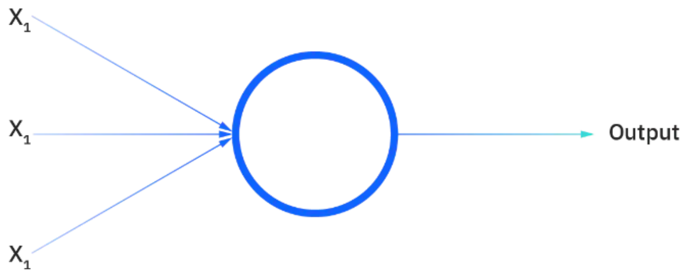
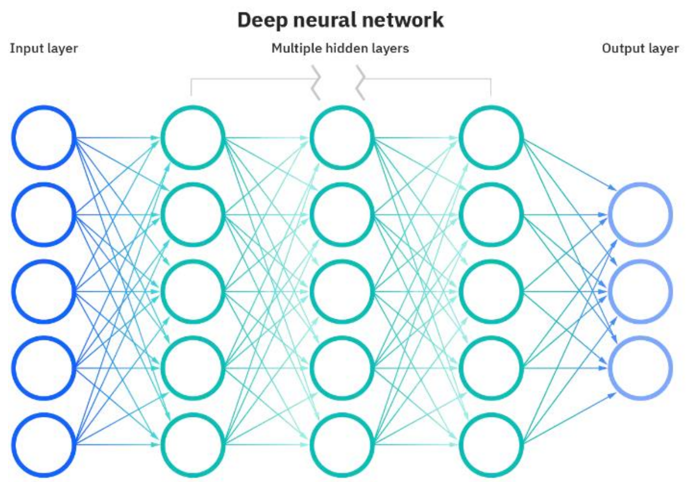




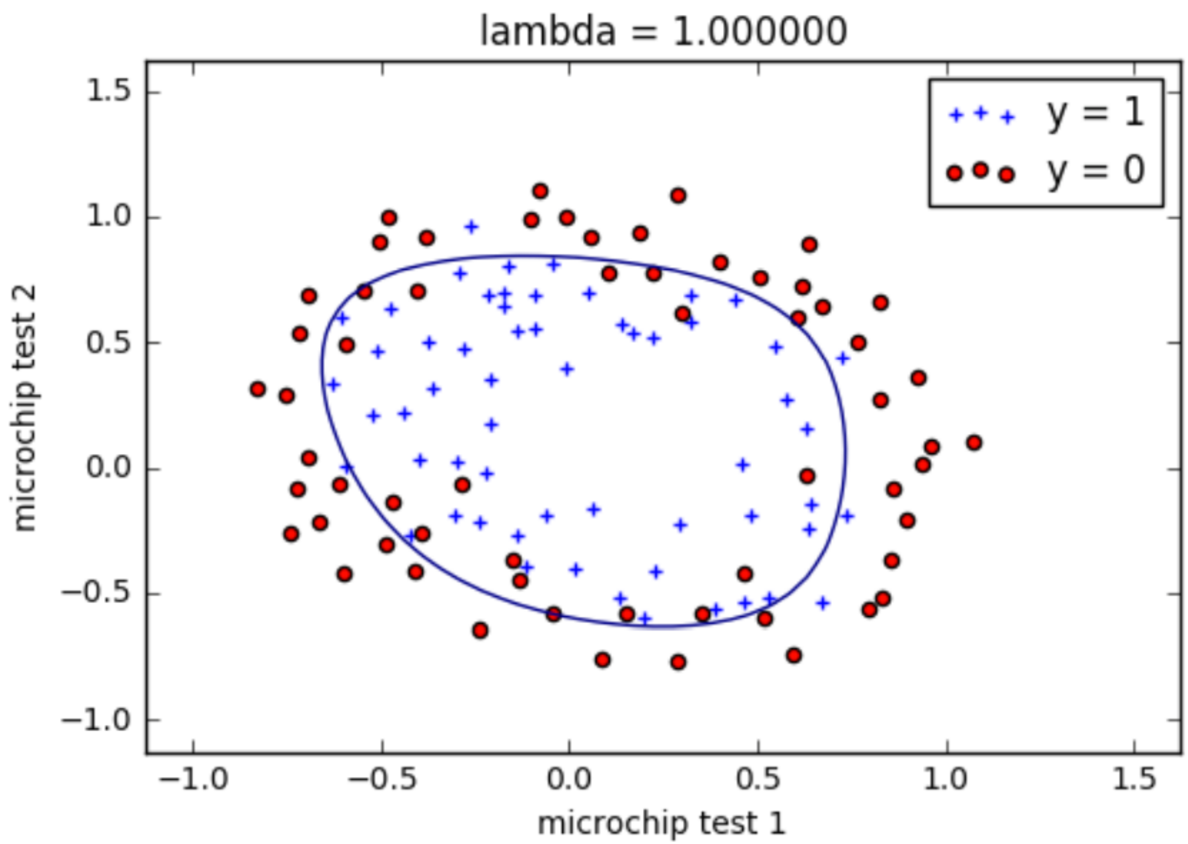
Lecture 19

Neural Network

Credit: Khola Naseem

Hypothesis

- Non Linear Hypothesis



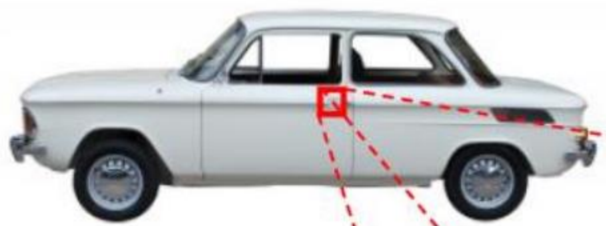
$$\begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \\ x_1^3 \\ \vdots \\ x_1 x_2^5 \\ x_2^6 \end{bmatrix}$$

Credit: Khola Naseem

Motivation

- however in most cases, we don’t know what features to choose, either because the prediction task is non-intuitive or the input vector is of very high dimension (e.g image)

You see this:



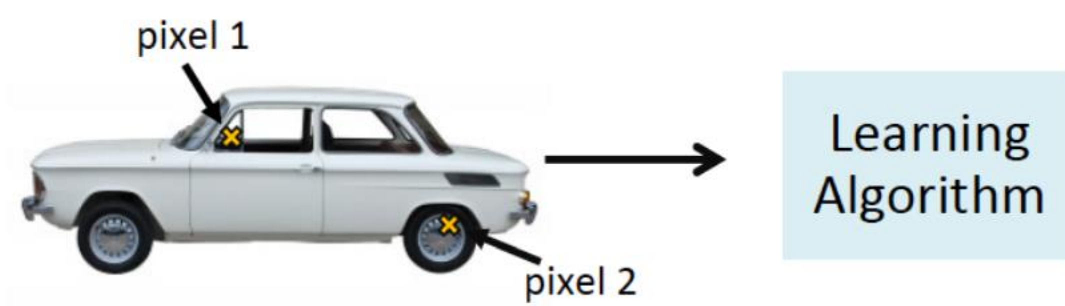
But the camera sees this:

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

Credit: Khola Naseem

Motivation

- however in most cases, we don't know what features to choose, either because the prediction task is non-intuitive or the input vector is of very high dimension (e.g image)



50 x 50 pixel images→ 2500 pixels

$\boxed{n = 2500}$ (7500 if RGB)

