Deep Learning : Dipping and Diving

Hi !!!

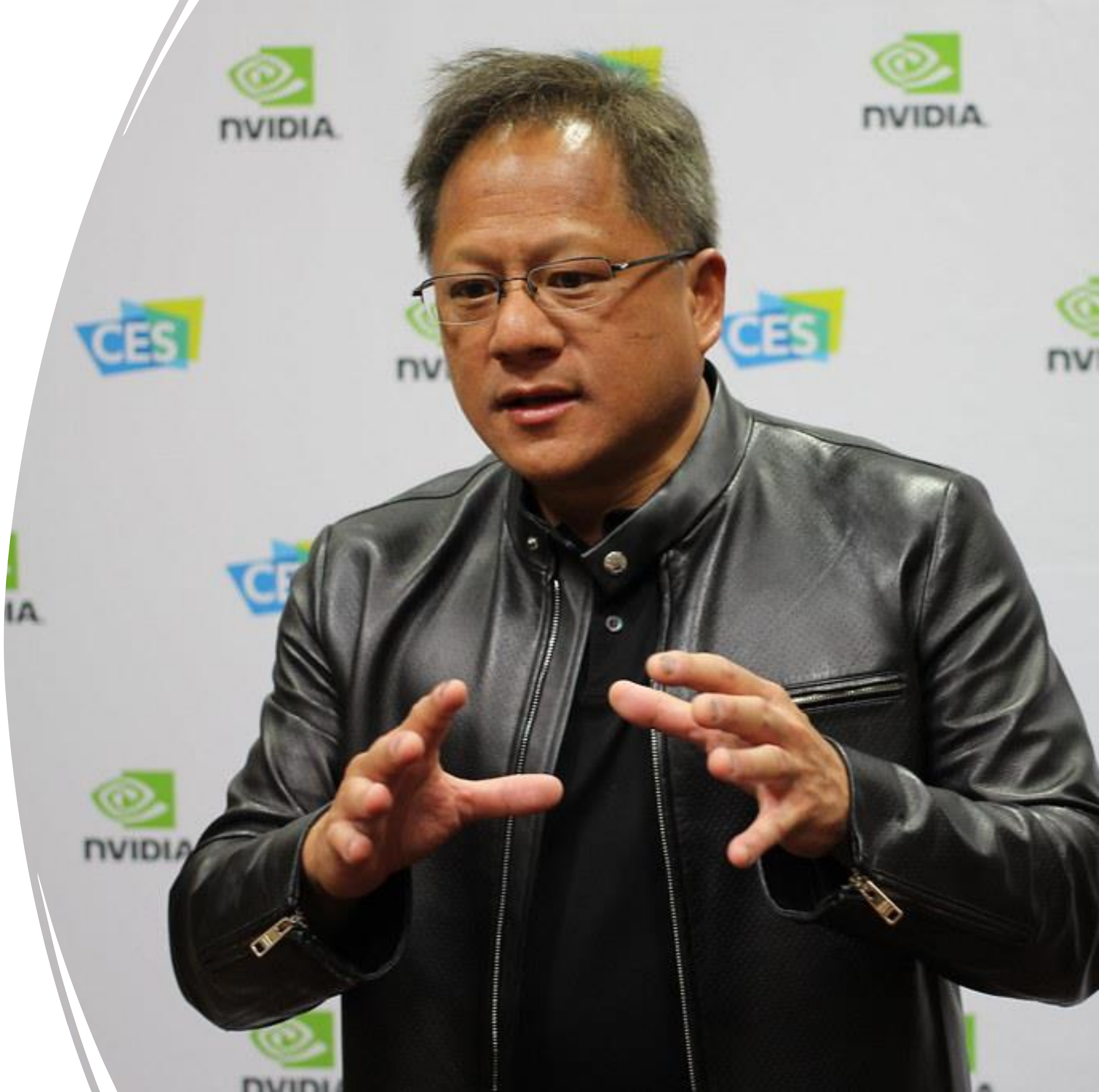
Rakesh

Systems Engineer

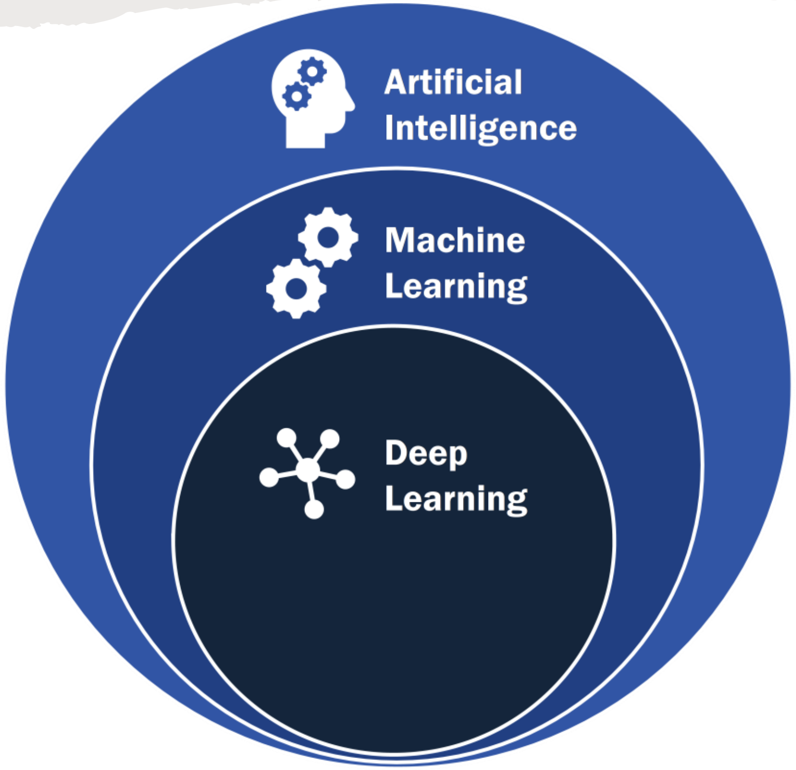
- Full-Time Developer
- AI Enthusiast
- Freelancer

