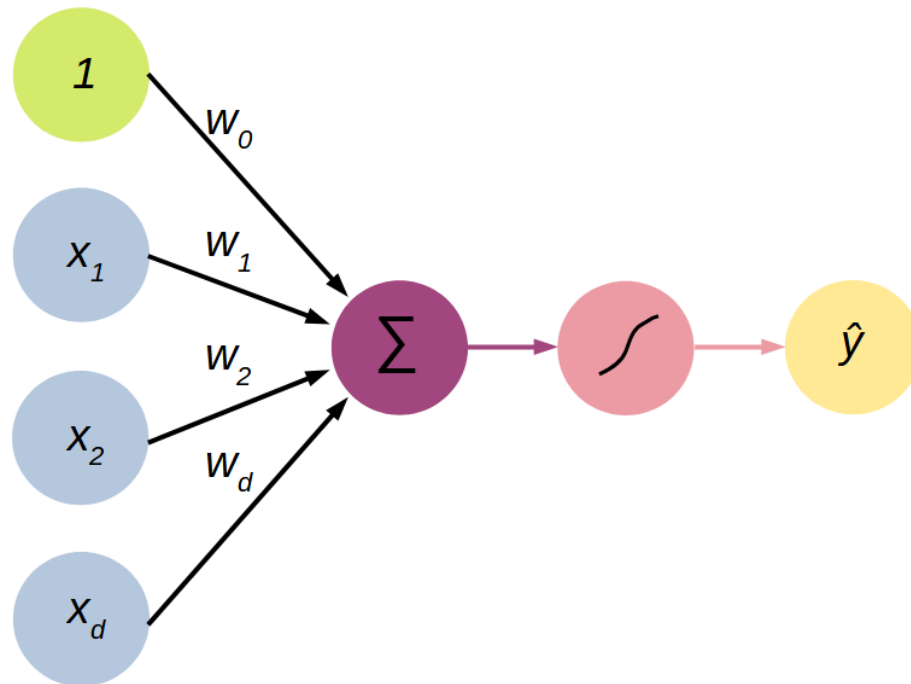

IMAGE ANALYTIC USING DEEP LEARNING

Mohd Halim Mohd Noor, PhD

Outline

- Perceptron
- Neural Networks
- Training Neural Networks
- Feature Extraction in Deep Learning
- Convolutional Neural Networks

Perceptron: Forward Propagation



Inputs Weights Sum Non-linearity Output

Linear combination of inputs

$$\hat{y} = a = g \left(w_0 + \sum_{j=0}^d w_j x_j \right)$$

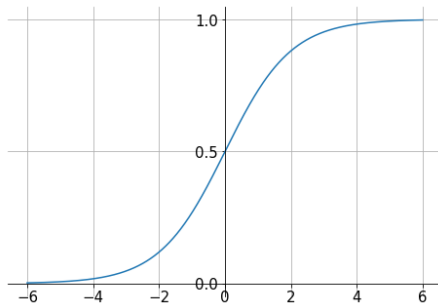
Activation function bias

Activation Function

- An activation function is a mathematical function that determines the output of a unit in the network
- It introduces non-linearities into the network, allowing it to learn and model complex relationships between inputs and outputs
- The common activation functions are Sigmoid, Hyperbolic Tangent and Rectified Linear Unit (ReLU)

Activation Function

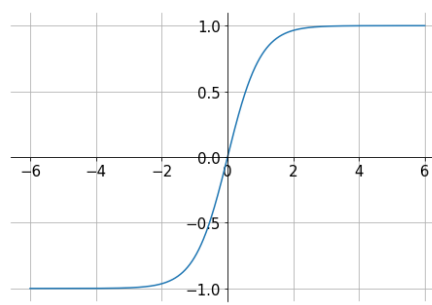
Sigmoid (Logistic)



$$g(z) = \frac{1}{1 + e^{-z}}$$

Sigmoid maps the input to a value in the range of 0 and 1

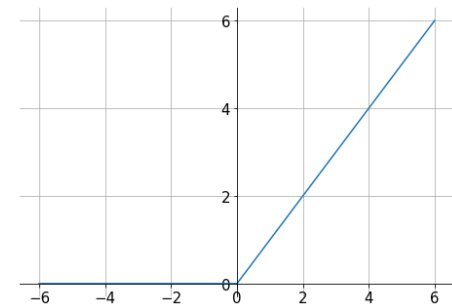
Hyperbolic tangent



$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

Hyperbolic tangent (tanh) is similar to sigmoid but the output is between -1 and 1

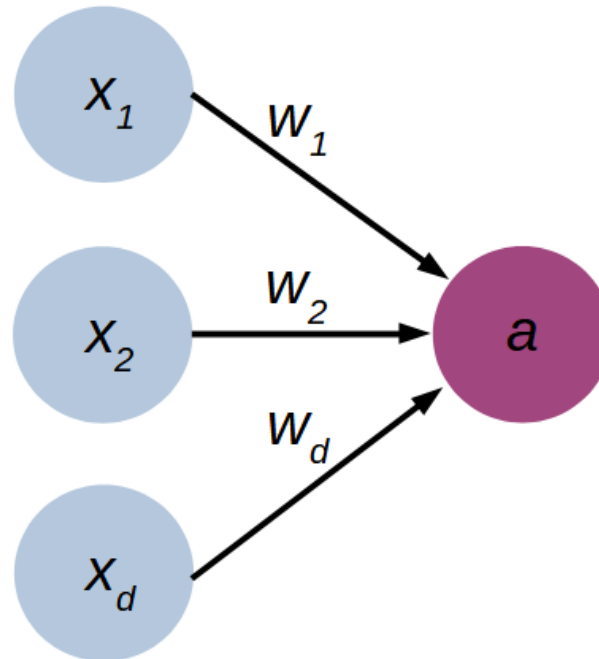
Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

ReLU is a piecewise linear function whereby the output is zero if the input is less than 0, and the output is the input itself if the input is more than 0

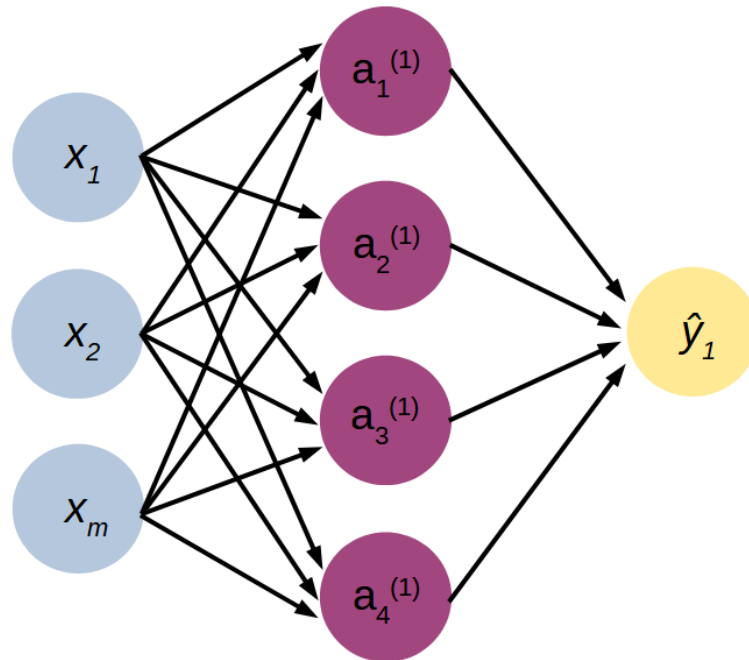
Perceptron (Simplified)



$$z = w_0 + \sum_{j=0}^d w_j x_j$$

$$a = g(z)$$

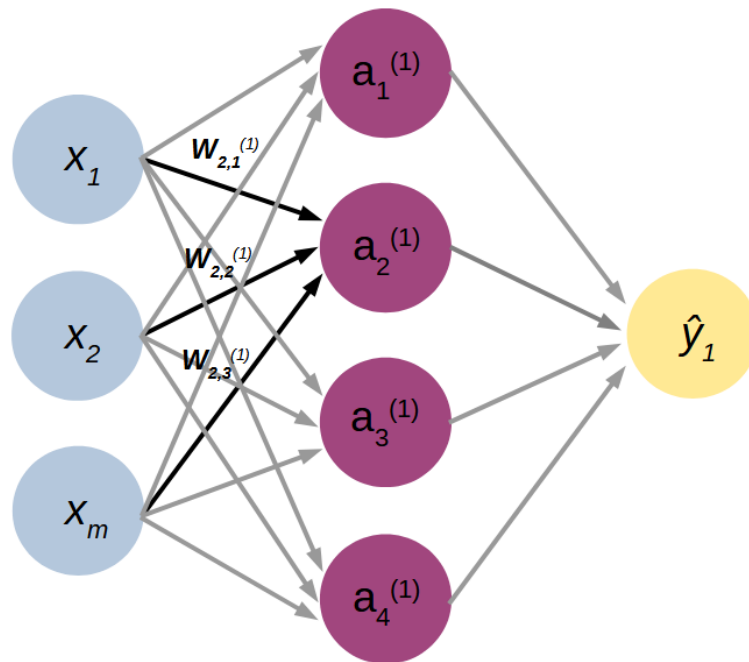
Multi-layer Perceptron



a.k.a. Neural Network

- The input is propagated to a layer of perceptrons (hidden layer)
- The output of the hidden layer is propagated to the output layer
- Because of the hidden layer, the state is not directly observable

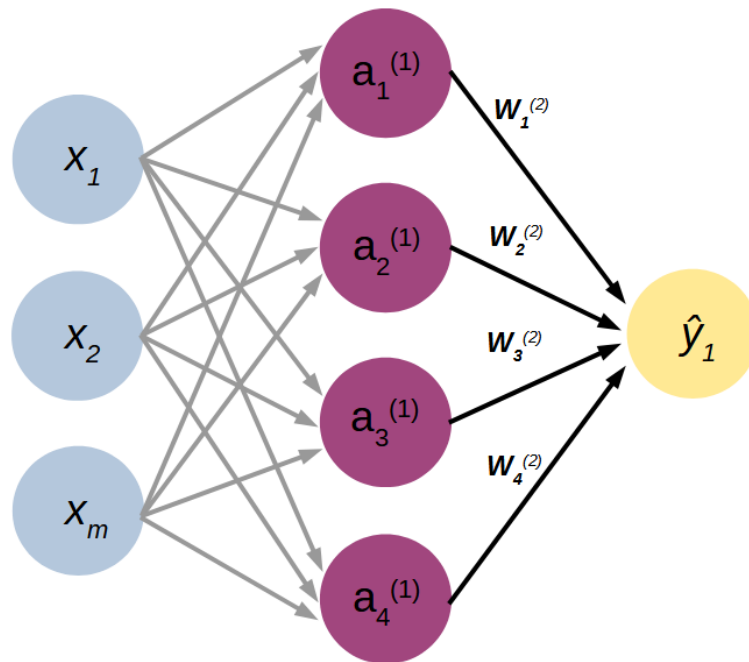
Multi-layer Perceptron



a.k.a. Neural Network

- The forward propagation is the same
- Each unit of the hidden layer takes the input, calculates the sum of weighted input and the activation

