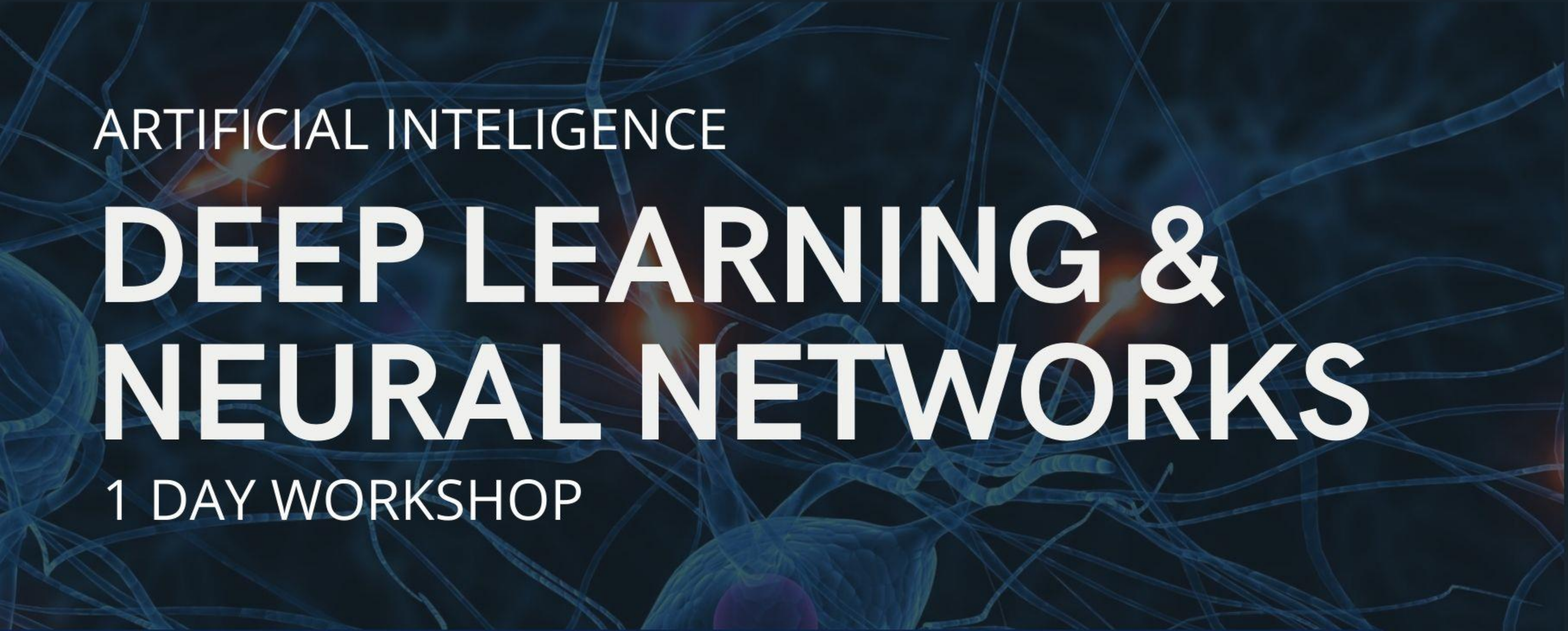


ARTIFICIAL INTELIGENCE

DEEP LEARNING & NEURAL NETWORKS

1 DAY WORKSHOP

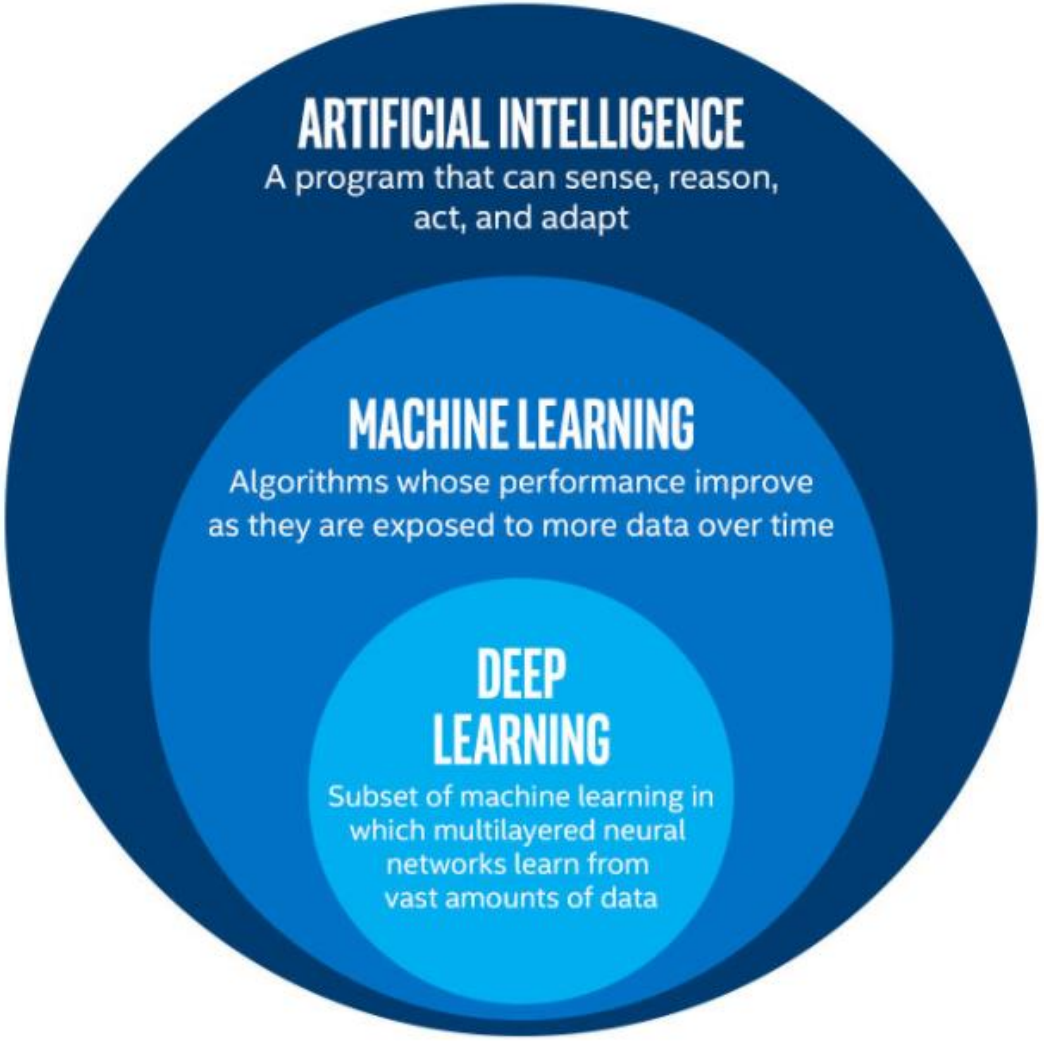
SPEAKER

Thakshila Dasun

BSc. Engineering Hons, CIMA UK, IEEE, IMech

Assistant Lecturer at SLIIT | Senior Lecturer at Academy of Innovative Education

AI vs ML vs DL


ARTIFICIAL INTELLIGENCE
A program that can sense, reason, act, and adapt
MACHINE LEARNING
Algorithms whose performance improve as they are exposed to more data over time
DEEP LEARNING
Subset of machine learning in which multilayered neural networks learn from vast amounts of data

Can you Drive Cars ?

So does AI (Convolutional Neural Networks)

Human-Centered
Autonomous
Vehicle

Can you Identify Objects ?

So does AI (Convolutional Neural Networks)

YOLO v2

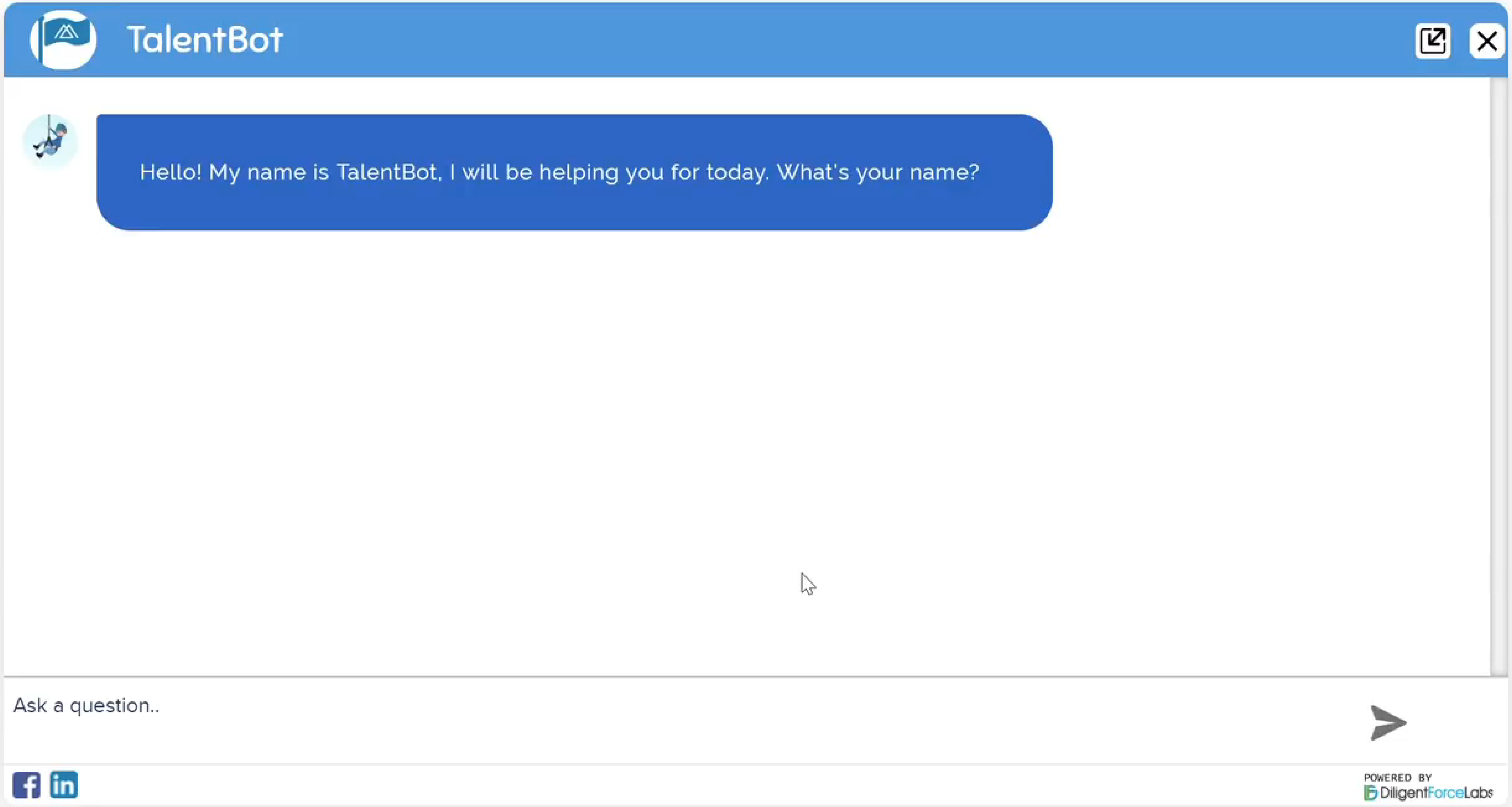
http://pjreddie.com/yolo

Can you Play Games?

So does AI (Deep Q Nets/ CNN)

Can you Read?

