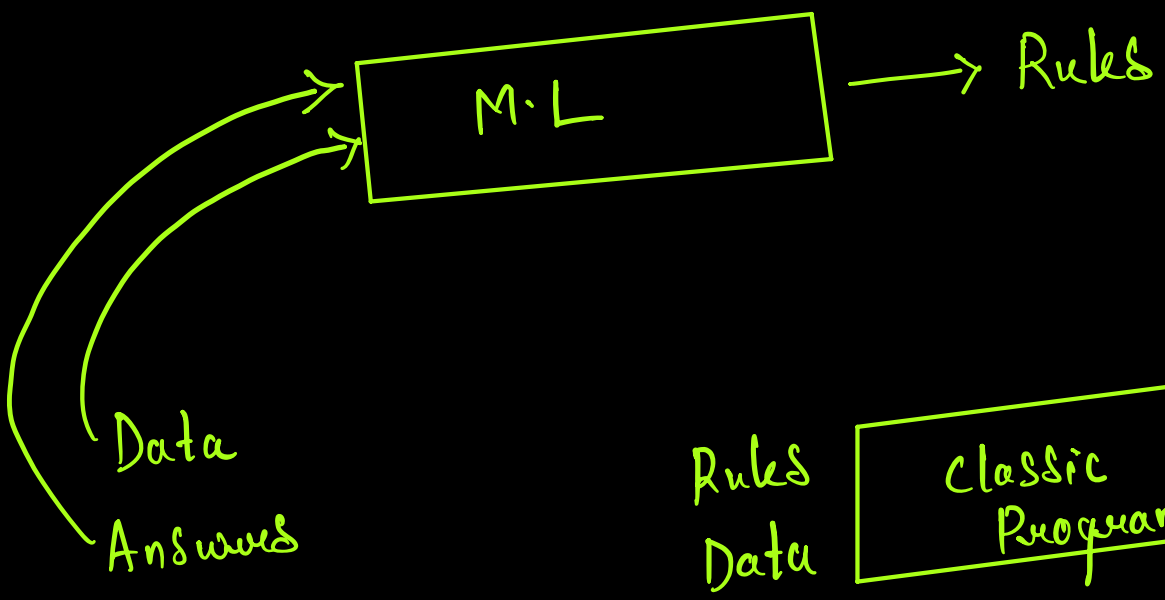
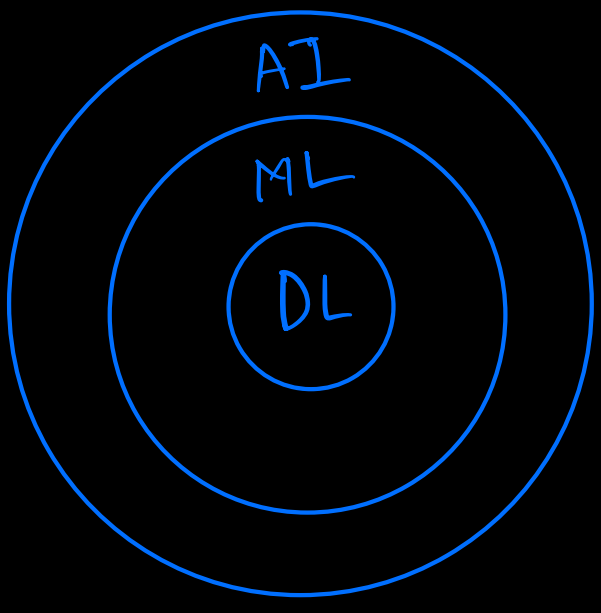
Deep Learning

Rule based Programming



Supervised Learning



Data :→ Network

Answer

Mapping Input → Output

↳ Mathematical Function

↳ Learning

Training

1) Approximation Function

Deep Learning :- Neural Networks

Universal Approximators

Mathematical Functions

1) loss function

Log loss (logistic)