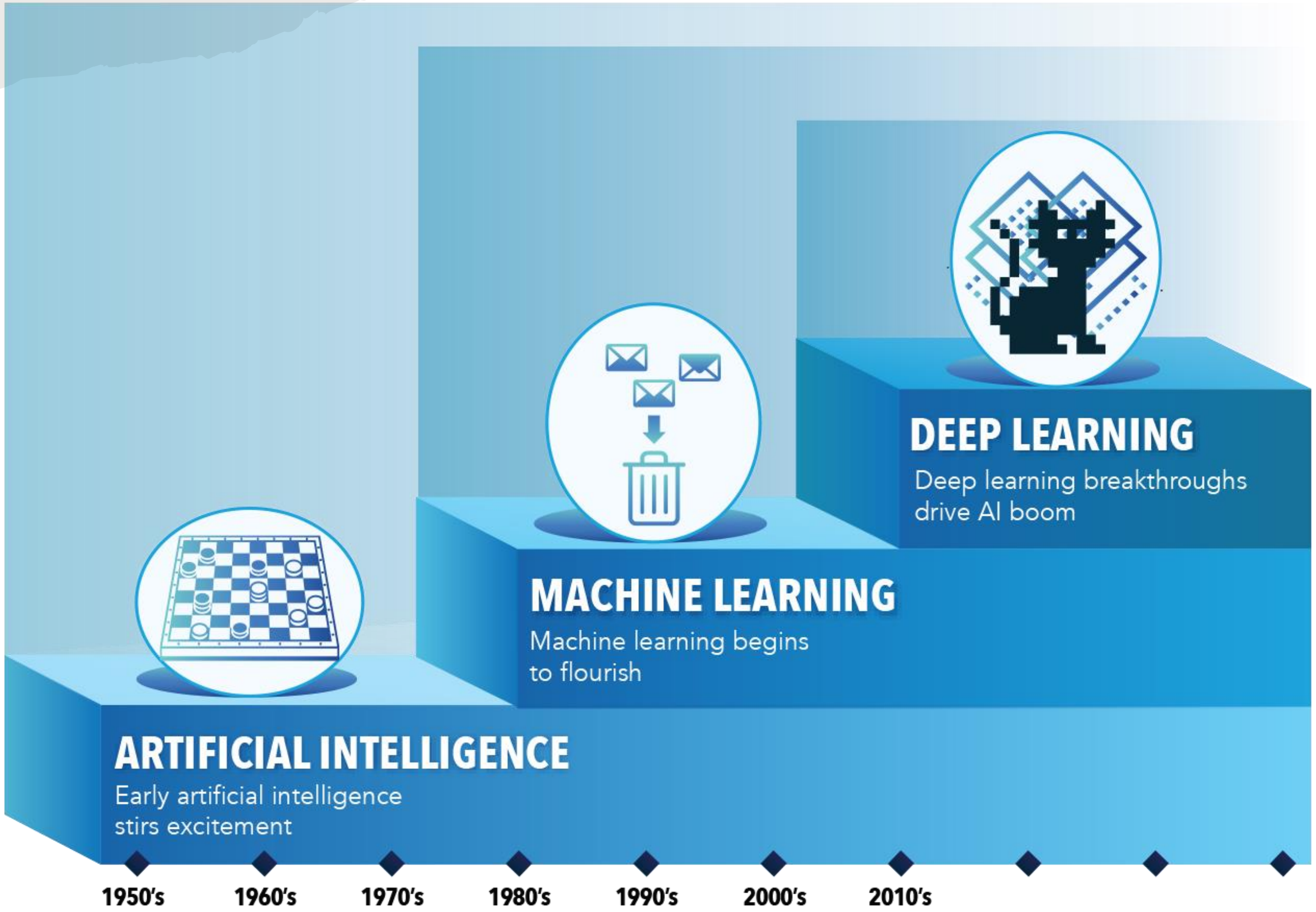
- It's our job to create computing technology such that nobody has to program, And that the programming language is Human. Everybody in the world is now a programmer.

- Jensen Huang, CEO, NVIDIA.