So does AI (Recurrent Neural Networks/LSTMs)



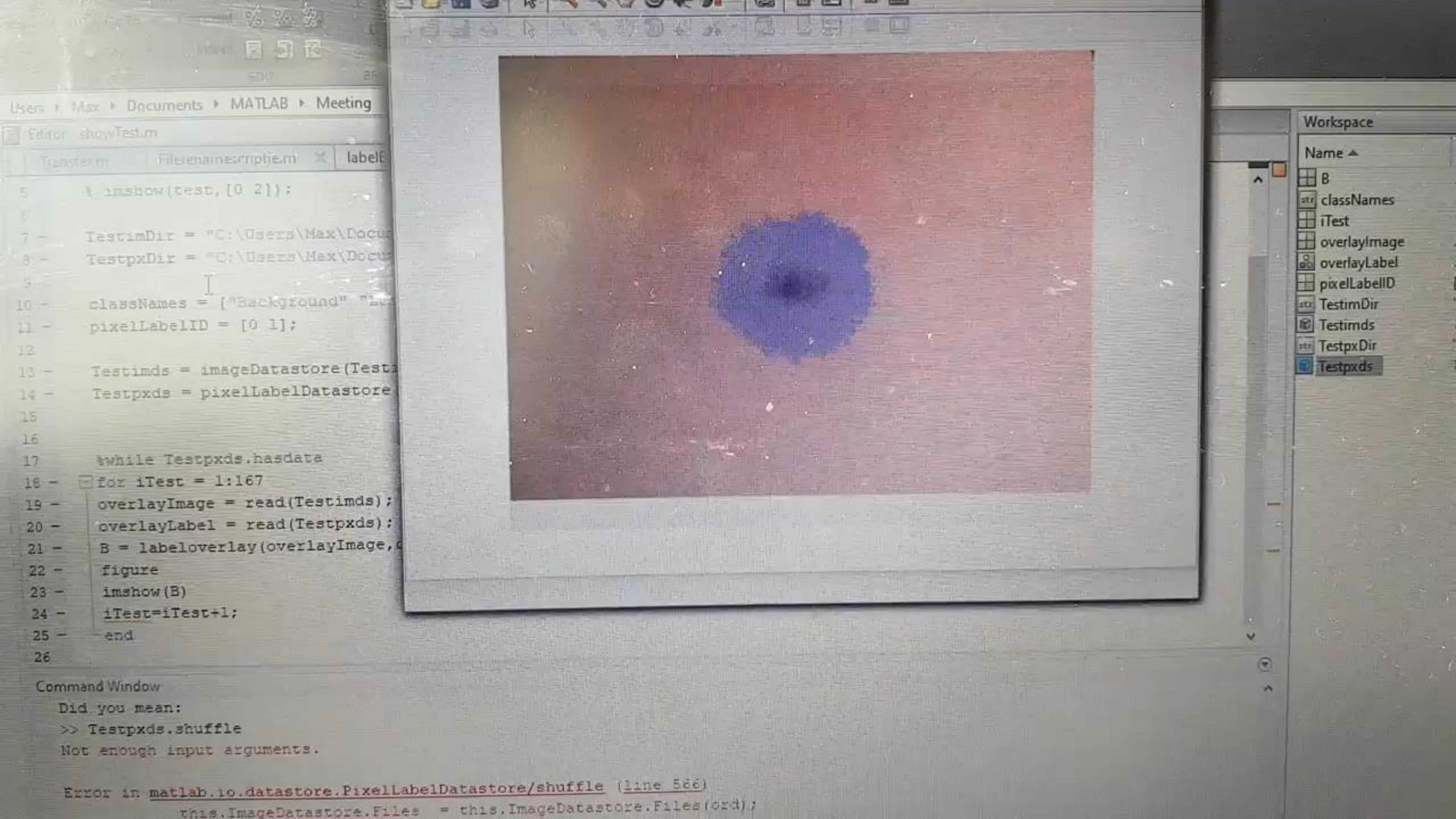
Hello! My name is TalentBot, I will be helping you for today. What's your name?

Music Composing?

So does AI (Autoencorders, GANs)

Health Care?

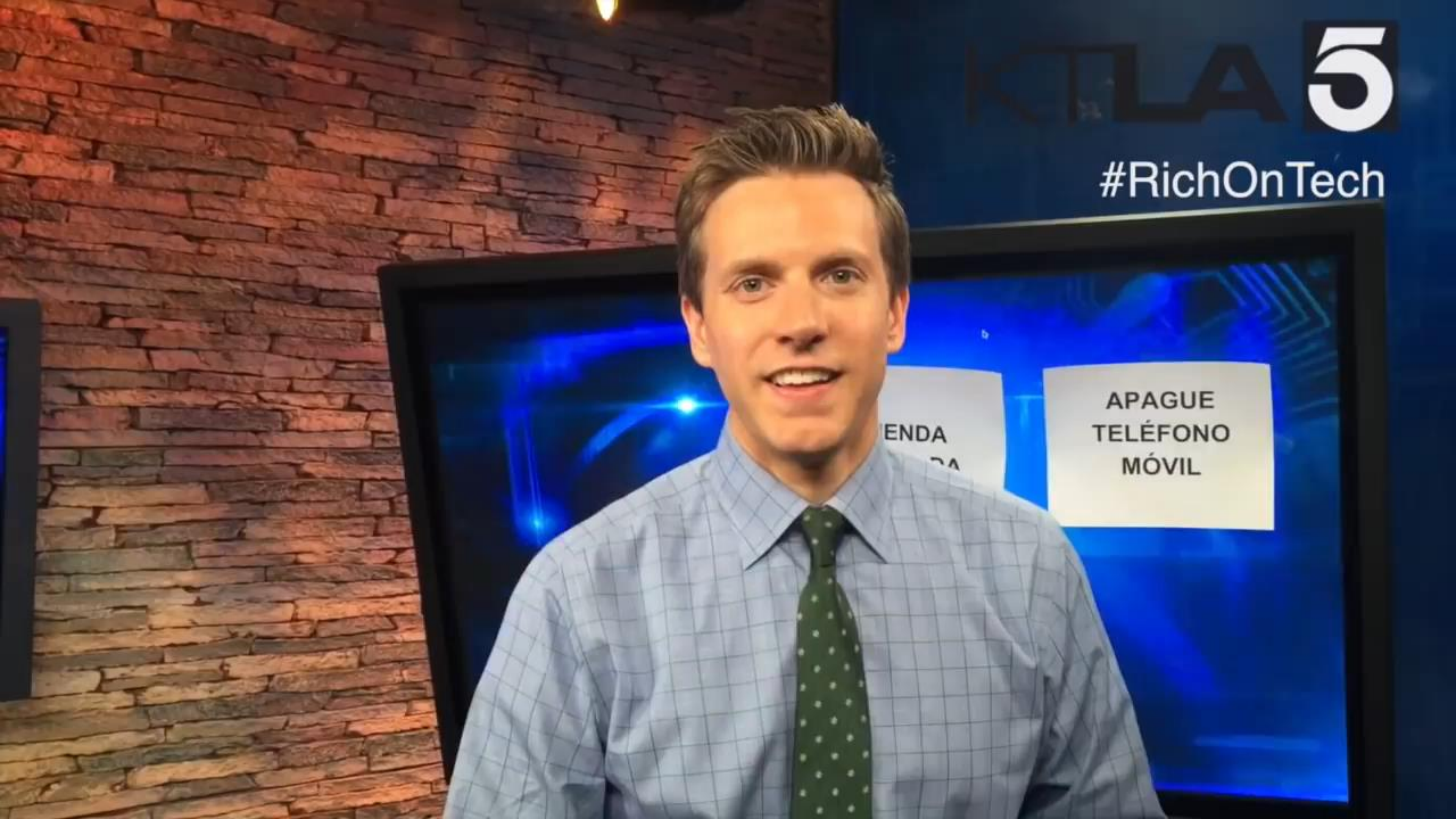
So does AI (Autoencorders, CNNs)



```
% imshow(test,[0 2]);

TestimDir = "C:\Users\Max\Docu
TestpxDir = "C:\Users\Max\Docu

classNames = ["Background" "
pixelLabelID = [0 1];

Testimds = imageDatastore(Test
Testpxds = pixelLabelDatastore


%while Testpxds.hasdata
for iTest = 1:167
 overlayImage = read(Testimds);
 overlayLabel = read(Testpxds);
 B = labeloverlay(overlayImage,
 figure
 imshow(B)
 iTest=iTest+1;
 end
```

```
Did you mean:
>> Testpxds.shuffle
Not enough input arguments.

Error in matlab.io.datastore.PixelLabelDatastore/shuffle (line 566)
        this.ImageDatastore.Files = this.ImageDatastore.Files(ord);
```

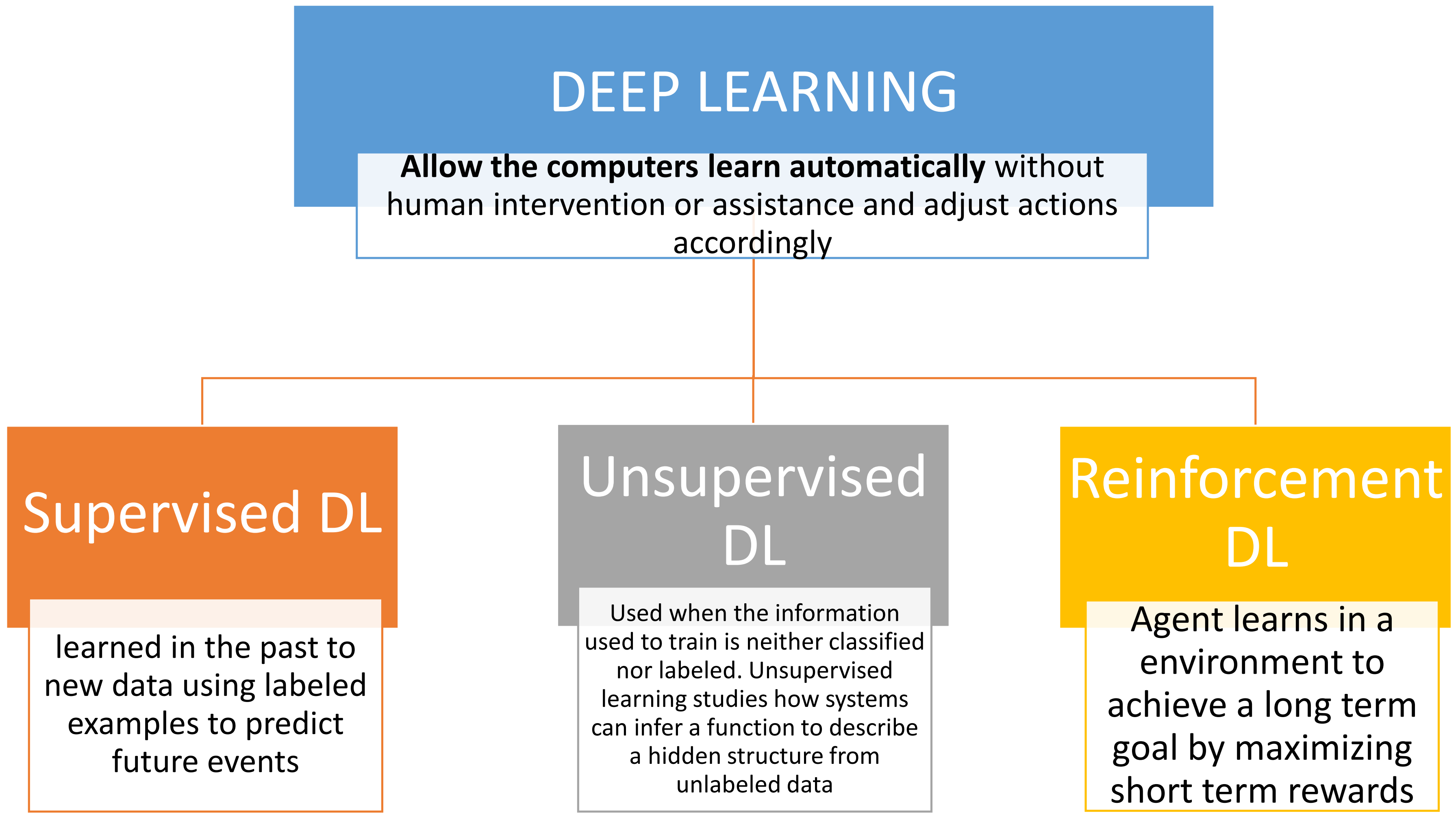
Machine Translation


KTLA 5
#RichOnTech
IENDA
DA
APAGUE
TELÉFONO
MÓVIL

What More?

