Machine Learning based Object Recognition

July. 11, 2020- Hoa Doan- Hoadnt@fe.edu.vn

Objectives

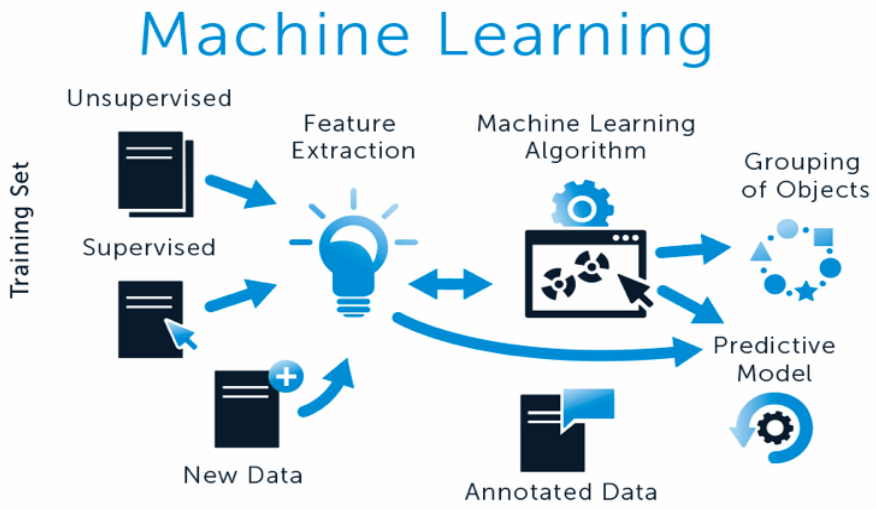
DO MORE

- ✓ Machine Learning
- ✓ Deep Learning
- ✓ Object Recognition
- ✓ Algorithms
- ✓ Demo

Machine Learning

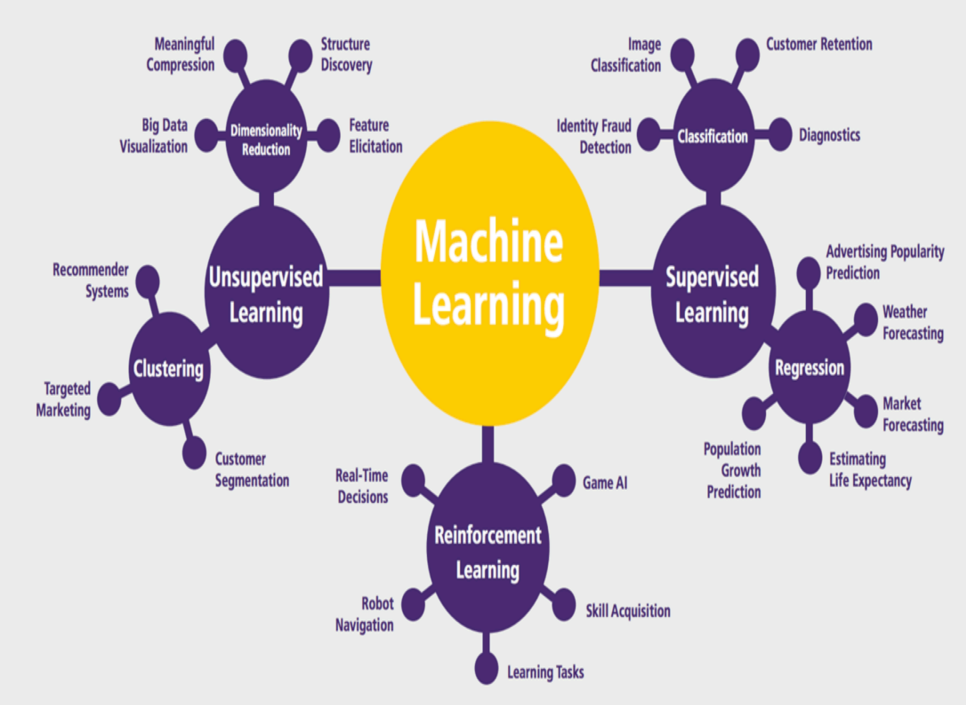
What is Machine Learning?

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.



Machine Learning

How many types?



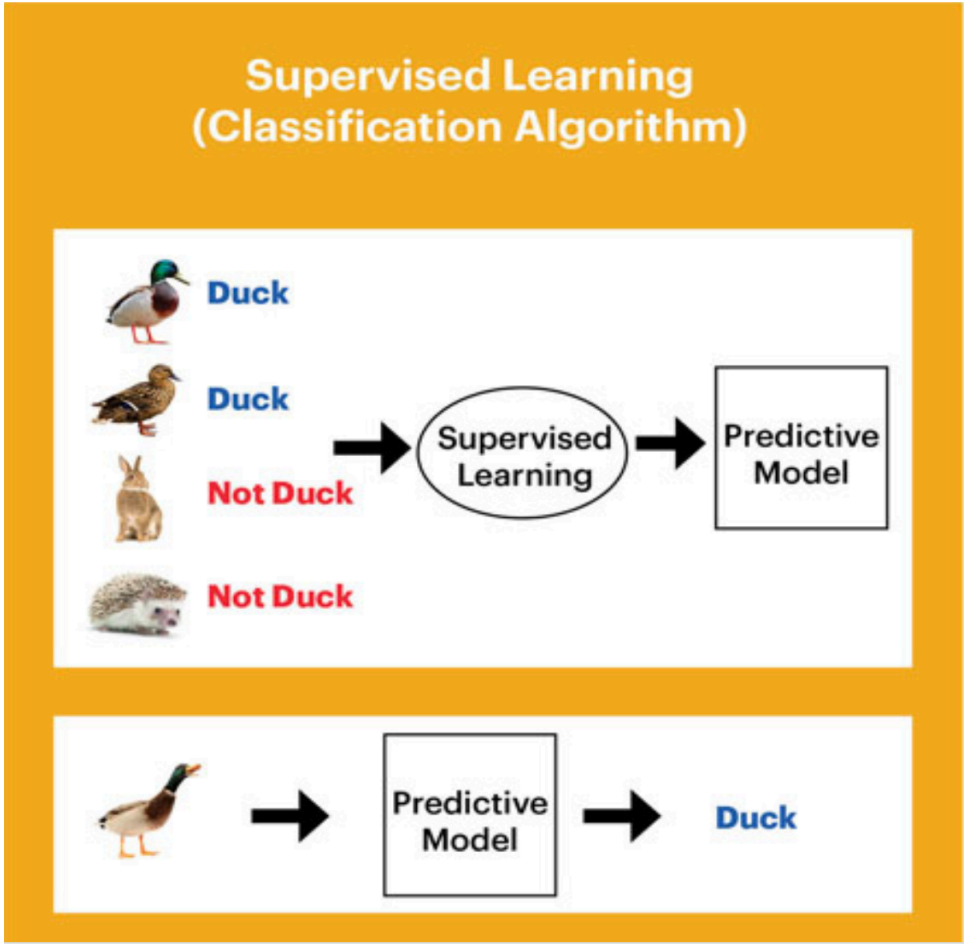
Machine Learning

Supervised ML:

Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

$$Y = f(X)$$

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

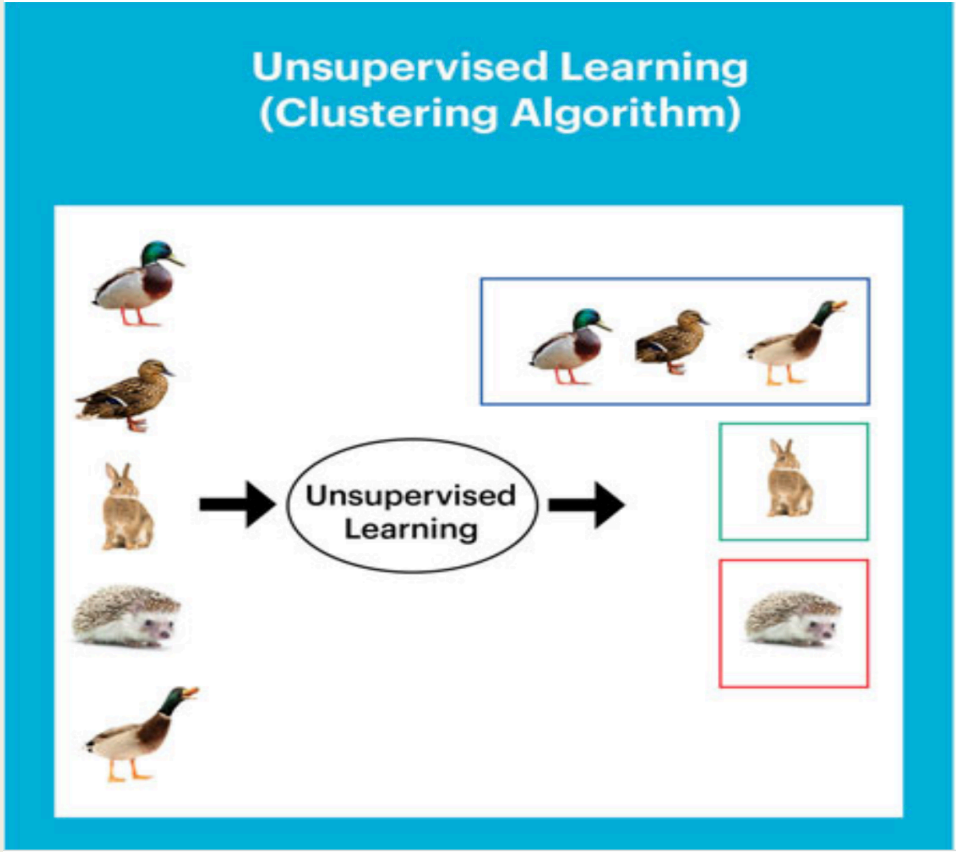


Machine Learning

Unsupervised ML:

Unsupervised learning is where you only have input data (X) and no corresponding output variables.