2) Activation Function

1) Linear Function

$f(x) = x$

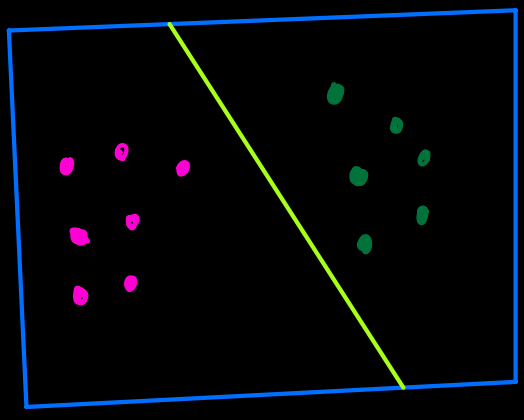
2) Step Functions

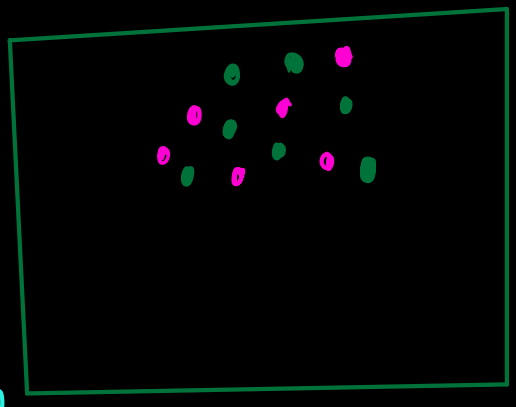
$$f(x) = \begin{cases} 0 \\ 1 \end{cases}$$

Pathways

1) Linear

2) Non - Linear





Deep Learning vs ML

Machine learning :- Shallow learning

Very Effective for Smaller Dataset

Data Increases ↓ ML Performance

Data Increase ↑ DL

DL

Data Hungry

2012

* Hardware

GPUs

Nvidia, AMD

Software :- Hardware

1) GPU
2) Edge Device

Deep learning Frameworks

1) TF (Keras)
   - TensorFlow Func
   - Production
   - Google

2) PyTorch
   - OOP
   - Research
   - Meta

- Mxnet
- Chainer
- Theano
- Sonnet

Deep learning

1) Artifical Neural Networks (ANN)

Multi-layer Perceptron (MLP)

1) Regression
2) Classification

2) Image Data / Video (Convolutional Neural Networks)

Vision Transformers

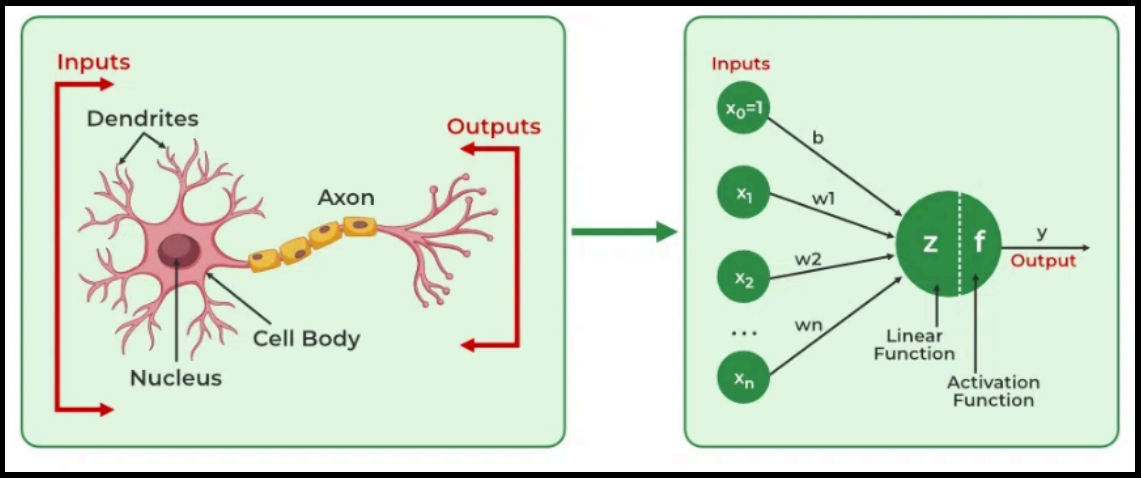
2) Text Data (Recurrent Neural Network)