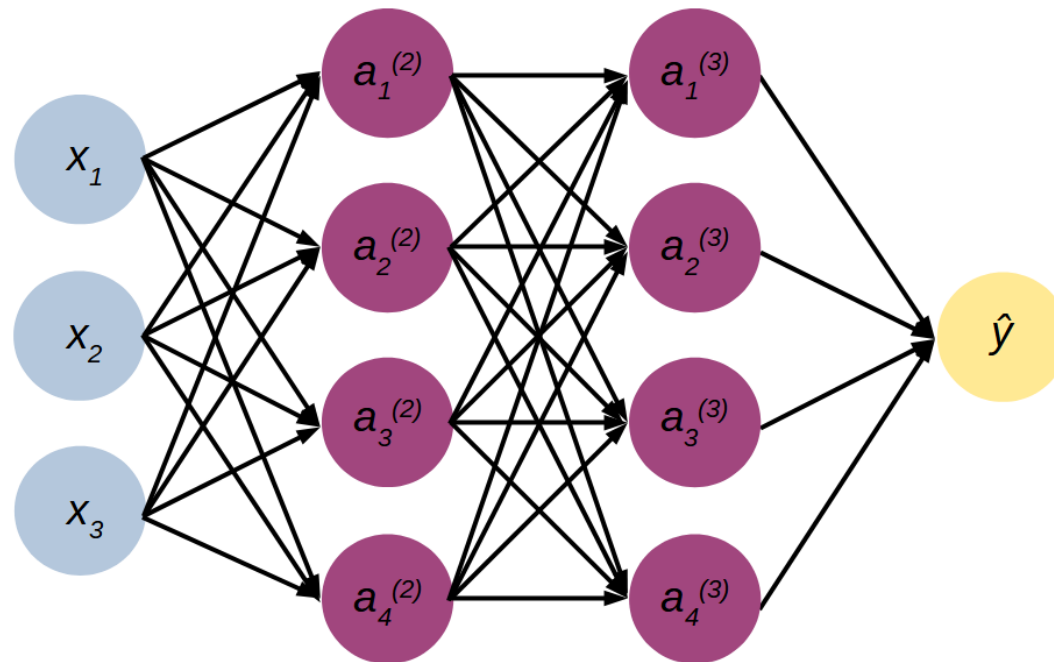
Multi-layer Perceptron



a.k.a. Neural Network

- The output of the hidden layer is used as input to the output layer
- The unit calculates the sum of weighted input and the activation

Deep Neural Network



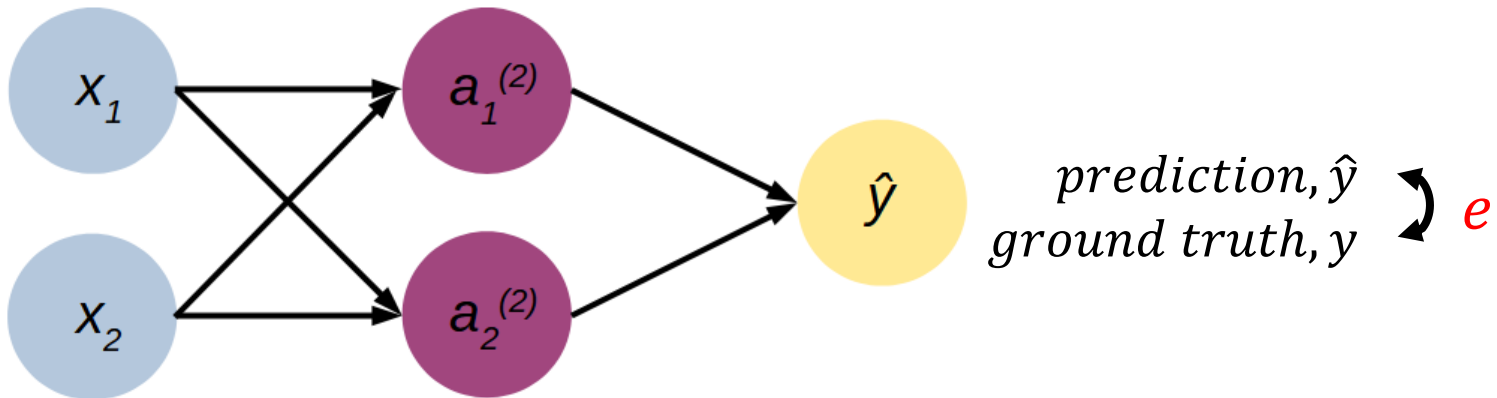
Deeper neural network architecture

More than one hidden layer can be stacked e.g. 2, 3 hidden layers

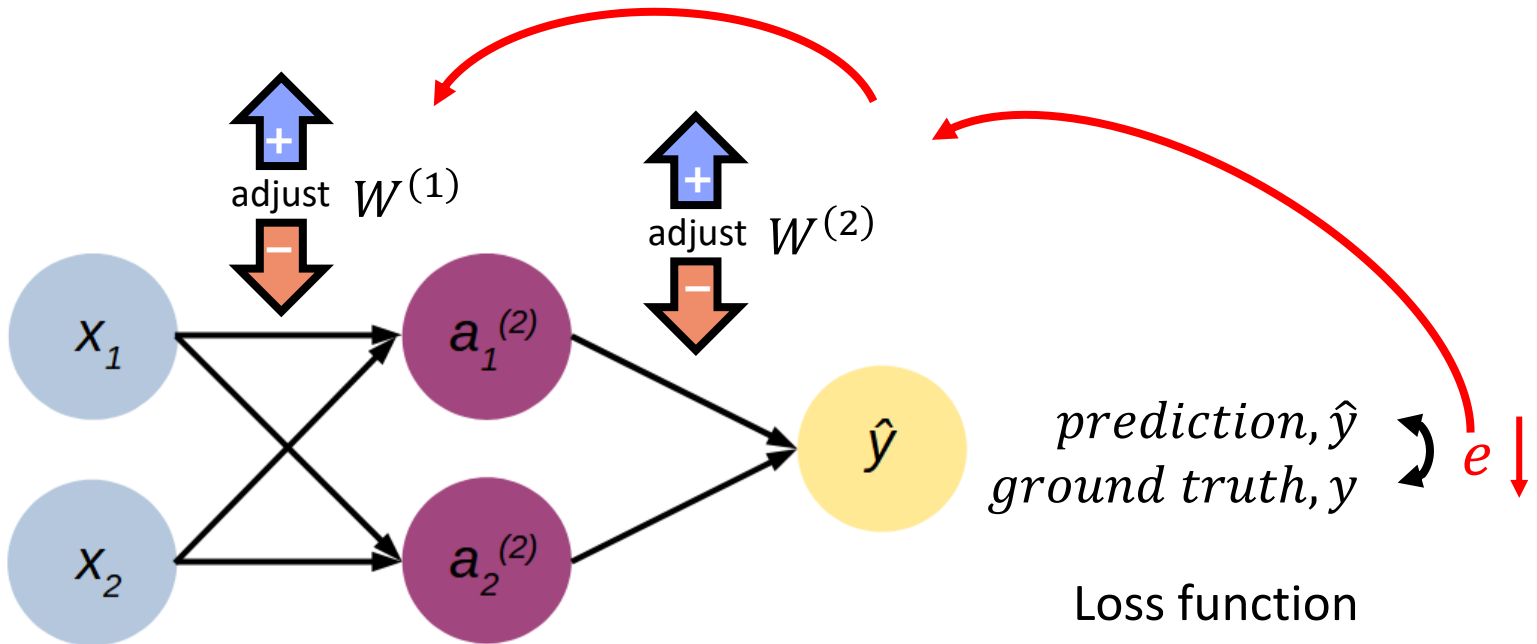
The forward propagation i.e. the output of a hidden layer is used as input to the next hidden layer

Training Neural Networks

Parameter Estimation



Parameter Estimation



Training Parameters

- Loss function
 - Regression: mean squared error
 - Classification: (binary) cross entropy
- Learning Rate, α
 - Default value: 0.001 (TensorFlow)
 - If α is too small, gradient descent can be slow
 - If α is too large, gradient descent can overshoot and may fail to converge

Feature Extraction in Deep Learning

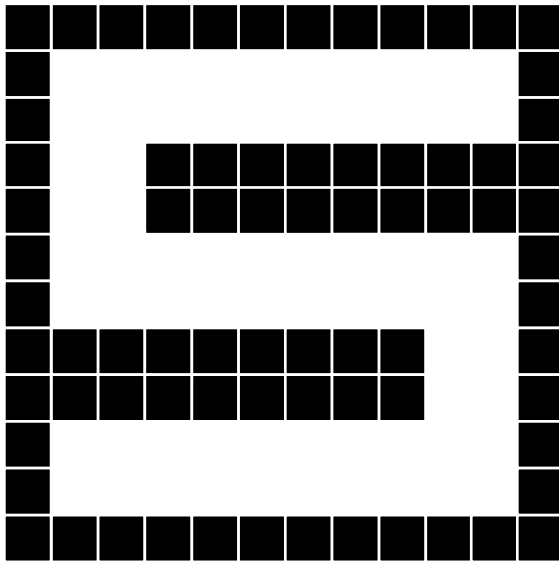
Using Spatial Structure



Process a region (patch of input data) to extract the details (local features)

Feature Extraction (Case Study)