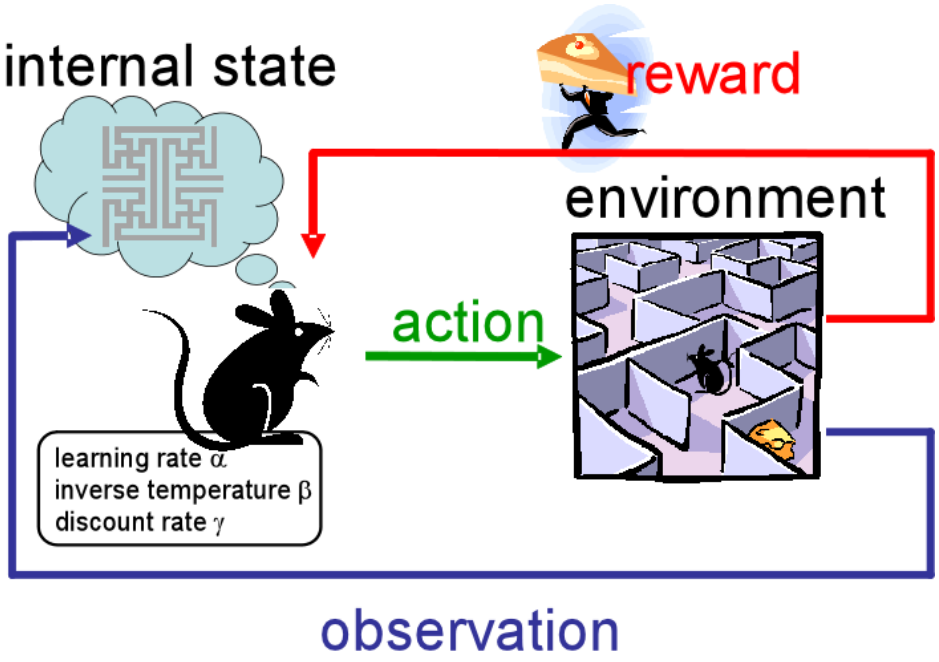
The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.



Machine Learning

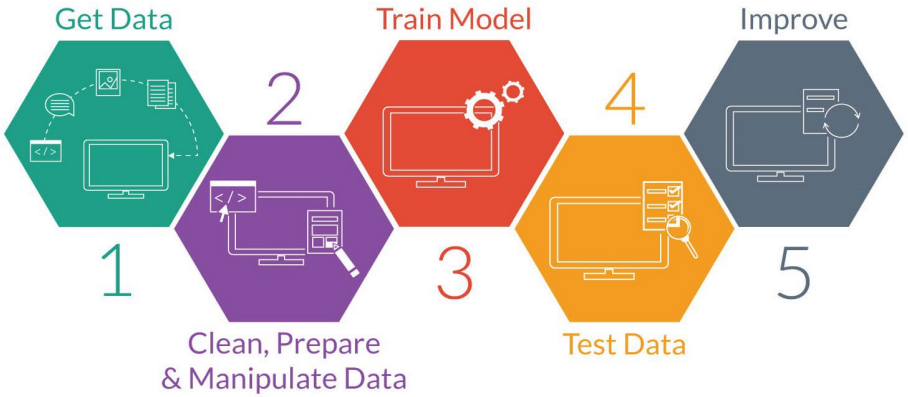
Reinforcement Learning:

Reinforcement Learning(RL) is a type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences.



Machine Learning

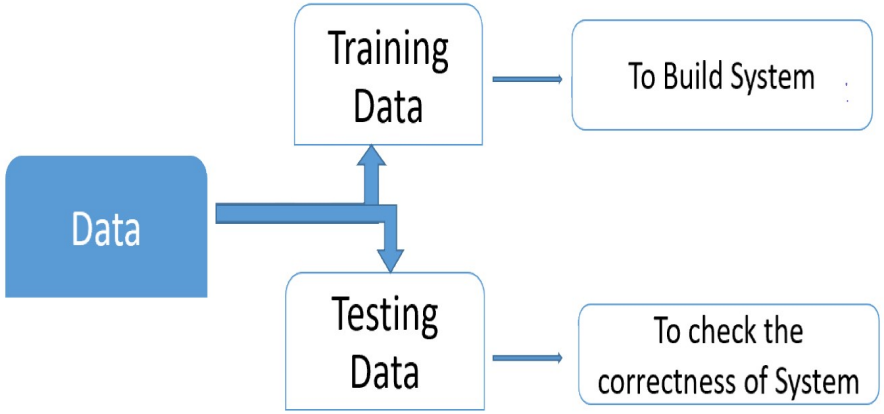
ML steps?



Machine Learning

1. GET DATA

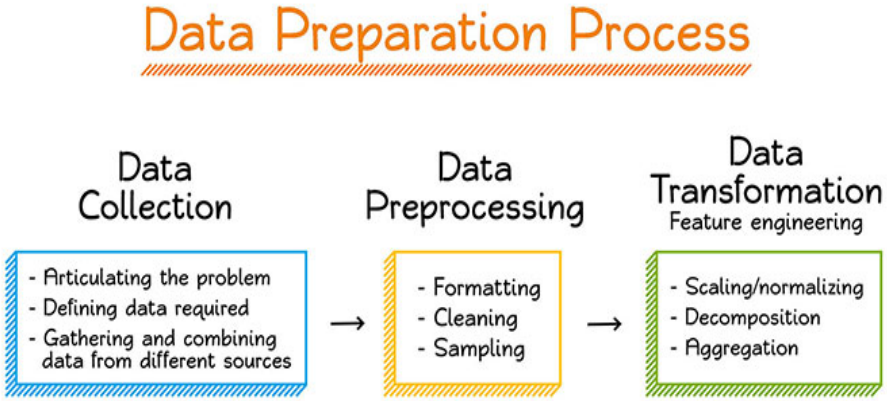
In machine learning, is more data always better than better algorithms?



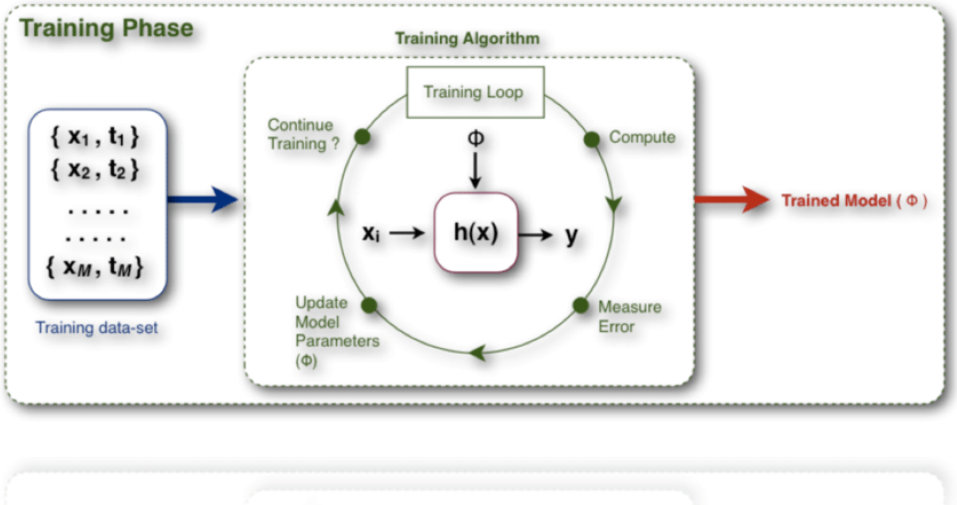
Machine Learning

2. PreProcessing DATA

Valued Data!



Machine Learning



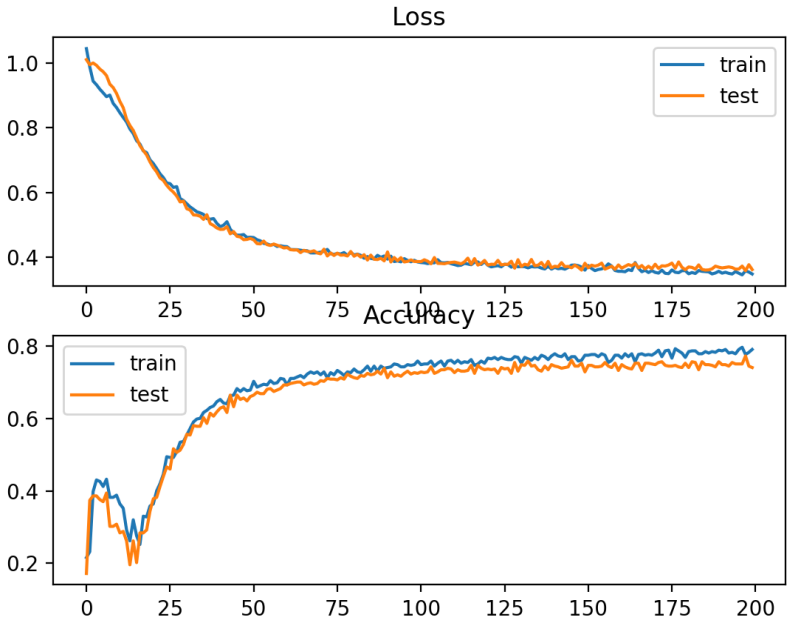
3. Traning

Training a model simply means learning (determining) good values for all the weights and the bias from labeled examples.

Machine Learning

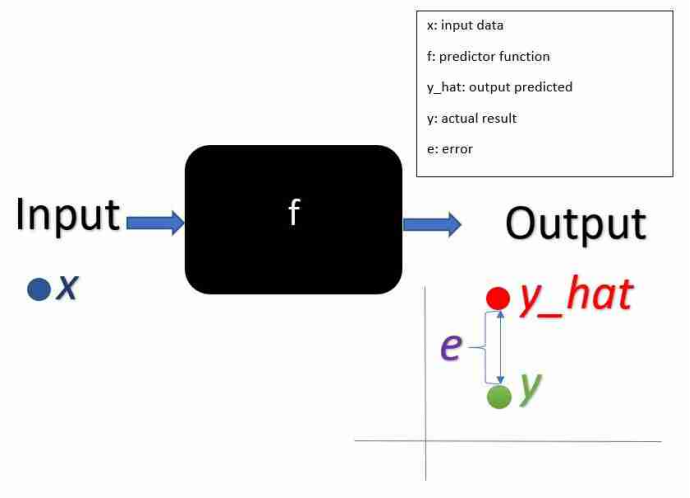
4. Validation

A **loss function** in Machine Learning is a measure of how accurately your ML model is able to predict the expected outcome i.e the ground truth.