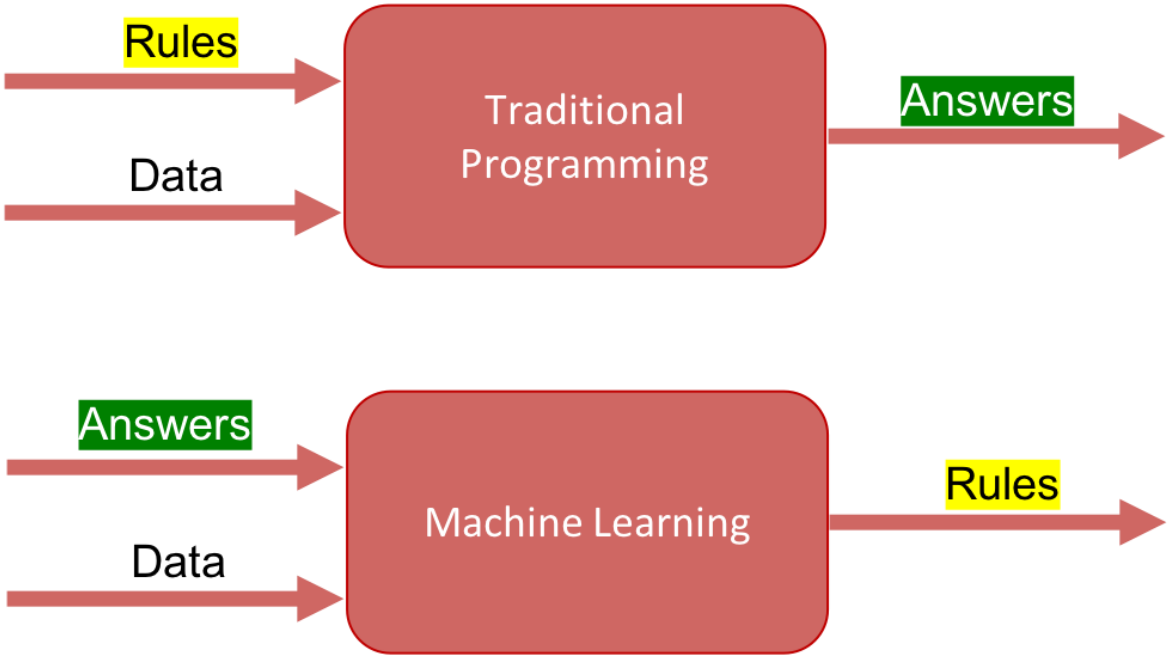


AI vs ML vs DL


Artificial Intelligence
Machine Learning
Deep Learning

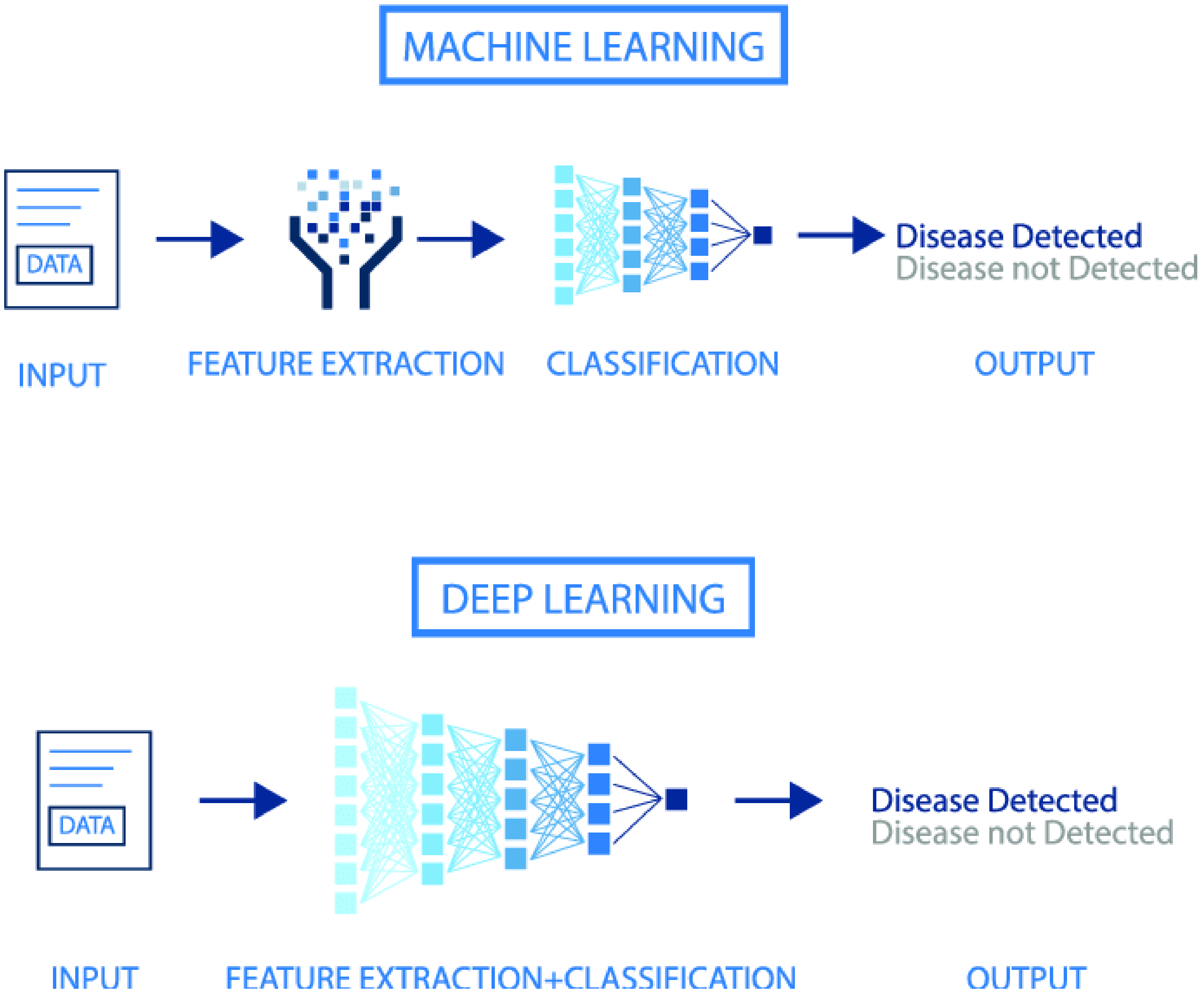

DEEP LEARNING
Deep learning breakthroughs drive AI boom
MACHINE LEARNING
Machine learning begins to flourish
ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement
1950's
1960's
1970's
1980's
1990's
2000's
2010's

Why it's different?



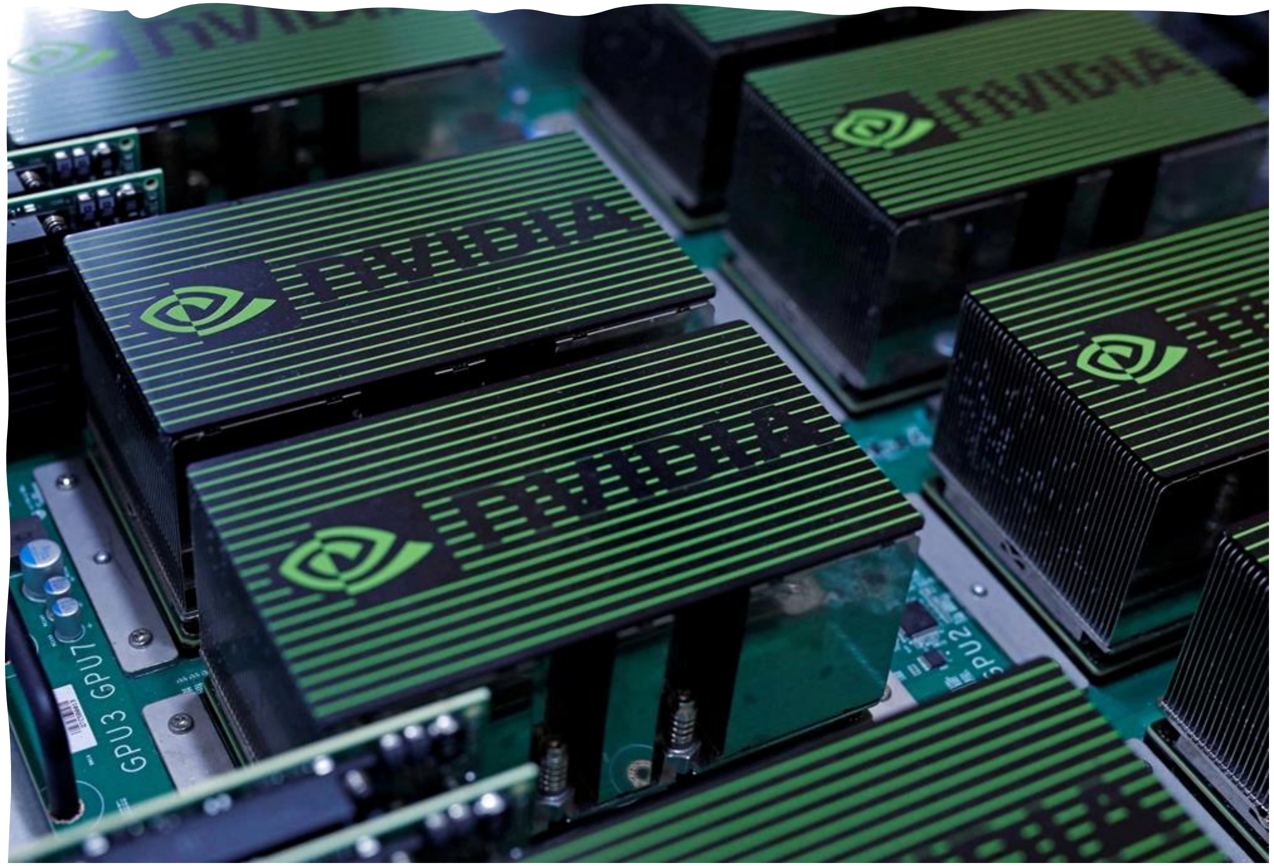
Deep Learning ?

- Deep learning is like teaching a computer to learn from experience by building increasingly complex patterns, much like how we learn through practice and exposure to difference situations.



Why Booming Now?

- Big Data - (Large Datasets)
- Parallel Computation Devices (GPU – Graphic Processing Unit)
- Softwares (Easy Implementation Eg: Pytorch, Tensorflow)

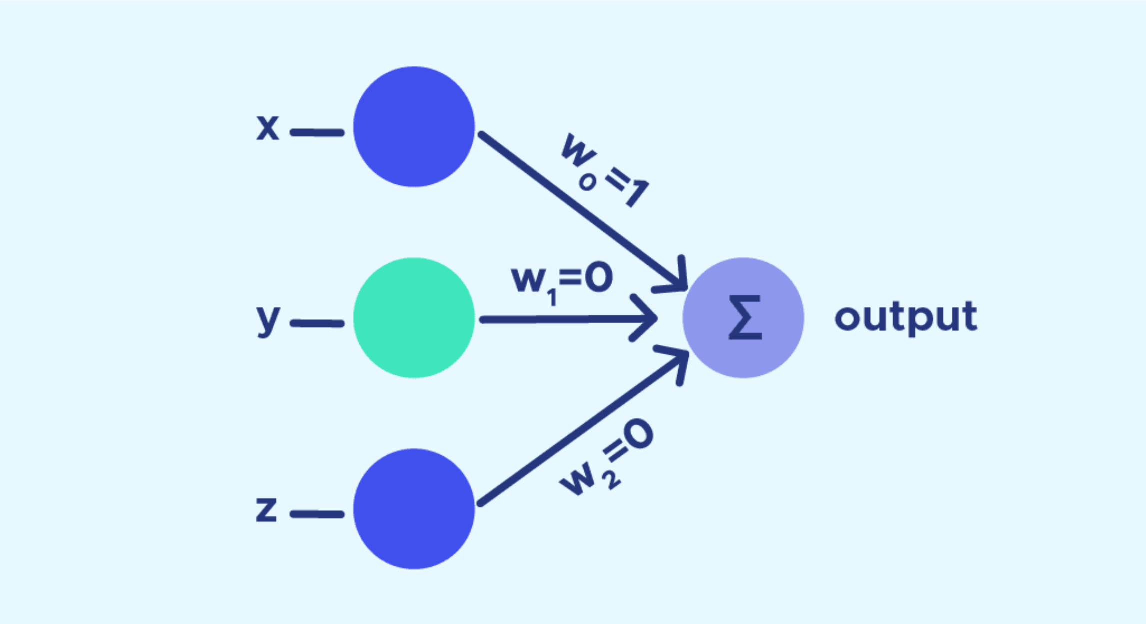


DL Concepts

- Perceptron
- Activation Function
- Neural Network
- Forward Propagation and Back Propagation?
  - Loss/Cost Function
  - Gradient Descent
- Overfitting and Underfitting Problems and its solution.