- Maps
- Stock Market Predictions
- Health Care
- Marketing/ Recommendations
- Sentiment Analysis
- Finance/ Fraud Detection
- Digital Assistants
- Ads, search, social recommendations

- Face Recognition
- Image Classification
- Speech Recognition
- Text-to-speech Generation
- Handwriting Transcription
- Machine Translation
- Medical Diagnosis
- Self Driving Cars
- Games with Deep Reinforcement Learning



DEEP LEARNING

Allow the computers learn automatically without human intervention or assistance and adjust actions accordingly

Supervised DL

learned in the past to new data using labeled examples to predict future events

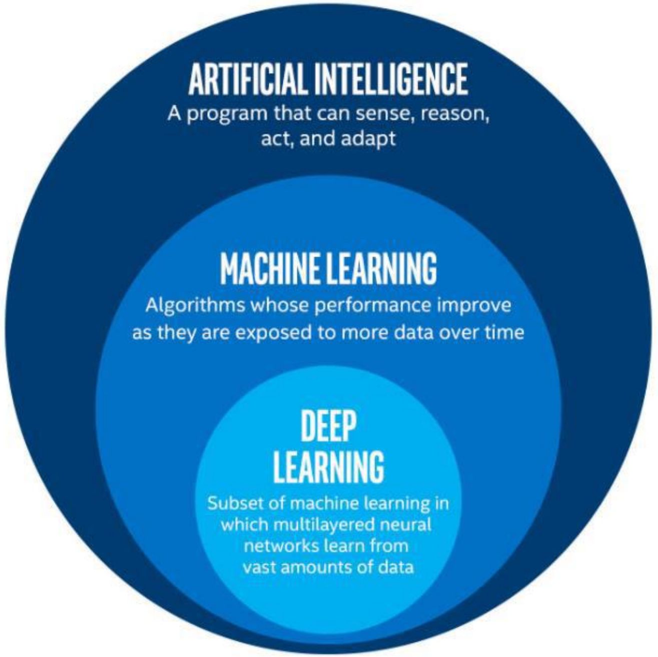
Unsupervised DL

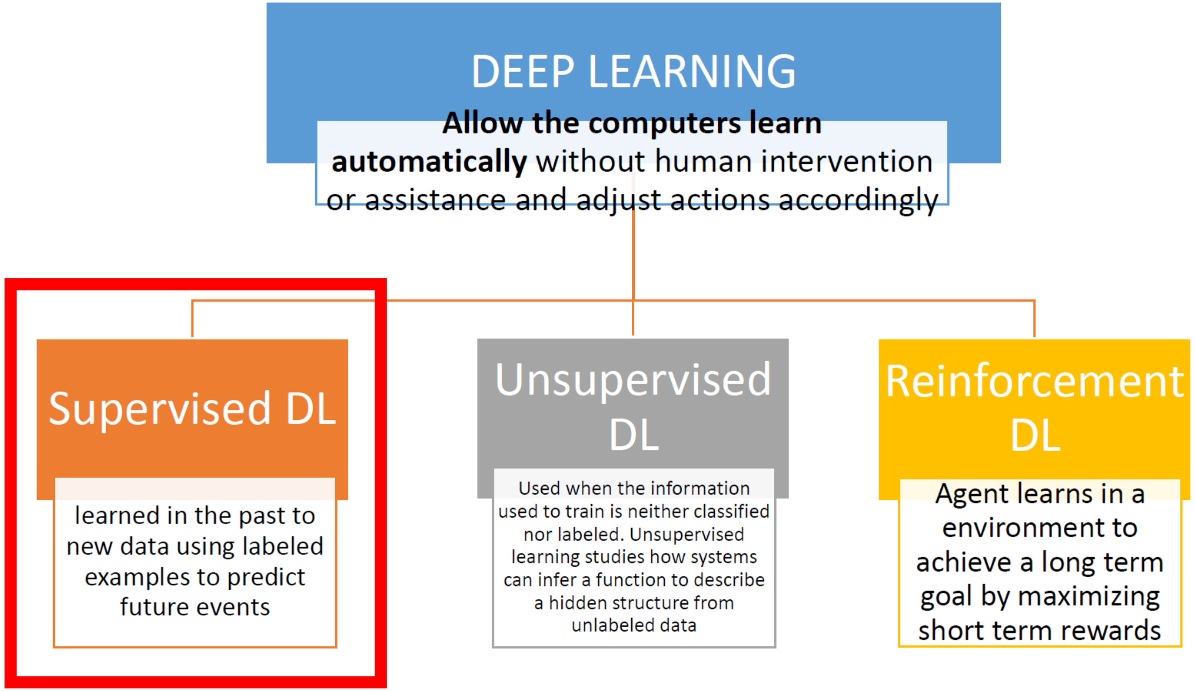
Used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data

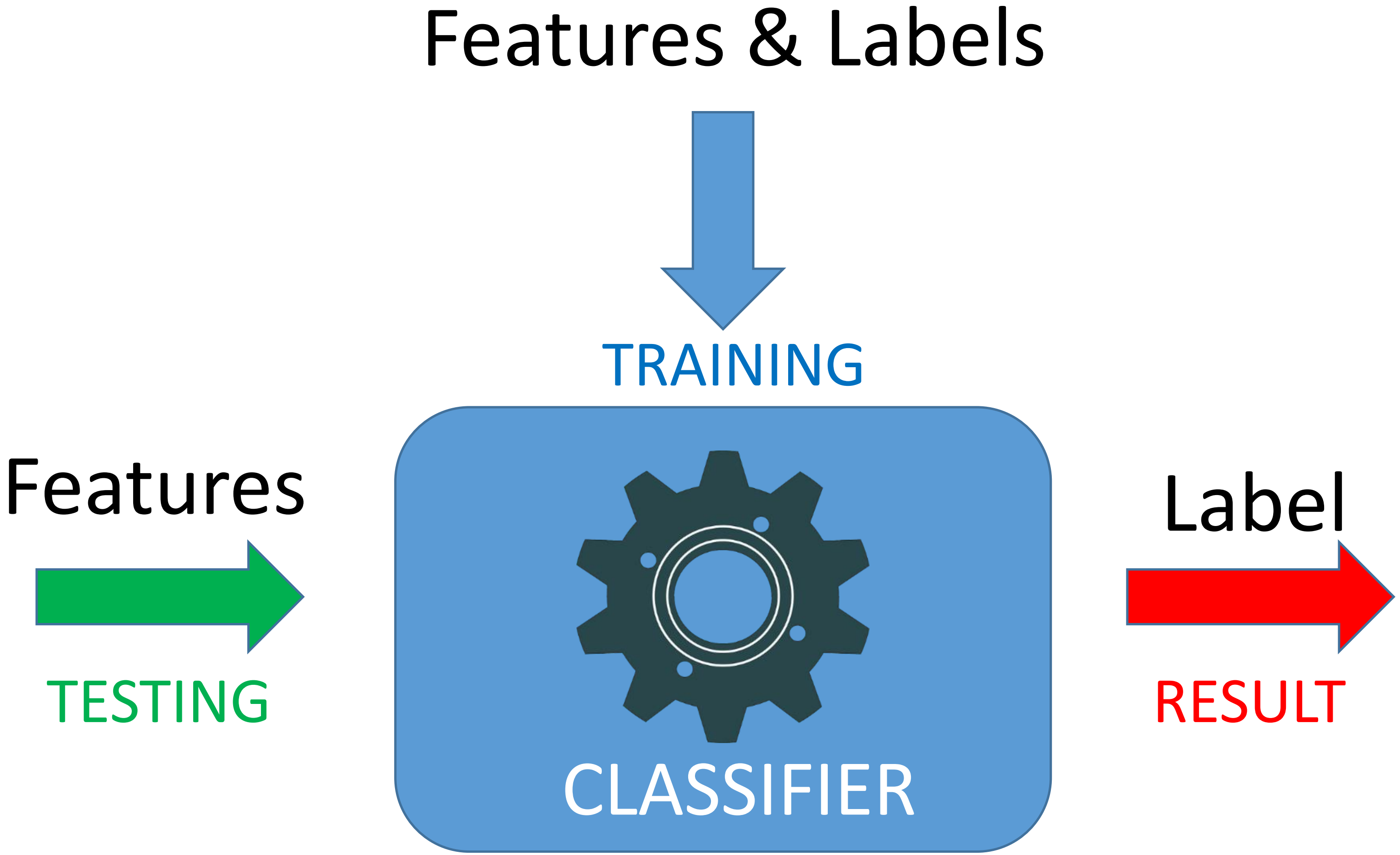
Reinforcement DL

Agent learns in a environment to achieve a long term goal by maximizing short term rewards

Focus Today


ARTIFICIAL INTELLIGENCE
A program that can sense, reason, act, and adapt
MACHINE LEARNING
Algorithms whose performance improve as they are exposed to more data over time
DEEP LEARNING
Subset of machine learning in which multilayered neural networks learn from vast amounts of data


DEEP LEARNING
Allow the computers learn automatically without human intervention or assistance and adjust actions accordingly
Supervised DL
learned in the past to new data using labeled examples to predict future events
Unsupervised DL
Used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data
Reinforcement DL
Agent learns in a environment to achieve a long term goal by maximizing short term rewards

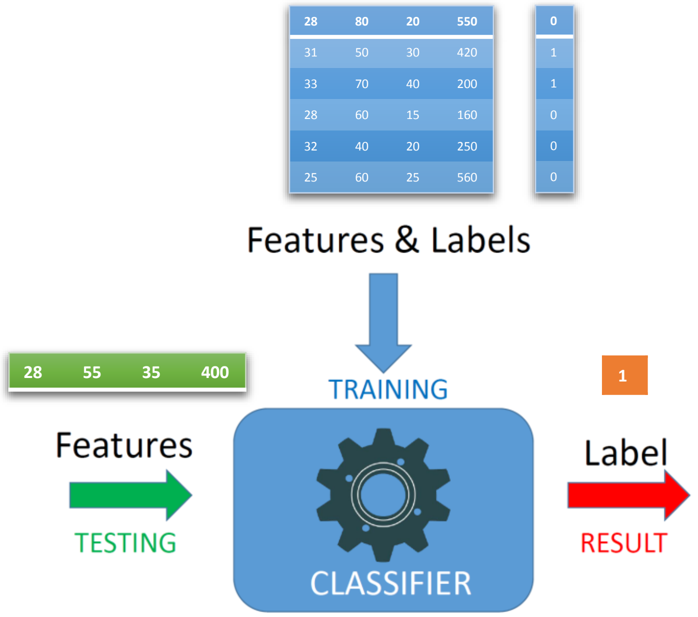

Features & Labels
TRAINING
Features
TESTING
CLASSIFIER
Label
RESULT

Predicting whether it is going to RAIN today or NOT

1. Identify Feature and Labels
   - Labels: Possible solution of the problem
   - Feature: Critical Attribute that decides the labels
2. Create a dataset