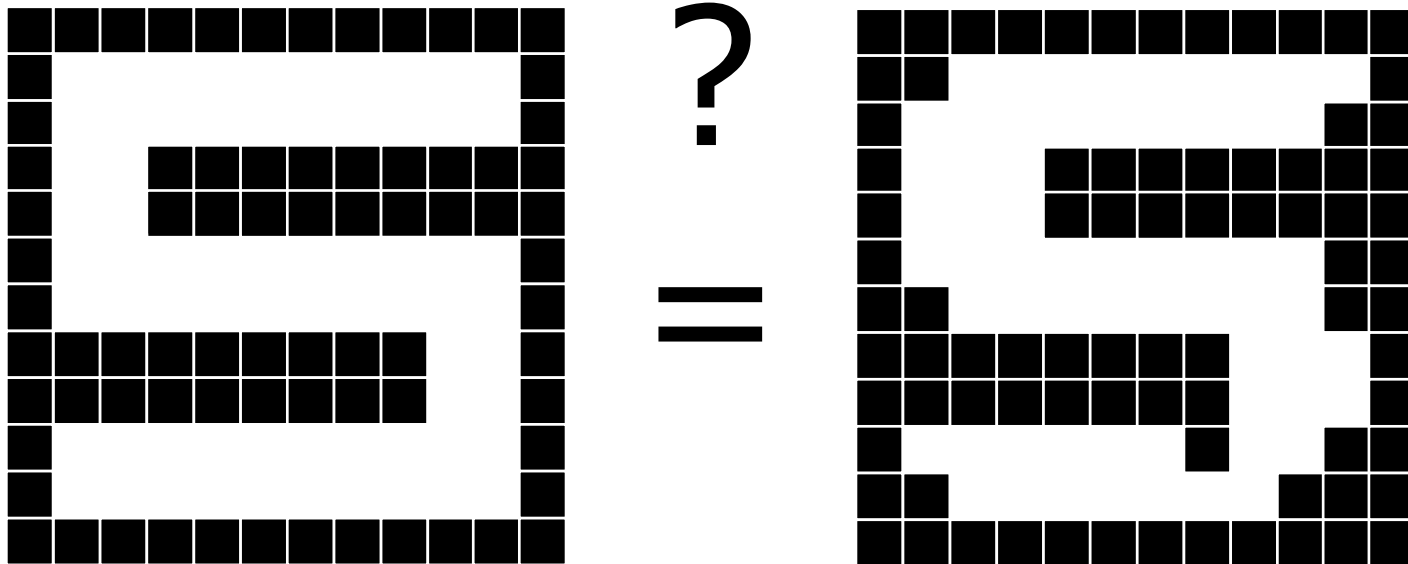
Given an image of five



We recognize this image is a digit five

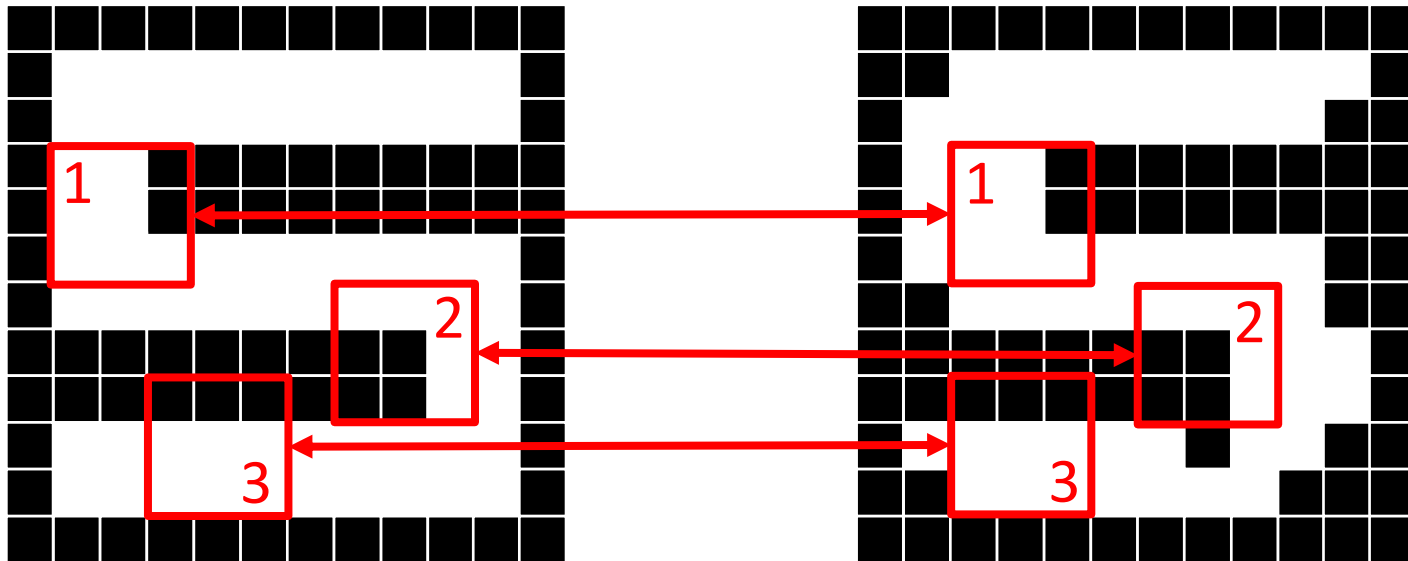
Feature Extraction (Case Study)

Can a computer recognize given a distorted/deformed image of 5?



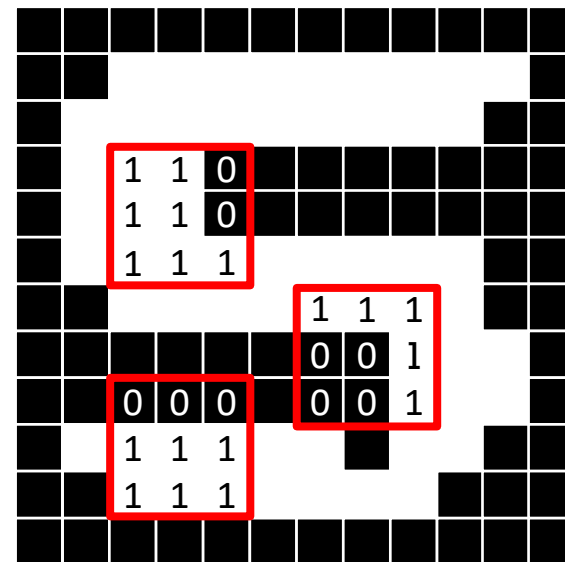
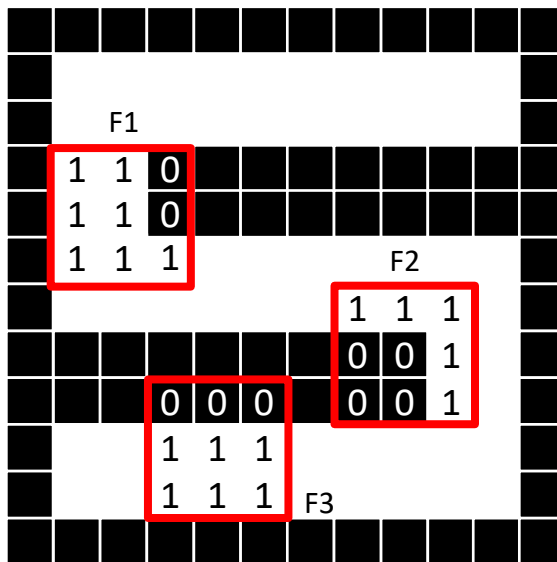
Feature Extraction

One way to recognize it is to capture/detect the features of the image – corner 1, corner 2 and horizontal line



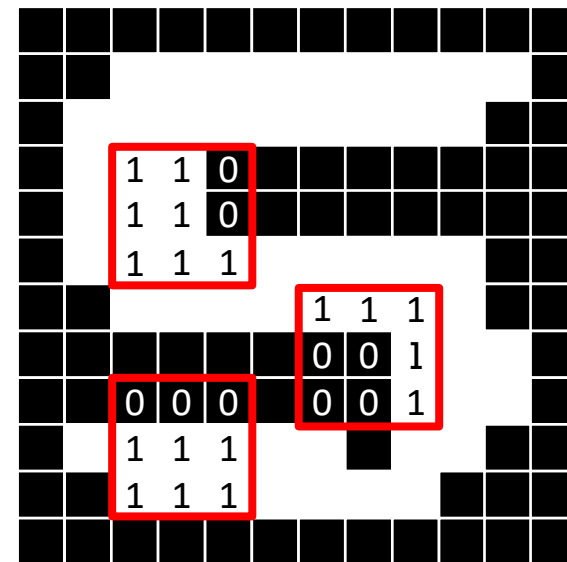
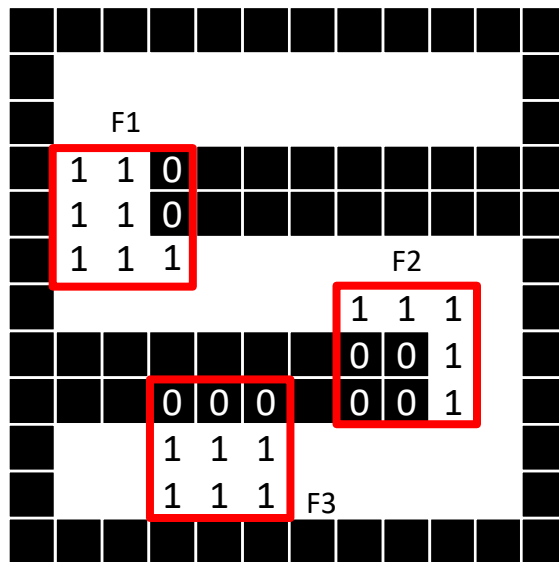
Feature Extraction

A feature is a mini-image – a small 2D array of pixels
Assuming black = 0 and white = 1