Machine Learning

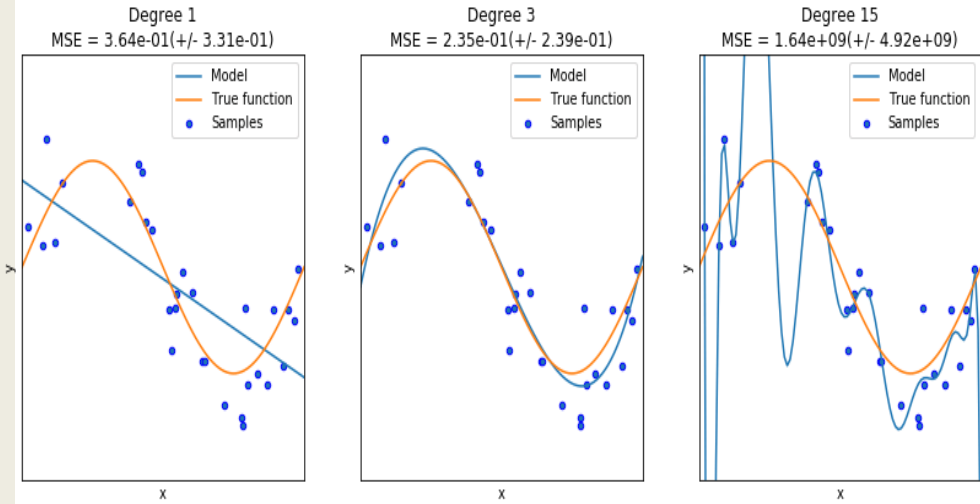
4. Validation



$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \hat{y}\right)}^{2}$$

The square of the difference between actual and predicted

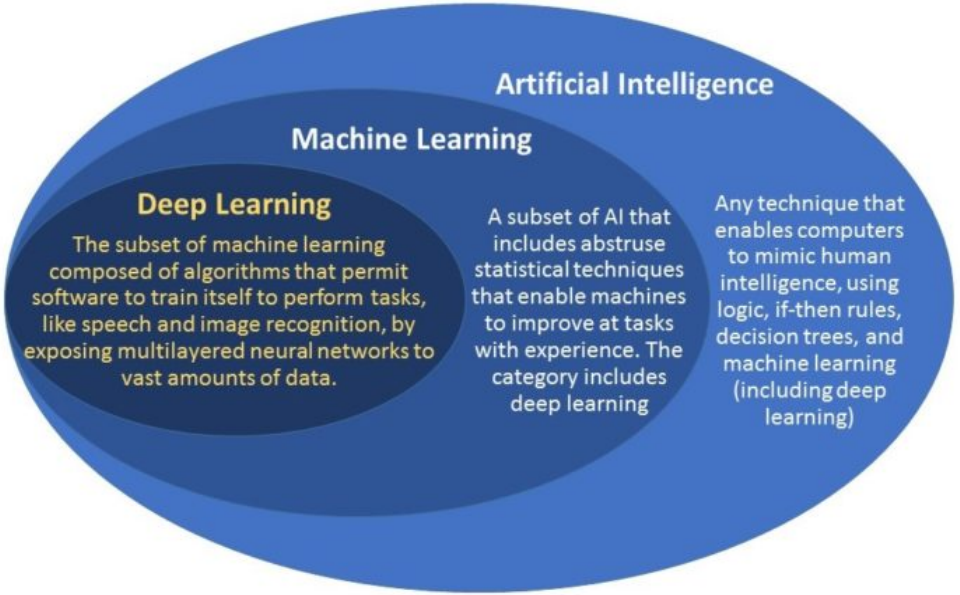
Machine Learning



4. Validation

- Under fitting
- Good fitting
- Over Fitting

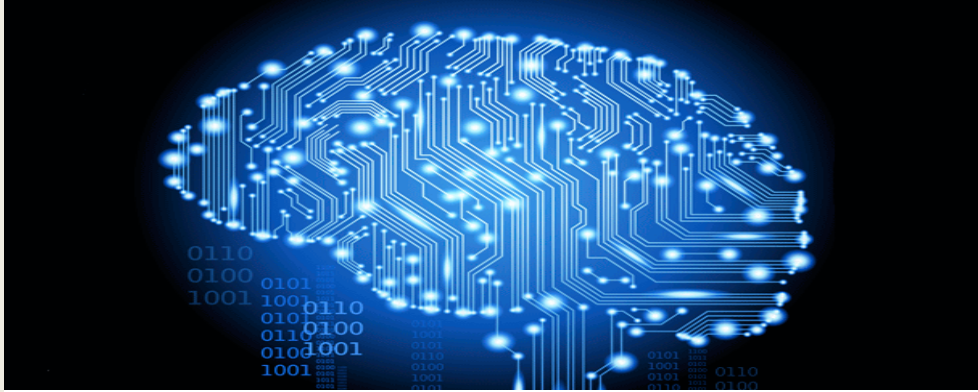
Deep Learning

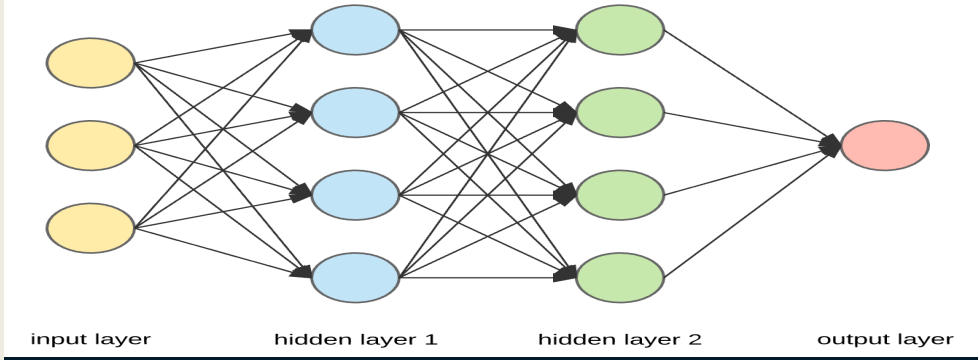


What is Deep Learning?

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks

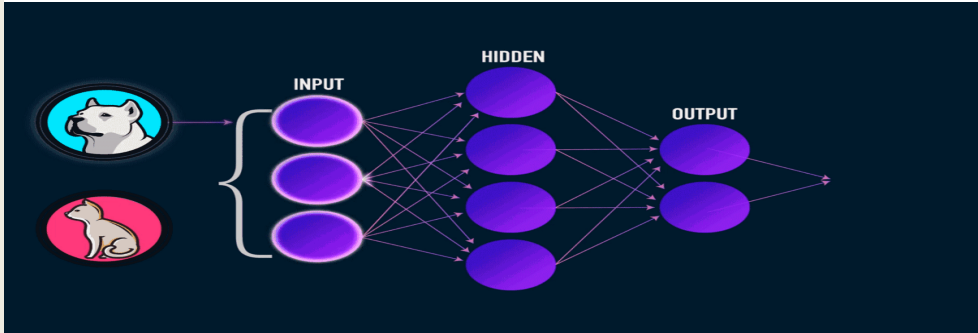
Deep Learning





Convolutional Neural Network

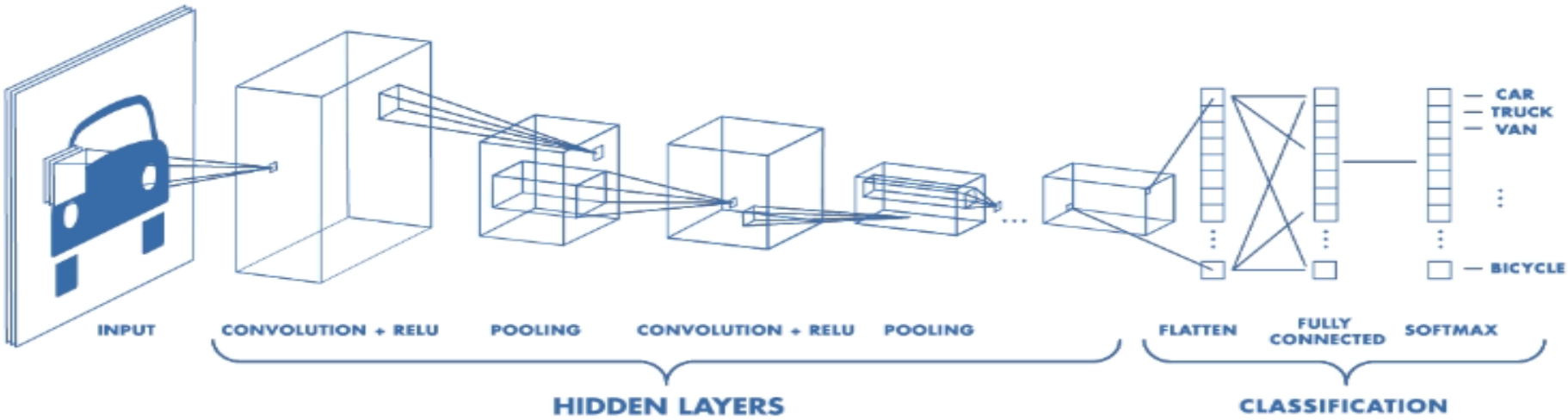
A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data.



Deep Learning

Convolutional Neural Network

Architecture



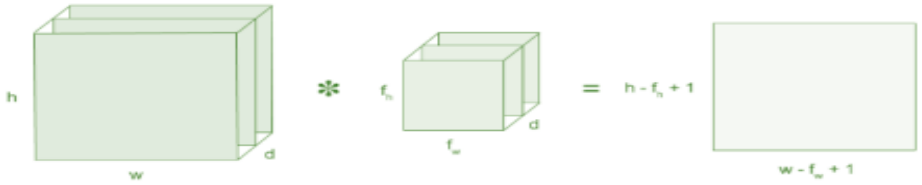
Deep Learning

Convolutinal Layer

A convolutional layer that extracts features from a source image. Convolution helps with blurring, sharpening, edge detection, noise reduction, or other operations that can help the machine to learn specific characteristics of an image.

- ✓ Input: image
- ✓ Output: Feature map

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f_h x f_w x d)**
- Outputs a volume dimension **(h - f_h + 1) x (w - f_w + 1) x 1**



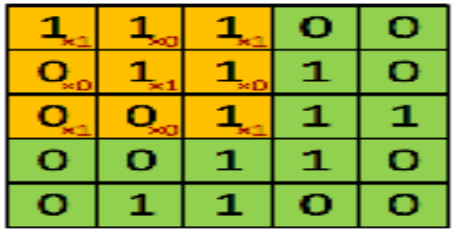
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 x 5 – Image Matrix

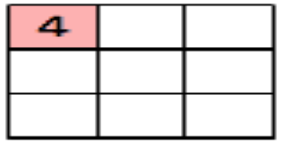
*

1	0	1
0	1	0
1	0	1

3 x 3 – Filter Matrix



Image

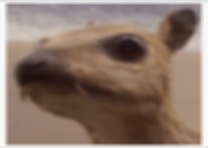


Convolved Feature

Deep Learning

Convolutinal Layer

Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying 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)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