0-255

$$x = \begin{bmatrix} \text{pixel 1 intensity} \\ \text{pixel 2 intensity} \\ \vdots \\ \text{pixel 2500 intensity} \end{bmatrix}$$

Credit: Khola Naseem

Motivation

- In the spirit of machine learning, we'd like to automate things as much as possible. In this context, it means creating algorithms that can take whatever crude features we have and turn them into refined predictions, thereby shifting the burden feature extraction and moving it to learning
- Neural networks allow one to automatically learn the features of a linear classifier which are geared towards the desired task, rather than specifying them all by hand

Credit: Khola Naseem

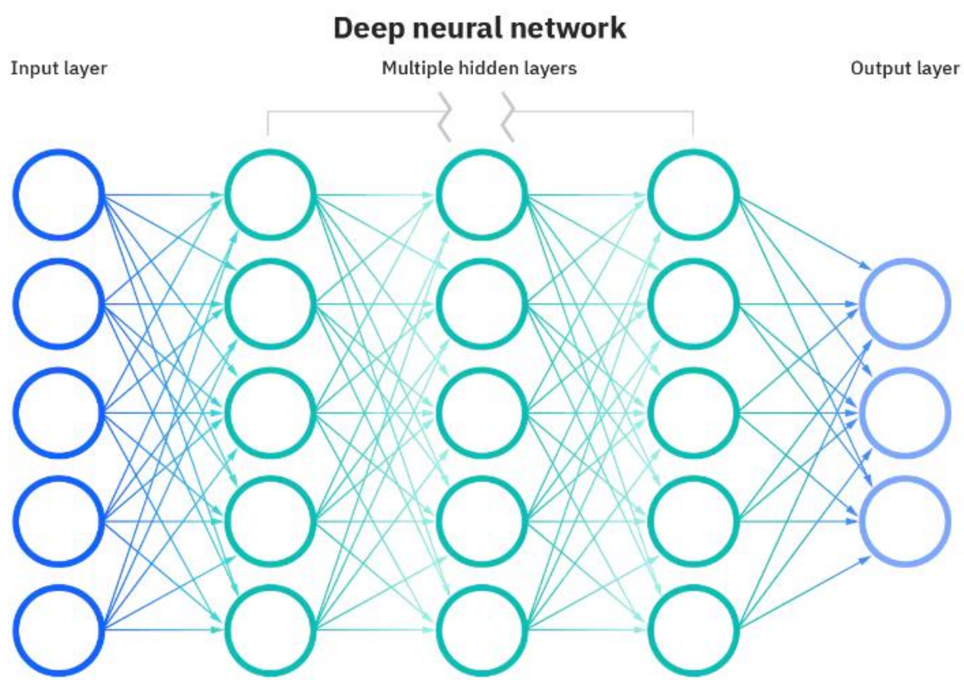
Neural Network

- Neural networks reflect the behavior of the human brain, allowing computer programs to recognize patterns and solve common problems in the fields of AI, machine learning, and deep learning
- Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms.
- In this sense, neural networks refer to systems of neurons, or artificial in nature.
- Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria.

Credit: Khola Naseem

Neural Network

- Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer.



Credit: Khola Naseem

Neural Network

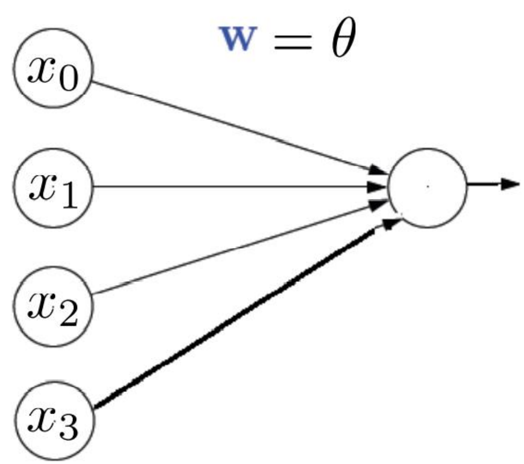
- Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.
- Neural networks rely on training data to learn and improve their accuracy over time. However, once these learning algorithms are fine-tuned for accuracy,

Neural Network

- Perceptrons

- The perceptron is the oldest neural network,It has a single neuron and is the simplest form of a neural network:

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$



$$h(x) = g(\theta^T x)$$

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

Credit: Khola Naseem

Neural Network

- Think of each individual node as its own linear regression model,
- composed of input data, weights, a bias (or threshold), and an output. The formula would look something like this:

$$\sum_{i=1}^{m} w_i x_i + bias = w_1 x_1 + w_2 x_2 + w_3 x_3 + bias$$

$$output = f(x) = \begin{cases} 1 \text{ if } \sum w_1 x_1 + b \geq 0 \\ 0 \text{ if } \sum w_1 x_1 + b < 0 \end{cases}$$

Credit: Khola Naseem

Neural Network

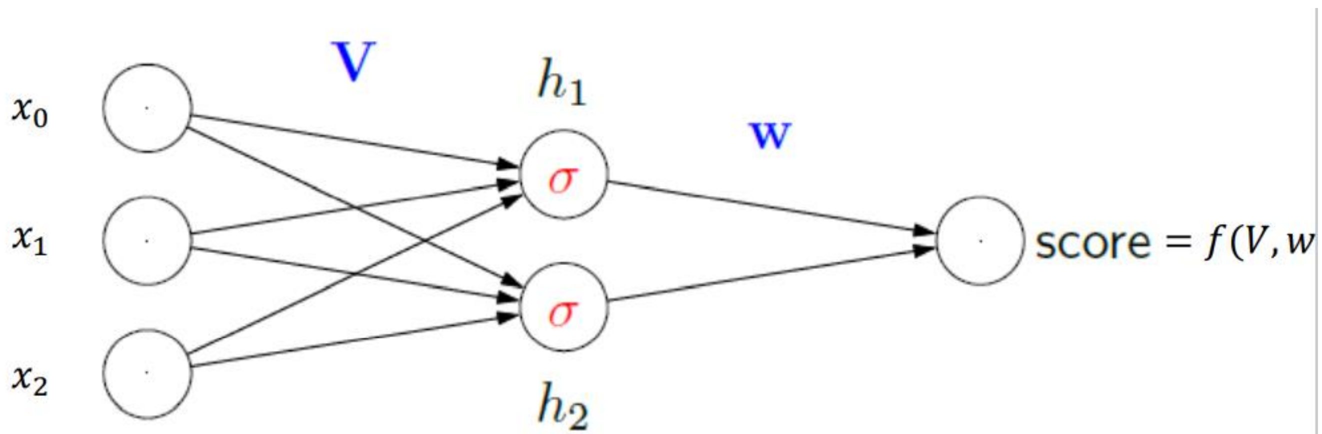
- Once an input layer is determined, weights are assigned. These weights help determine the importance of any given variable, with larger ones contributing more significantly to the output compared to other inputs.

- All inputs are then multiplied by their respective weights and then summed. Afterward, the output is passed through an activation function, which determines the output. If that output exceeds a given threshold, it "fires" (or activates) the node, passing data to the next layer in the network. This results in the output of one node becoming in the input of the next node. This process of passing data from one layer to the next layer defines this neural network as a feedforward network.

Credit: Khola Naseem

Neural Network

- Once an input layer is determined, weights are assigned. These
  - weights help determine the importance of any given variable, with larger ones contributing more significantly to the output compared to other inputs.