Feature Extraction

The three filters are designed to detect three types of features



1	1	0
2	1	0
2	1	1

1	2	2
0	0	2
0	0	1

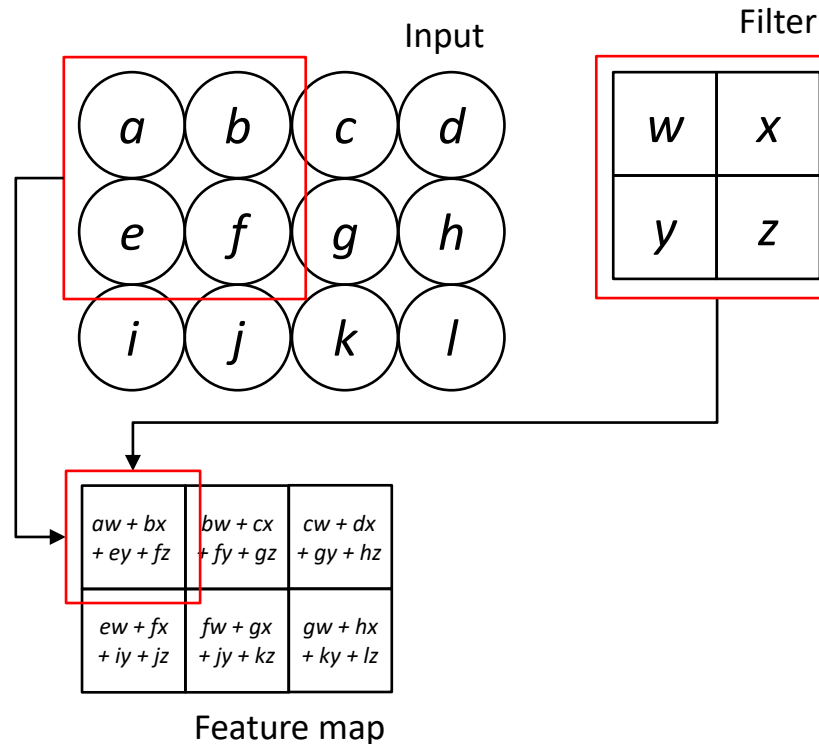
0	0	0
1	1	1
2	2	2

1

2

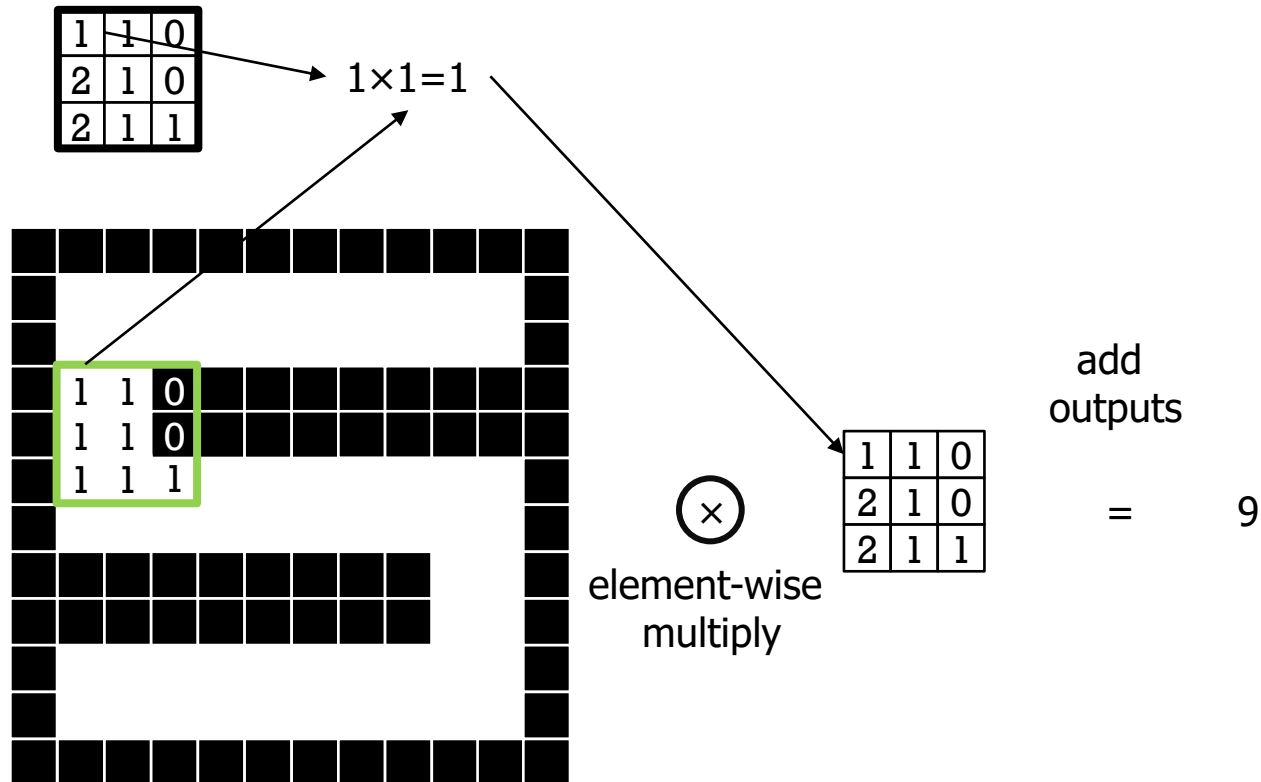
3

Convolution Operation



How do filters/convolution pick up the features?
Apply convolution operation

Convolution Operation



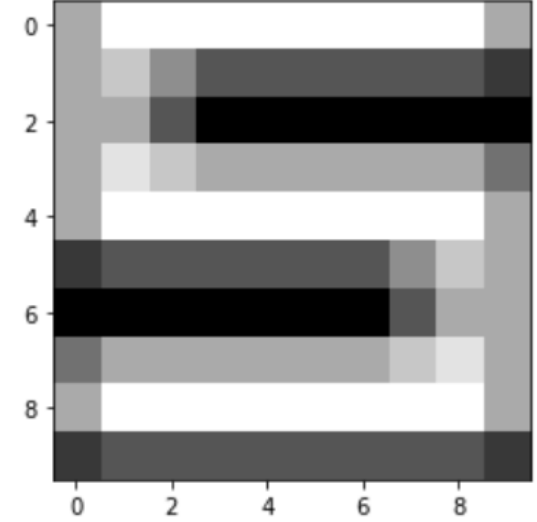
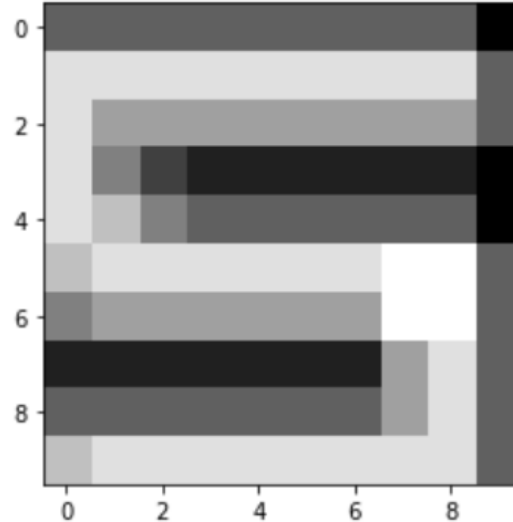
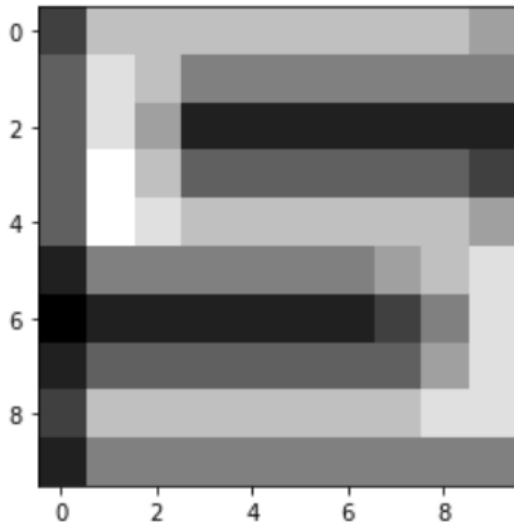
How do filters/convolution pick up the features?
Apply convolution operation

Convolution Operation

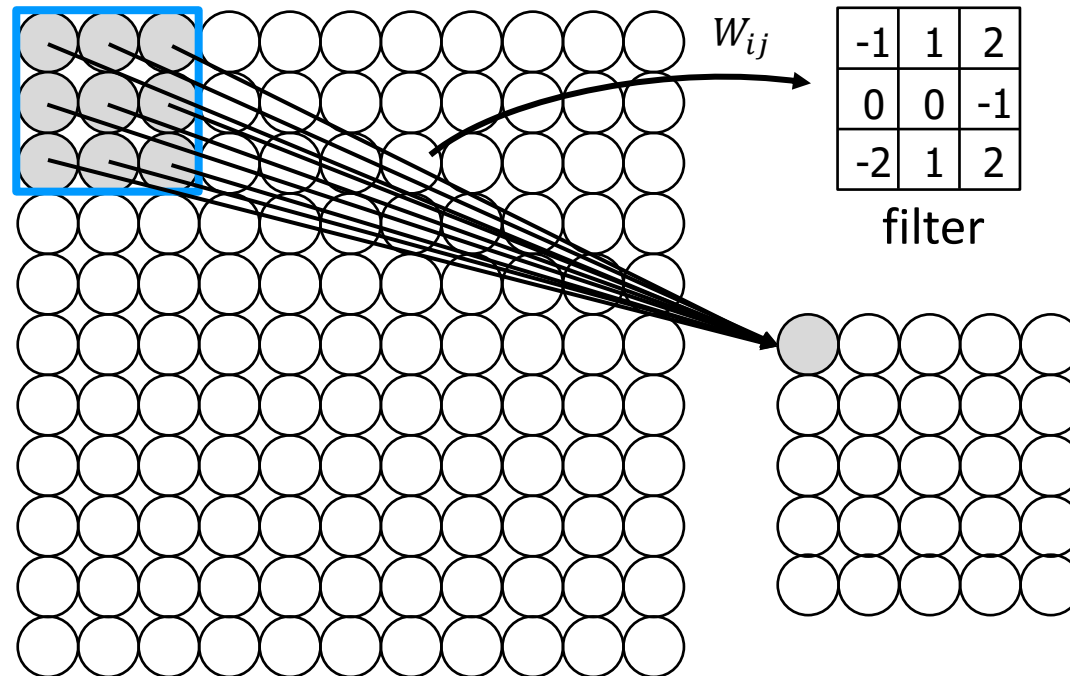
```
[[3. 7. 7. 7. 7. 7. 7. 7. 6.]
 [4. 8. 7. 5. 5. 5. 5. 5. 5.]
 [4. 8. 6. 2. 2. 2. 2. 2. 2.]
 [4. 9. 7. 4. 4. 4. 4. 4. 3.]
 [4. 9. 8. 7. 7. 7. 7. 7. 6.]
 [2. 5. 5. 5. 5. 5. 5. 6. 7. 8.]
 [1. 2. 2. 2. 2. 2. 2. 3. 5. 8.]
 [2. 4. 4. 4. 4. 4. 4. 4. 6. 8.]
 [3. 7. 7. 7. 7. 7. 7. 7. 8. 8.]
 [2. 5. 5. 5. 5. 5. 5. 5. 5. 5.]]
```

```
[[3. 3. 3. 3. 3. 3. 3. 3. 0.]
 [7. 7. 7. 7. 7. 7. 7. 7. 3.]
 [7. 5. 5. 5. 5. 5. 5. 5. 3.]
 [7. 4. 2. 1. 1. 1. 1. 1. 0.]
 [7. 6. 4. 3. 3. 3. 3. 3. 0.]
 [6. 7. 7. 7. 7. 7. 7. 8. 8. 3.]
 [4. 5. 5. 5. 5. 5. 5. 8. 8. 3.]
 [1. 1. 1. 1. 1. 1. 1. 5. 7. 3.]
 [3. 3. 3. 3. 3. 3. 3. 5. 7. 3.]
 [6. 7. 7. 7. 7. 7. 7. 7. 7. 3.]]
```

```
[[6. 9. 9. 9. 9. 9. 9. 9. 6.]
 [6. 7. 5. 3. 3. 3. 3. 3. 2.]
 [6. 6. 3. 0. 0. 0. 0. 0. 0.]
 [6. 8. 7. 6. 6. 6. 6. 6. 4.]
 [6. 9. 9. 9. 9. 9. 9. 9. 6.]
 [2. 3. 3. 3. 3. 3. 3. 5. 7. 6.]
 [0. 0. 0. 0. 0. 0. 0. 3. 6. 6.]
 [4. 6. 6. 6. 6. 6. 6. 7. 8. 6.]
 [6. 9. 9. 9. 9. 9. 9. 9. 6.]
 [2. 3. 3. 3. 3. 3. 3. 3. 3. 2.]]
```