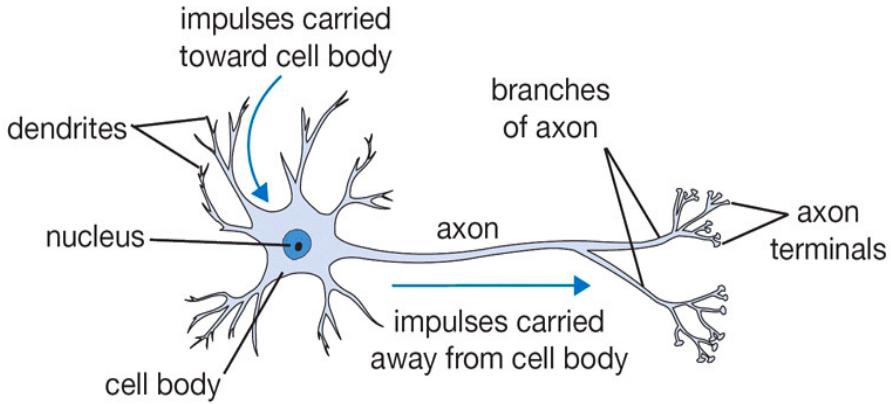
Deep Learning

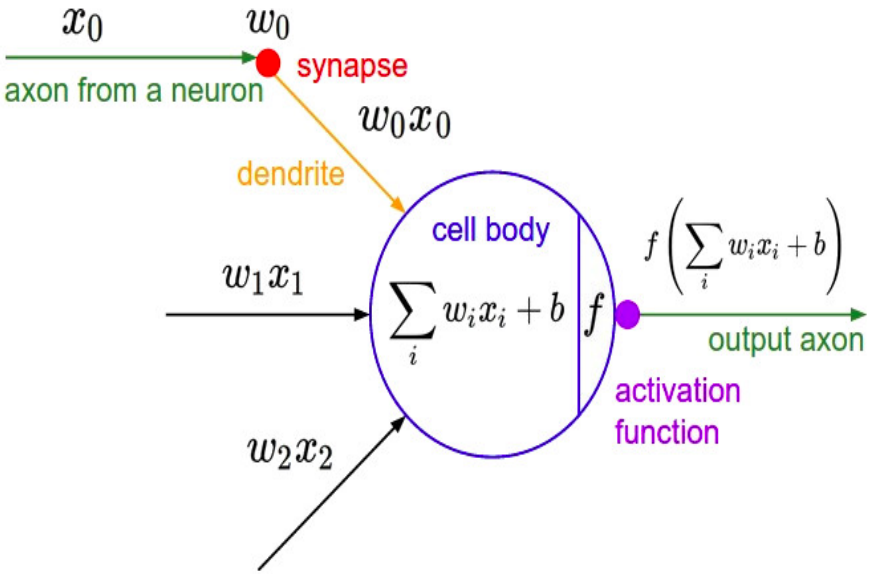
Activation Function

The activation function of a node defines the output of that node given an input or set of inputs. A standard integrated circuit can be seen as a digital network of activation functions that can be "ON" (1) or "OFF" (0), depending on input.

✓ Input: neuron

✓ Output: Active/ Deative





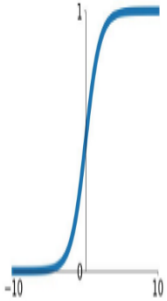
Deep Learning

Activation Function

- ✓ Sigmoid: input: real number → output: number [0,1]
- ✓ Tanh: input: real number → output: number [-1,1]
- ✓ **ReLU: input: real number → output= max(0,number)**
- ✓ Leaky ReLu:
- ✓ Maxout:
- ✓ ELU: input:

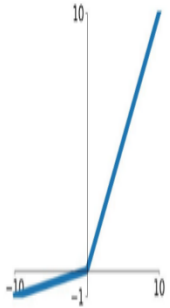
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



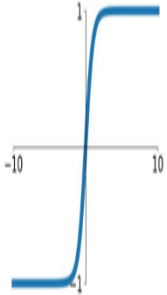
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

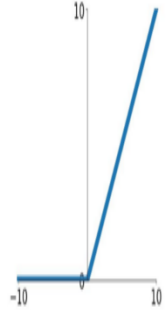


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

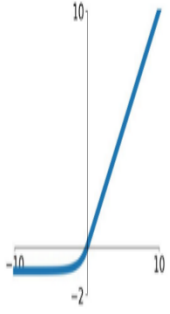
ReLU

$$\max(0, x)$$



ELU

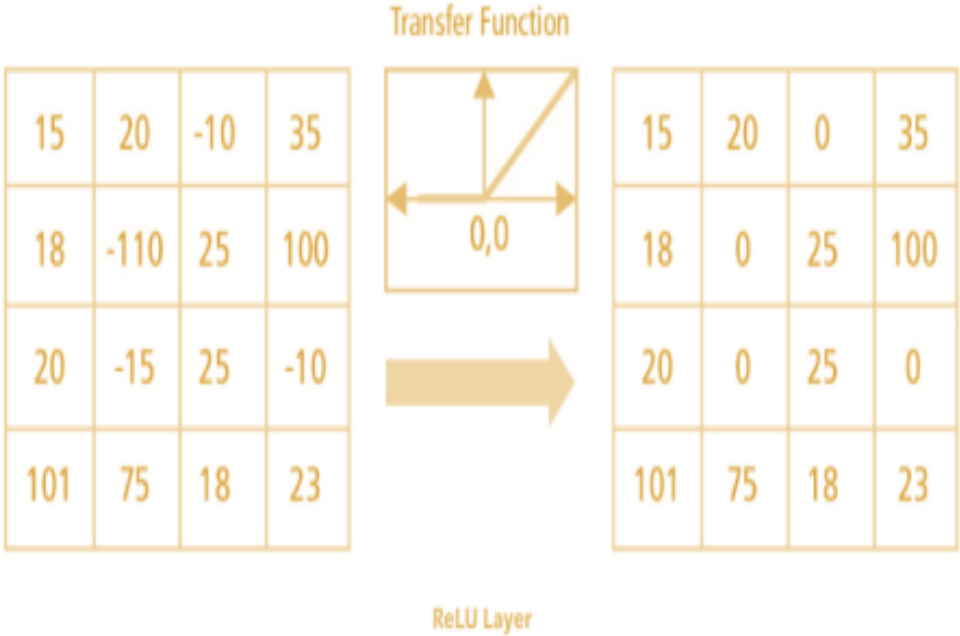
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Deep Learning

Activation Function

- ✓ Sigmoid: input: real number → output: number [0,1]
- ✓ Tanh: input: real number → output: number [-1,1]
- ✓ **ReLU: input: real number → output= max(0,number)**
- ✓ Leaky ReLu:
- ✓ Maxout:
- ✓ ELU: input:

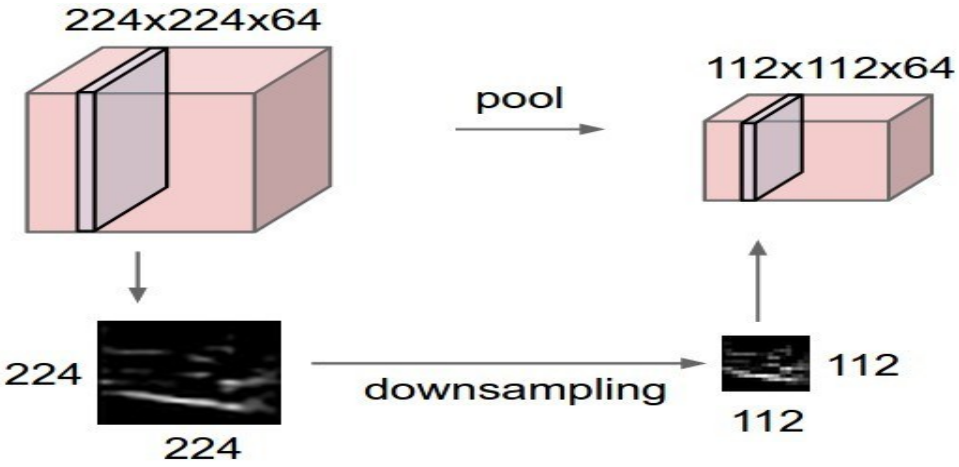


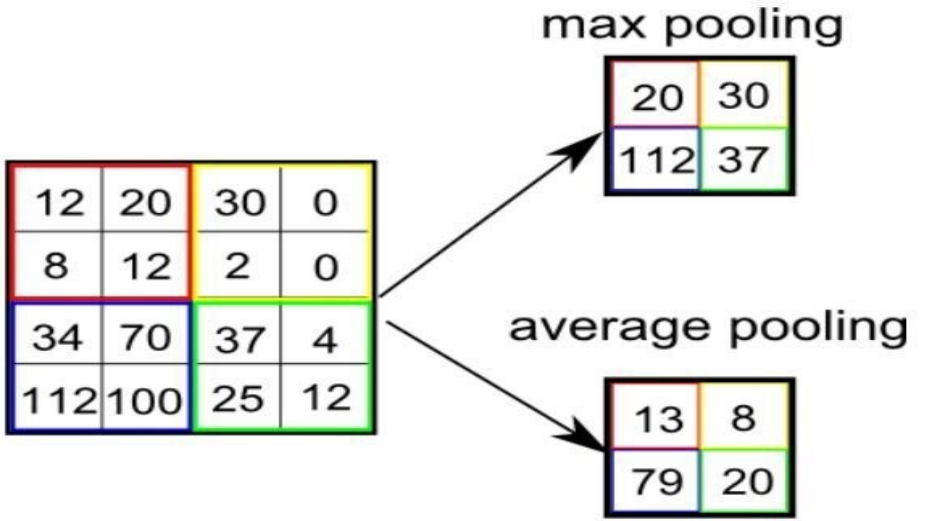
Deep Learning

Pooling Layer

A pooling layer that **reduces the image dimensionality without losing important features or patterns**.

- ✓ Max Pooling
- ✓ Average Pooling
- ✓ Sum Pooling



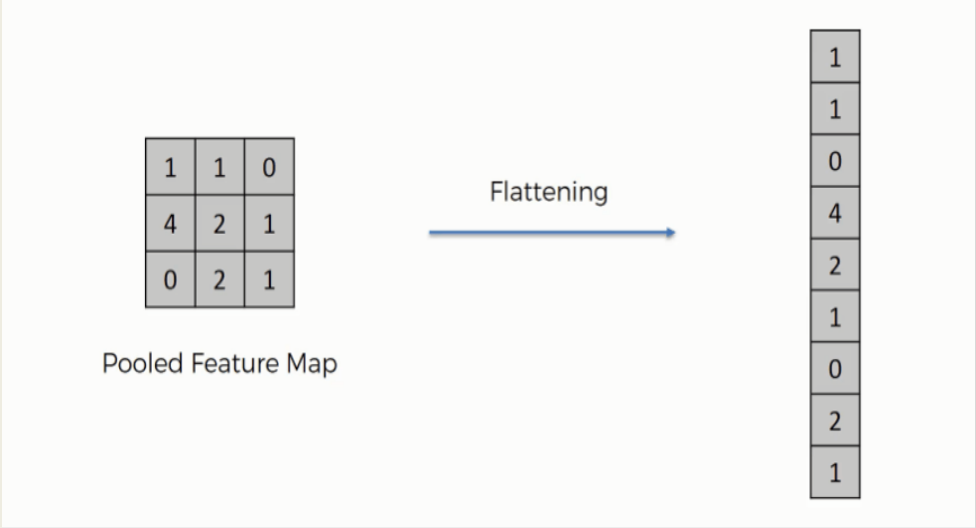


Deep Learning

Flattening

In between the convolutional layer and the fully connected layer, there is a 'Flatten' layer. **Flattening transforms a two-dimensional matrix of features into a vector** that can be fed into a fully connected neural network classifier.