Credit: Khola Naseem

Neural Network

- Optimization problem:

$$\min_{\mathbf{V},\mathbf{w}} \text{TrainLoss}(\mathbf{V},\mathbf{w})$$

$$\text{TrainLoss}(\mathbf{V},\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} \text{Loss}(x,y,\mathbf{V},\mathbf{w})$$

$$\text{Loss}(x,y,\mathbf{V},\mathbf{w}) = (y - f_{\mathbf{V},\mathbf{w}}(x))^2$$

.

Goal: compute gradient

$$\nabla_{\mathbf{V},\mathbf{w}} \text{TrainLoss}(\mathbf{V},\mathbf{w})$$

Credit: Khola Naseem

Neural Network

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Loss(x, y, \theta)$$

Gradient Descent:

$$\theta = \theta - \alpha \nabla_{\theta} J(x, y, \theta)$$

Stochastic Gradient Descent:

For each training data (x, y)

$$\theta = \theta - \alpha \nabla_{\theta} Loss(x, y, \theta)$$

Credit: Khola Naseem

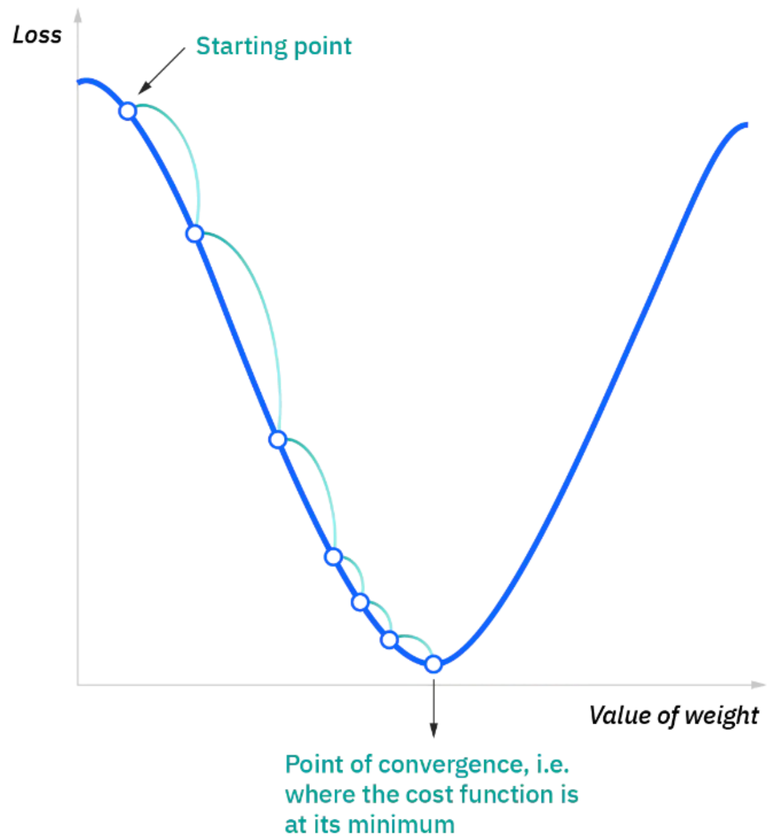
Neural Network

- The goal is to minimize our cost function to ensure correctness of fit for any given observation.
- As the model adjusts its weights and bias, it uses the cost function to reach the point of convergence, or the global minimum. The process in which the algorithm adjusts its weights is through gradient descent, allowing the model to determine the direction to take to reduce errors (or minimize the cost function). With each training example, the parameters of the model adjust to gradually converge at the minimum.

Credit: Khola Naseem

Neural Network

- The goal is to minimize our cost function to ensure correctness of fit for any given observation.



Credit: Khola Naseem

Neural Network

- Types of neural networks:
  - Perceptron
  - multi-layer perceptrons (MLPs)
  - Convolutional neural networks (CNNs)
    - they're usually utilized for image recognition, pattern recognition, and/or computer vision.
  - Recurrent neural networks (RNNs)
    - These learning algorithms are primarily leveraged when using time-series data to make predictions about future outcomes, such as stock market predictions or sales forecasting.

Neural Network

➢Pros and cons

Pros

- Can often work more efficiently and for longer than humans
- Can be programmed to learn from prior outcomes to strive to make smarter future calculations
- Often leverage online services that reduce (but do not eliminate) systematic risk
- Are continually being expanded in new fields with more difficult problems

Cons

- Still rely on hardware that may require labor and expertise to maintain
- May take long periods of time to develop the code and algorithms
- May be difficult to assess errors or adaptions to the assumptions if the system is self-learning but lacks transparency
- Usually report an estimated range or estimated amount that may not actualize

Credit: Khola Naseem

Neural Network

Application of Neural Networks

- Neural networks are broadly used, with applications for financial operations, trading, business analytics, business applications such as forecasting and marketing research solutions, fraud detection, and risk assessment,
- solving complex signal processing or pattern recognition problems. Examples of significant commercial applications
- since 2000 include handwriting recognition for check processing, speech-to-text transcription, oil-exploration data analysis, weather prediction and facial recognition.