Perceptron

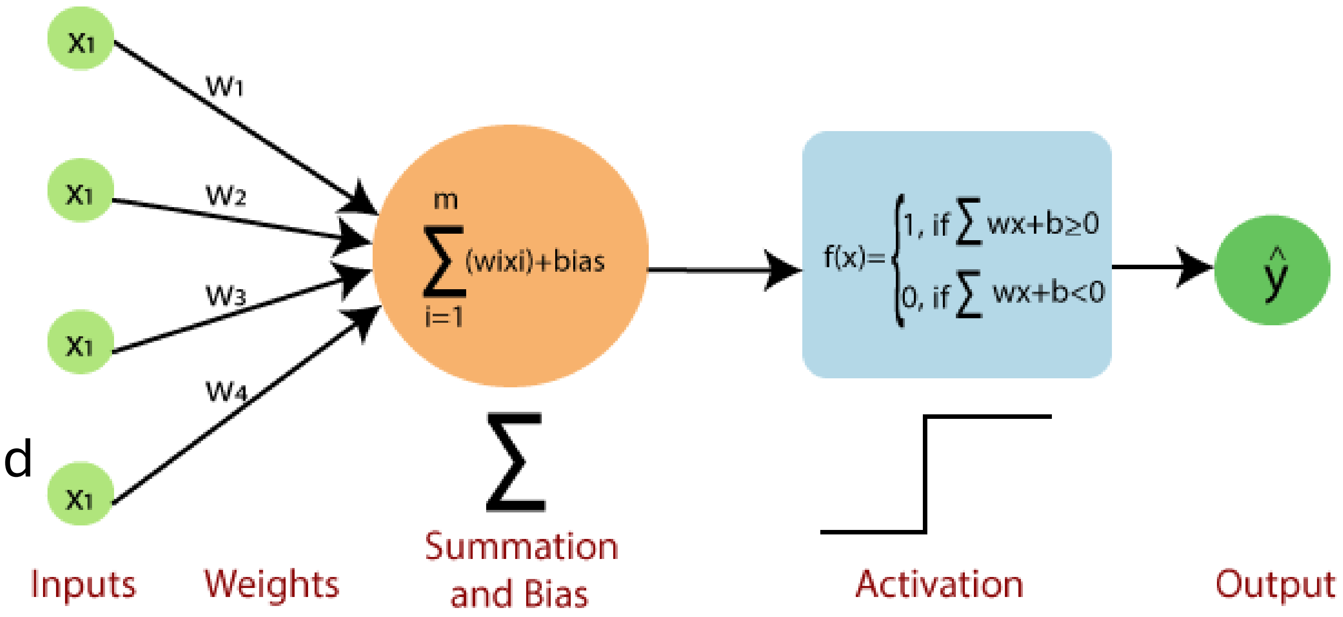
- A Perceptron is an **artificial neuron**, and thus a neural network unit. It performs computations to detect features or patterns in the input data. It is an algorithm for supervised learning of binary classifiers. It is this algorithm that **allows artificial neurons to learn** and **process features in a data set**
- Two Types: **Single Layer Perceptron** and **Multi Layer Perceptron**
- Introduced in **1958**.



Perceptron Learning Rule

Intuition behind Perceptron

- The way neurons in the brain work. It’s a simple model that learns to classify input data by adjusting weights and biases to minimize errors, mimicking how the brain strengthens and weakens connection between neurons based on experience.

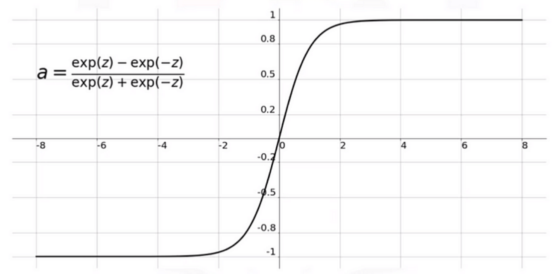


x1
x1
x1
x1
w1
w2
w3
w4
$\sum_{i=1}^{m}$(wixi)+bias
f(x)= {1, if ∑ wx+b≥0 ; 0, if ∑ wx+b<0
ŷ
∑
Inputs
Weights
Summation and Bias
Activation
Output

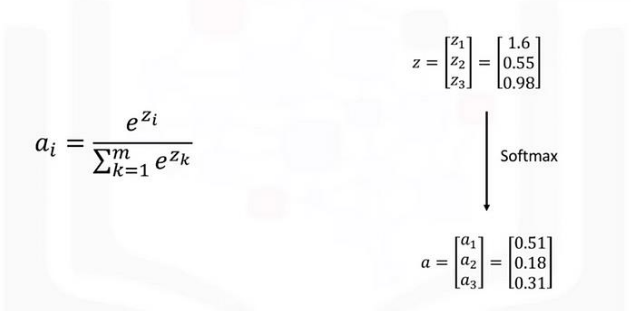
Activation Functions

- Sigmoid
- Softmax
- ReLU (Rectified Linear Unit)
- Hyperbolic Tangent Function

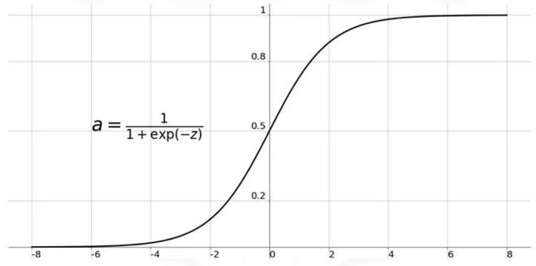
Hyperbolic Tangent Function



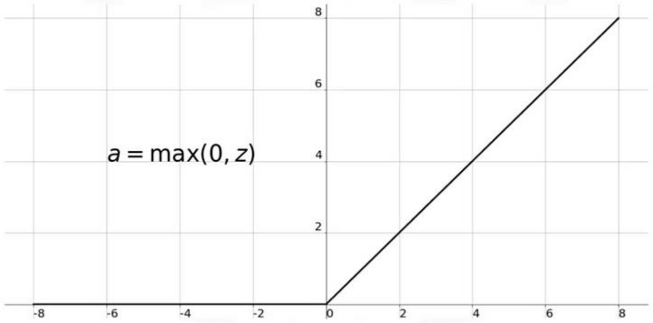
Softmax Function


$a_i = \frac{e^{z_i}}{\sum_{k=1}^{m} e^{z_k}}$
$z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 1.6 \\ 0.55 \\ 0.98 \end{bmatrix}$
Softmax
$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0.51 \\ 0.18 \\ 0.31 \end{bmatrix}$