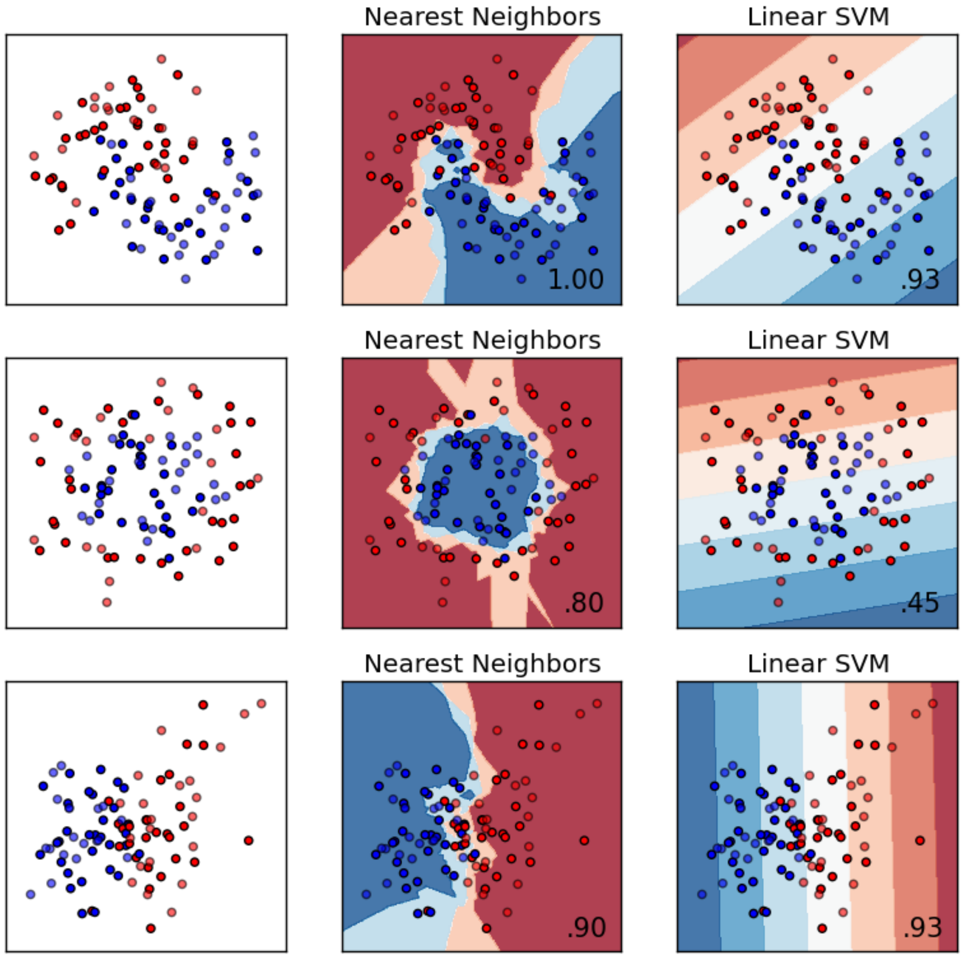
Features				Labels
Temperature C	Humidity %	Wind Speed/ Kmph	Light Intensity/ lux	
28	80	20	550	0
31	50	30	420	1
33	70	40	200	1
28	60	15	160	0
32	40	20	250	0
25	60	25	560	0

3. Train the NN
4. Test & Results



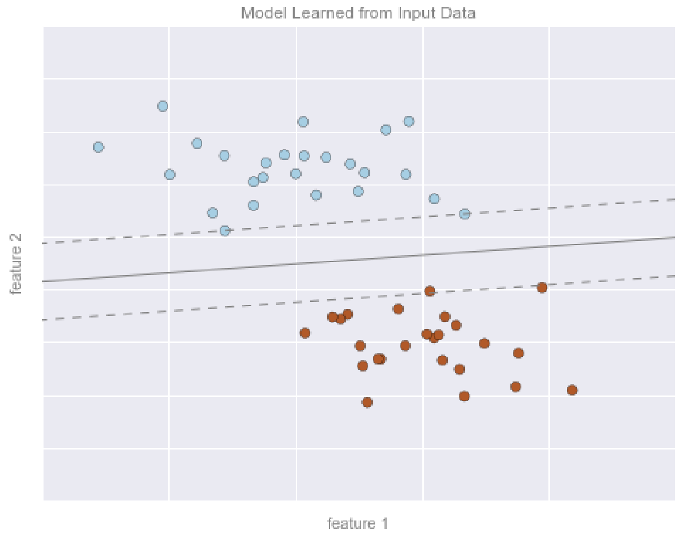
Classification & Regression

2 Types of Predictions in Machine Learning, Qualitative and Quantitative



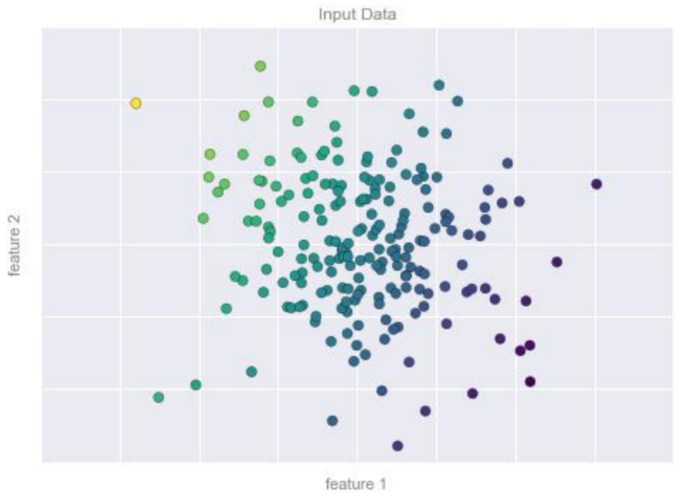
1. Classification: Predicting discrete labels

- “Classification" indicates that the data has discrete class label.
- Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y) or classes.
- The output variables are often called labels or categories. The mapping function predicts the class or category for a given observation



2. Regression: Predicting continuous labels

- Regression predictive modeling is the task of approximating a mapping function (f) from input variables (X) to a continuous output variable (y).
- A continuous output variable is a real-value, such as an integer or floating point value. These are often quantities, such as amounts and sizes.
- For example, a house may be predicted to sell for a specific dollar value, perhaps in the range of $100,000 to $200,000.



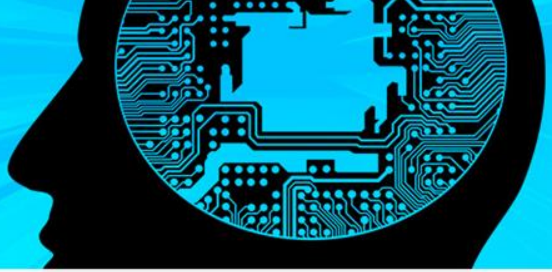
Why You Should Learn ML

- It is the Future Industry
- Research Opportunities for Undergraduates/ Post Graduates
- Data Scientist
- Big Data analysis
- AI Engineers
- AI based IOT and Mobile App Development
- Finance and Statistics Sector
- Stock Market Data Analysis, share value Prediction
- Automobile Industry
- Medical Industry

DEEP LEARNING

“In a neural network we don't tell the computer how to solve our problem. Instead, it learns from observational data, figuring out its own solution to the problem at hand.”

-In 2006 was the discovery of techniques for learning in so-called deep neural networks. These techniques are now known as deep learning. They've been developed further, and today deep neural networks and deep learning achieve outstanding performance on many important problems in computer vision, speech recognition, and natural language processing-



They use Deep Learning

airbnb · Coca-Cola · DeepMind · GE Healthcare · Google · intel · NERSC

AIRBUS DEFENCE & SPACE · AMD · arm · BAKER HUGHES a GE company · BITMAIN · Bloomberg · caicloud 才云 · CAMBIA HEALTH SOLUTIONS

Castbox · CEVA · cheetahmobile · CI&T · Dropbox · ebay · 流利说 · IBM Power Systems

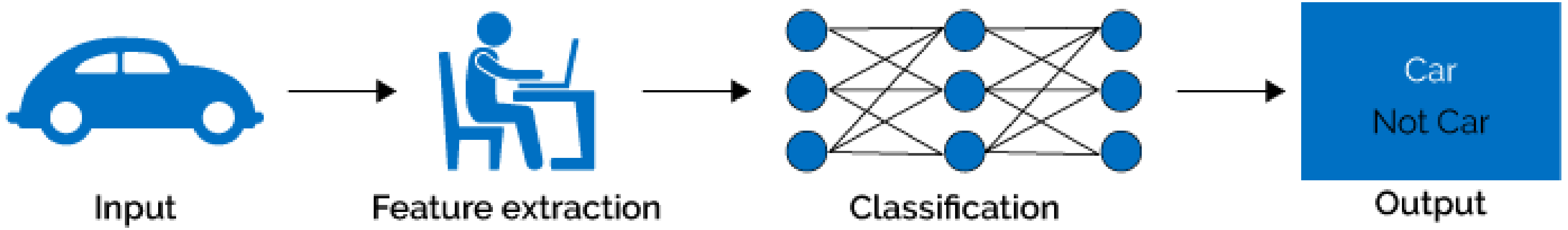
JD.COM · kakao · kika · Lenovo · LinkedIn · Matroid · MEDIATEK · 美团 meituan.com

mi · Mobvoi · Movidius an Intel company · musical.ly · 網易 NetEase · NVIDIA · PayPal · Qualcomm

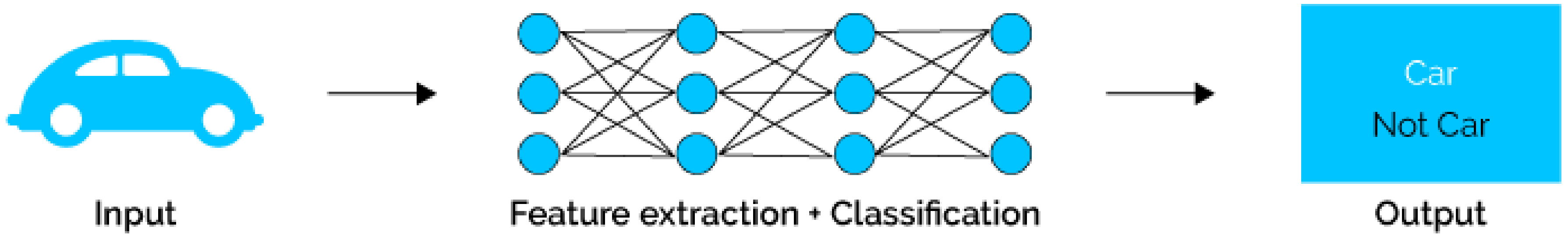
quantiphi · SAP · SINOVATION VENTURES · Sogou 搜狗 · 腾讯社交广告 Tencent Social Ads · Uber · 云知声 Unisound