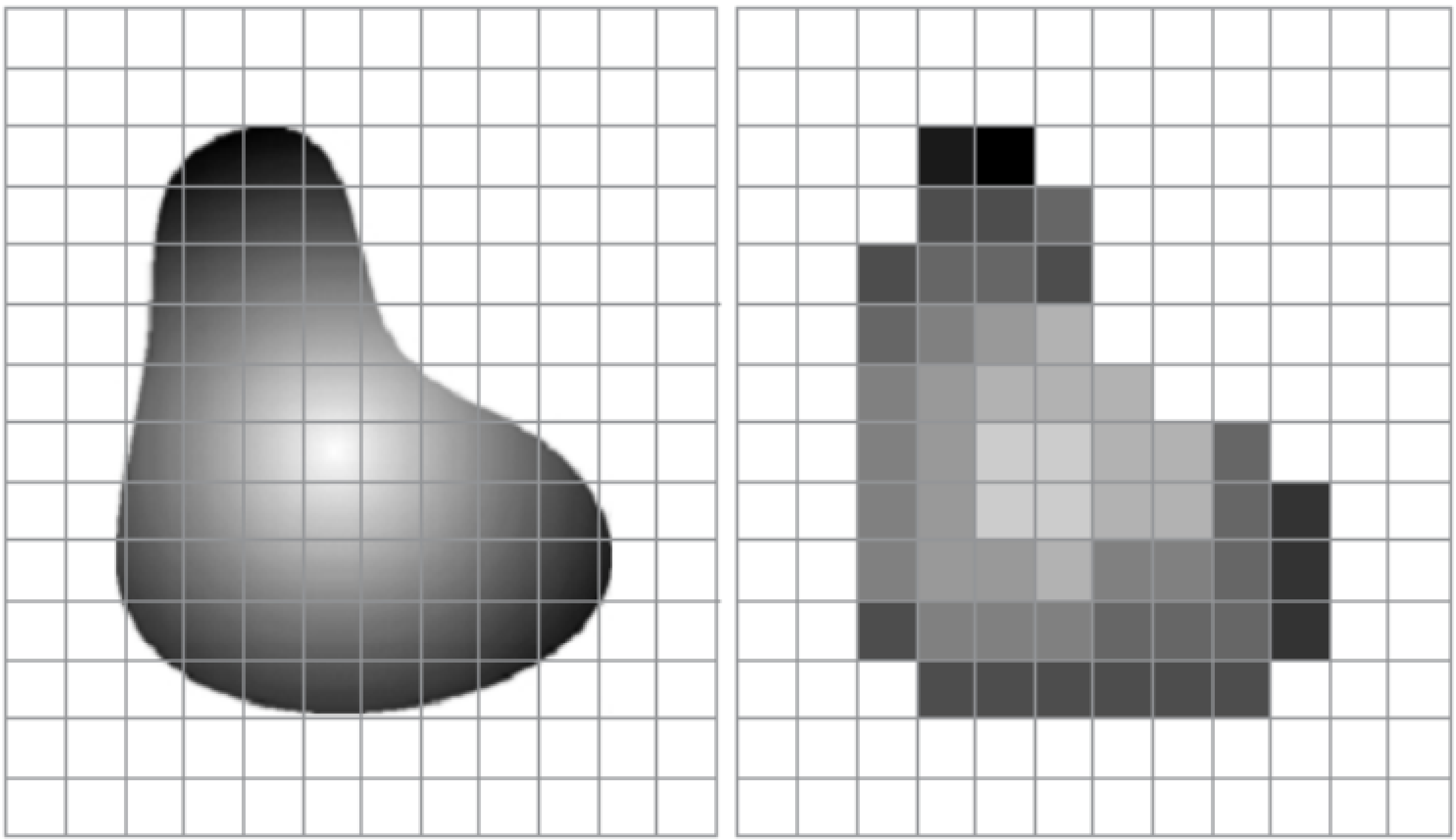
Credit: Khola Naseem

Visual Object Recognition

Credit: Khola Naseem

Image Sensing

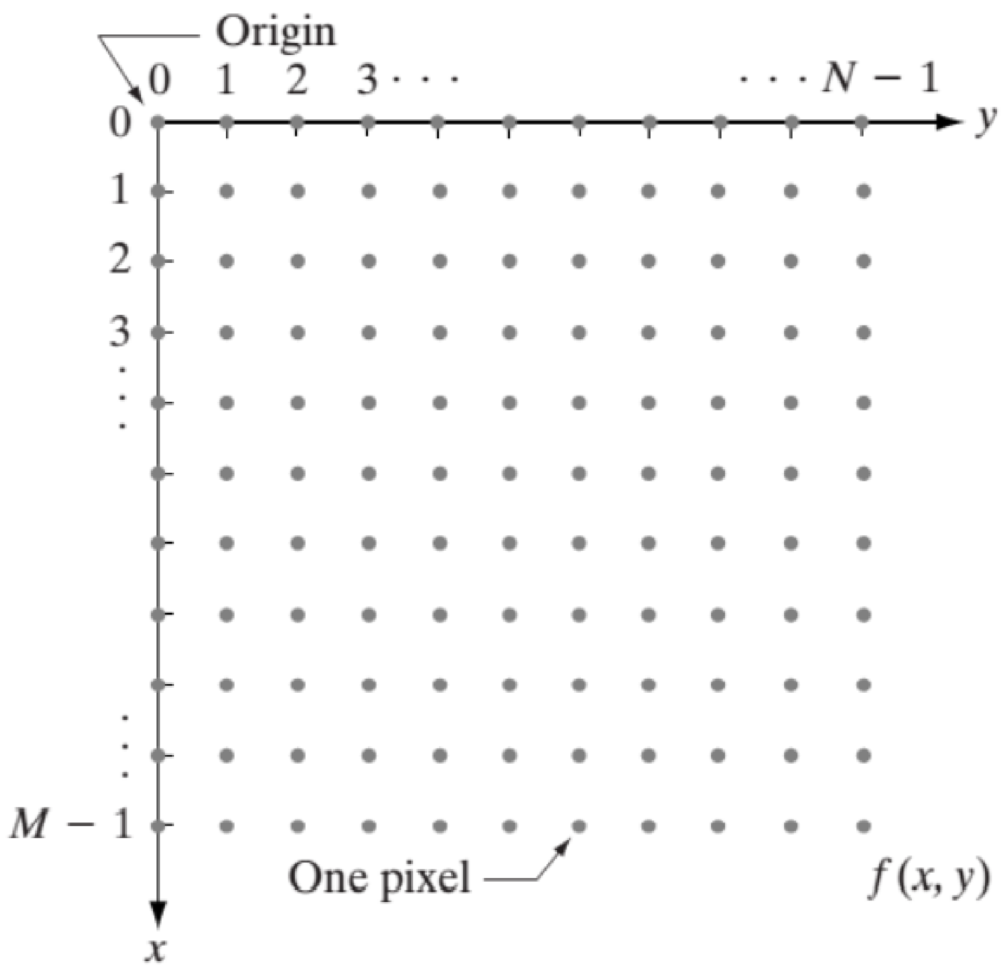
- Image Sensing: Continuous Image Projected onto a Array



Credit: Khola Naseem

Representing Image as a Matrix

- Image Sensing: Continuous Image Projected onto a Array



Credit: Khola Naseem

Representing Image as a Matrix

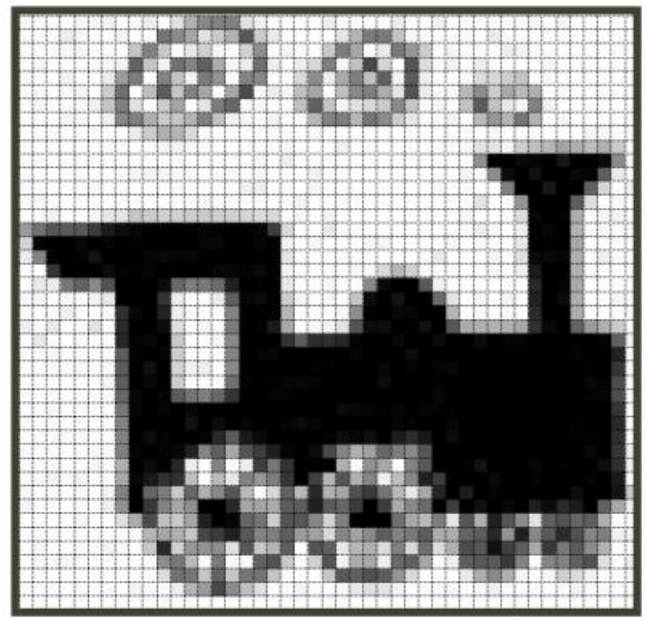
$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix}$$

Credit: Khola Naseem

image

- Computer Vision – Make Sense of Numbers

$$
\begin{matrix}
255 & 255 & 240 & \cdots & 255 \\
255 & 248 & 232 & \cdots & 255 \\
252 & 247 & 238 & \cdots & 239 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
255 & 255 & 255 & \cdots & 255
\end{matrix}
$$



Credit: Khola Naseem

image

- Visual Recognition
- Design algorithms that are capable of
  - Classifying images or videos
  - Detect and localize image
  - Classify human activity and events
- Why is this challenging?

Credit: Khola Naseem

Challenges

- How many Object Categories are there?

