a node is path-connected to.

Introducing Non-Linearity

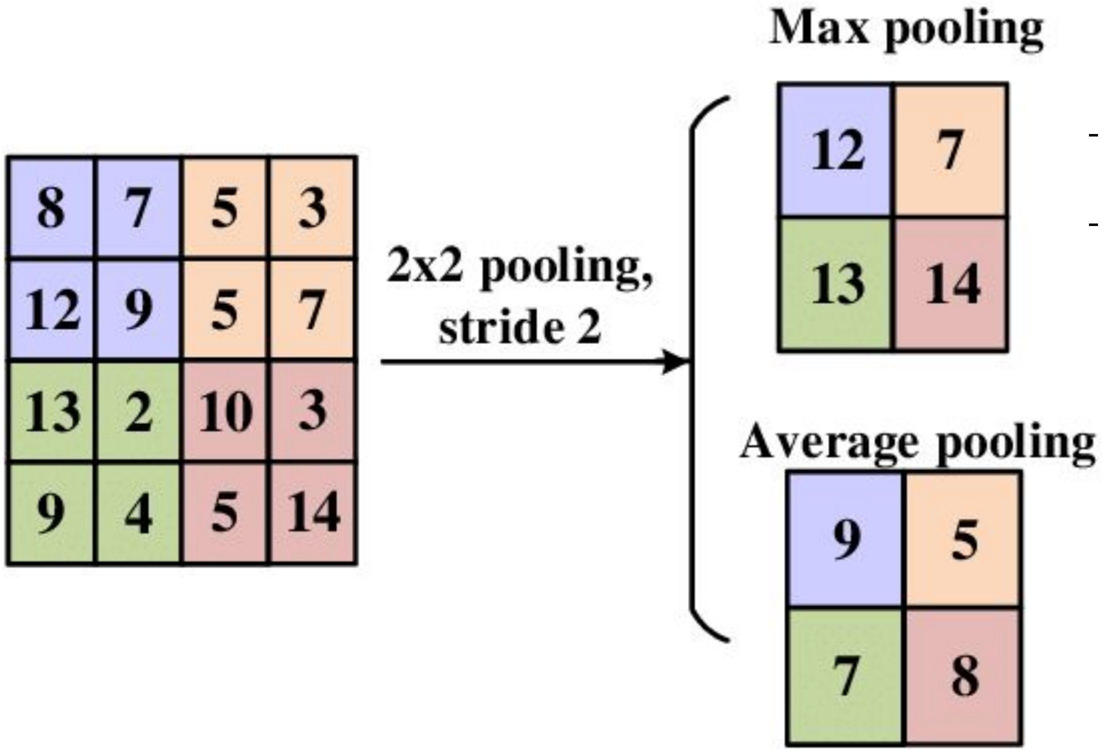
- Apply after every convolution operation (i.e., after convolutional layers)
- ReLU: pixel-by-pixel operation that replaces all negative values by zero. **Non-linear operation**!





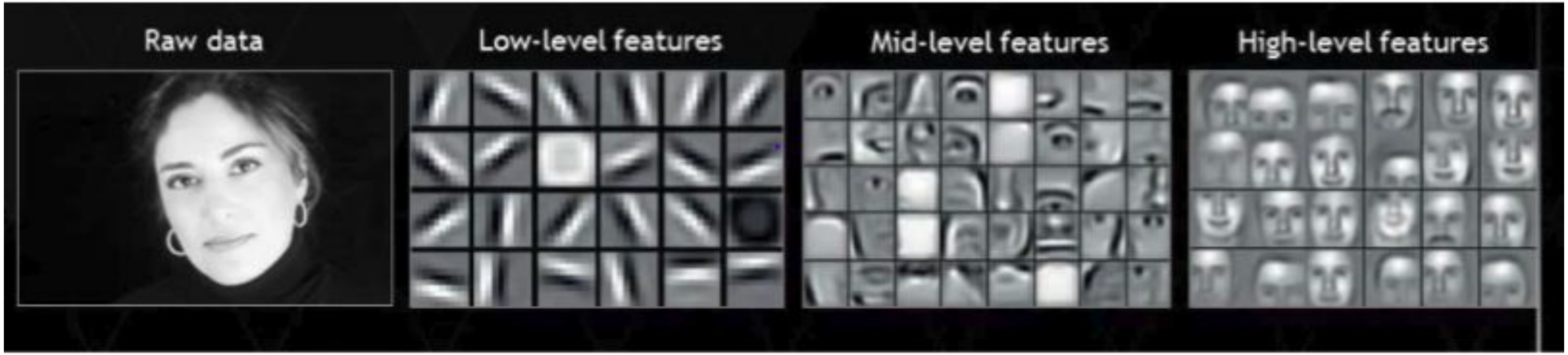
Rectified Linear Unit (ReLU)