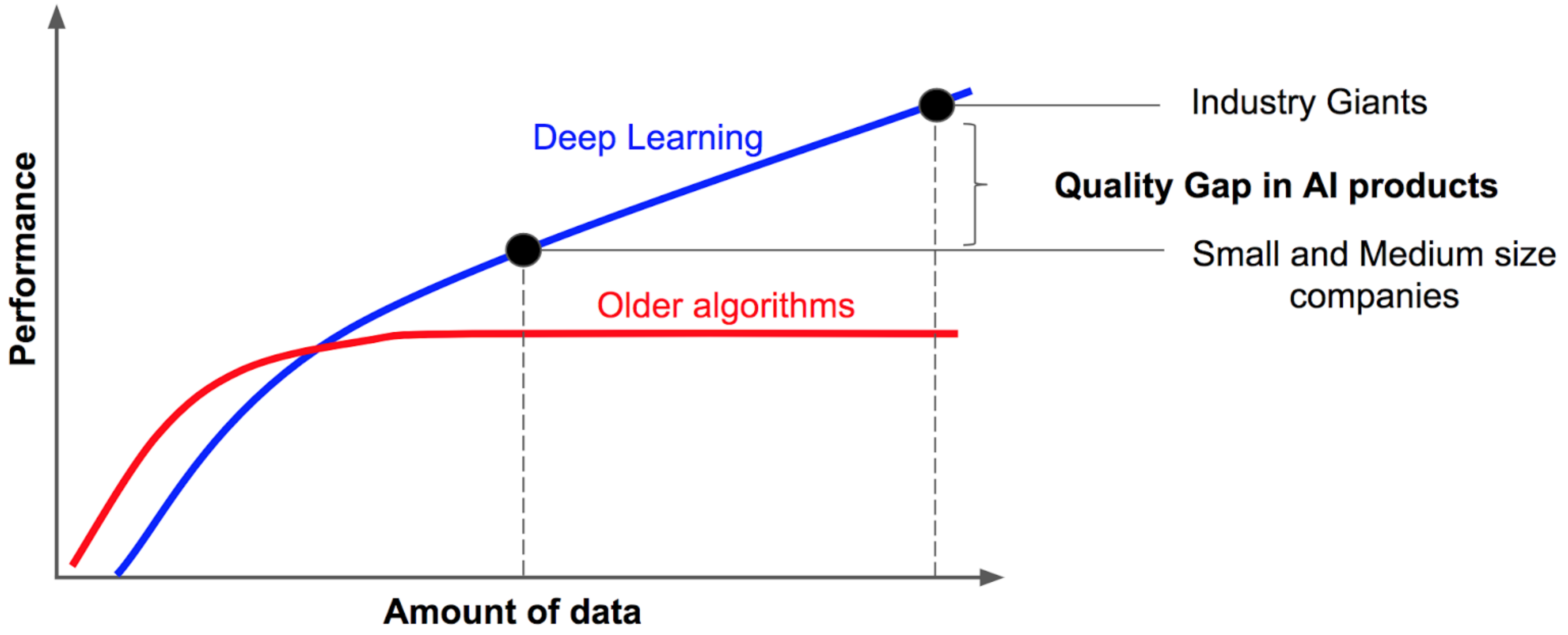
360 TOTAL SECURITY · 3DR

Machine Learning



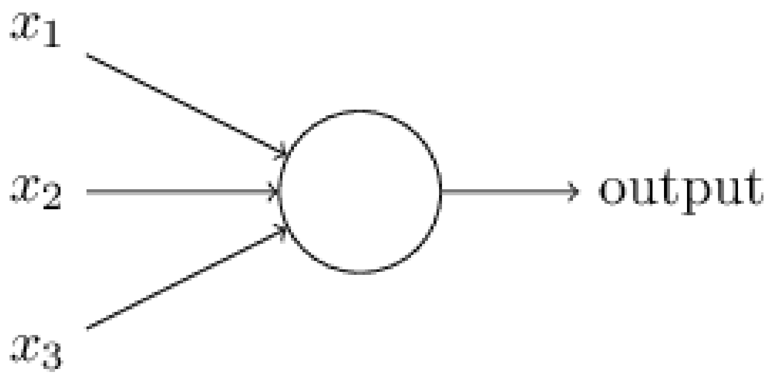
Deep Learning




Performance
Amount of data
Deep Learning
Older algorithms
Industry Giants
Quality Gap in AI products
Small and Medium size companies

History In Brief (1)

- The idea of neural networks began unsurprisingly as a model of how neurons in the brain function.
  - 1943: Portrayed with a simple electrical circuit by neurophysiologist Warren McCulloch and mathematician Walter Pitt
  - 1950-1960: Perceptrons were developed by the scientist Frank Rosenblatt,



History In Brief (2)

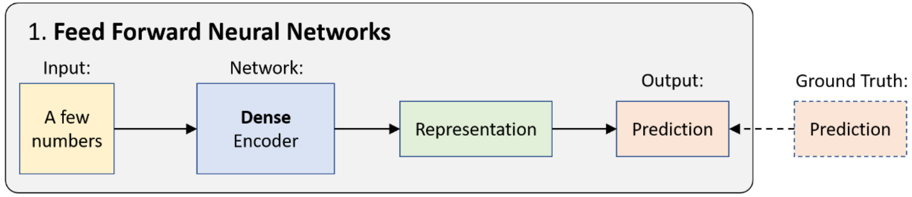
- 1974-86: Backpropagation Algorithm, Recurrent NL
- 1989-98: Convolutional Neural Networks, Bi Directional RNN, Long Short Term Memory (LSTM), MNIST Data Set
- 2006: “Deep Learning” Concept
- 2009: ImageNet
- 2012: AlexNet, Dropout
- 2014: DeepFace
- 2016: AlphaGo
- 2017: AlphaGo Zero
- 2018: BERT

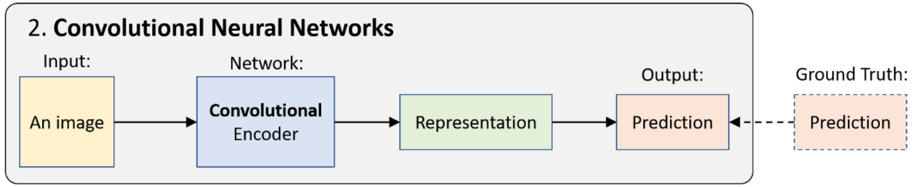
History In Brief (3)

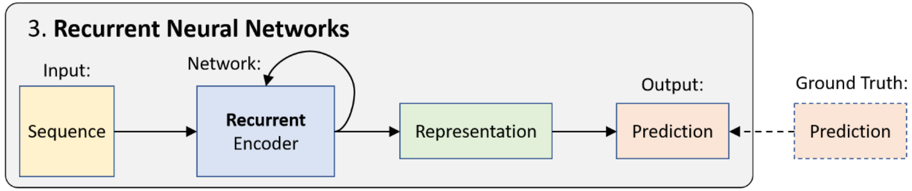
- Mark 1 Perceptron: 1960
- Torch: 2002
- CUDA: 2007
- Theano: 2008
- Caffe: 2015
- Tensorflow 0.1: 2015
- PyTorch 0.1: 2017
- TensorFlow 1.0: 2017
- Tensorflow 2.0: 2019

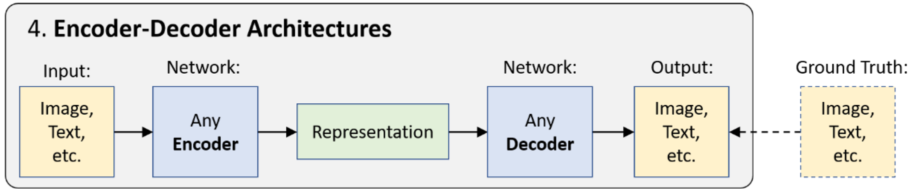
Neural Network Architecture (3)

Supervised Learning

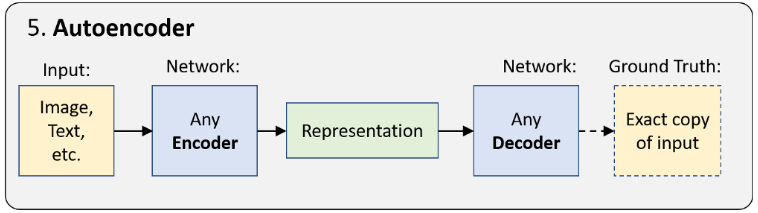


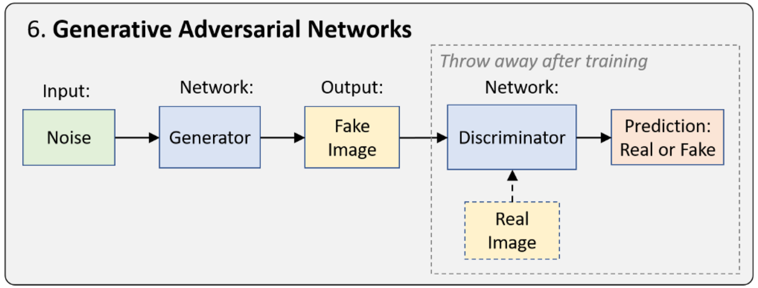




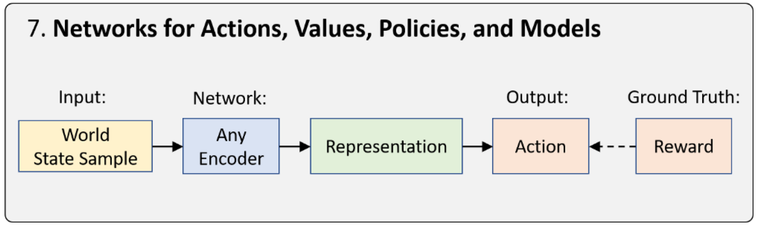


Unsupervised Learning

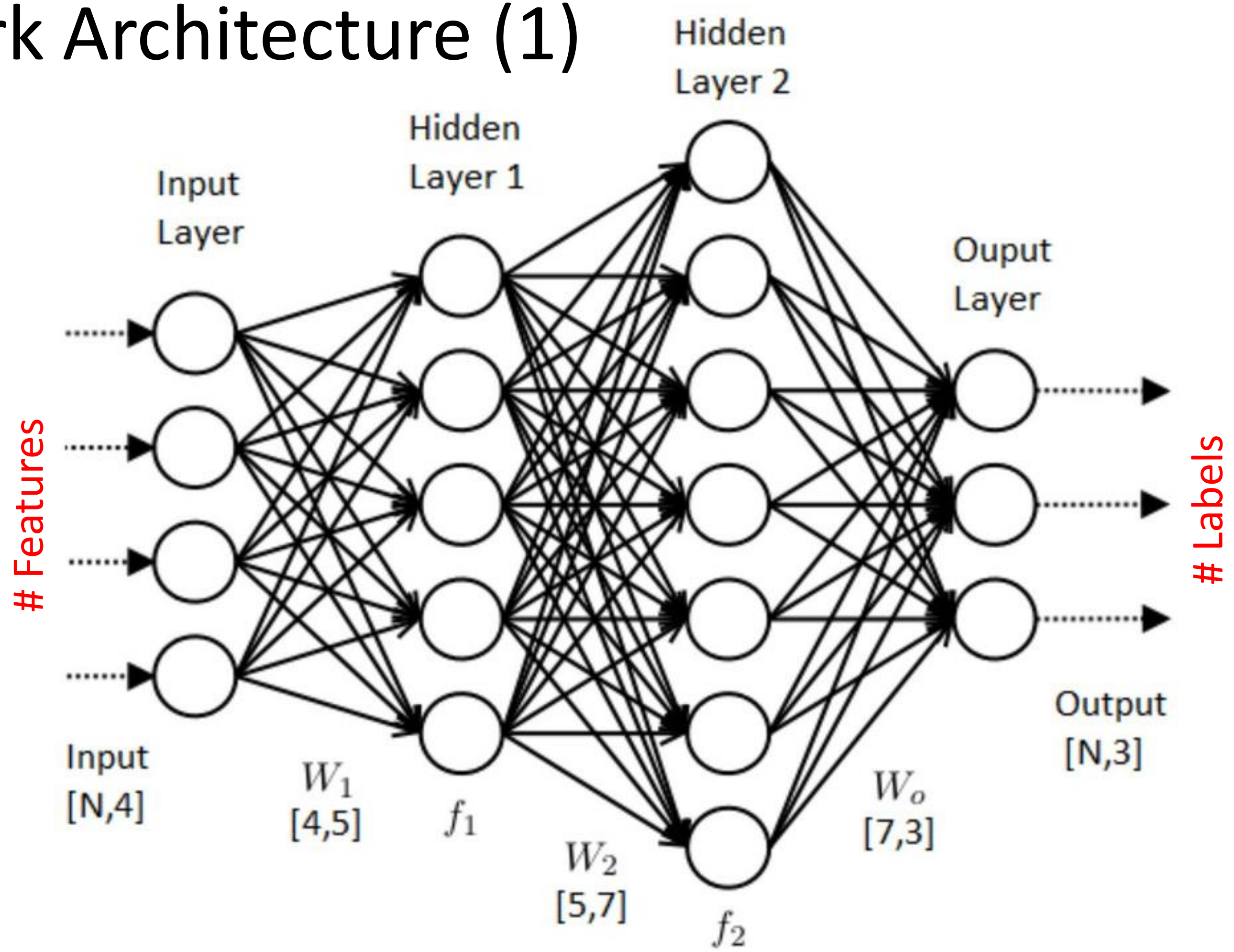




Reinforcement Learning

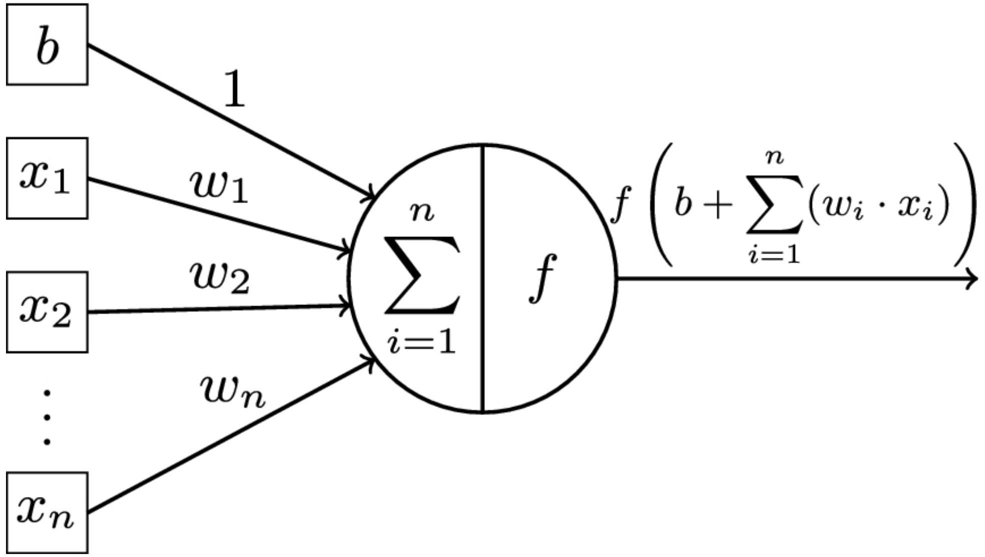


Neural Network Architecture (1)