Credit: Khola Naseem

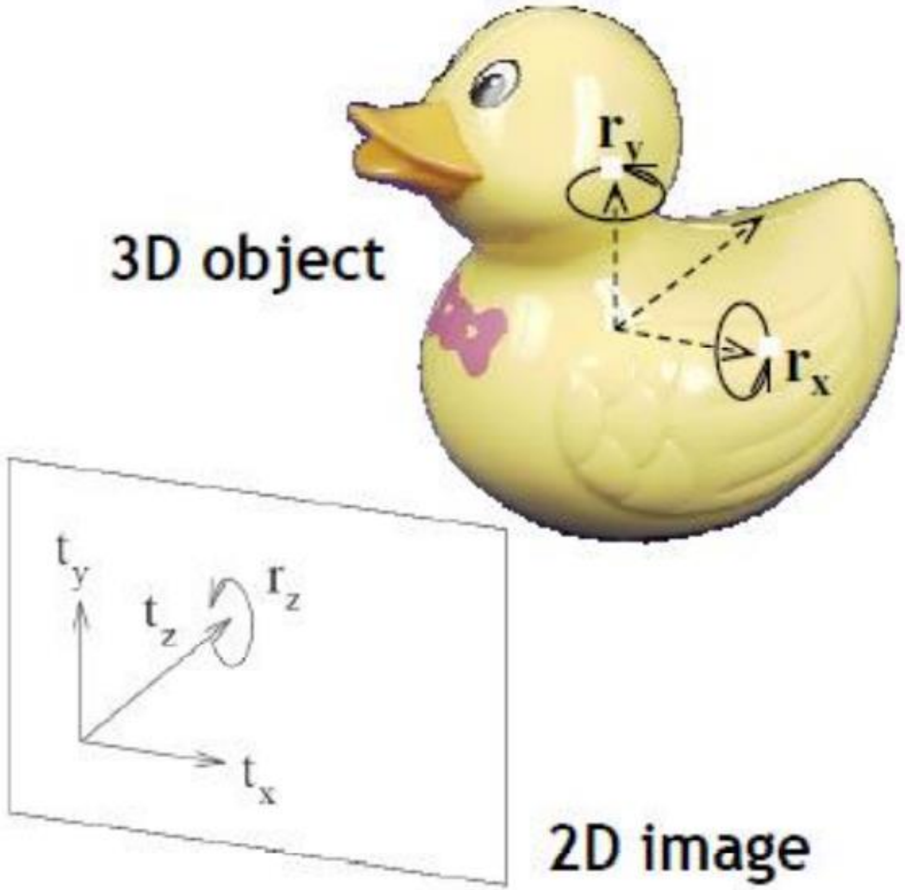
Challenges

- Challenges – Shape and Appearance Variations?