Pooling



- Reduced Dimensionality
- Spatial Invariance

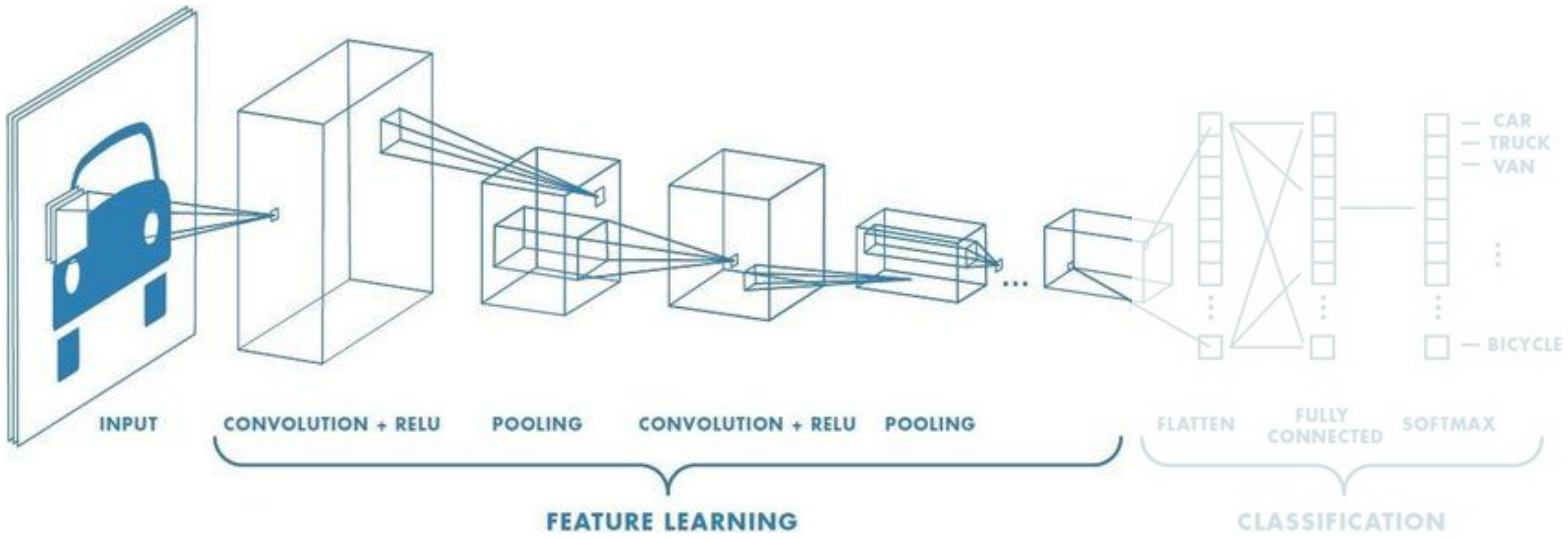
Representation Learning in Deep CNNs



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

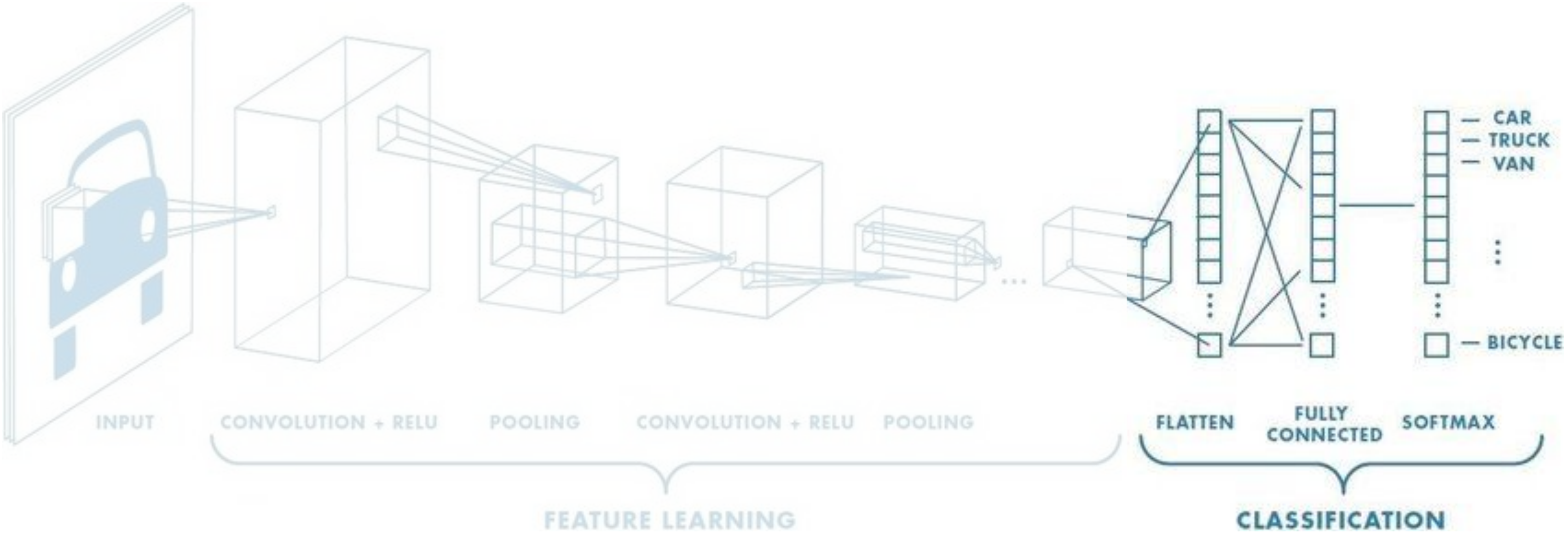
Conv Layer 1 | Conv Layer 2 | Conv Layer 3

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**