Sigmoid Function

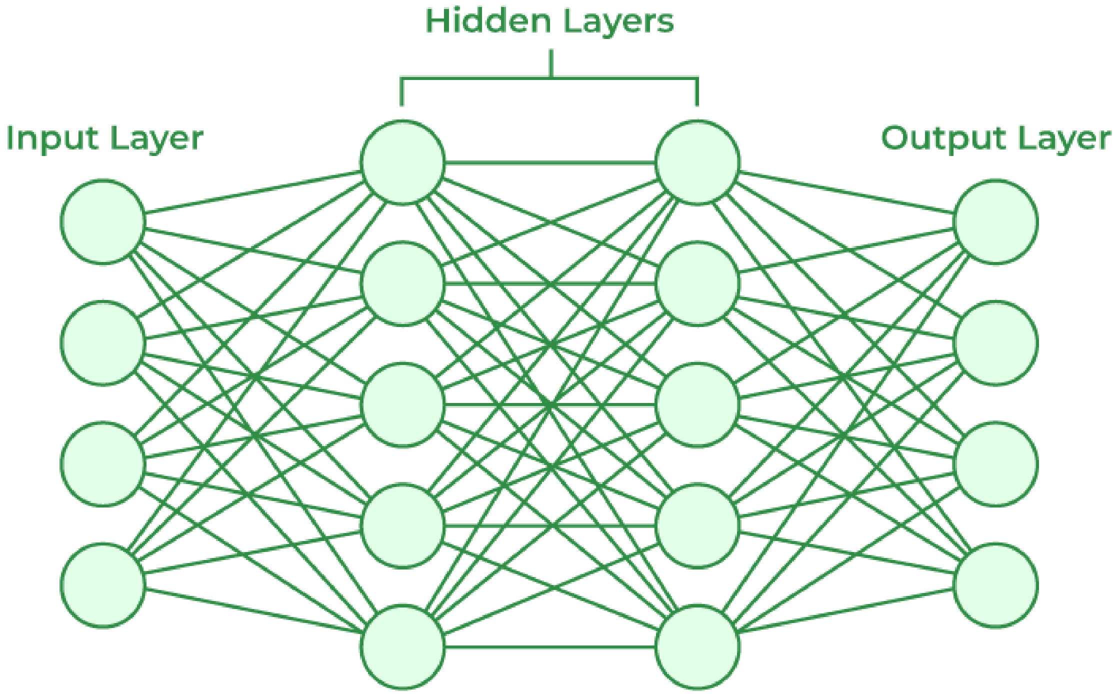


ReLU Function



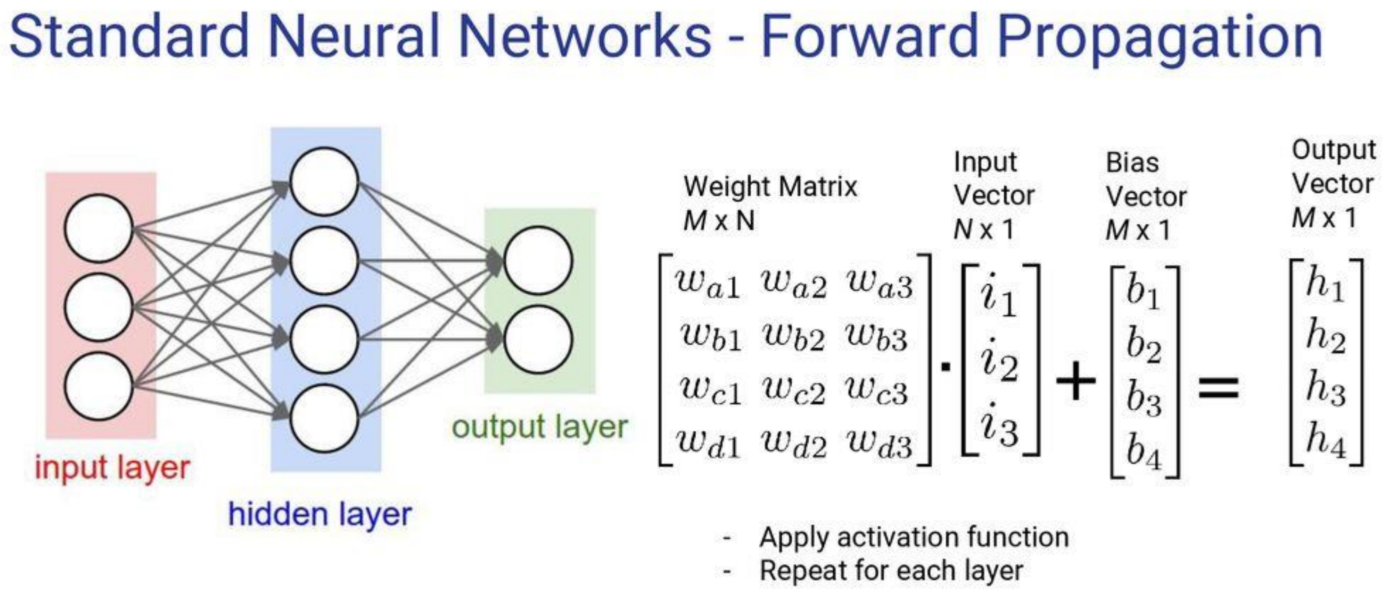
Neural Network

- Neural Networks are like digital brains that learn, adapt and recognize patterns, paving the way for computers to think and understand like humans.
- A Neural network is a set of **interconnected Perceptrons (Multi Layer Perceptron)**



Forward Propagation

- One step of calculation through the neural network.
- Initially random weights are assigned and



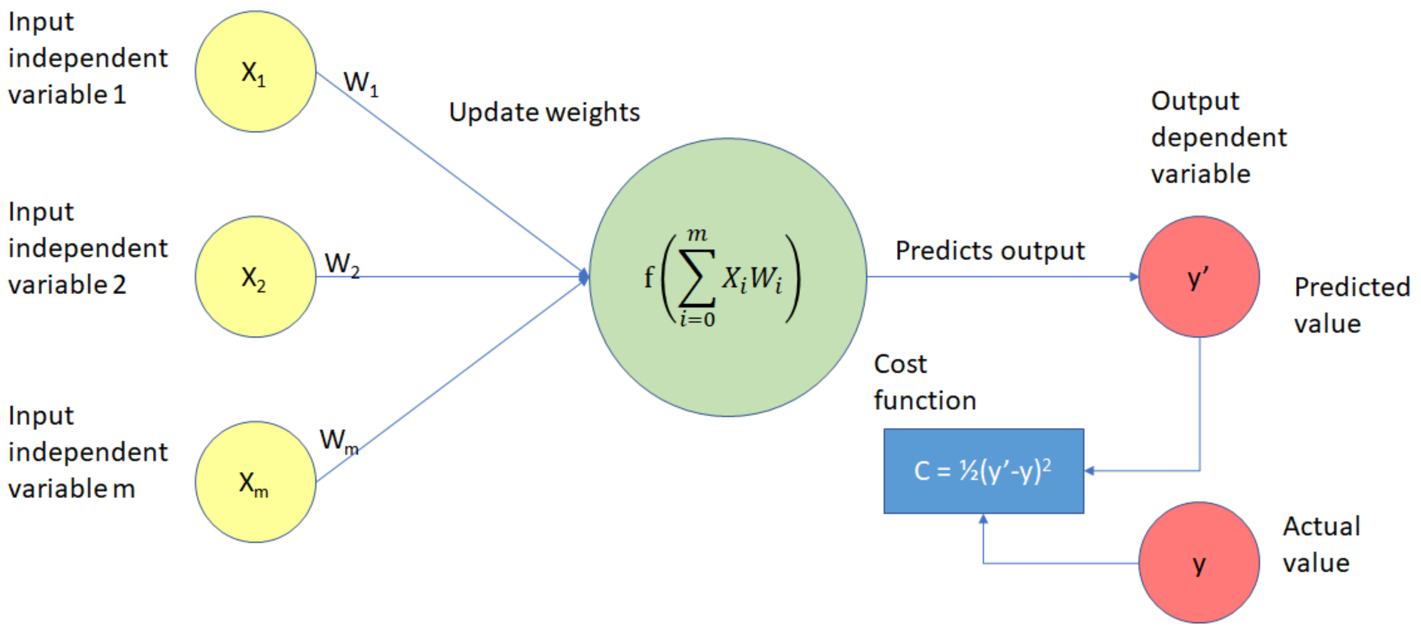
Standard Neural Networks - Forward Propagation

$$\begin{array}{ccccccc} \text{Weight Matrix } M \times \text{N} & & \text{Input Vector } N \times 1 & & \text{Bias Vector } M \times 1 & & \text{Output Vector } M \times 1 \\ \begin{bmatrix} w_{a1} & w_{a2} & w_{a3} \\ w_{b1} & w_{b2} & w_{b3} \\ w_{c1} & w_{c2} & w_{c3} \\ w_{d1} & w_{d2} & w_{d3} \end{bmatrix} & \cdot & \begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} & + & \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} & = & \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} \end{array}$$