Credit: Khola Naseem

Challenges

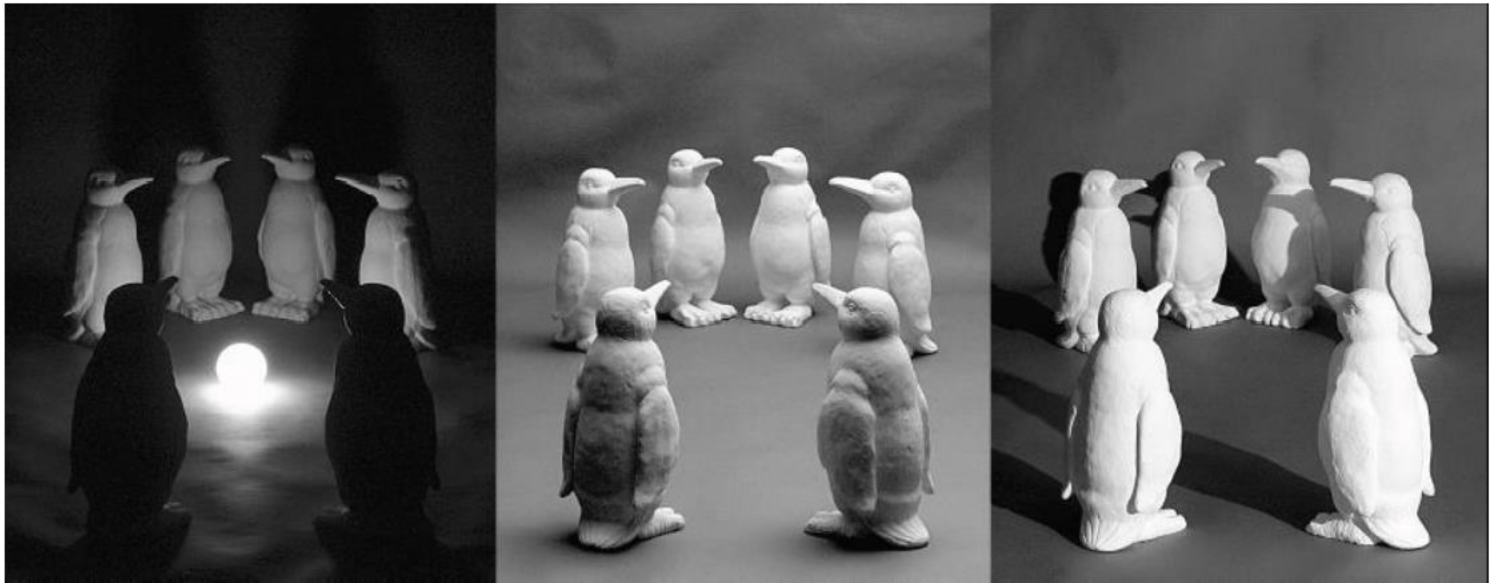
- Challenges – Viewpoint Variations



Credit: Khola Naseem

Challenges

- Challenges – Illumination



Credit: Khola Naseem

Challenges

- Challenges – Background Clutter

Credit: Khola Naseem

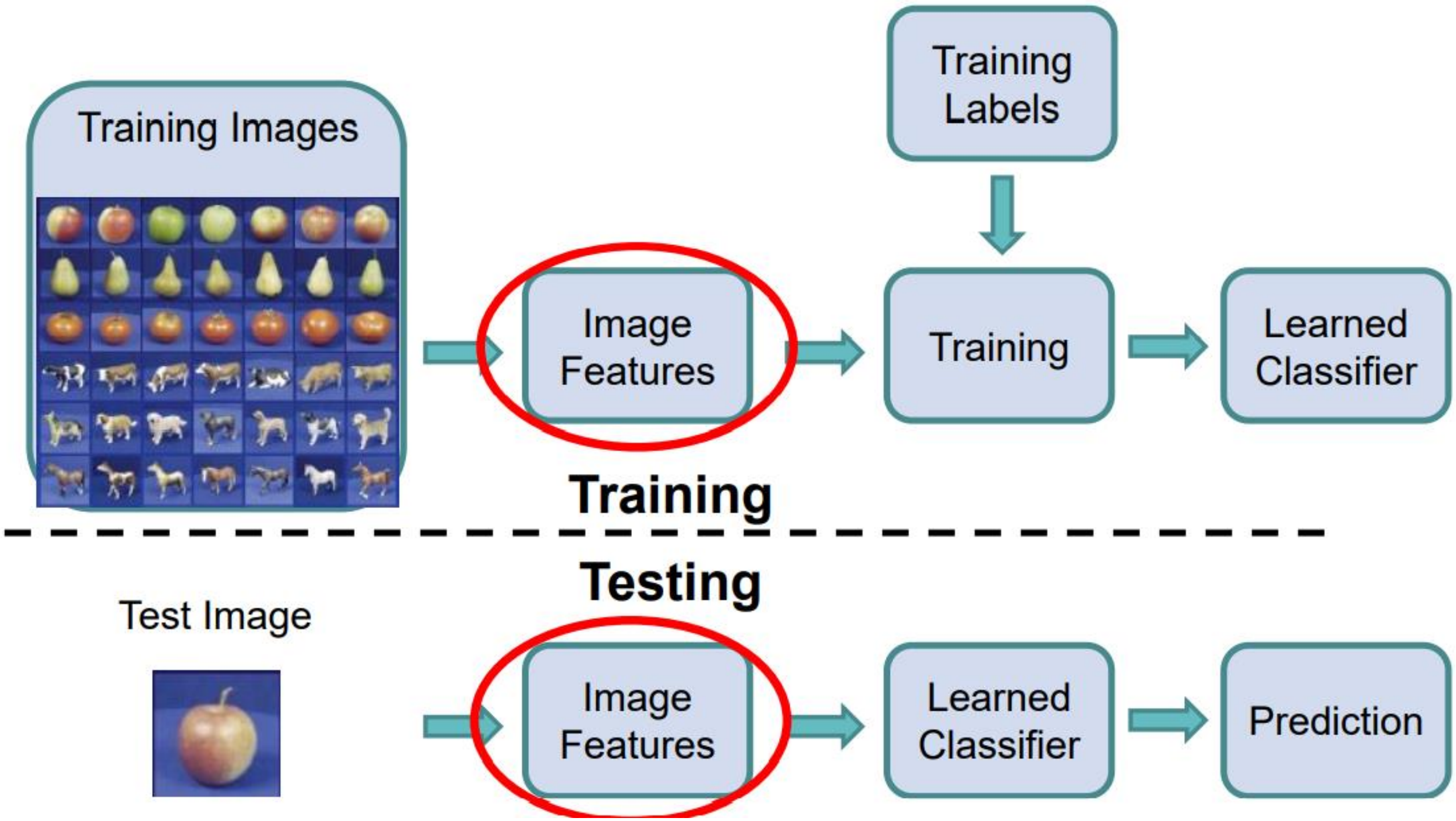
Challenges

- Challenges – Occlusion

Credit: Khola Naseem

Object Recognition Pipeline

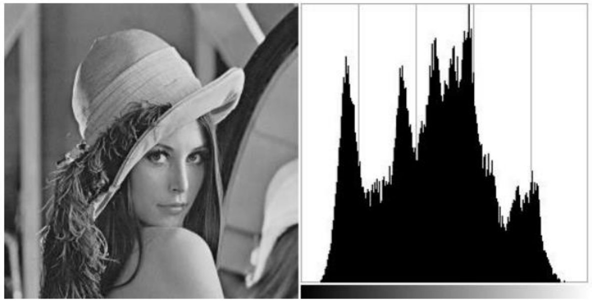
- A simple Object Recognition Pipeline



Credit: Khola Naseem

Image Features

- Histogram



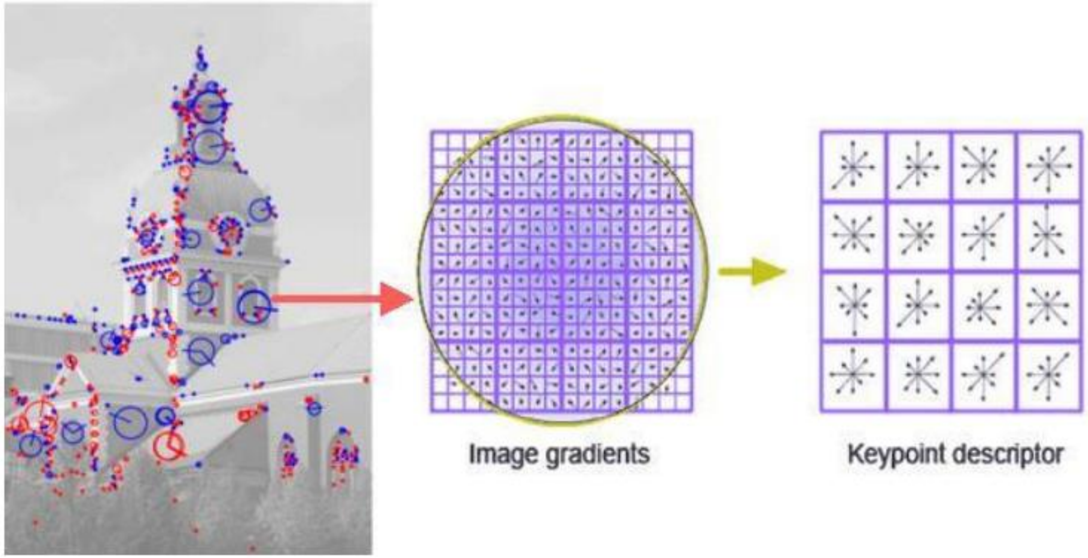
  - This histogram is a graph showing the number of pixels in an image at each different intensity value found in that image. For an 8-bit grayscale image there are 256 different possible intensities, and so the histogram will graphically display 256 numbers showing the distribution of pixels amongst those grayscale value

- SIFT (Scale Feature Invariant Transform)

Image Features

- Histogram

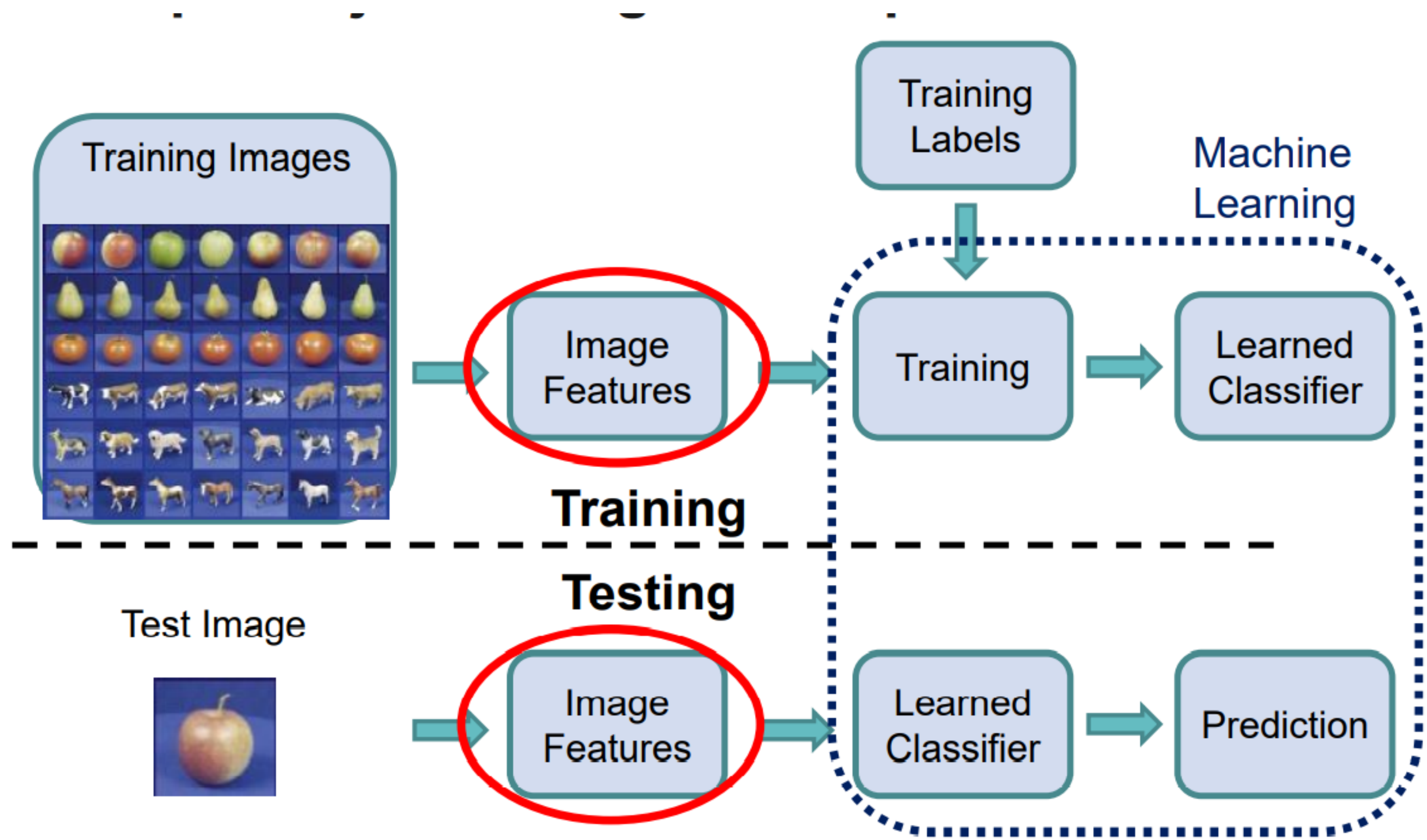
- SIFT (Scale Feature Invariant Transform)