- ✓ Input: Matrix[H,W,D]
- ✓ Output: Vector (H*W*D)

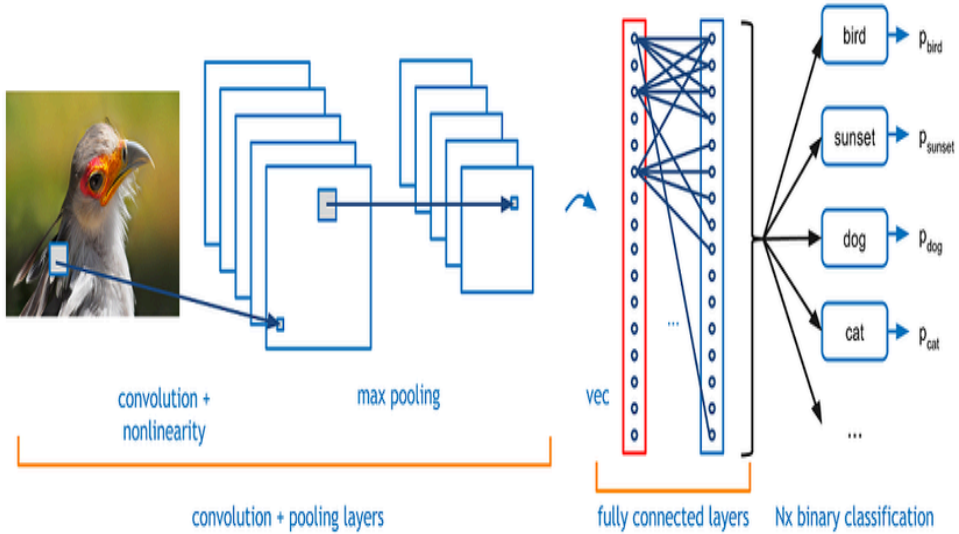


Deep Learning

Fully connected layer

A fully connected layer also known as the dense layer, in which the results of the convolutional layers are fed through one or more neural layers to generate a prediction.

- ✓ The first fully connected layer takes the inputs from the feature analysis and applies weights to predict the correct label.
- ✓ Fully connected output layer gives **the final probabilities for each label**.



Deep Learning

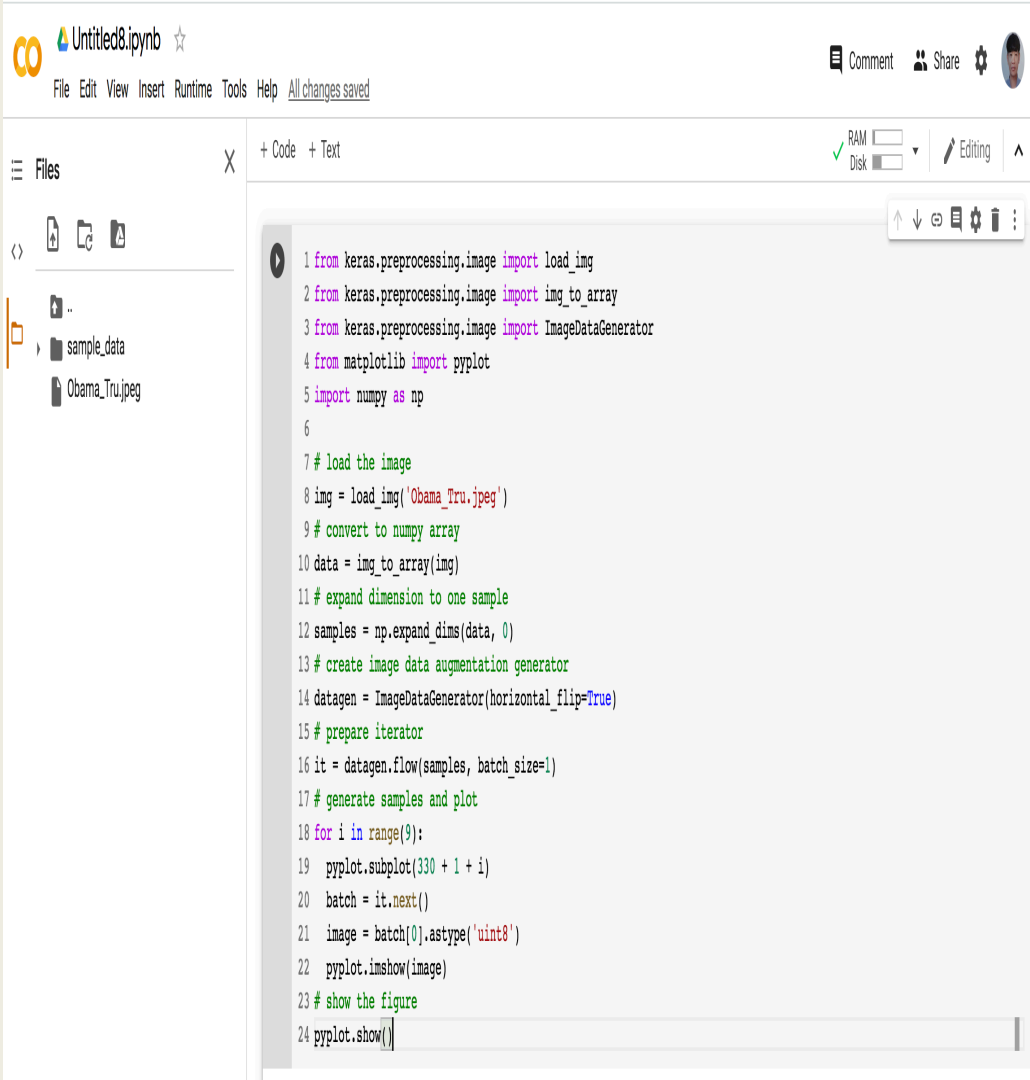
How to choose the parameters?

- ✓ Number of convolution layers: the more convolution layers the better the performance.
- ✓ Filter size: size 5×5 or 3×3
- ✓ Pooling size: size 2×2 or 4×4
- ✓ The final way is to perform multiple train tests to select the best param.

Colaboratory Project

PaaS: Platform as a Service

!pip install matplotlib-venn



Colaboratory Project

PaaS: Platform as a Service

What is Colaboratory?

Colaboratory is a research tool for machine learning education and research. It's a Jupyter notebook environment that requires no setup to use.

Is it free to use?

Yes. Colaboratory is a research project that is free to use.

Where are my notebooks stored, and can I share them?

All Colaboratory notebooks are stored in Google Drive. Colaboratory notebooks can be shared just as you would with Google Docs or Sheets.

What about Python3? (or R, Scala, ...)

Colaboratory supports Python 2.7 and Python 3.6.

How may I use GPUs and why are they sometimes unavailable?

Colaboratory is intended for interactive use. Long-running background computations, particularly on GPUs, may be stopped.

Google Colab gives you a free K80 GPU for up to 12hrs at a time.

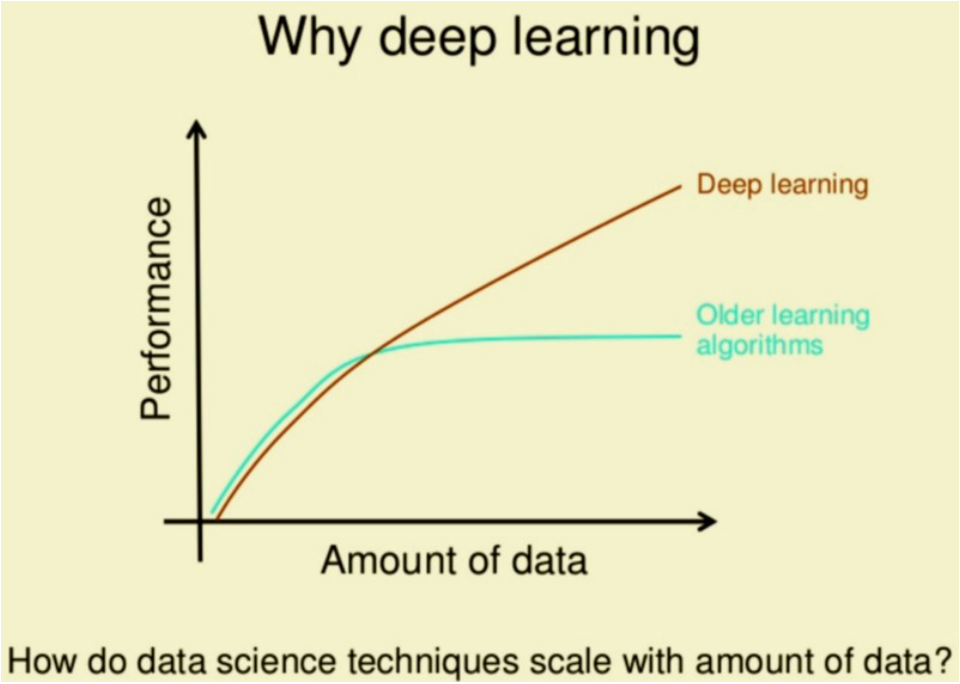
Tesla K80 GPU- using Keras, Tensorflow and PyTorch, CV2.

Training time is up 15-20 times compare with CPU.

Deep Learning

Data Augmentation?

Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data. Data augmentation techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks



Deep Learning

Data Augmentation**?**

Three ways to improve data

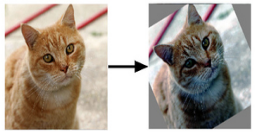
1 - Collect more

- expensive
- requires manual labor

2 - Synthesize

- complicated
- might not truly represent the real data

3 - Augment



- simple
- but finding a good augmentation strategy takes lots of trial & error (**=time of AI engineers**)

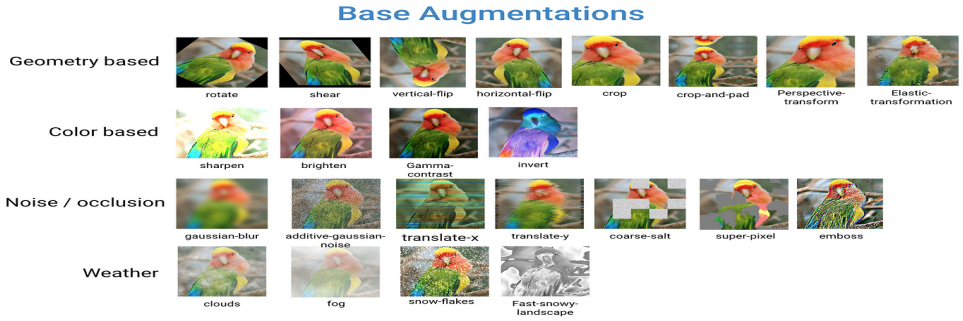
Deep Learning

Data Augmentation

Keras lib

Original	Flip	Rotation	Random crop
			
• Image without any modification	• Flipped with respect to an axis for which the meaning of the image is preserved	• Rotation with a slight angle • Simulates incorrect horizon calibration	• Random focus on one part of the image • Several random crops can be done in a row

Color shift	Noise addition	Information loss	Contrast change
			
• Nuances of RGB is slightly changed • Captures noise that can occur with light exposure	• Addition of noise • More tolerance to quality variation of inputs	• Parts of image ignored • Mimics potential loss of parts of image	• Luminosity changes • Controls difference in exposition due to time of day



Deep Learning

Demo 1: Data Augmentation

- ✓ Input: your image
- ✓ Output: generate 9 images of yours

Keras lib

!pip install numpy

```
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot
import numpy as np

# load the image
img = load_img('Obama_Tru.jpeg')
# convert to numpy array
data = img_to_array(img)
# expand dimension to one sample
samples = np.expand_dims(data, 0)
# create image data augmentation generator
datagen = ImageDataGenerator(horizontal_flip=True)
# prepare iterator
it = datagen.flow(samples, batch_size=1)
# generate samples and plot
for i in range(9):
  pyplot.subplot(330 + 1 + i)
  batch = it.next()
  image = batch[0].astype('uint8')
  pyplot.imshow(image)
# show the figure
pyplot.show()
```

Deep Learning

Application

Machine learning use cases

Manufacturing

- Predictive maintenance or condition monitoring
- Warranty reserve estimation
- Propensity to buy
- Demand forecasting
- Process optimization
- Telematics

Retail

- Predictive inventory planning
- Recommendation engines
- Upsell and cross-channel marketing
- Market segmentation and targeting
- Customer ROI and lifetime value

Healthcare and Life Sciences

- Alerts and diagnostics from real-time patient data
- Disease identification and risk satisfaction
- Patient triage optimization
- Proactive health management
- Healthcare provider sentiment analysis

Travel and Hospitality

- Aircraft scheduling
- Dynamic pricing
- Social media – consumer feedback and interaction analysis
- Customer complaint resolution
- Traffic patterns and congestion management

Financial Services

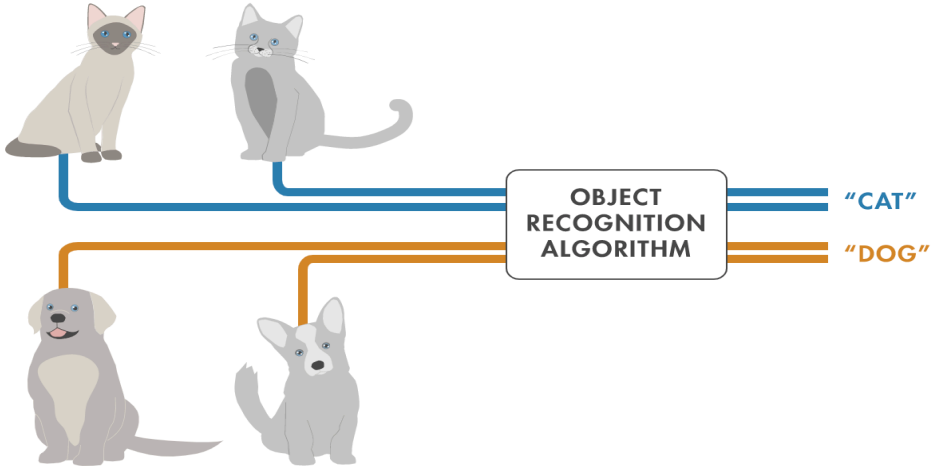
- Risk analytics and regulation
- Customer Segmentation
- Cross-selling and up-selling
- Sales and marketing campaign management
- Credit worthiness evaluation

Energy, Feedstock and Utilities

- Power usage analytics
- Seismic data processing
- Carbon emissions and trading
- Customer-specific pricing
- Smart grid management
- Energy demand and supply optimization

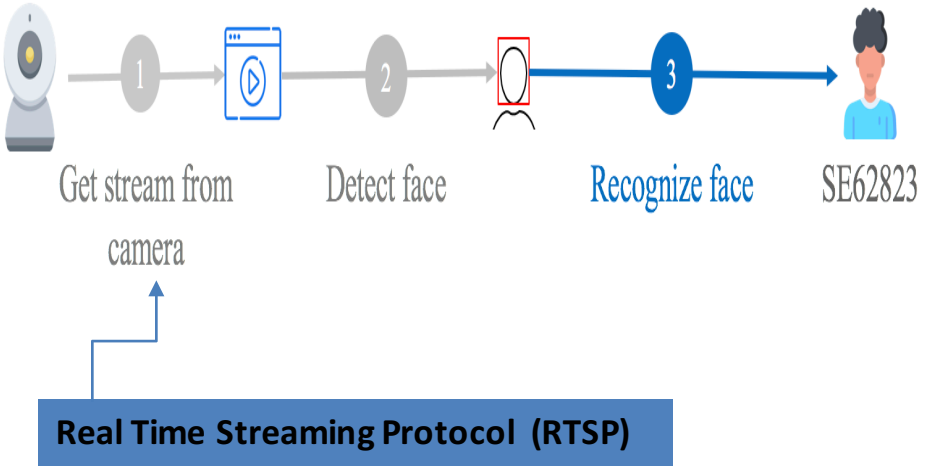
Object Recognition

Object recognition is a computer vision technique for identifying objects in images or videos



Object Recognition

Processing

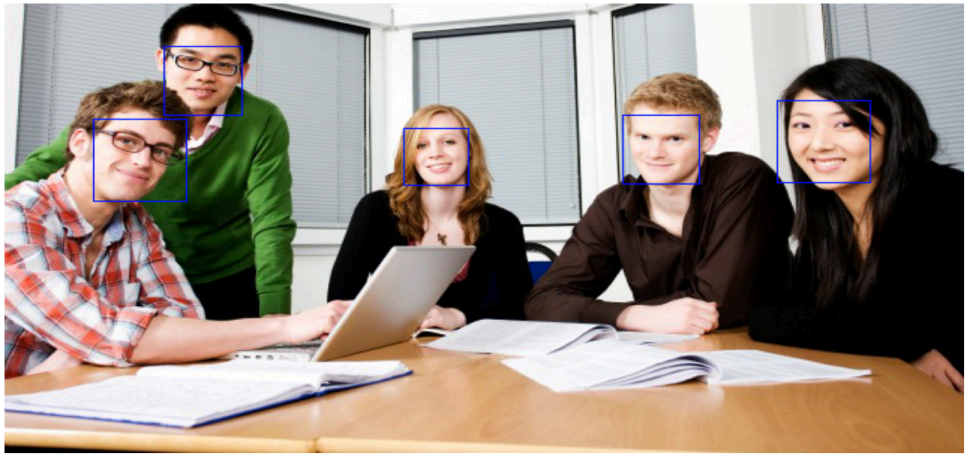


Object Recognition

Face Detection

Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images

Each face has the property such as **eye, nose, mouth** which is **possible to detect**



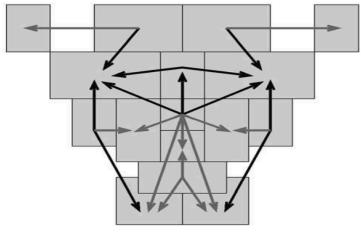


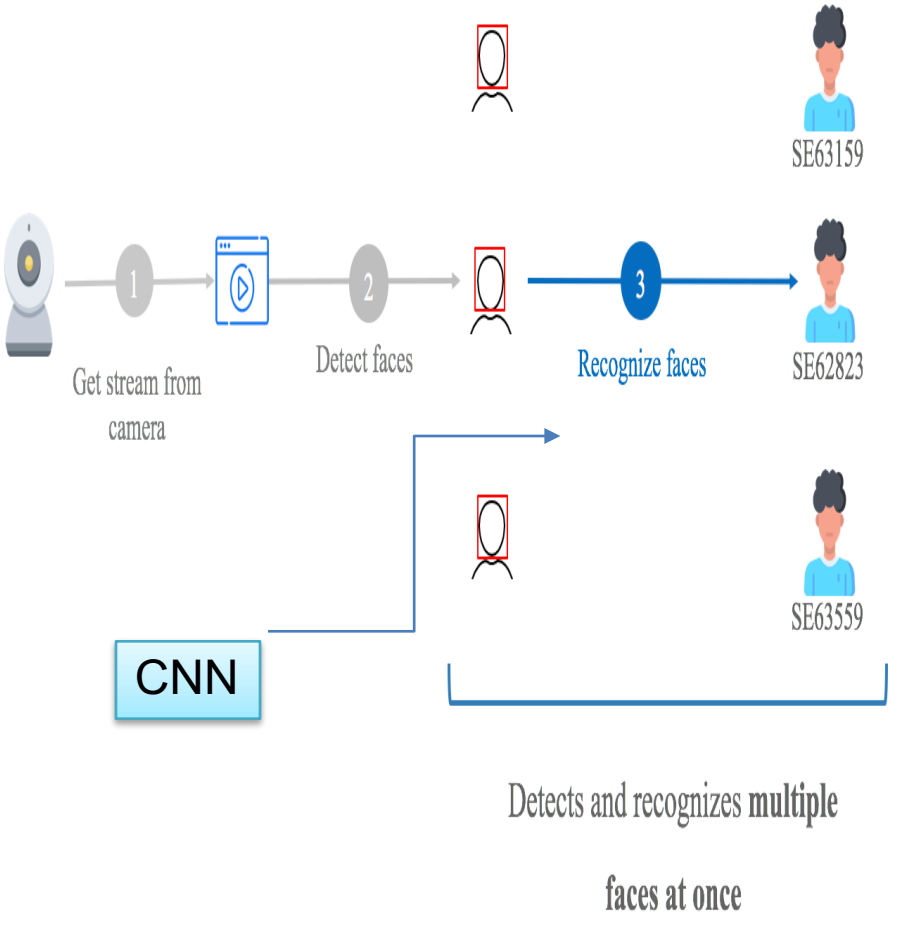
Fig. 5. A 14x16 pixel ratio template for face localization based on Sinha method. The template is composed of 16 regions (the gray boxes) and 23 relations (shown by arrows) [139] (Courtesy of B. Scassellati).