Perceptron

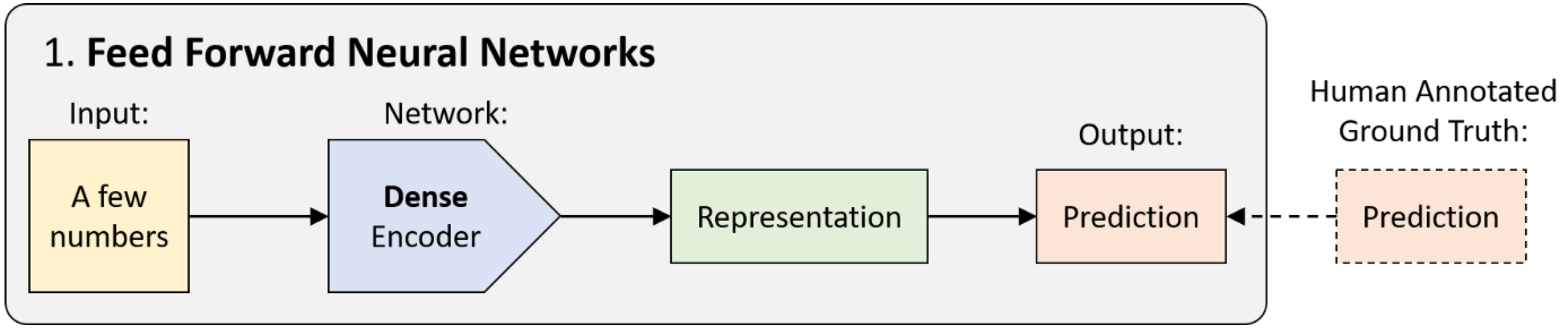
- Building Model of Classic Artificial Neural Networks
- Inputs, x1,x2,x3
- weights, w1,w2,…, real numbers expressing the importance of the respective inputs to the output



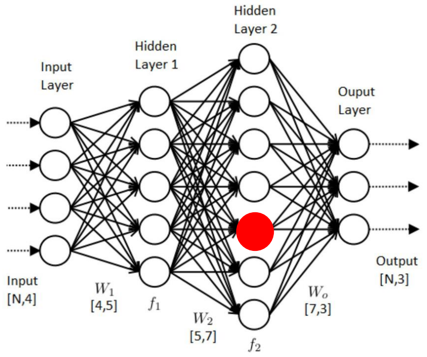
“Perceptrons were developed in the 1950s and 1960s by the scientist Frank Rosenblatt, inspired by earlier work by Warren McCulloch and Walter Pitts”

Feed Forward Neural Networks (FFNNs)

- Data passes from input to output in a single pass without any “state memory” of what came before.
- “FFNN” refers to its simplest variant: a densely-connected multilayer perceptron (MLP).
- Dense encoders are used to map an already compact set of numbers on the input to a prediction: either a classification (discrete) or a regression (continuous).

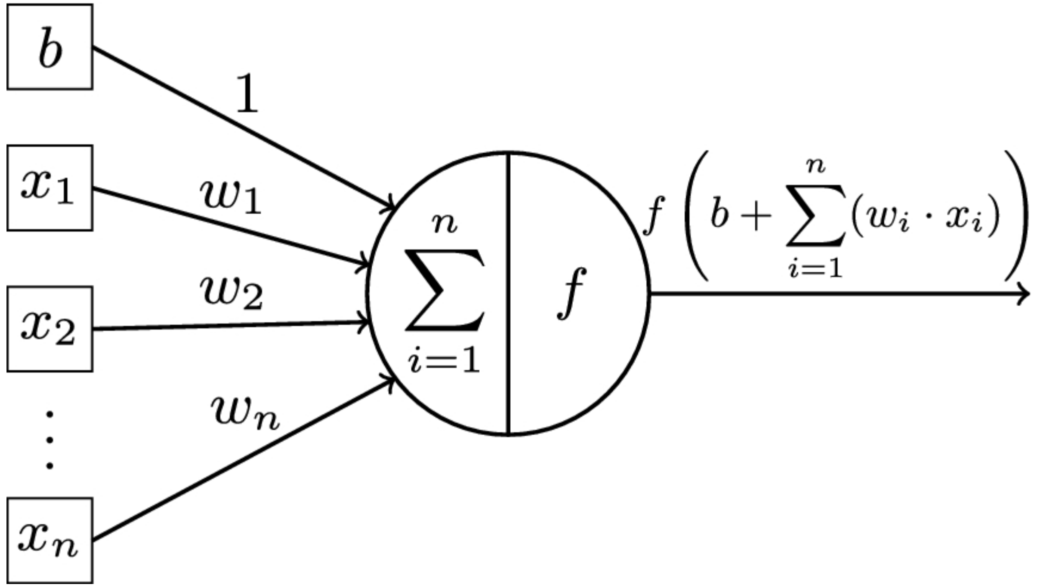


Activation Functions(1)



$$Z = \left(\sum_{i=1}^{n} wi \cdot xi \right) + \text{b}$$

$$Y = F(Z)$$



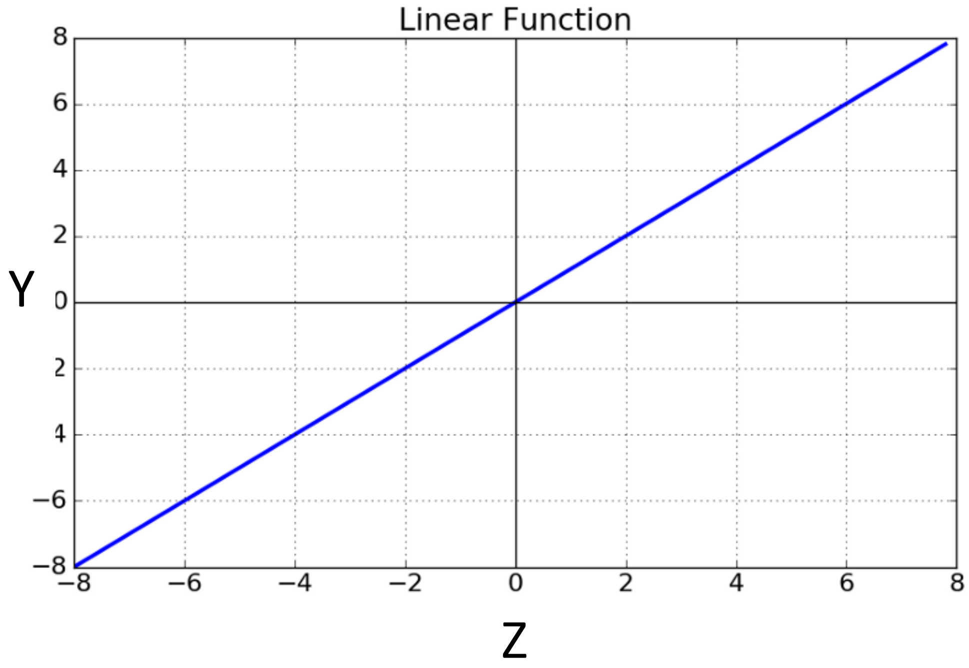
Types of activation functions

- The Activation Functions can be basically divided into 2 types
  - Linear or Identity Activation Function
  - Non-linear Activation Functions

Linear or Identity Activation Function

- The activation is proportional to the input. . This can be applied to various neurons and multiple neurons can be activated at the same time

- **Equation : Y=** F(Z) = Z
- **Range :** (-infinity to infinity)



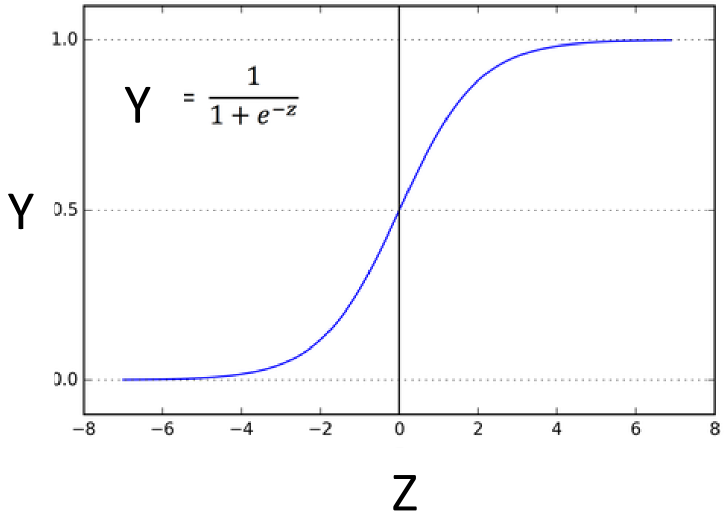
Non-linear Activation Functions

- The Nonlinear Activation Functions are the most used activation functions. It makes it easy for the model to generalize or adapt with variety of data and to differentiate between the output.

1. **Sigmoid or Logistic Activation Function**
2. **Tanh or hyperbolic tangent Activation Function:**
3. **ReLU (Rectified Linear Unit) Activation Function**
4. **Leaky ReLU**
5. **Softmax**

Sigmoid or Logistic Activation Function

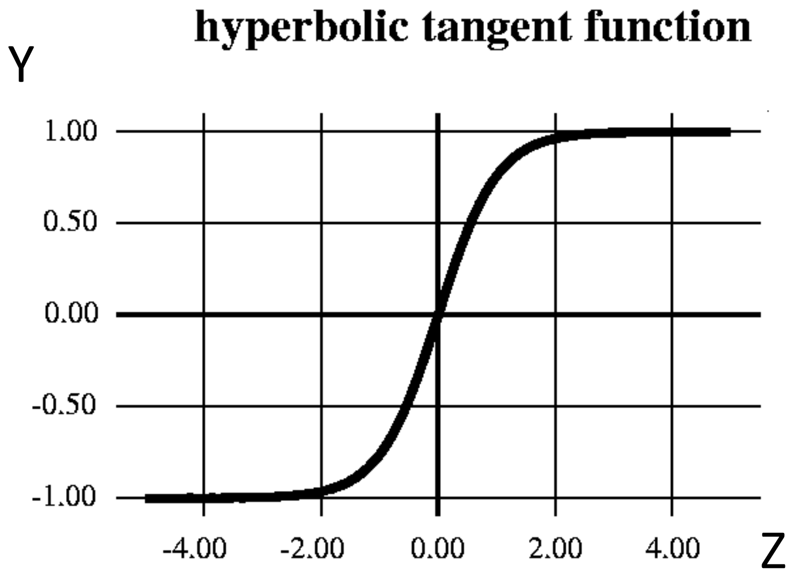
- **Equation : Y=** F(Z) = $\frac{1}{1+e^{-z}}$
- **Range :** (0 to 1)
- It gives rise to a problem of "**vanishing gradients**", since the Y values tend to respond very less to changes in X
- It saturate and kill gradients.



Tanh or hyperbolic tangent Activation Function

- **Equation : Y=** F(Z) = $\frac{1-e^{-2z}}{1+e^{-2z}}$

- **Range :** (-1 to 1)
- It also suffers vanishing gradient problem
- It saturate and kill gradients.