- Apply activation function
- Repeat for each layer

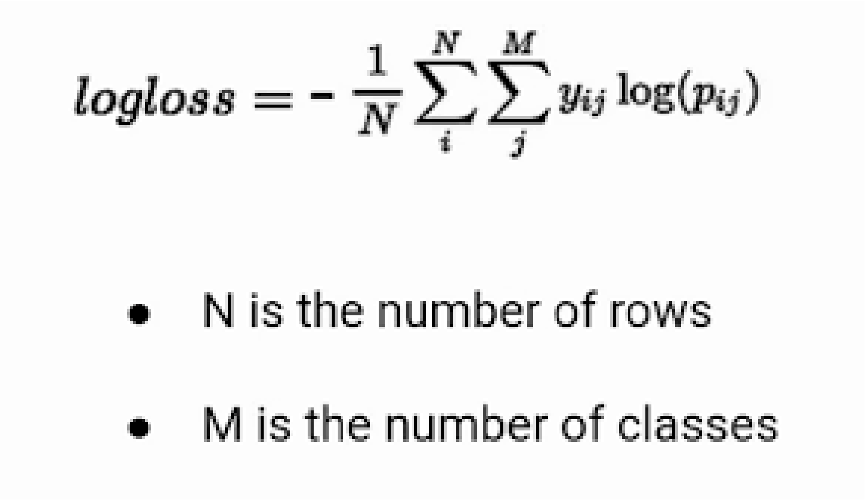
Back Propagation ?

- As weights are initially random, network will obviously produce a wrong output.
- So, upon each input and calculation, there's need to adjust weights based on the output.
- This is performed through Back Propagation.
- The current output is compared with expected/ real output and a difference is calculated using loss/cost function.


Input independent variable 1
X_1
Input independent variable 2
X_2
Input independent variable m
X_m
W_1
W_2
W_m
Update weights
$f\left(\sum_{i=0}^{m} X_i W_i\right)$
Predicts output
Output dependent variable
y'
Predicted value
Cost function
$C = ½(y'-y)^2$
y
Actual value

Loss/Cost Function

- The loss function is a measurement of error which defined the precision lost on comparing the predicted output to the actual output.
- Few examples are, Mean-Squared Error (MSE) and Binary-Cross Entropy Loss



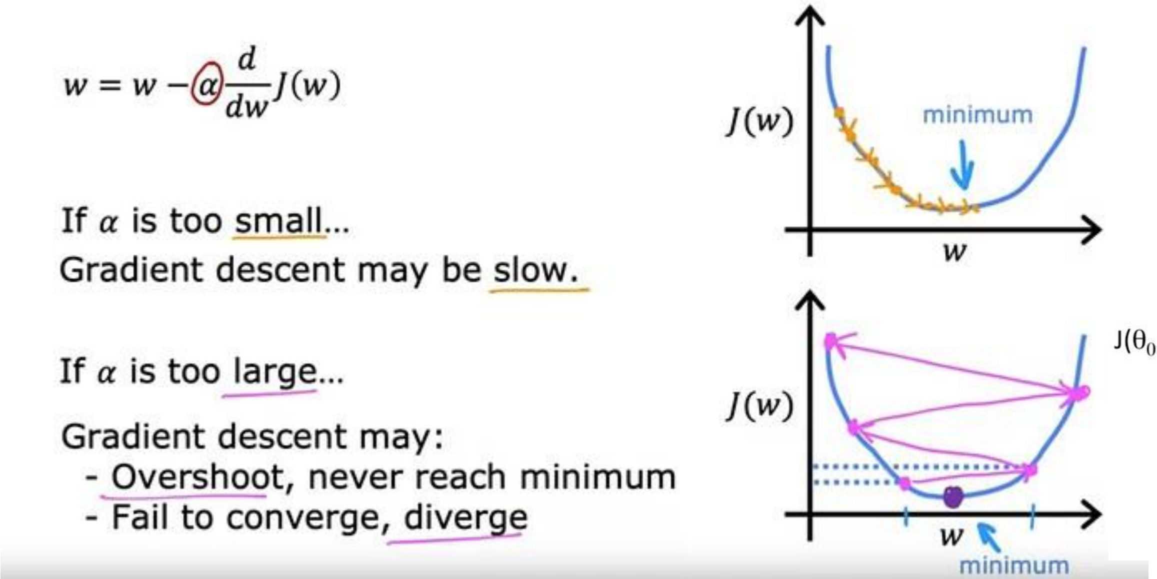
$$logloss = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{M} y_{ij}\log(p_{ij})$$

- N is the number of rows
- M is the number of classes

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Gradient Descent

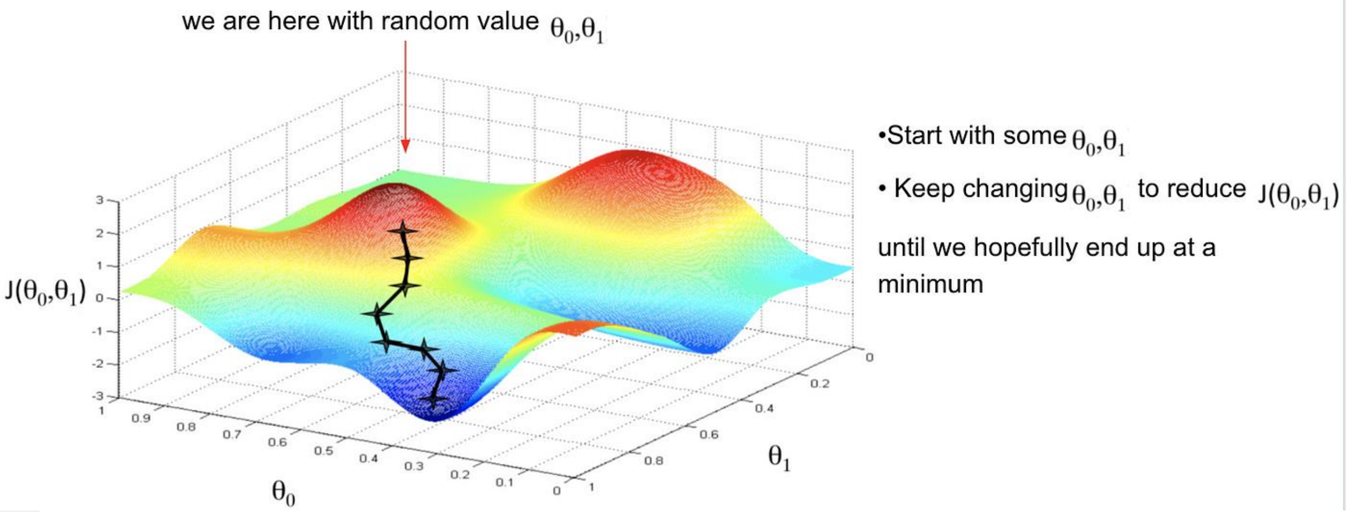
- Now, think finding the lowest change in weight as a problem/ value space.
- This becomes a mathematical problem where the lowest point in the space has to be find.
- We can find gradient which contributes to the highest point in the space, so we move the values to the opposite to find the lowest point.



$$w = w - \alpha \frac{d}{dw} J(w)$$

If α is too small...
Gradient descent may be slow.