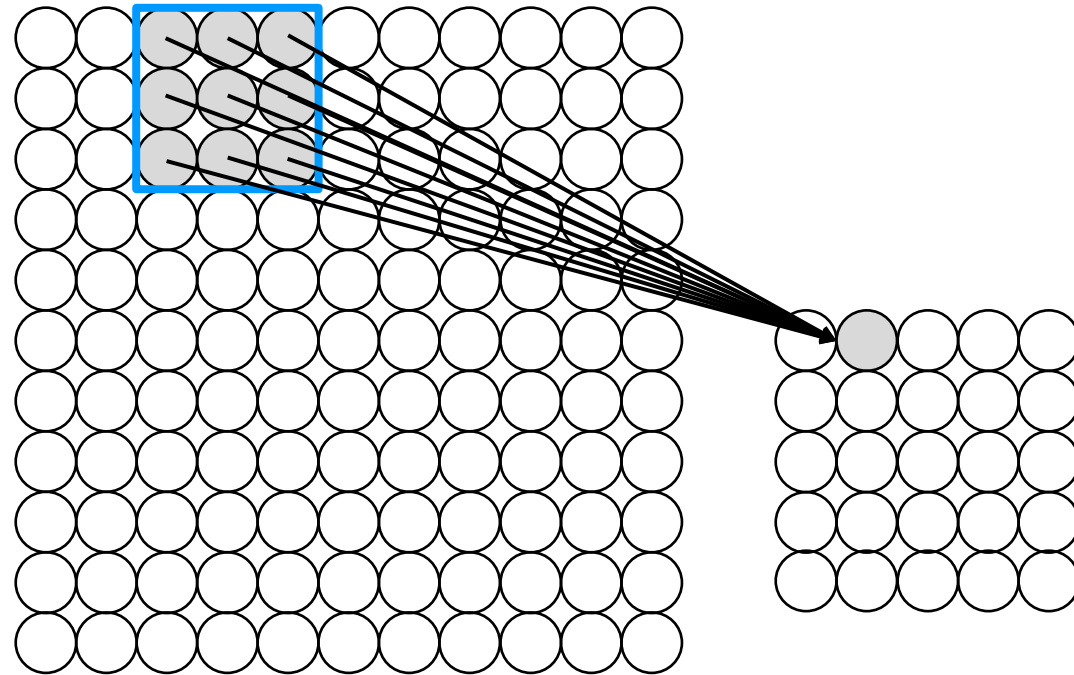


Feature Extraction



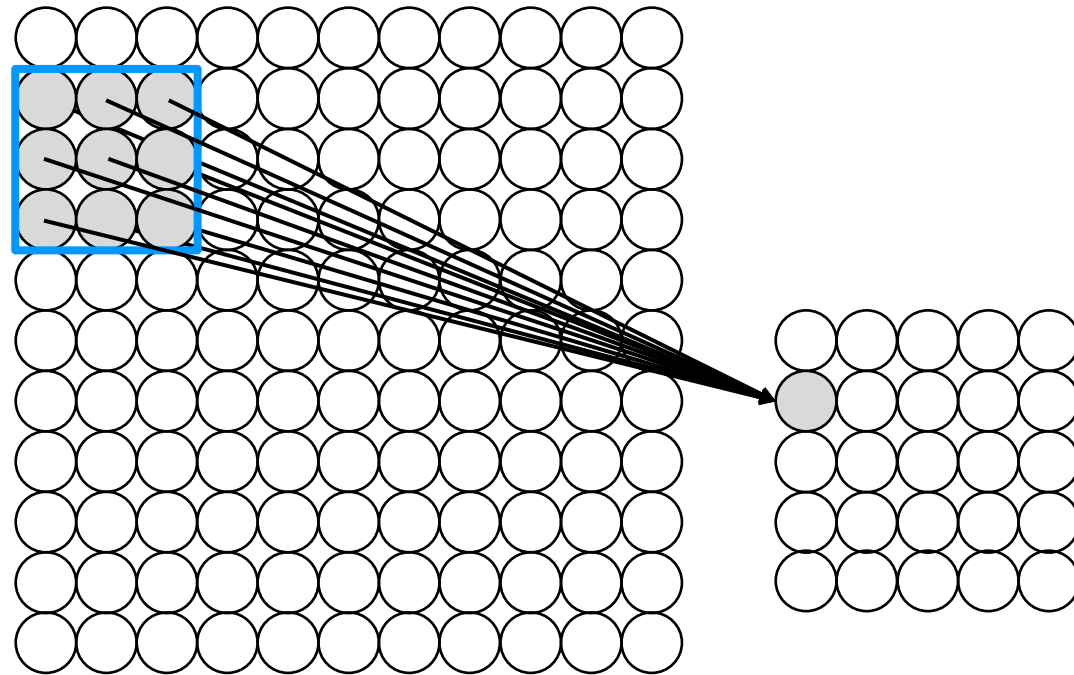
Connect a patch of input neurons to a neuron in hidden layer

Feature Extraction



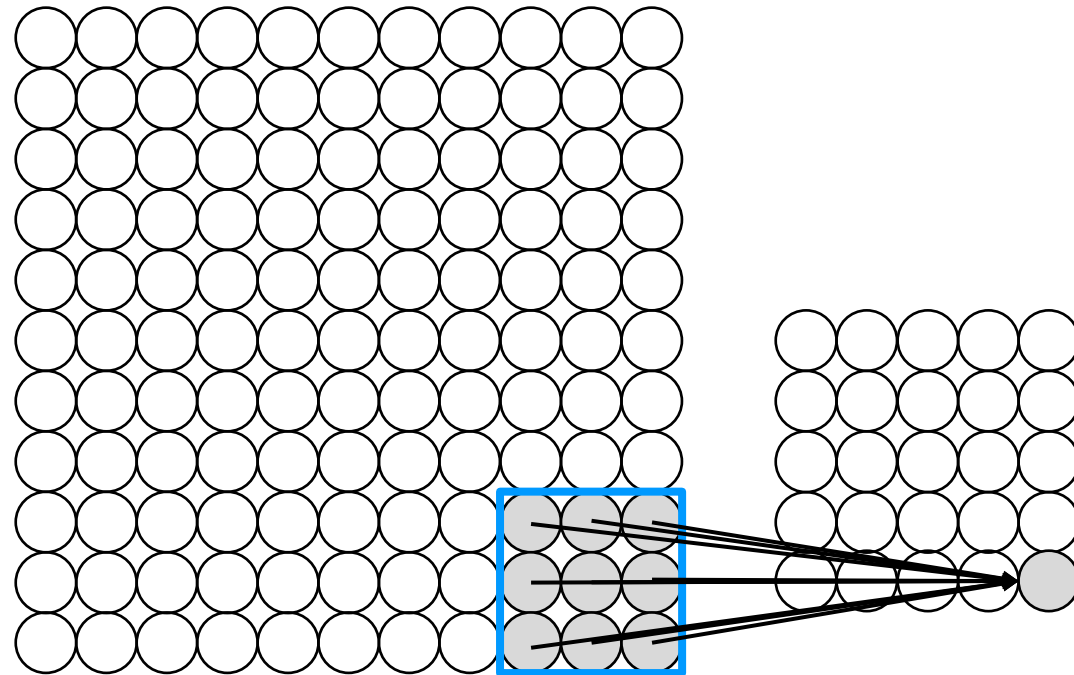
Connect a patch of input neurons to a neuron in hidden layer
Move the filter across the image from top left to bottom right

Feature Extraction



Connect a patch of input neurons to a neuron in hidden layer

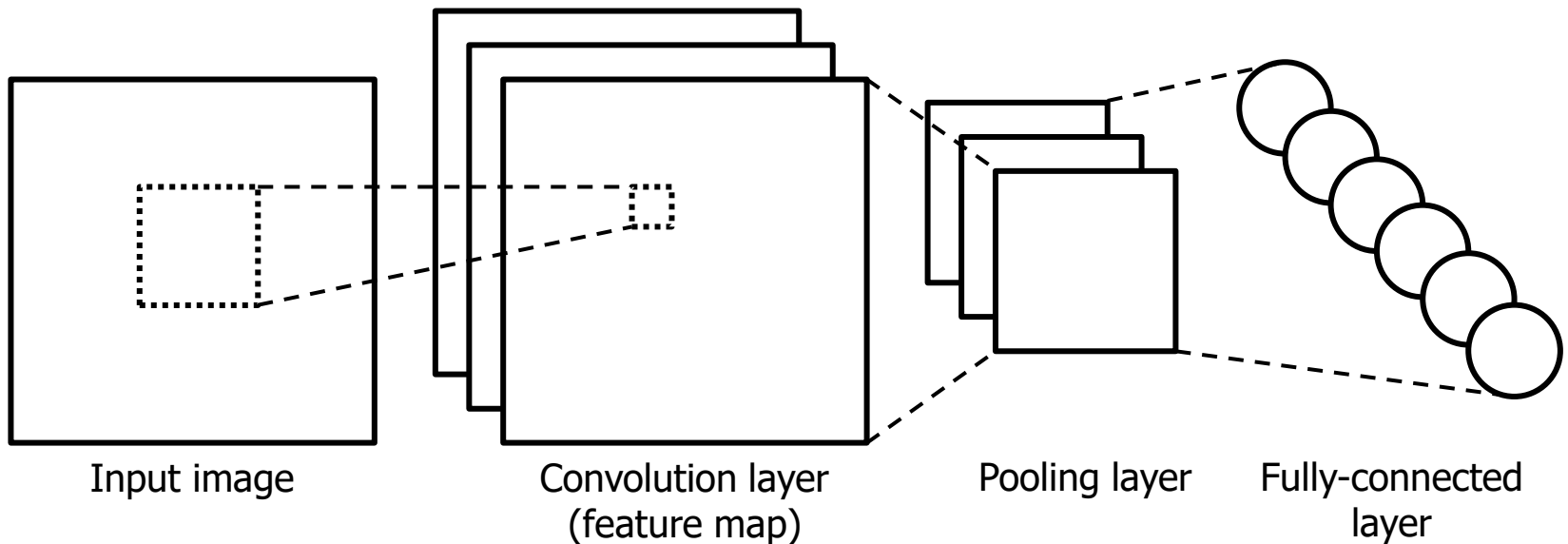
Feature Extraction



Connect a patch of input neurons to a neuron in hidden layer

Convolutional Neural Networks (CNN)