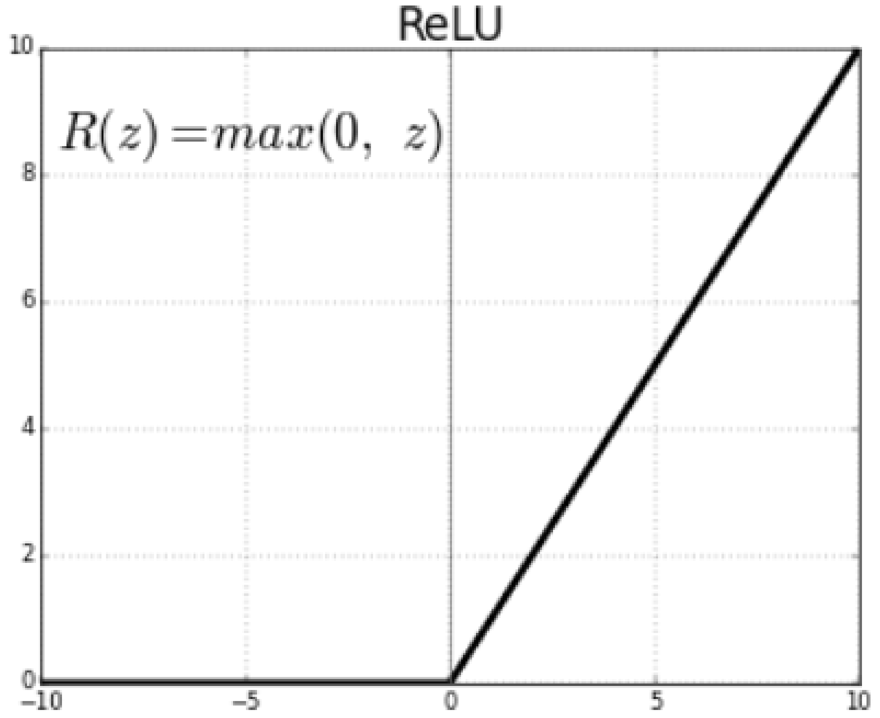


ReLU (Rectified Linear Unit)

- The ReLU is the most used activation function in the world right now

- **Equation : f(Z) = max(0,Z)**
- **Range :** (0 to infinity)

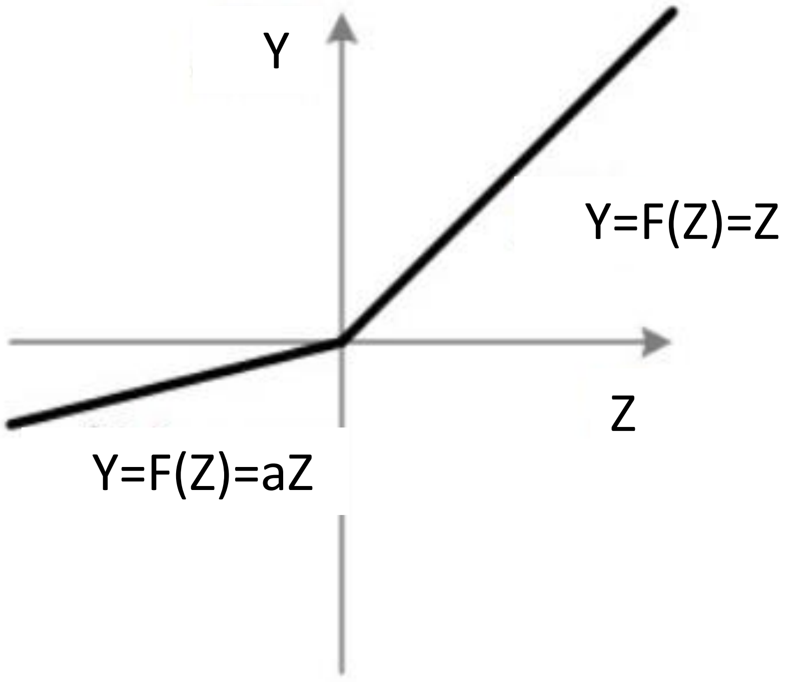
- The outputs are not zero centered similar to the sigmoid activation function
- When the gradient hits zero for the negative values, it does not converge towards the minima which will result in a dead neuron while back propagation



Leaky ReLU

- To solve the ReLU problem we have leaky ReLU

- **Equation :** f(x) = ax for x<0 and x for x>0
- **Range :** (0.01 to infinity)



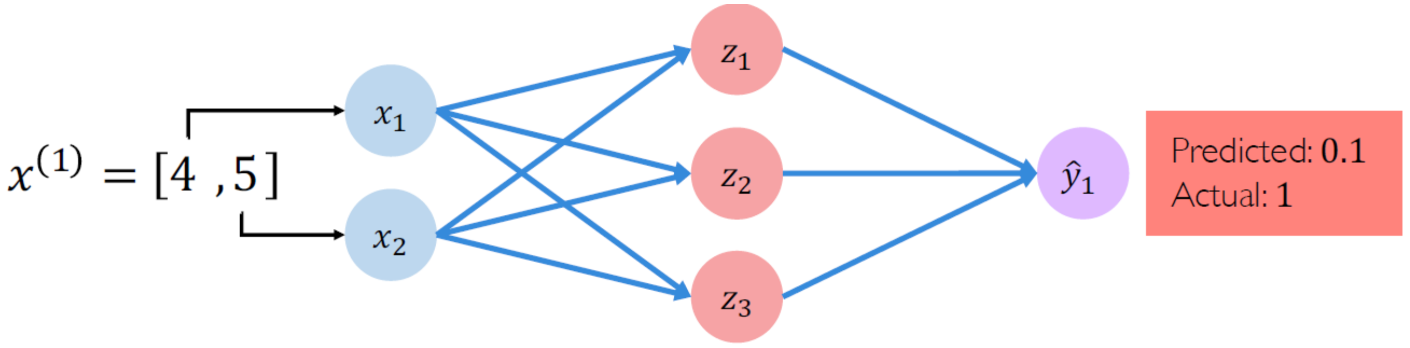
Softmax

- The softmax function is also a type of sigmoid function but it is very useful to handle classification problems having multiple classes.
- The softmax function is shown above, where z is a vector of the inputs to the output layer
- The softmax function is ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Loss Functions

The loss of our network measures the cost incurred from incorrect predictions

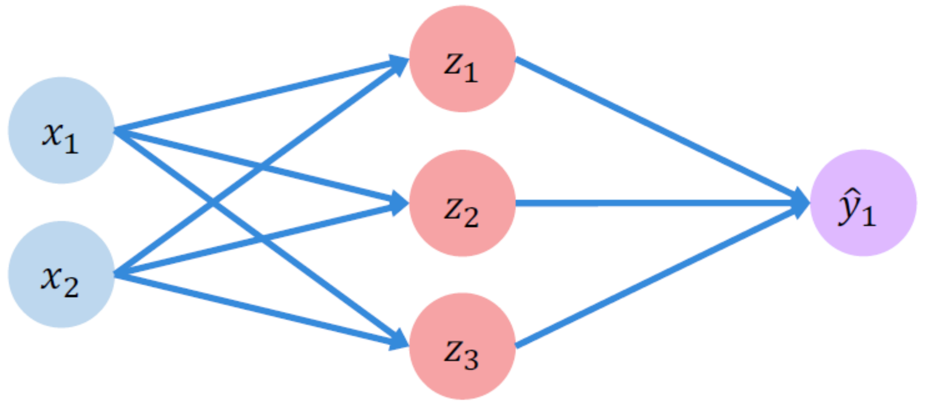


$$\mathcal{L}\left(\underbrace{f\left(x^{(i)}; \boldsymbol{W}\right)}_{\text{Predicted}}, \underbrace{y^{(i)}}_{\text{Actual}}\right)$$

Mean Squared Error

Mean squared error loss can be used with regression models that output continuous real numbers

$$\boldsymbol{X} = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$

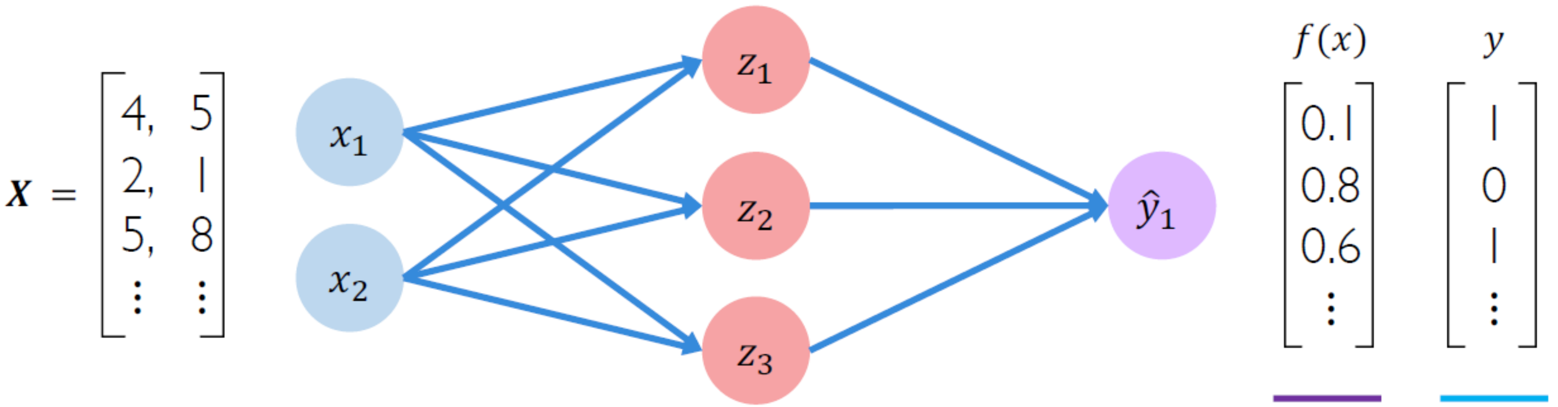


$$\underbrace{\begin{array}{cc} f(x) & y \\ \begin{bmatrix} 30 \\ 80 \\ 85 \\ \vdots \end{bmatrix} & \begin{bmatrix} 90 \\ 20 \\ 95 \\ \vdots \end{bmatrix} \end{array}}_{\text{Final Grades (percentage)}}$$

$$\boldsymbol{J}(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n}\left(\underbrace{y^{(i)}}_{\text{Actual}} - \underbrace{f\left(x^{(i)};\boldsymbol{W}\right)}_{\text{Predicted}}\right)^2$$

Cross Entropy

Cross entropy loss can be used with models that output a probability between 0 and 1


$X = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$
x_1
x_2
z_1
z_2
z_3
$\hat{y}_1$
$f(x)$
$\begin{bmatrix} 0.1 \\ 0.8 \\ 0.6 \\ \vdots \end{bmatrix}$
y
$\begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$

$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{y^{(i)}}_{\text{Actual}} \log\left(\underbrace{f\left(x^{(i)};\boldsymbol{W}\right)}_{\text{Predicted}}\right) + (1 - \underbrace{y^{(i)}}_{\text{Actual}}) \log\left(1 - \underbrace{f\left(x^{(i)};\boldsymbol{W}\right)}_{\text{Predicted}}\right)$$

Loss Optimization

We want to find the network weights that achieve the lowest loss

$$\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f\left(x^{(i)}; \boldsymbol{W}\right), y^{(i)}\right)$$