Object Recognition

Face Recognition

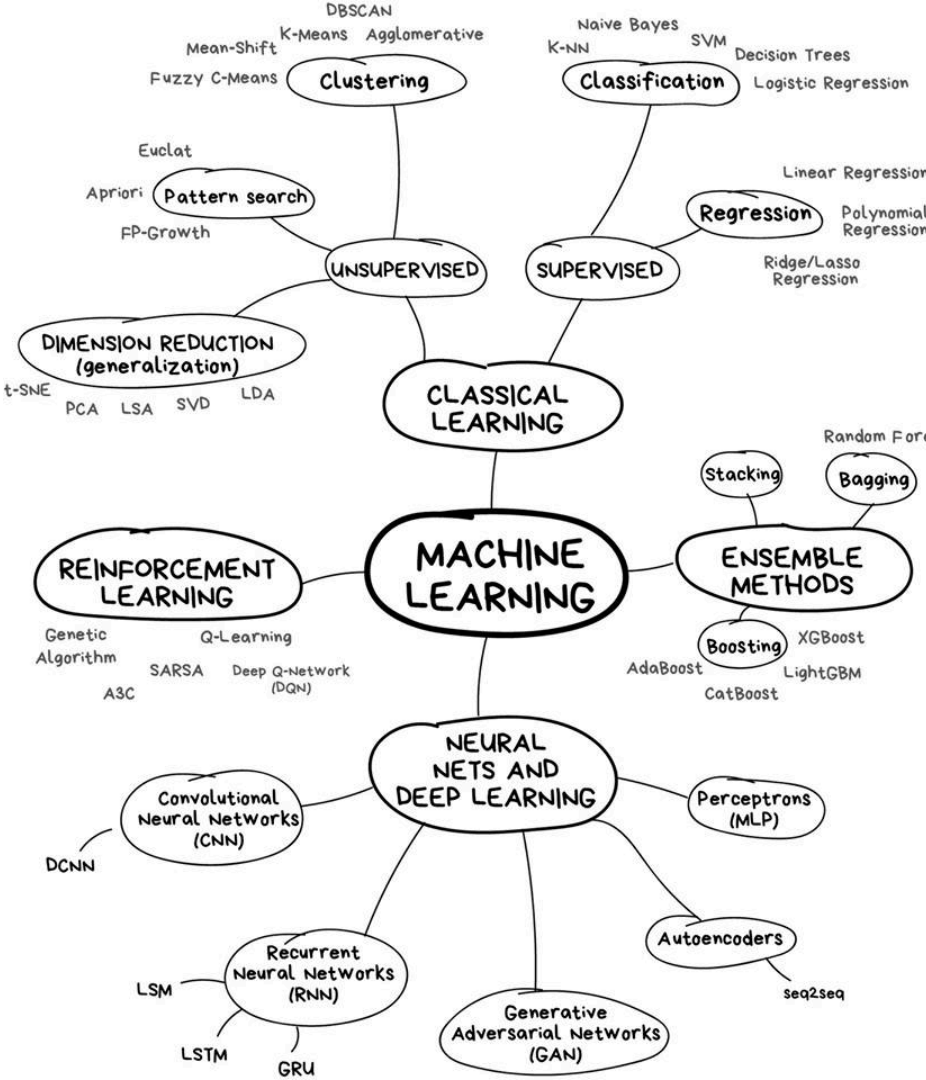
Facial recognition is the process of identifying or verifying the identity of a person using their face. It captures, analyzes, and compares patterns based on the person's facial details.

- ✓ The face detection process is an essential step as it detects and locates human faces in images and videos.
- ✓ The face capture process transforms analogue information (a face) into a set of digital information (data) based on the person's facial features.
- ✓ The face match process verifies if two faces belong to the same person.



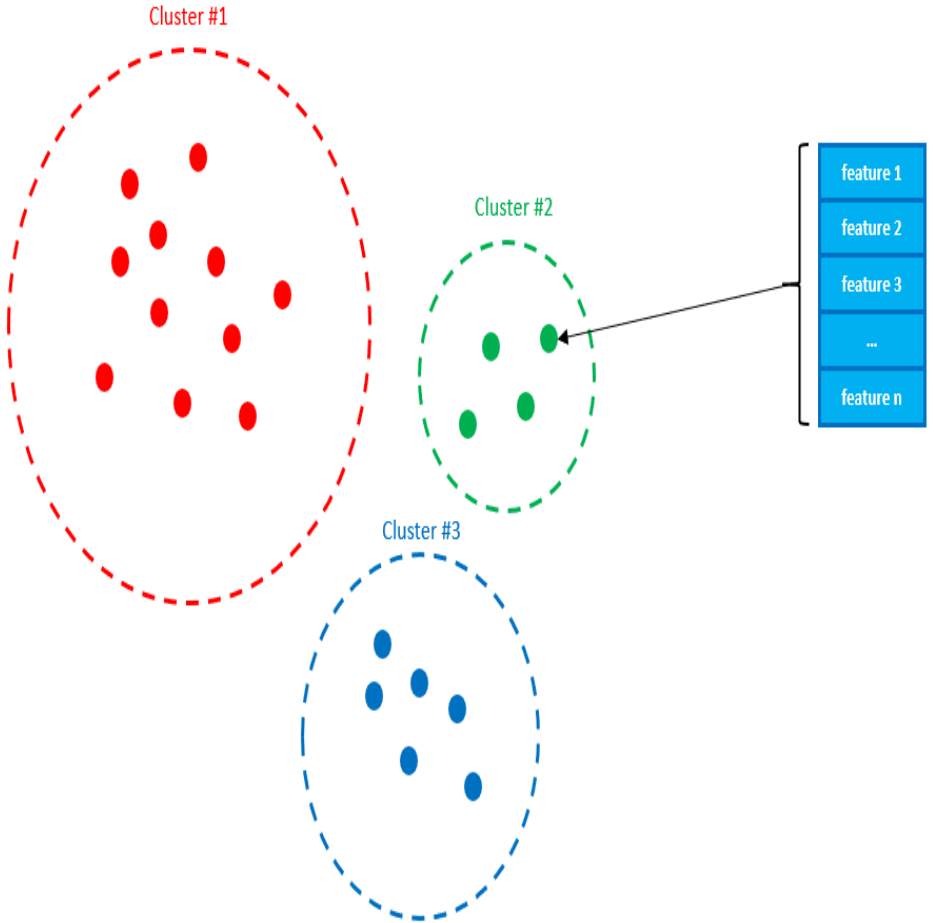
Algorithms

Explain why you're focusing on a particular part of the problem or a particular subset of users.



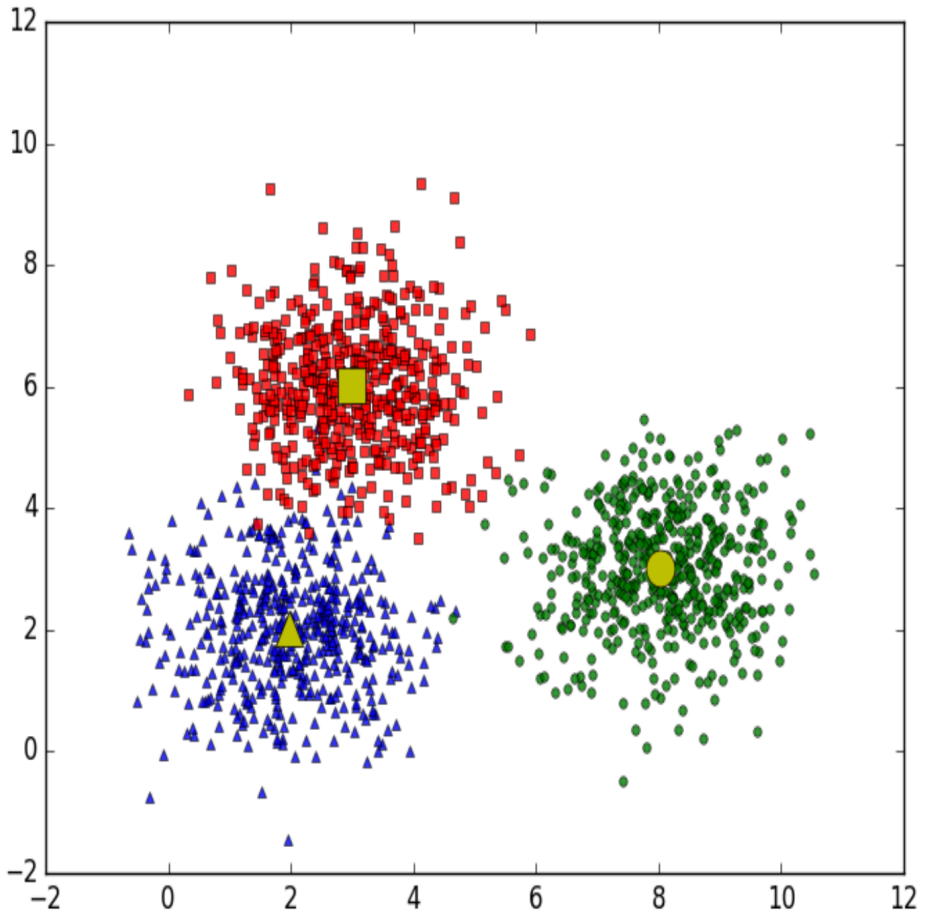
Clustering

Clustering is the grouping of a particular set of objects based on their characteristics, aggregating them according to their similarities.



K-Means Clustering

The K-means algorithm identifies **k** number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.



K-Means Clustering

How does it works ?