Convolutional Neural Networks

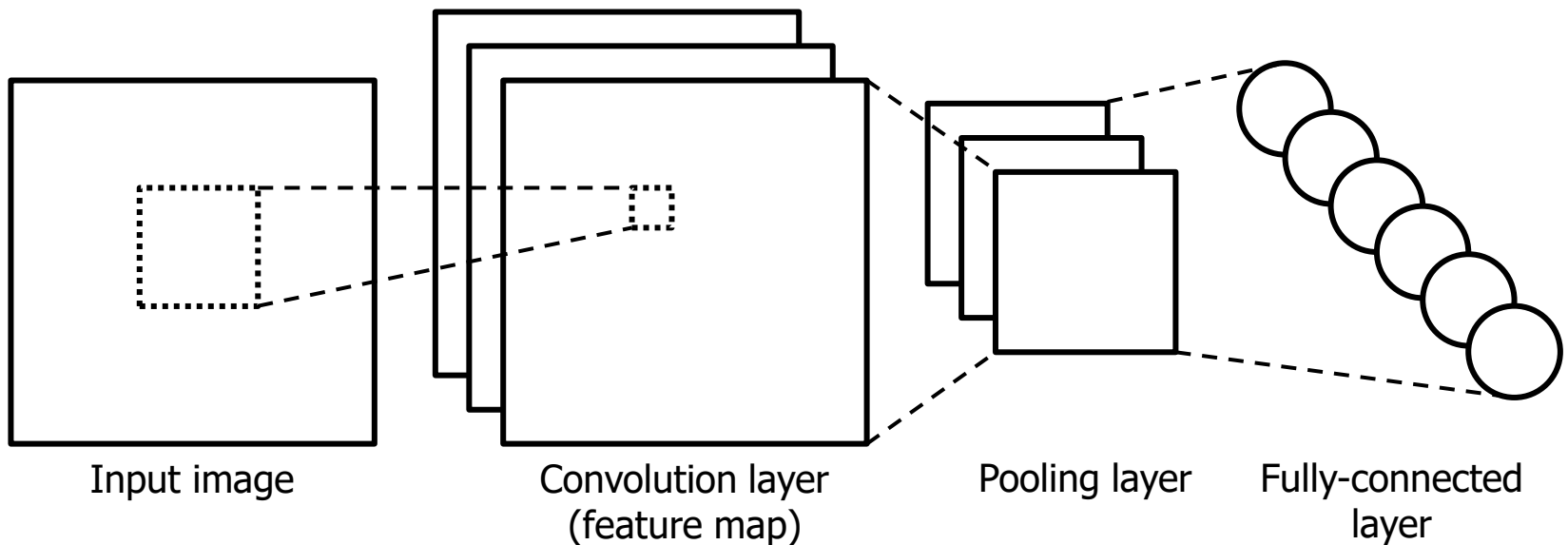


Convolution: apply filters to extract local features

Non-linearity: non-linear activation function e.g. ReLU

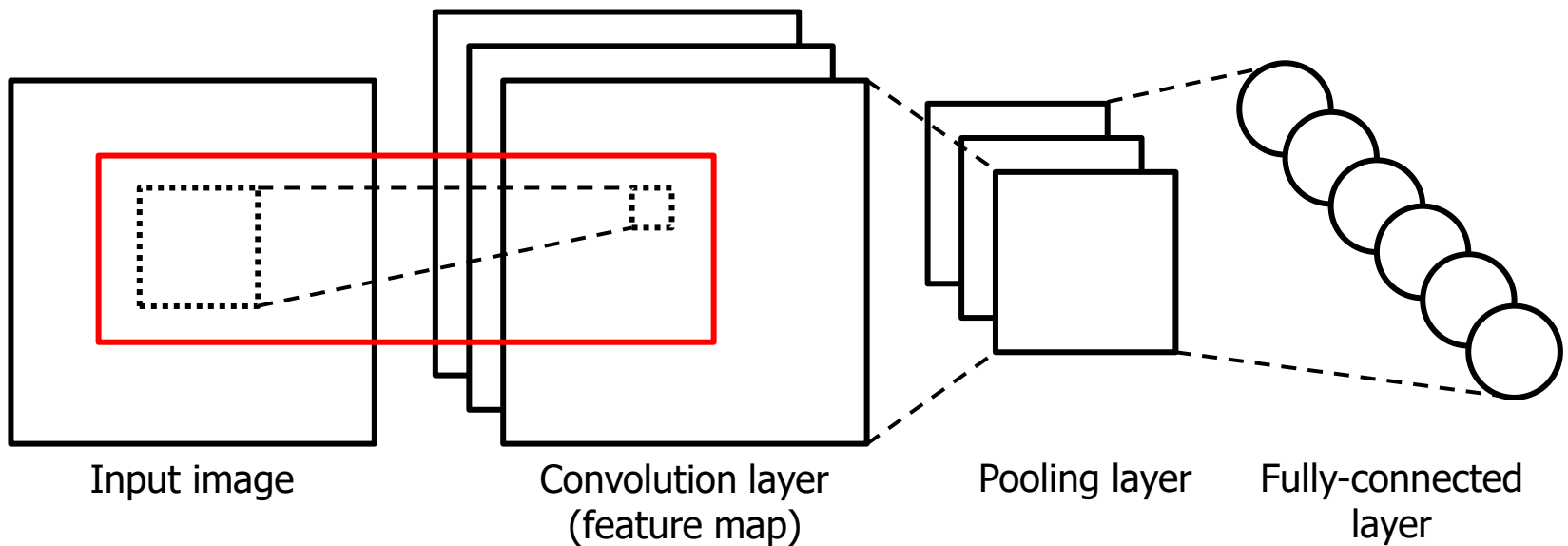
Pooling: downsampling operation on each feature map

Convolutional Neural Networks



Train CNN model with image data
Learn the **weights** of **filters** in convolutional layers
Learn the **weights** of **fully-connected layer**

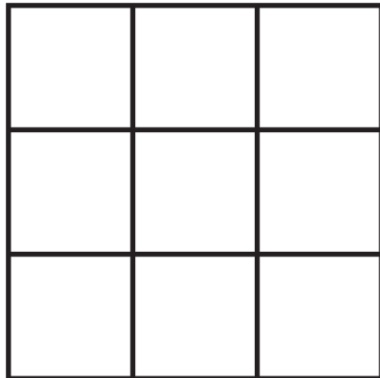
Parameters of Convolutional Layers



Filter (Kernel) Size

- Kernel height and width
- The number of pixels a kernel “sees” at once
- Typically use odd numbers so that there is a “center” pixel
- Kernel does not need to be square

Height: 3, Width: 3



Height: 1, Width: 3



Height: 3, Width: 1



Without Padding

1	2	0	3	1
1	0	0	2	2
2	1	2	1	1
0	0	1	0	0
1	2	1	1	1

input

-1	1	2
1	1	0
-1	-2	0

kernel

-2		

output

There will be an “edge effect” if we apply the filter directly to input data
Pixels near the edge will not be the “center pixels”
The pixels will be processed only once (we are losing information)

With Padding

0	0	0	0	0	0	0
0	1	2	0	3	1	0
0	1	0	0	2	2	0
0	2	1	2	1	1	0
0	0	0	1	0	0	0
0	1	2	1	1	1	0
0	0	0	0	0	0	0

input

-1	1	2
1	1	0
-1	-2	0

kernel

-1				

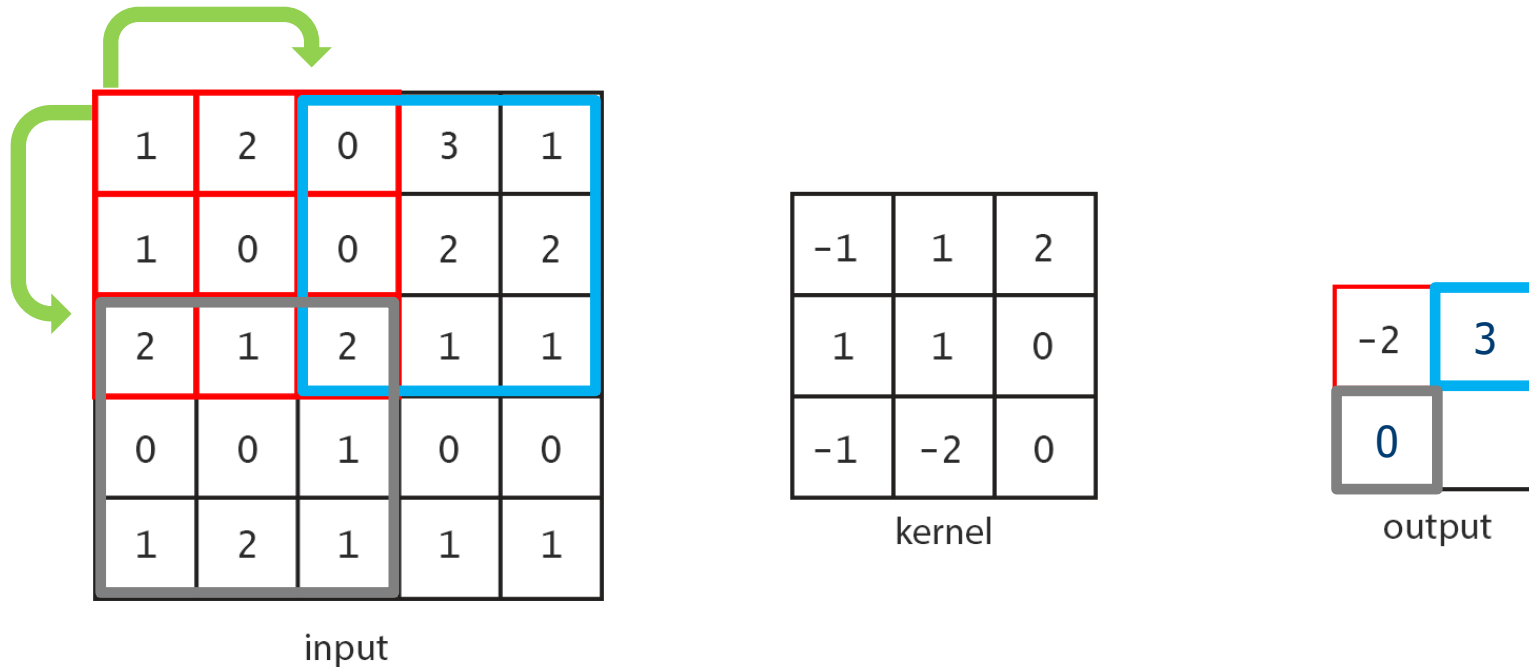
output

Pad extra pixels around the frame

The pixels at the edge will be the “center pixels”

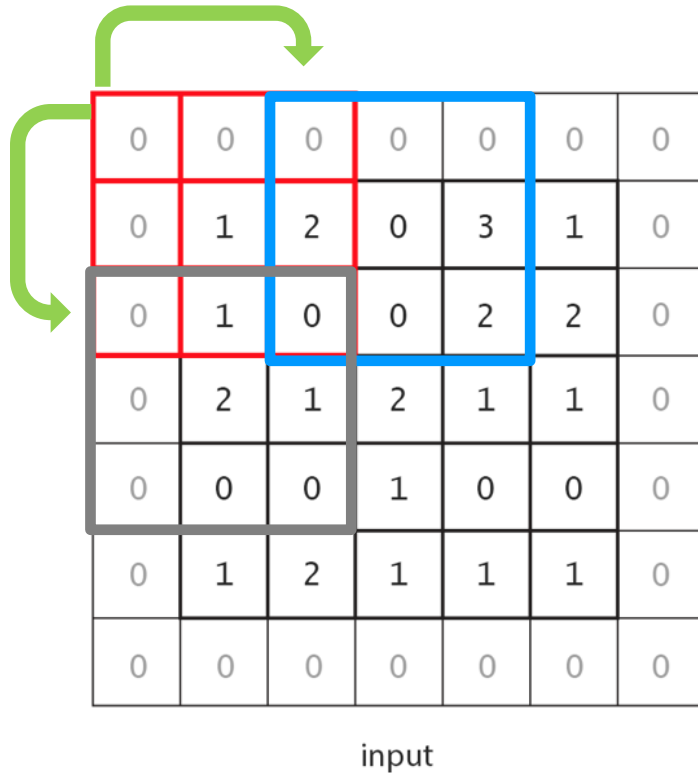
Typically the value of padding is zero (zero-padding)