Credit: Khola Naseem

Object Recognition Pipeline

- Object Recognition Pipeline



Credit: Khola Naseem

Object Recognition Pipeline

- **Training data** consists of data samples and the target vectors
- **Learning / Training:** Machine takes training data and automatically learns mapping from data samples to target vectors

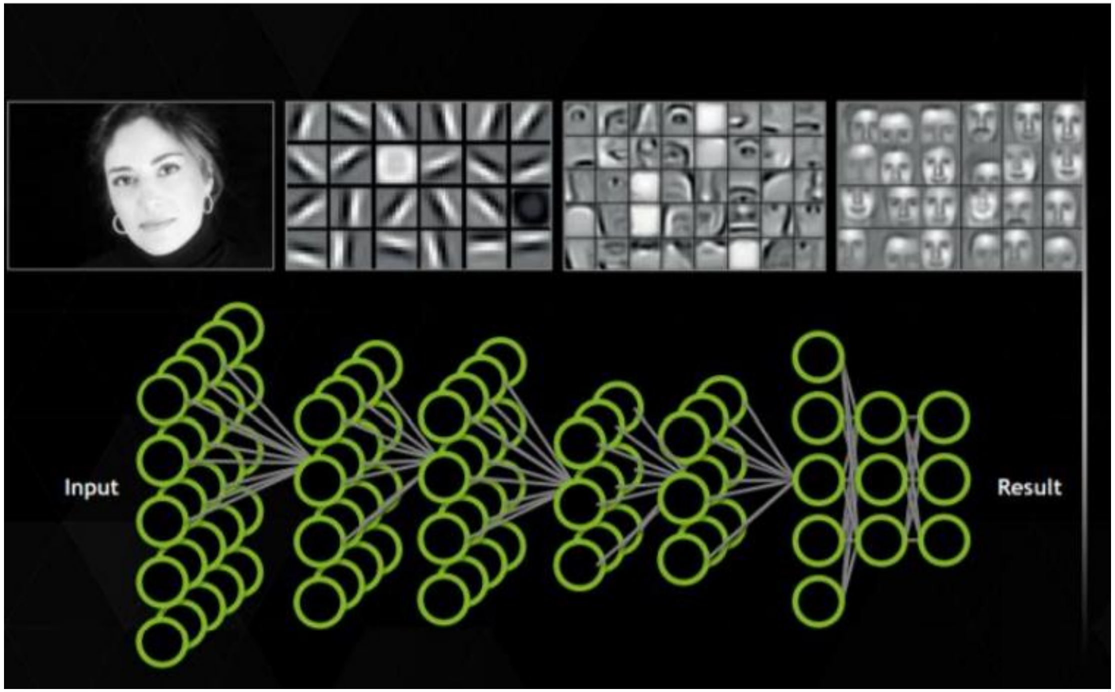
$$\underbrace{y}_{\text{Output Label}} = \underbrace{f}_{\text{Prediction Function}}(\underbrace{x}_{\text{Image Feature}})$$

- **Test data**
  - Target vectors are concealed from the machine
  - Machine predicts the target vectors based on previously learned model
  - Accuracy can be evaluated by comparing the predicted vectors to the actual vectors

Credit: Khola Naseem

Deep Learning Visual Recognition Pipelin

- Instead of using hand tuned features (SIFT, histogram etc.), let the machine find features useful for recognition (End to end learning)



Credit: Khola Naseem

Deep Learning

- CNN

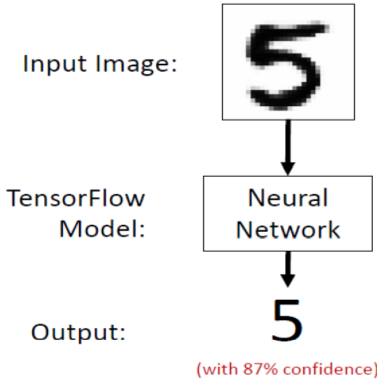
$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \boxed{\eta} \frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$$

How can we set the learning rate?