If α is too large...
Gradient descent may:
- Overshoot, never reach minimum
- Fail to converge, diverge



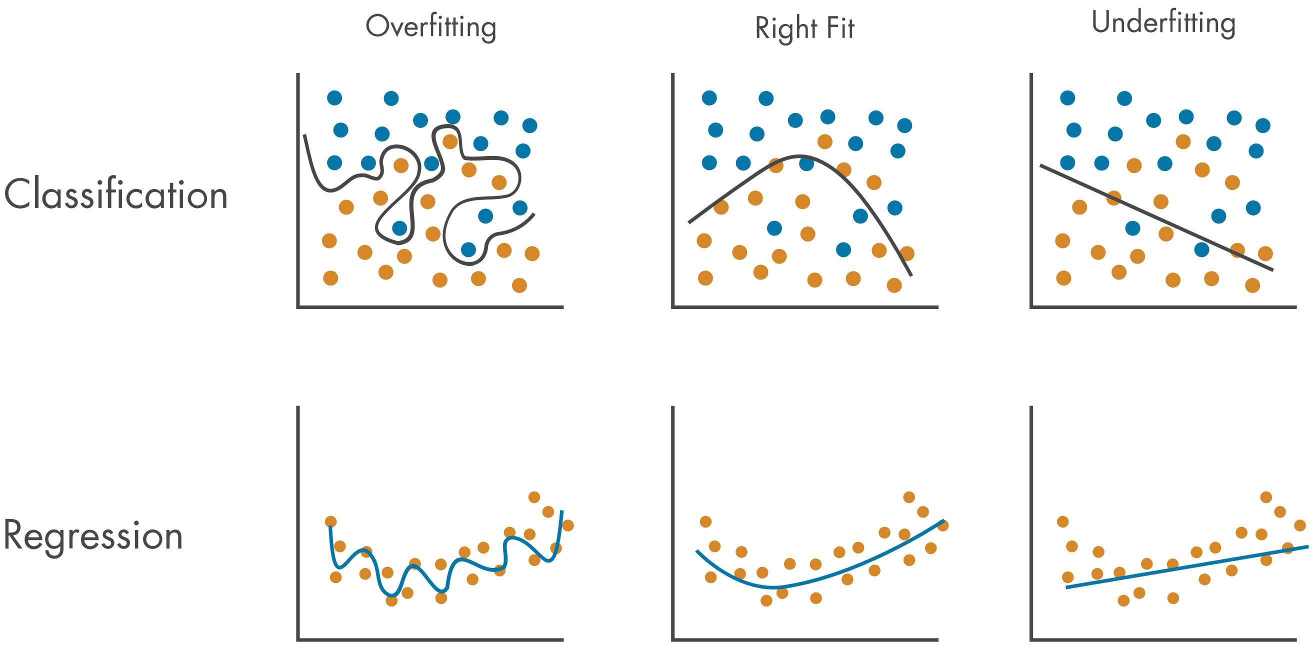
we are here with random value θ_0,θ_1

- Start with some θ_0,θ_1
- Keep changing θ_0,θ_1 to reduce $J(\theta_0,\theta_1)$

until we hopefully end up at a minimum

Overfitting and Underfitting

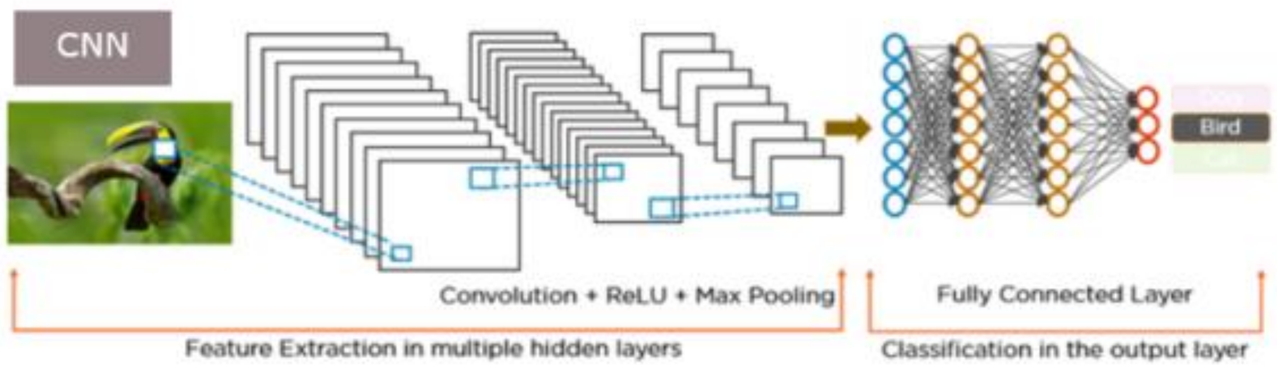
- Underfit models experience high bias—they give inaccurate results for both the training data and test set.
- On the other hand, overfit models experience high variance—they give accurate results for the training set but not for the test set.



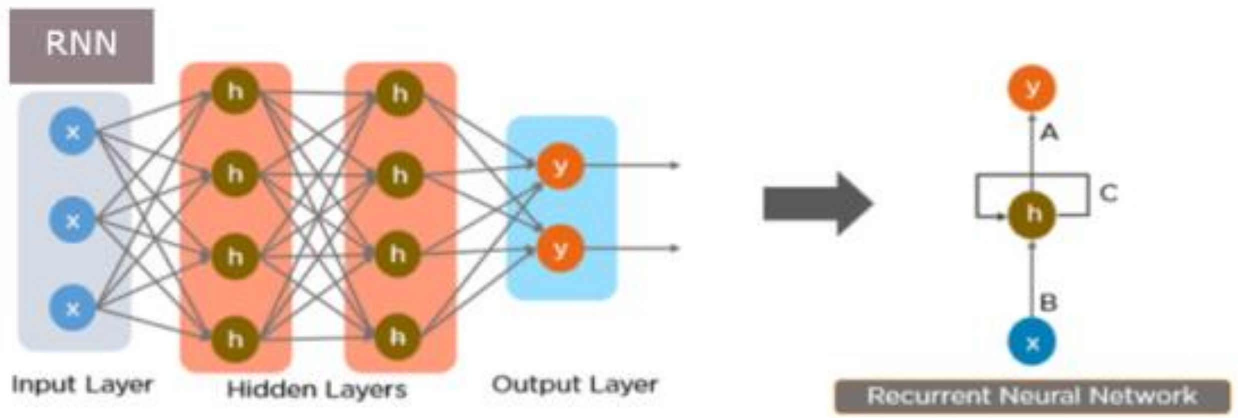
Model Architecture

- Artificial Neural Network (ANN).
- Convolutional Neural Network (CNN).
- Recurrent Neural Network (RNN).
- LSTM (Long Short Term Memory) model.
- Generative Adversarial Network.
- Transformers.