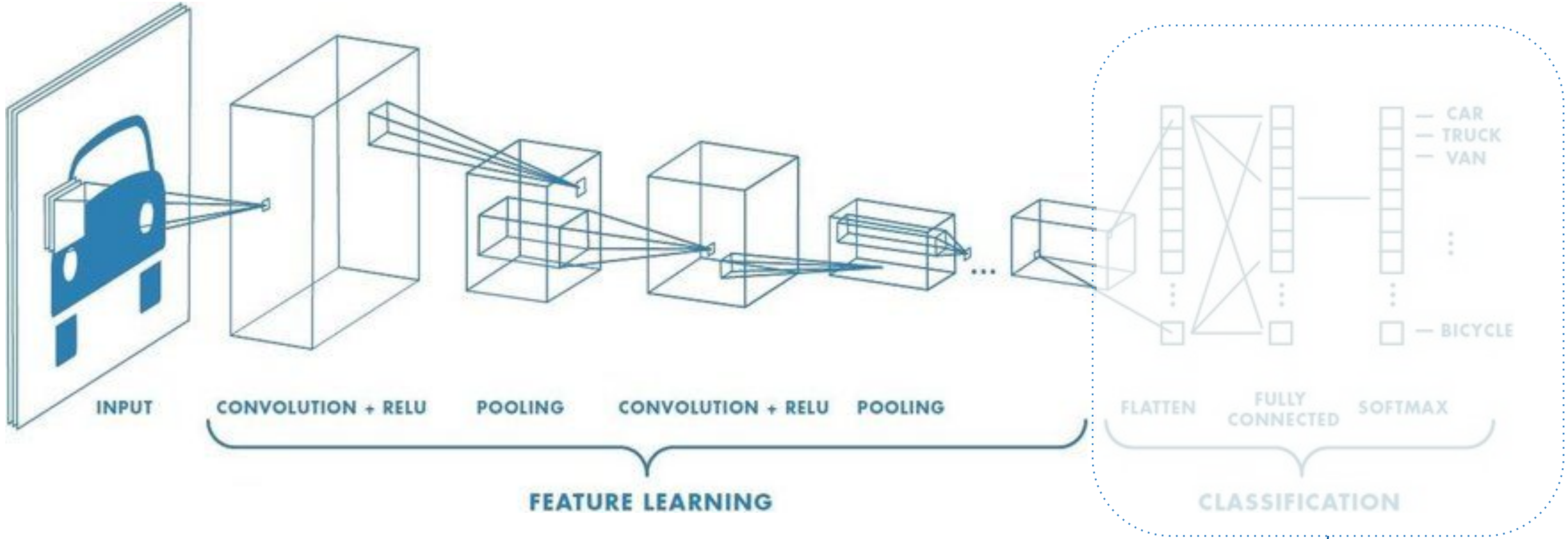
CNNs for Classification: Class Probabilities



- 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

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

CNNs for Classification: Feature Learning



Classification
Object Detection
Segmentation
Generative AI

Next Lecture: Different CNN Architectures, Transfer Learning!