Credit: Khola Naseem

Deep Learning

- CNN

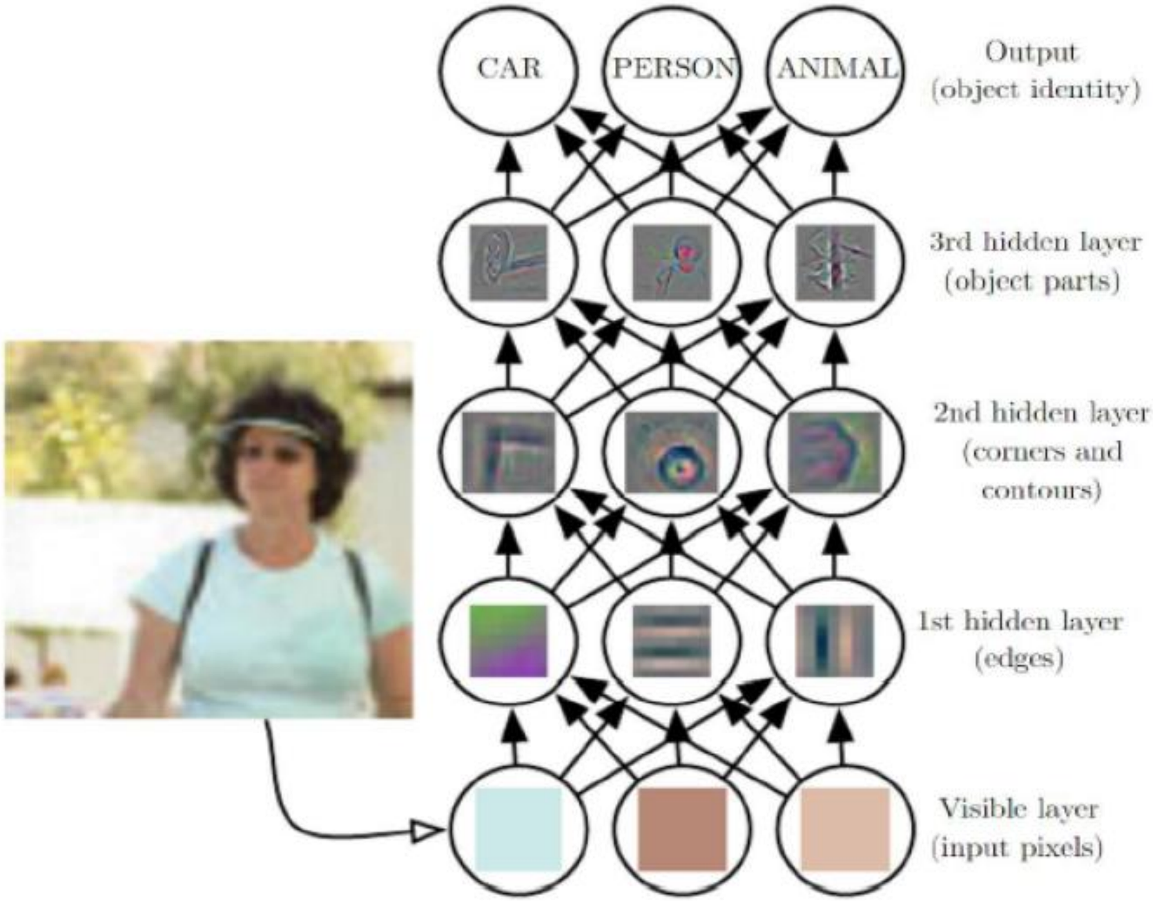



```
# import tensorflow and keras (tf.keras not "vanilla" Keras)
import tensorflow as tf
from tensorflow import keras

# get data
(train_images, train_labels), (test_images, test_labels) = \
keras.datasets.mnist.load_data()

# setup model
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])

model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# train model
model.fit(train_images, train_labels, epochs=5)

# evaluate
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('test accuracy:', test_acc)

# make predictions
predictions = model.predict(test_images)
```

Credit: Khola Naseem

Deep Learning

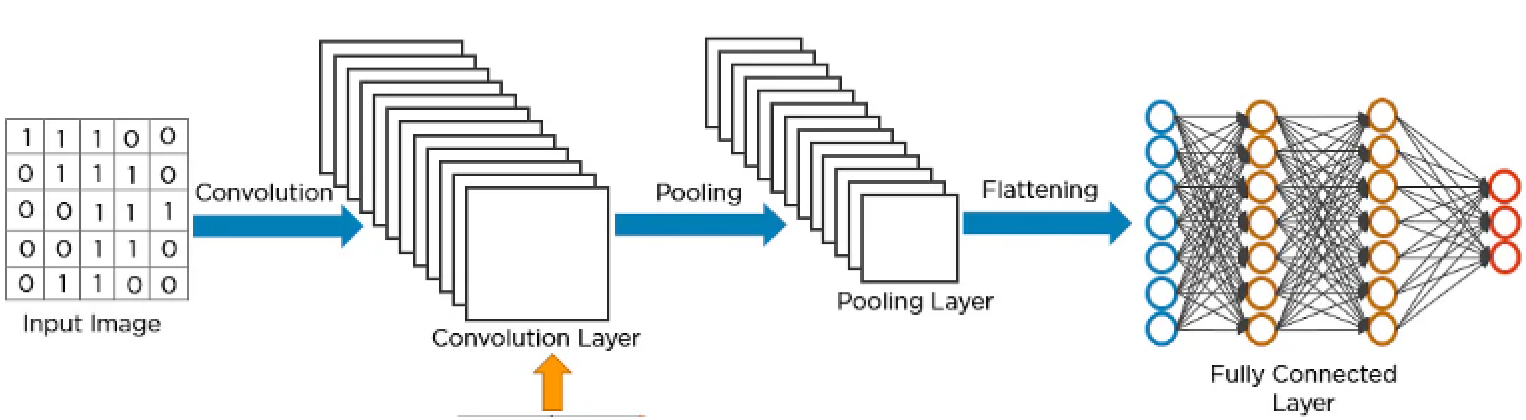
- CNN



Credit: Khola Naseem

Deep Learning

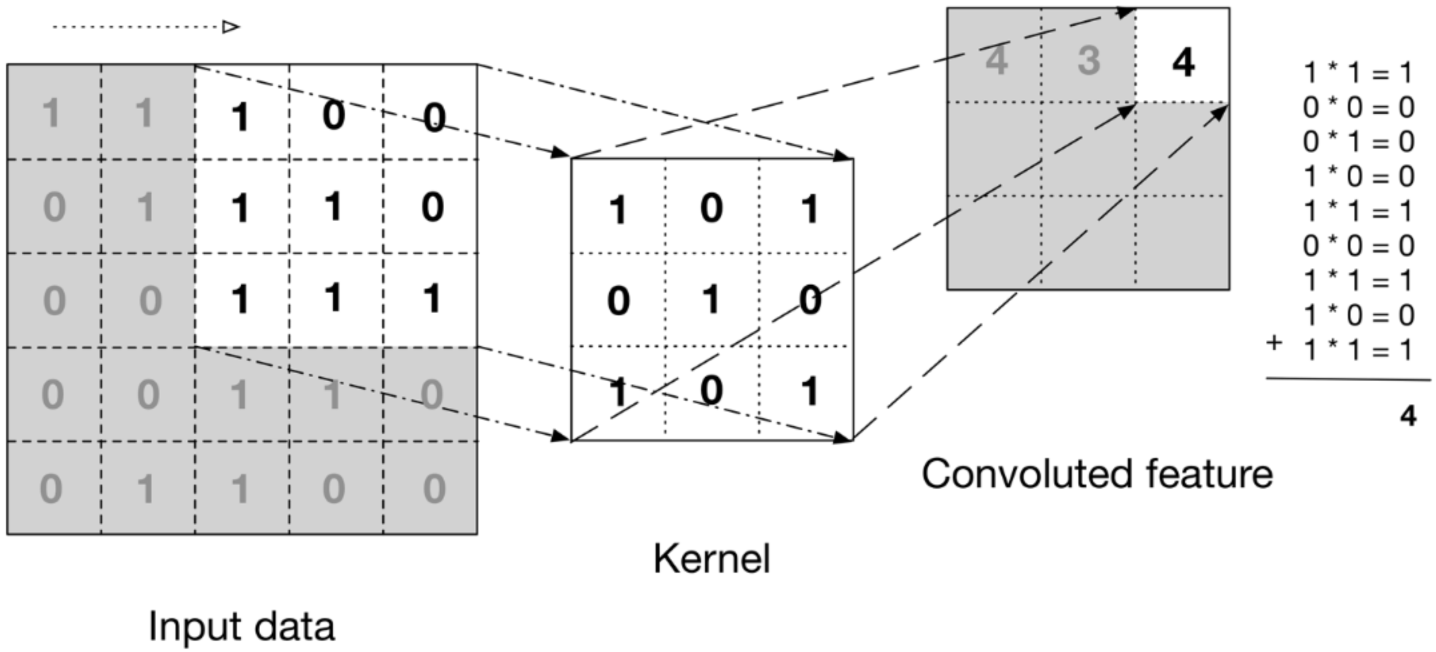
- CNN



Credit: Khola Naseem

Deep Learning

➤ CNN



Credit: Khola Naseem