Stride – No Padding



The "step size" as the kernel moves across the image
Can be different for vertical and horizontal steps
When stride is greater than 1, it scales down the output dimension

Stride – With Padding



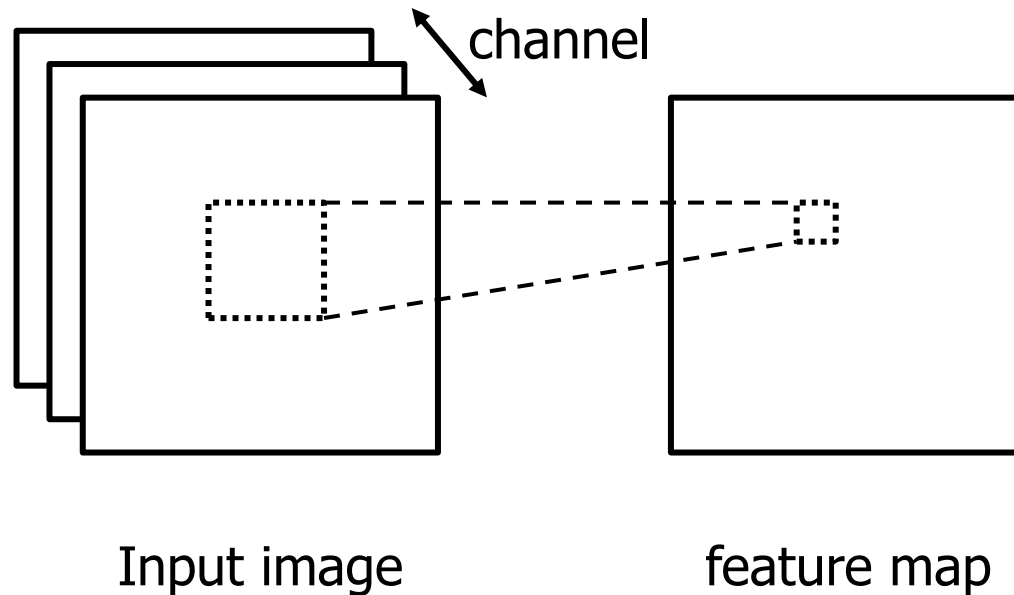
-1	1	2
1	1	0
-1	-2	0

kernel

-1	2	
3		

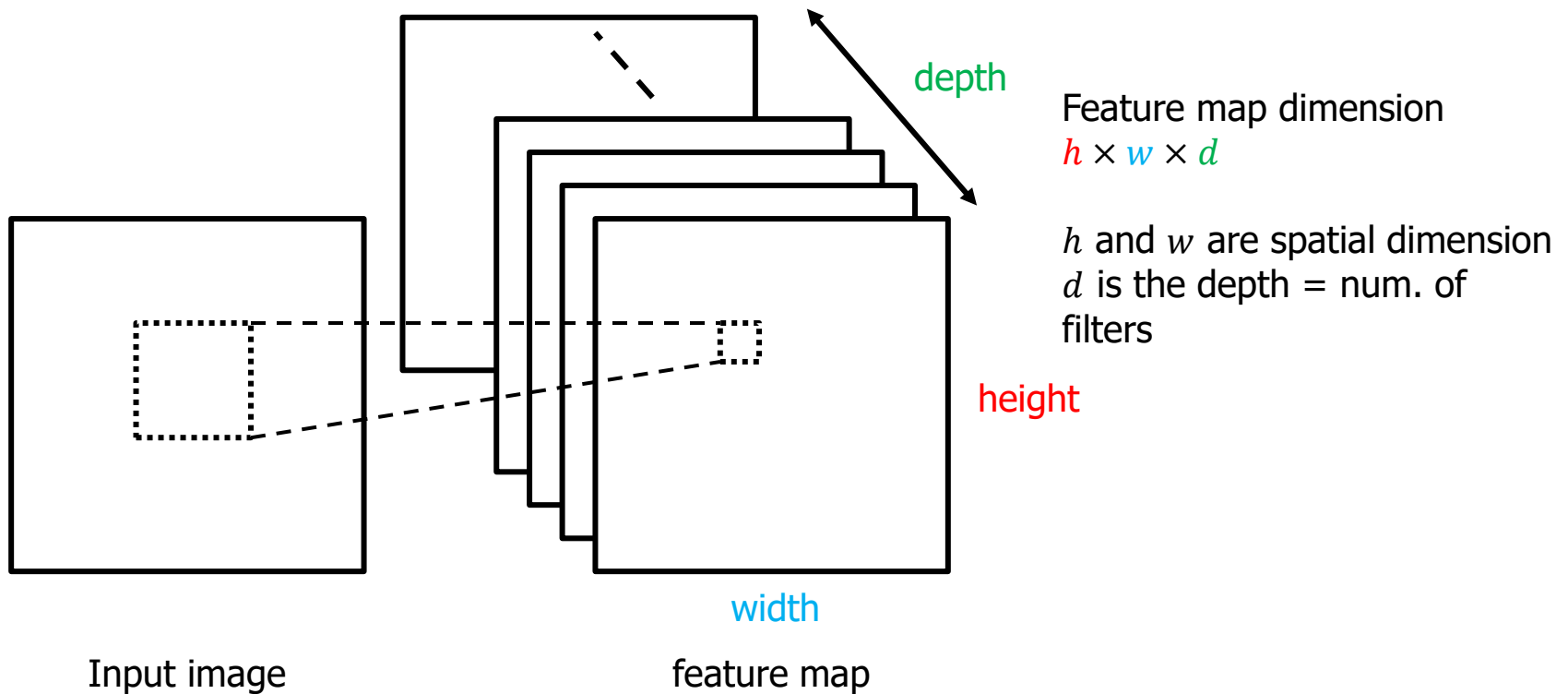
output

Depth – Input Channel

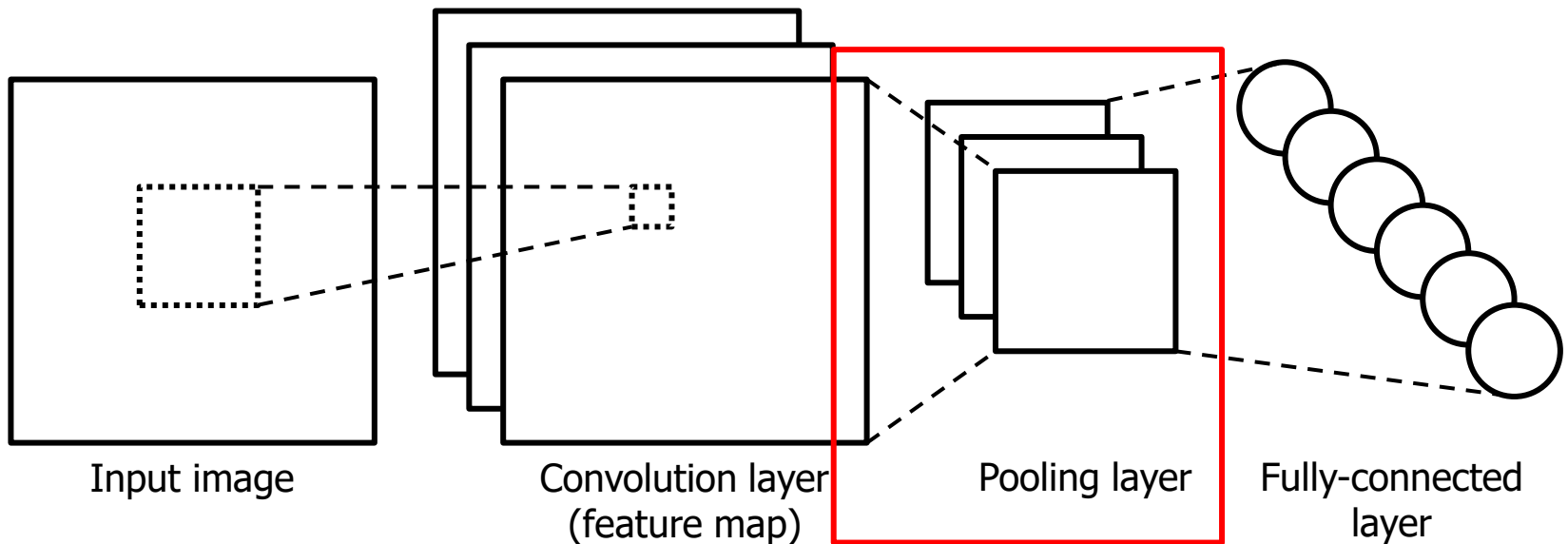


The pixel may be associated with more than one value
The values referred to as “channel” e.g. RGB image (3 channels)
The number of channels is referred to as the “depth”
Each channel has a filter (kernel)

Depth – Output Channel (Volume)



Pooling Layers



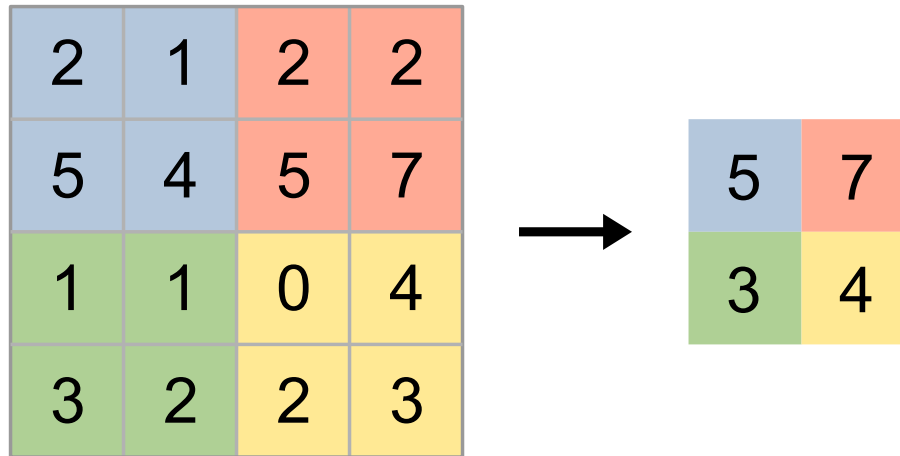
Reduce the feature map size by mapping a patch of pixels to a single value
Does not have parameters
There are different types of pooling operations