Convolutional Neural Network

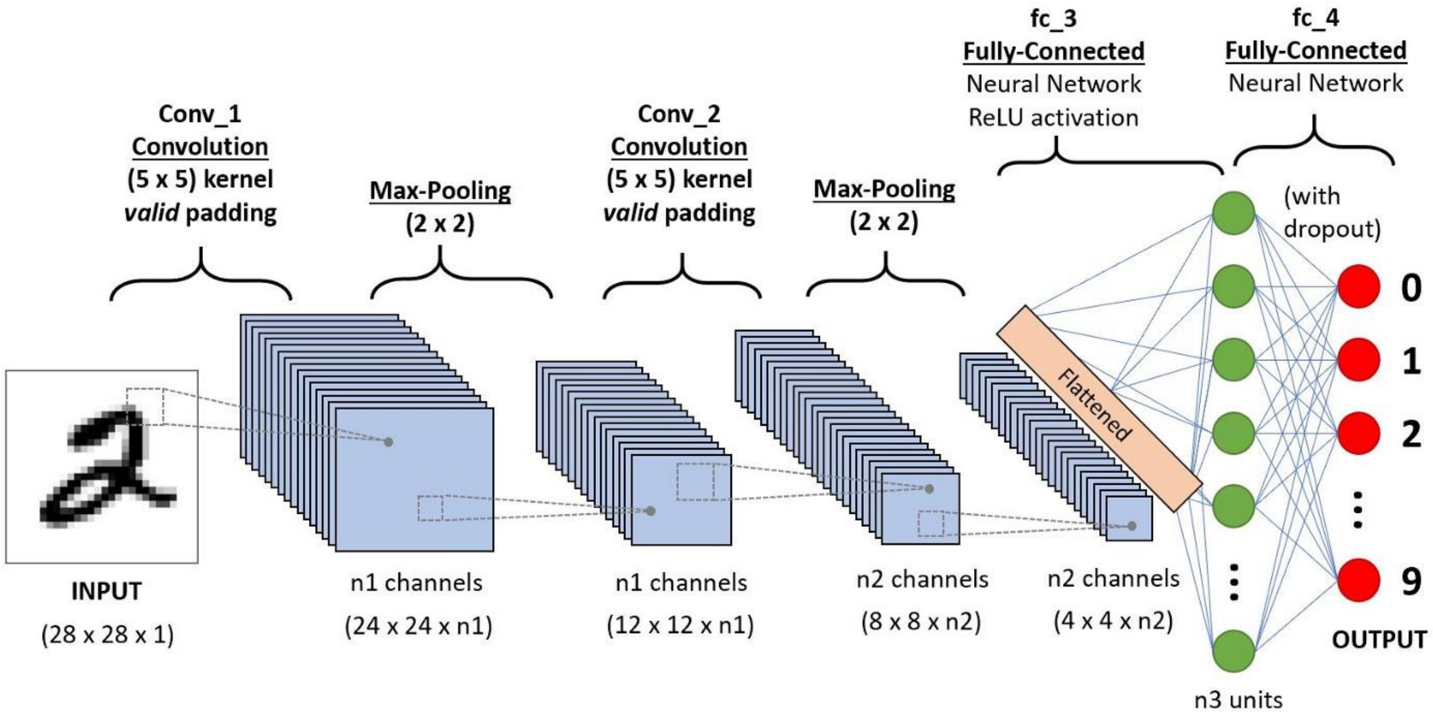

CNN
Bird
Convolution + ReLU + Max Pooling
Fully Connected Layer
Feature Extraction in multiple hidden layers
Classification in the output layer

Recurrent Neural Network



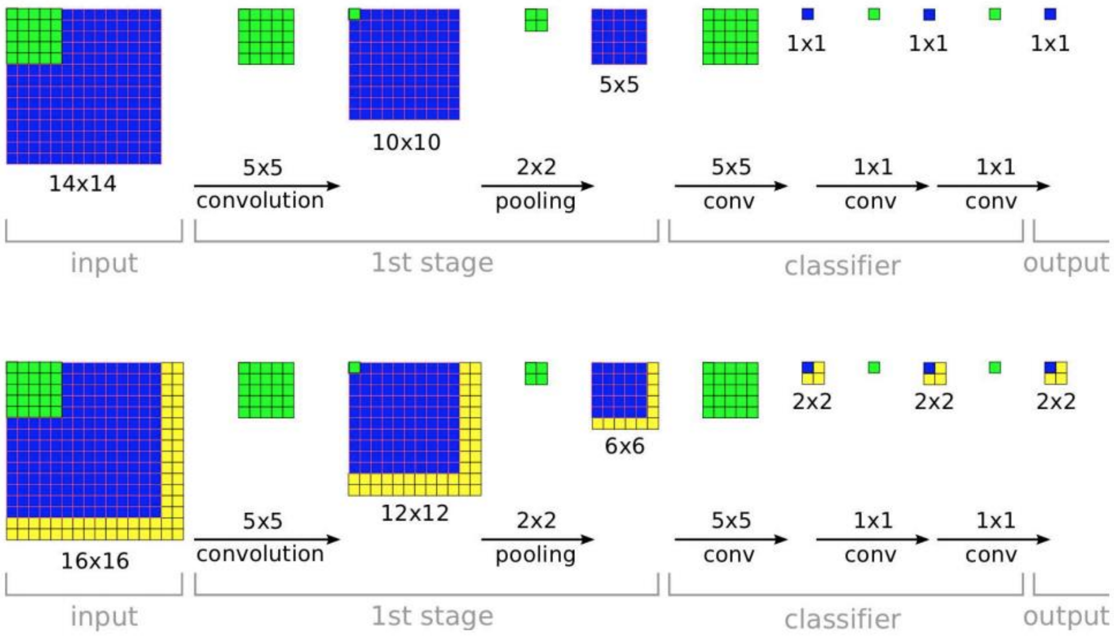
Convolutional Neural Network (CNN)

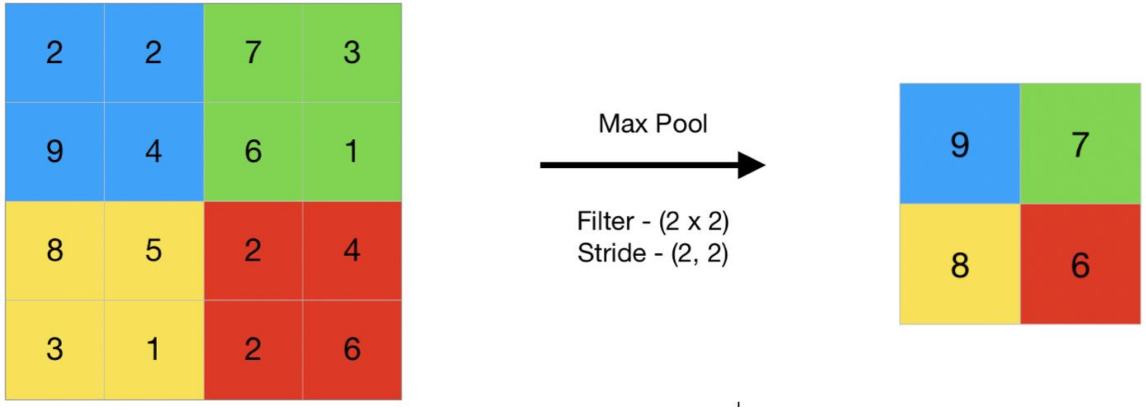
- Convolutional Neural Network (CNN) is a feed-forward neural network that is generally used to analyze visual images by processing data with grid-like topology. A CNN is also known as "ConvNet"



Convolution and Pooling

- 2D-Conv layer.
- Max-Pooling.
- Min-Pooling.
- Avg-Pooling.





2	2	7	3
9	4	6	1
8	5	2	4
3	1	2	6

Max Pool

Filter - (2 x 2)
Stride - (2, 2)

9	7
8	6

References

- 1. MIT 6.S191: Introduction to Deep Learning - https://www.youtube.com/playlist?list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI
- 2. Deep Learning Blog - https://gayathri-siva.medium.com/deep-learning-f52ae2ddef2
- 3. RNN Blog - https://gayathri-siva.medium.com/recurrent-neural-networks-rnn-4a5c547e949f
- 4. CNN Blog - https://gayathri-siva.medium.com/convolutional-neural-network-cnn-71c85e8c4ec2
- 5. ANN Example - https://www.kaggle.com/code/efekurdoglu/ann-using-pytorch
- 6. CNN Example - https://www.kaggle.com/code/masfour/99-7-accuracy-top-10-digit-classifier-tutorial
- 7. CNN Example Pytorch - https://www.kaggle.com/code/marcpaulo/pytorch-cnn-tutorial
- 8. Loss functions - https://builtin.com/machine-learning/common-loss-functions

LinkedIn : https://www.linkedin.com/in/rakes-me/

Github : https://github.com/rakes-me