- ✓ N data points: X=[x1, x2, …, xn] belong Rdxn
- ✓ K cluster: K<N
- ➢ Find the K center: m1, m2, …, mk belong Rdx1
- ➢ Find group of each data point that belong to

- ✓ Label vector of each data point Xi: yi=[yi1, yi2, …, yiK]
- ✓ If xi belong cluster k → yik=1 and yij=0 for j#k

Examples:

- xi belong cluster 1→ yi=[1,0, …, 0]
- xi belong cluster 2→ yi=[0,1, …, 0]

$$y_{ik} \in \{0,1\}, \quad \sum_{k=1}^{K} y_{ik} = 1 \quad (1)$$

K-Means Clustering

Loss Function

- ✓ Mk : center of each cluster
- ✓ Xi belong cluster k→ error= (xi - mk)

Purpose: minimize the error $\|\mathbf{x}_i - \mathbf{m}_k\|_2^2$

xi belong cluster k → $y_{ik} = 1, y_{ij} = 0, \forall j \neq k$

The error : $y_{ik}\|\mathbf{x}_i - \mathbf{m}_k\|_2^2 = \sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$

Total Error: $\mathcal{L}(\mathbf{Y}, \mathbf{M}) = \sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$

Optimization:

$$\mathbf{Y}, \mathbf{M} = \arg\min_{\mathbf{Y},\mathbf{M}} \sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2 \quad (2)$$

$$\text{subject to: } y_{ij} \in \{0, 1\} \ \forall i, j; \quad \sum_{j=1}^{K} y_{ij} = 1 \ \forall i$$

K-Means Clustering

Optimization: $\mathcal{L}(\mathrm{M}, Y)$

- ✓ M=[m1, m2, …, mK]: Center of each cluster.
- ✓ Y=[y1, y2, …, yN]: Label vector of each data point
  - ➢ Fixed M, find Y
  - ➢ Fixed Y, find M

- ➢ Fixed M, find Y:
  - ✓ Find cluster for each data point

$$\mathbf{y}_i = \arg\min_{\mathbf{y}_i} \sum_{j=1}^{K} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2 \quad (3)$$

$$\text{subject to: } y_{ij} \in \{0, 1\} \ \forall j; \quad \sum_{j=1}^{K} y_{ij} = 1$$

  - ✓ Only one yi =1: $j = \arg\min_j \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$
  - ✓ Distance xi to mj: $\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$ **~~ xi belong nearest center**

- ➢ Fixed Y, find M
  - ✓ Find center for each cluster

$$\mathbf{m}_j = \arg\min_{\mathbf{m}_j} \sum_{i=1}^{N} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2 \quad (4)$$

  - ✓ Derivative: $\frac{\partial l(\mathbf{m}_j)}{\partial \mathbf{m}_j} = 2 \sum_{i=1}^{N} y_{ij} (\mathbf{m}_j - \mathbf{x}_i) \quad (5)$

$$\mathbf{m}_j \sum_{i=1}^{N} y_{ij} = \sum_{i=1}^{N} y_{ij} \mathbf{x}_i \quad (6)$$

  - ✓ Derivative =0 : $\Rightarrow \mathbf{m}_j = \frac{\sum_{i=1}^{N} y_{ij} \mathbf{x}_i}{\sum_{i=1}^{N} y_{ij}}$
  - ✓ mj is the average of the points in the cluster j.

K-Means Clustering

Processing

- **Input:**
  - ✓ Data- X
  - ✓ Number of cluster- K
- **Output**: the Centers- **M** and Label vector – **Y**

1. Select random K centers.
2. Assign point to cluster that has nearest center.
3. Check: Is there a change from the previous step?
   - ✓ No→ Stop.
   - ✓ Yes-> Go to next step.
4. Update the center for each cluster by taking the average of all data points assigned that cluster after step 2.
5. Go to step 2.

Naive Bayes

Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong independence assumptions between the features.

Bayes Equation:

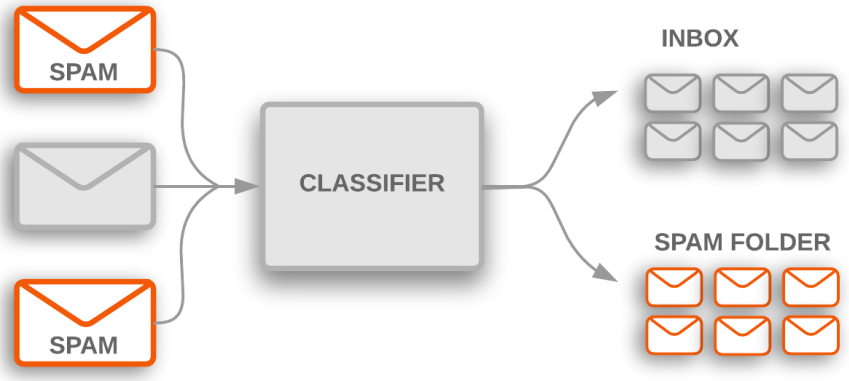
$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

- ✓ P(A|B): probability event A happen when event B happened
- ✓ P(B|A): probability event B happen when event A happened
- ✓ P(A): probability event A happen
- ✓ P(B): probability event B happen

Classification

Classification is the process of predicting the class of given data points.

- Input:
  - ✓ A fixed set of classes C={c1, c2, …, cn}
  - ✓ A new point X belong R domain
- Output: calculate probability that X belong ci: P(ci|X)
- How to select the class that X belong to ?



Naive Bayes Classifier

How to select the class c that X belong to ?

$$\hat{c} = \arg \max_{c \in \{1,\ldots,C\}} p(c|\mathbf{x}) \quad (2)$$

$$\hat{c} = \arg \max_{c} p(c|\mathbf{x}) \quad (3)$$

$$= \arg \max_{c} \frac{p(\mathbf{x}|c)p(c)}{p(\mathbf{x})} \quad 4)$$

$$= \arg \max_{c} p(\mathbf{x}|c)p(c) \quad (5)$$

Machine Learning based Object Recognition

Demo 2

- ✓ Train : CNN + your image + your mentor's Image
- ✓ Input: your group image
- ✓ Output: Identify you and your mentor

Machine Learning based Object Recognition

Demo 2

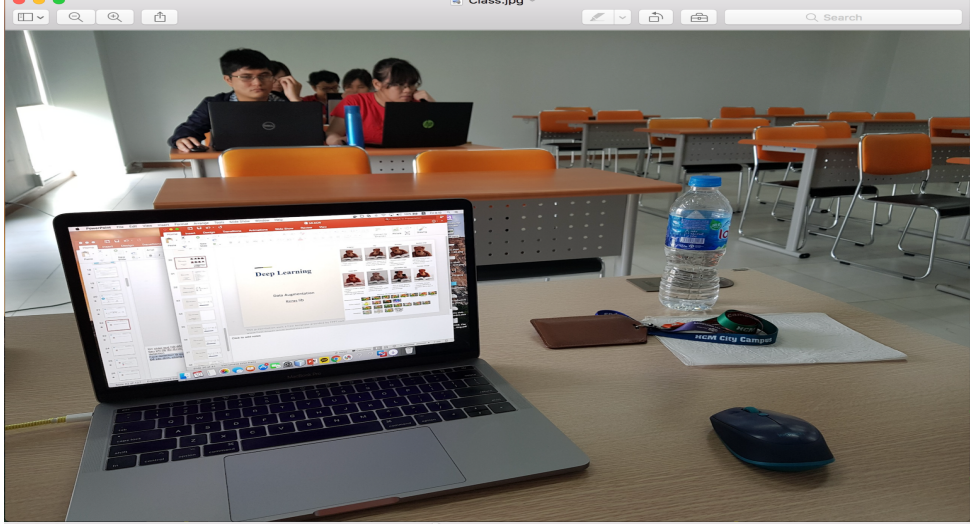
- ✓ Train : CNN + your image + your menor's image
- ✓ Input: your group image
- ✓ Output: Identify you and your mentor

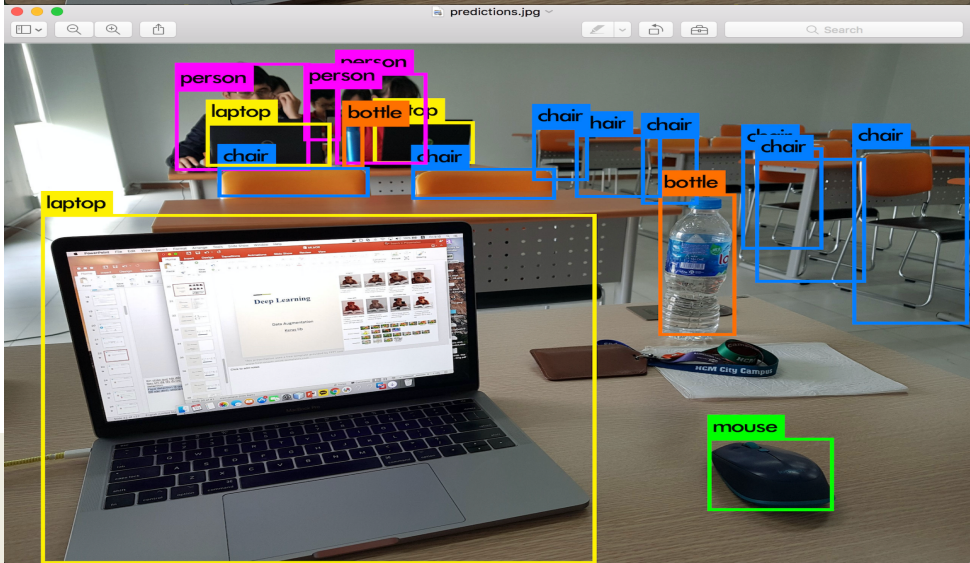
[13]

```python
import face_recognition
import numpy as np
from PIL import Image, ImageDraw
from IPython.display import display

obama_image = face_recognition.load_image_file("Kelly.png")
obama_face_encoding = face_recognition.face_encodings(obama_image)[0]
# Load a second sample picture and learn how to recognize it.
biden_image = face_recognition.load_image_file("Ken.png")
biden_face_encoding = face_recognition.face_encodings(biden_image)[0]

# Create arrays of known face encodings and their names
known_face_encodings = [
    obama_face_encoding,
    biden_face_encoding
]
known_face_names = [
    "Kelly Hu",
    "Ken Lu"
]
# Load an image with an unknown face
unknown_image = face_recognition.load_image_file("group.jpg")
# Find all the faces and face encodings in the unknown image
face_locations = face_recognition.face_locations(unknown_image)
face_encodings = face_recognition.face_encodings(unknown_image, face_locations)
pil_image = Image.fromarray(unknown_image)
# Create a Pillow ImageDraw Draw instance to draw with
draw = ImageDraw.Draw(pil_image)
# Loop through each face found in the unknown image
for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
    # See if the face is a match for the known face(s)
    matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
    name = "Unknown"
    # Or instead, use the known face with the smallest distance to the new face
    face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
    best_match_index = np.argmin(face_distances)
    if matches[best_match_index]:
        name = known_face_names[best_match_index]
    # Draw a box around the face using the Pillow module
    draw.rectangle(((left, top), (right, bottom)), outline=(0, 0, 255))
    # Draw a label with a name below the face
    text_width, text_height = draw.textsize(name)
    draw.rectangle(((left,bottom - text_height - 10),(right, bottom)),fill=(0, 0, 255),outline=(0, 0, 255))
    draw.text((left + 6, bottom - text_height - 5), name, fill=(255, 255, 255, 255))

# Remove the drawing library from memory as per the Pillow docs
del draw
# Display the resulting image
display(pil_image)
```

Machine Learning based Object Recognition

- ✓ laptop: 100%
- ✓ chair: 71%
- ✓ chair: 68%
- ✓ chair: 65%
- ✓ person: 100%
- ✓ person: 88%
- ✓ bottle: 100%
- ✓ chair: 98%
- ✓ bottle: 61%
- ✓ person: 67%
- ✓ chair: 97%
- ✓ chair: 93%
- ✓ chair: 92%
- ✓ chair: 82%
- ✓ laptop: 100%
- ✓ laptop: 100%
- ✓ mouse: 100%





Demo 3: Semantic Segmentation

- ✓ Get existed mode: YOLO
- ✓ Input: your screen image
- ✓ Output: **What does the image contain?**

Thank You