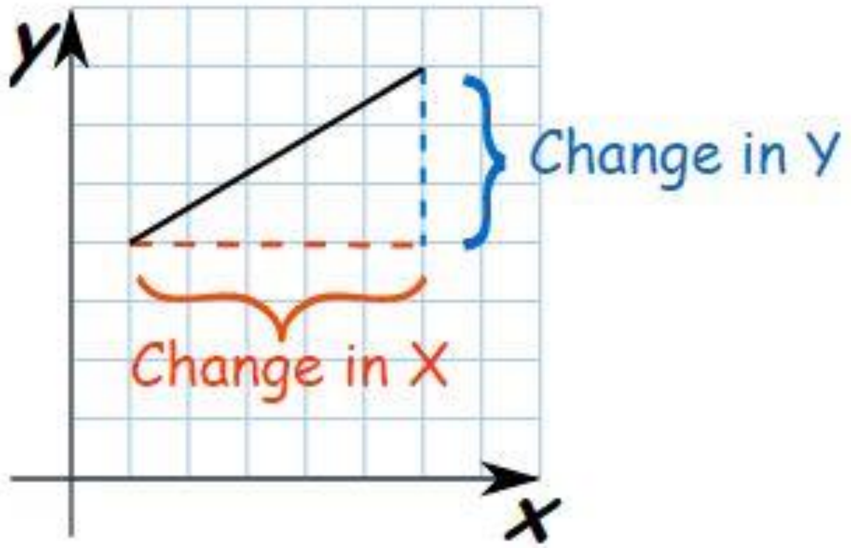
$$\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\operatorname{argmin}}\, J(\boldsymbol{W})$$

Remember:

$$\boldsymbol{W} = \left\{\boldsymbol{W}^{(0)}, \boldsymbol{W}^{(1)}, \cdots\right\}$$

Gradient

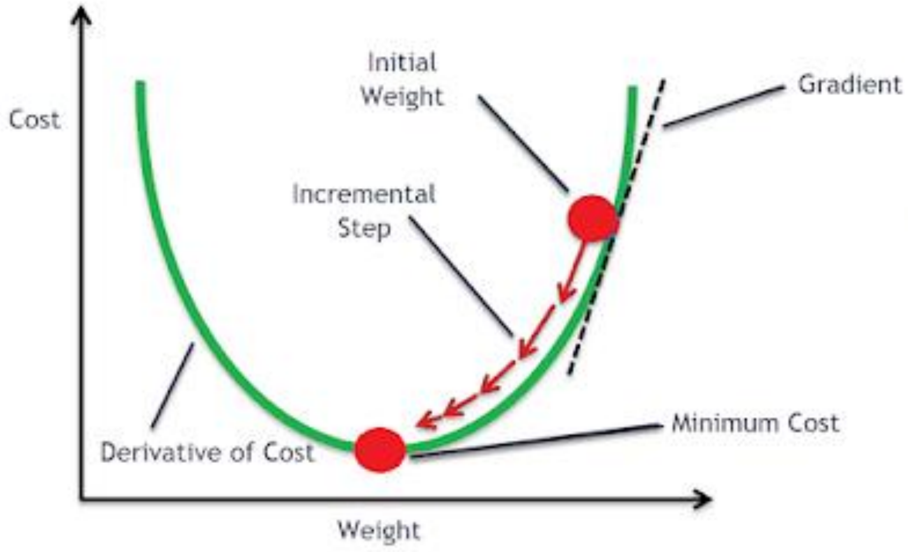
$$\textbf{Gradient} = \frac{\text{Change in Y}}{\text{Change in X}}$$

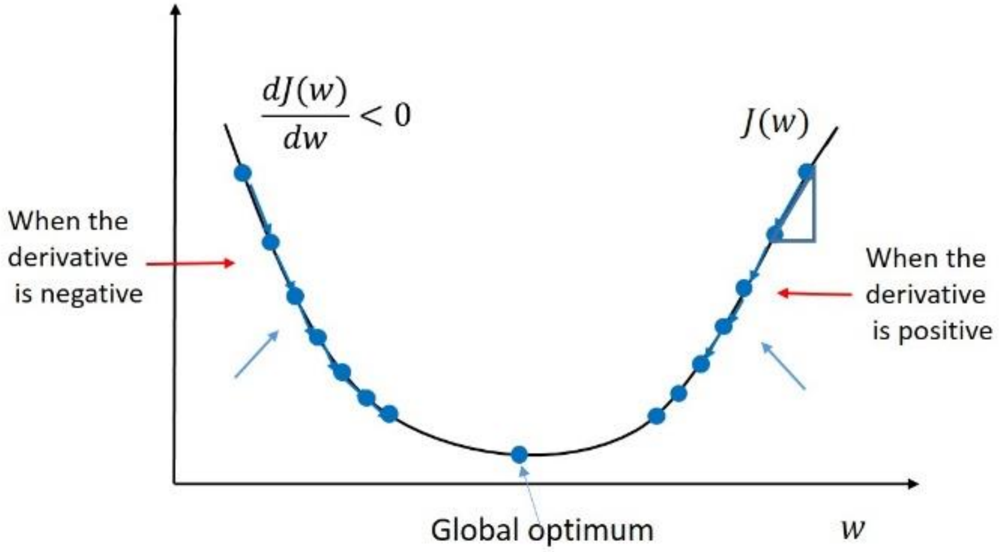


Gradient Descent

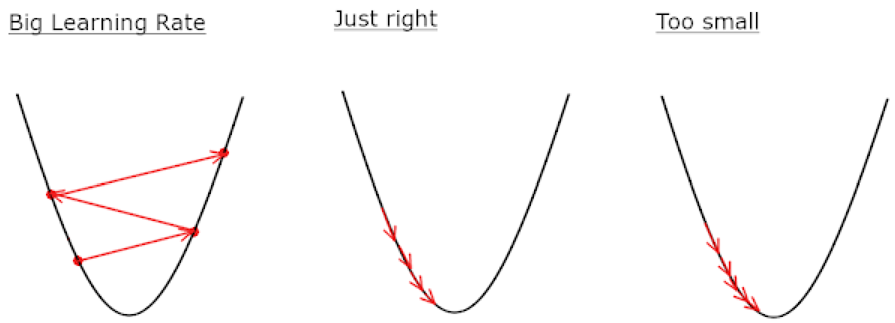
Compute gradient, $\frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$

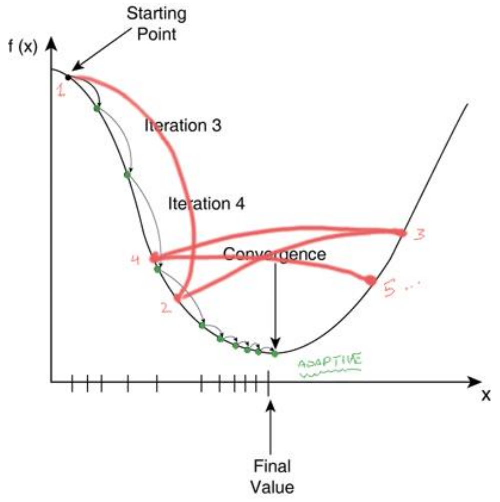
Update weights, $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$

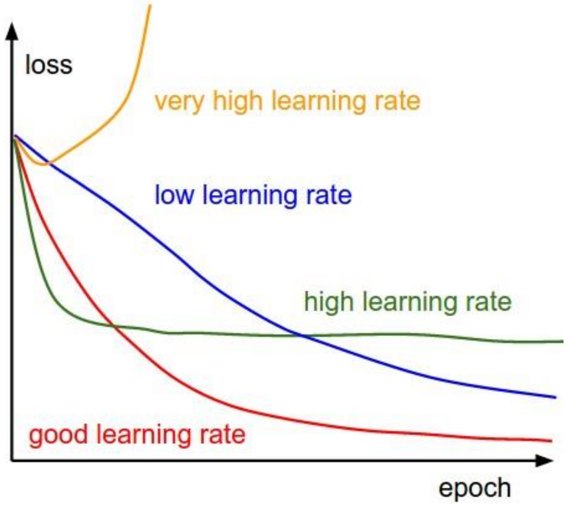



$\frac{dJ(w)}{dw} < 0$
$J(w)$
When the derivative is negative
When the derivative is positive
Global optimum
w

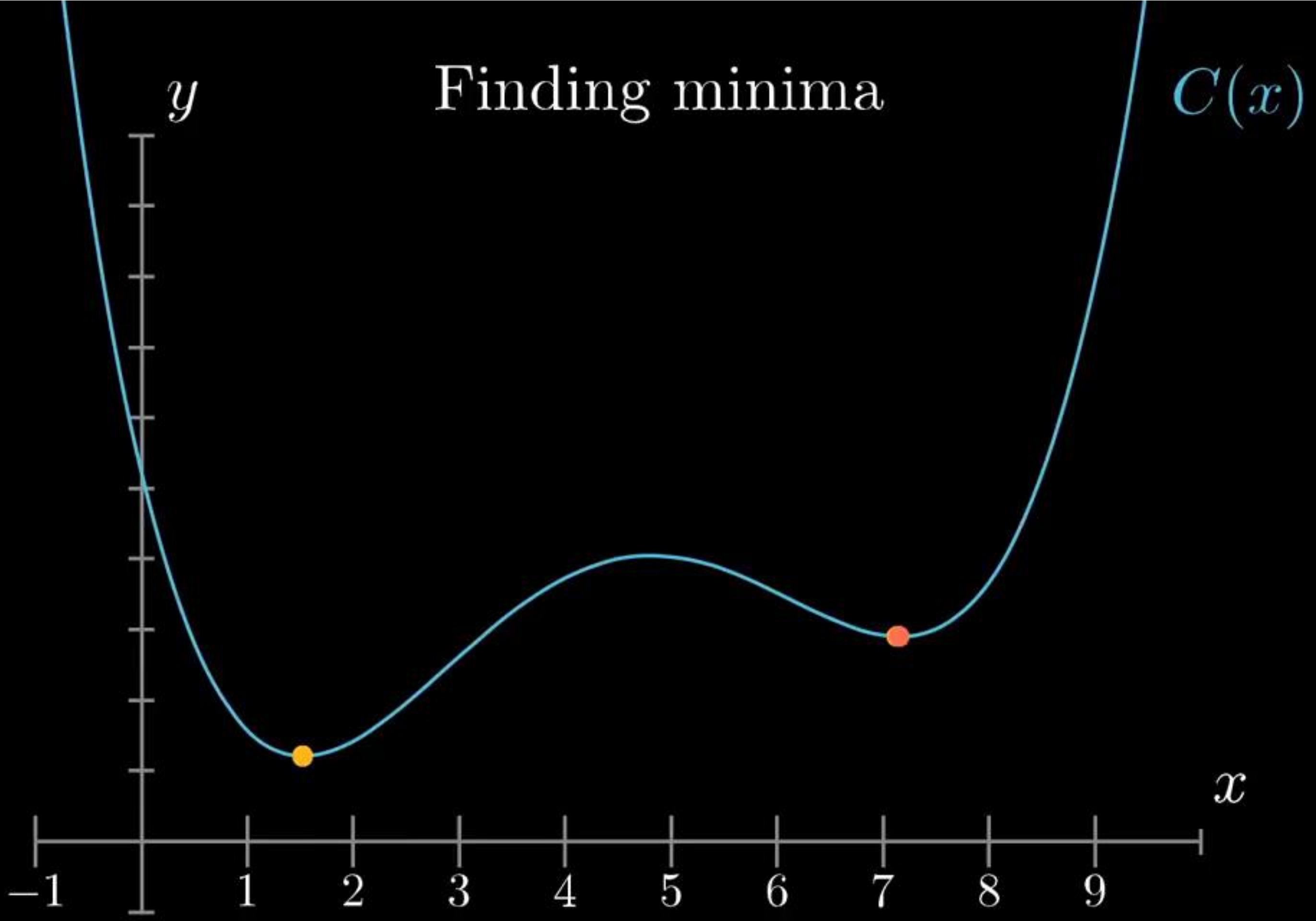
The Learning Rate (η)







Finding minima



Adam

- Adam stands for **Adaptive Moment Estimation.** Adaptive Moment Estimation (Adam) is another method that computes adaptive learning rates for each parameter.

AdaDelta

- It is an extension of **AdaGrad** which tends to remove the *decaying learning Rate* problem of it. Instead of accumulating all previous squared gradients, ***Adadelta*** limits the window of accumulated past gradients to some fixed size **w**.

Adagrad

- It simply allows the learning Rate -**η** to **adapt** based on the parameters. So it makes big updates for infrequent parameters and small updates for frequent parameters. For this reason, it is well-suited for dealing with sparse data.

TensorFlow

"TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications."

K Keras

"Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research"*

python™

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.