Pooling Layers

- Reduce the feature map size
- Parameters: pooling size
- There are different types of pooling operations

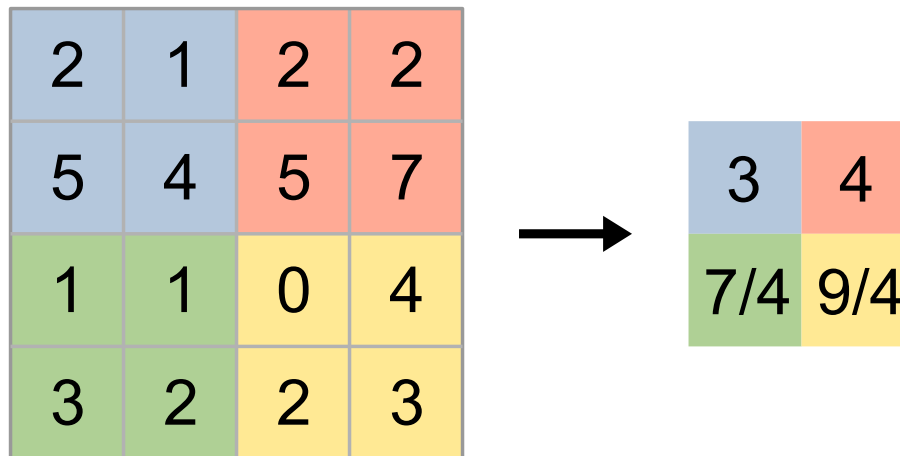
Pooling Layers – Max-pool

- For each distinct patch, represent it by the maximum
- 2x2 max-pool



Pooling Layers – Average-pool

- For each distinct patch, represent it by the average
- 2x2 average-pool



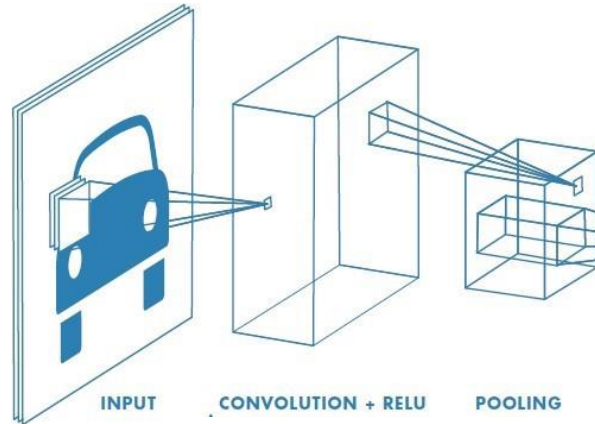
CNN for Classification

- Learn features in input image through **convolution**
- Introduce **non-linearity** through activation function (real world data is non-linear)
- Reduce dimensionality with **pooling**

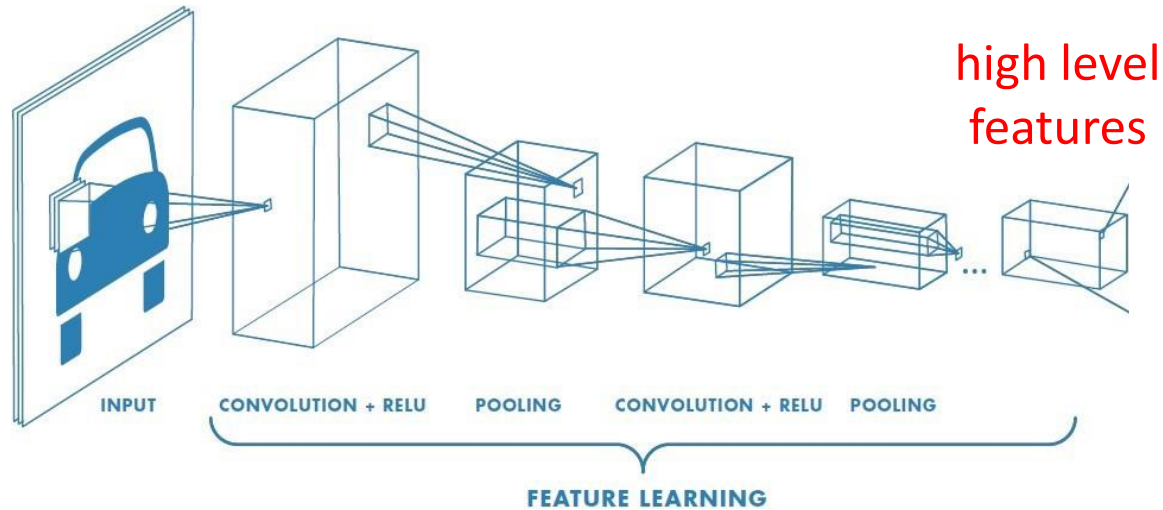
CNN for Classification



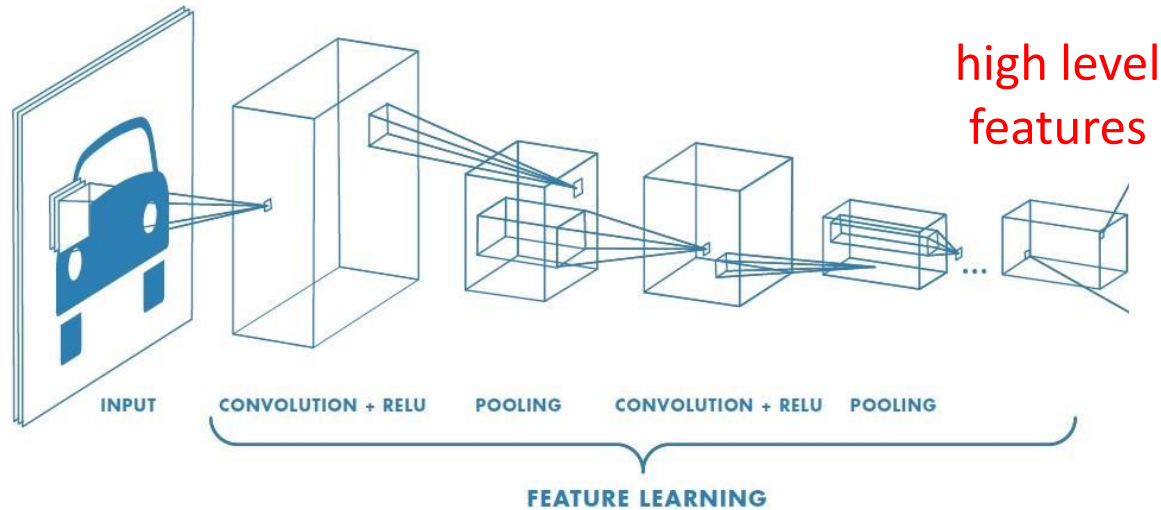
CNN for Classification



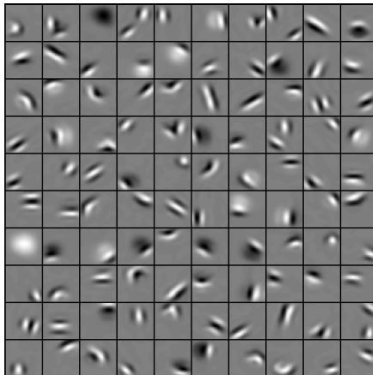
CNN for Classification



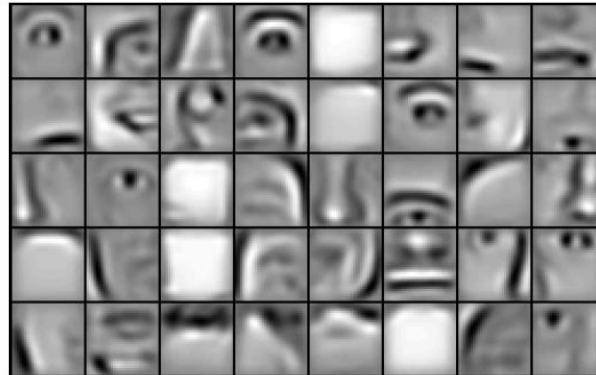
CNN for Classification



Learn features in **stages** or **hierarchical** manner



Low Level Features
Lines & Dots

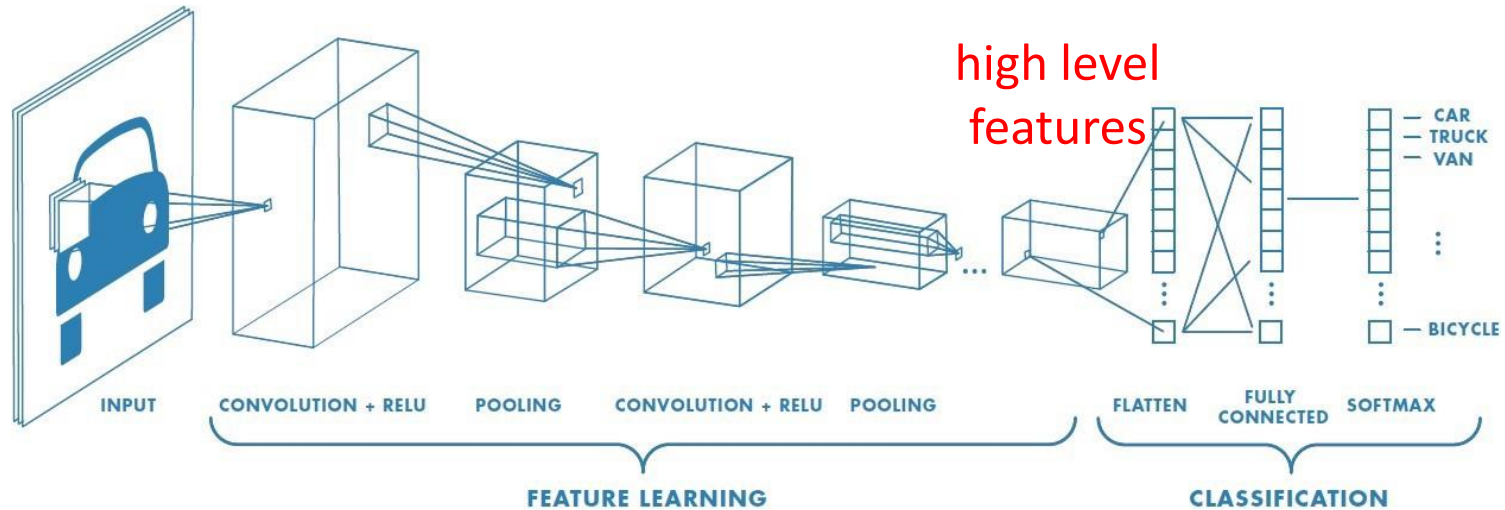


Mid Level Features
Eyes, Nose & Mouth



High Level Features
Facial Structure

CNN for Classification



The high level features are fed to the classification part for classifying the image into the corresponding classes.

End