Long Short Term Memory

Transformers

3) GAN :- Generative Adversial Network

Neuron



Bio

NN

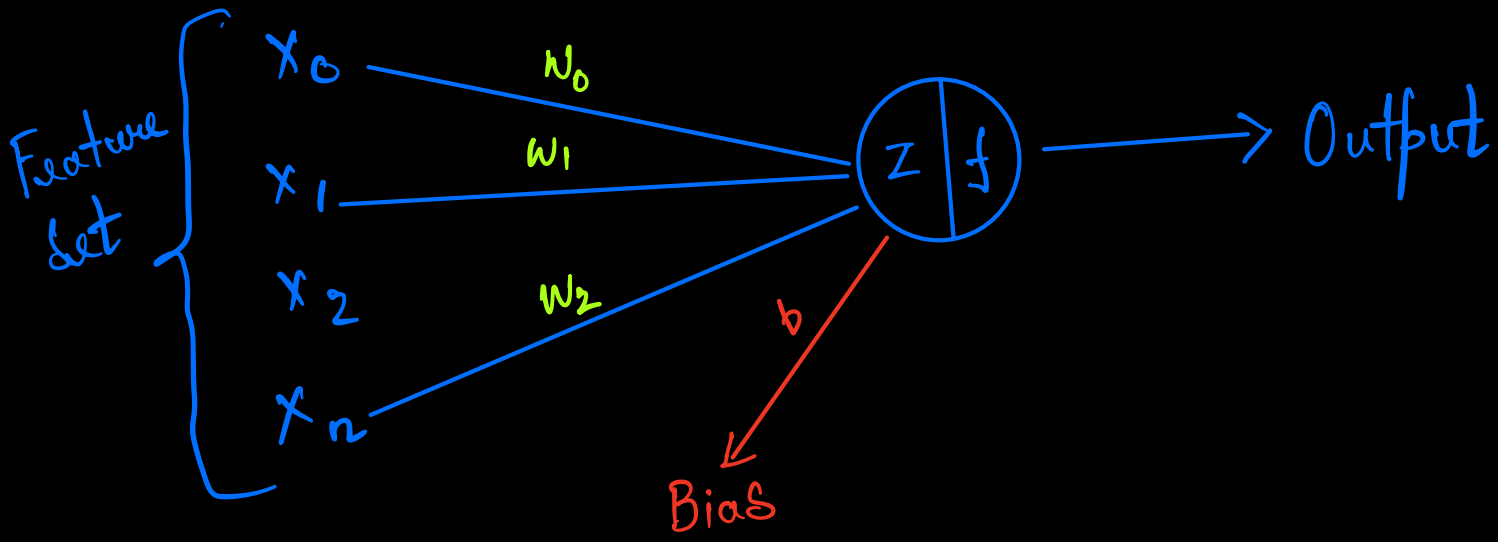
Signals

Intution

Perceptron

* Perceptron *

Inputs :-



Input :- x_0, x_1, x_2, x_2

x_0 ⇓ 0.1 , x_1 ⇓ 0.2 , x_2 ⇓ 0.79 , x_2 ⇓ 0.8

Weights = Learnable parameters

Bias = Learnable parameter

Model = Weights + Bias + Architecture

$z = \text{Summat}(w \cdot x)$

$z = x_0 \cdot w_0 + x_1 \cdot w_1 + x_2 \cdot w_2 + x_n w_n$

$(z) = \sum_{i=1}^{n} w_i \cdot x_i$

$y = f\left(\sum_{i=1}^{n} w_i \cdot x_i + b\right)$

$y = f(w \cdot x + b)$

Perceptron Model

$y = f(wx + b)$

f = Activation Function

Linear Function

$= f(z)$

$= z$

Step Function

$$f = \begin{cases} 0 \\ 1 \end{cases}$$

$= 1$

Learnable Params

Purpose :- Reduce the Error Rate

Loss Reduce / Decrease

(w and b) → change

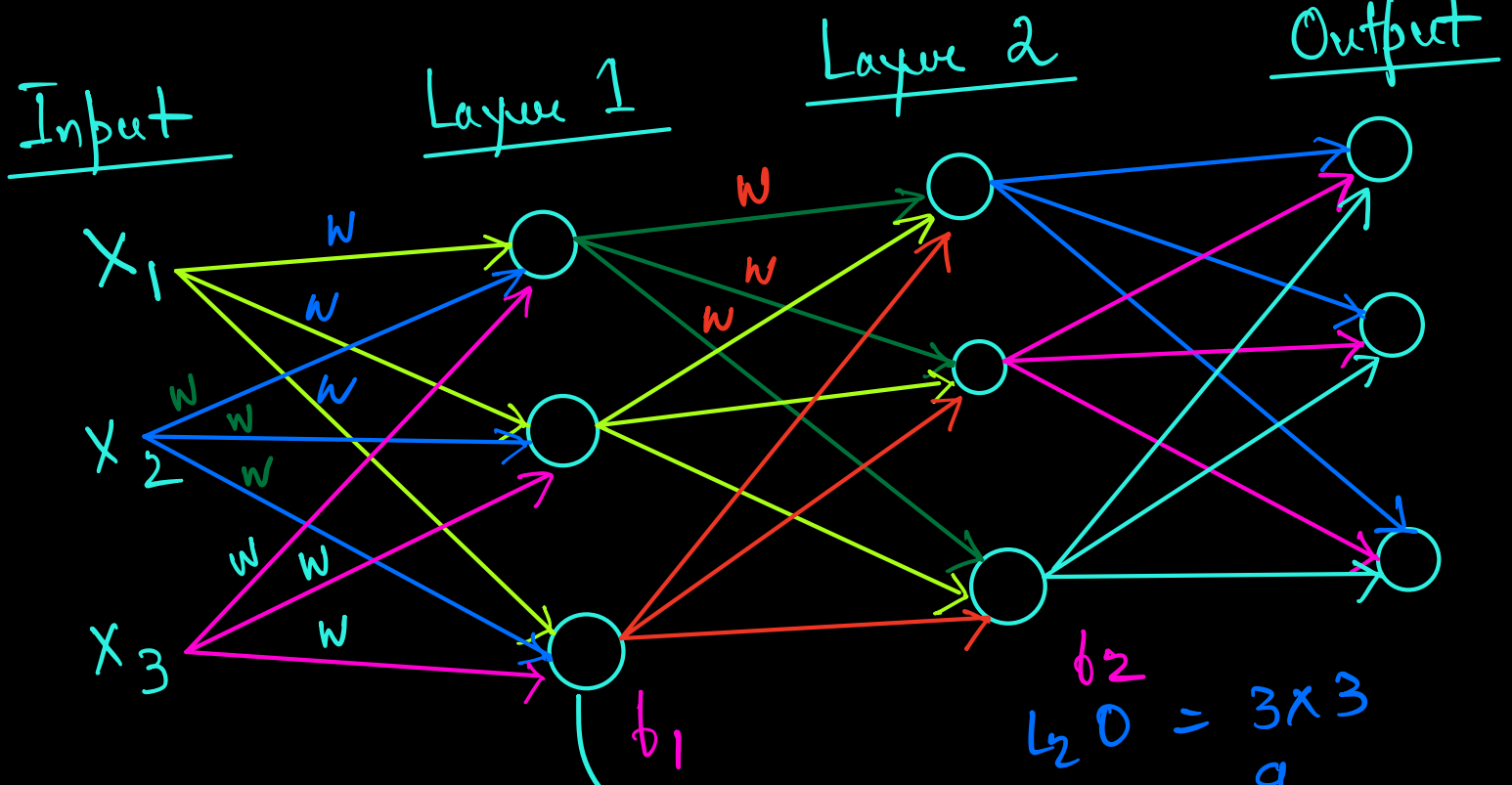
Frozen Weights Bias → learning is done

→ Share with others.

Pre Trained Models

Perceptron

Smallest unit / type of Neural Network.



$IL_1 = 3 \times 3 = 9$

$L_1 L_2 = 3 \times 3 = 9$

$L_2 O = 3 \times 3 = 9$

Node

Layer 1, 2 → Hidden layers

Dense layer (Keras)

Linear layer (Pytorch)

Fully Connected layer

Total :- 27

Input ⟶ n(Hidden layer) ⟶ Output

Hidden layer ⟶ Nodes

Selecting Layers / no of layers / nodes within a layer

⇓

* Hyperparameter *

1) Forward Propagation

Multi layer Perceptron

Artificial Neural Network (ANN)

Limitations

1) Availability of Data

Small Text

Structure Data

Images / Audio

2) Hardware Limitation

High Computation Power

(CPUs) → Very low

GPUs, TPUs → Costly

→ Google Cloud

Colab

Nvidia

3) Complexity of model

Simple Model :- Underfitting

Complex Model :- Overfitting

§ Balanced Dataset §

Vision → Image Classification

dog cat house

SMOTE

Dog - 300 (100) → Data Augmentation

Cat - 300

House - 300

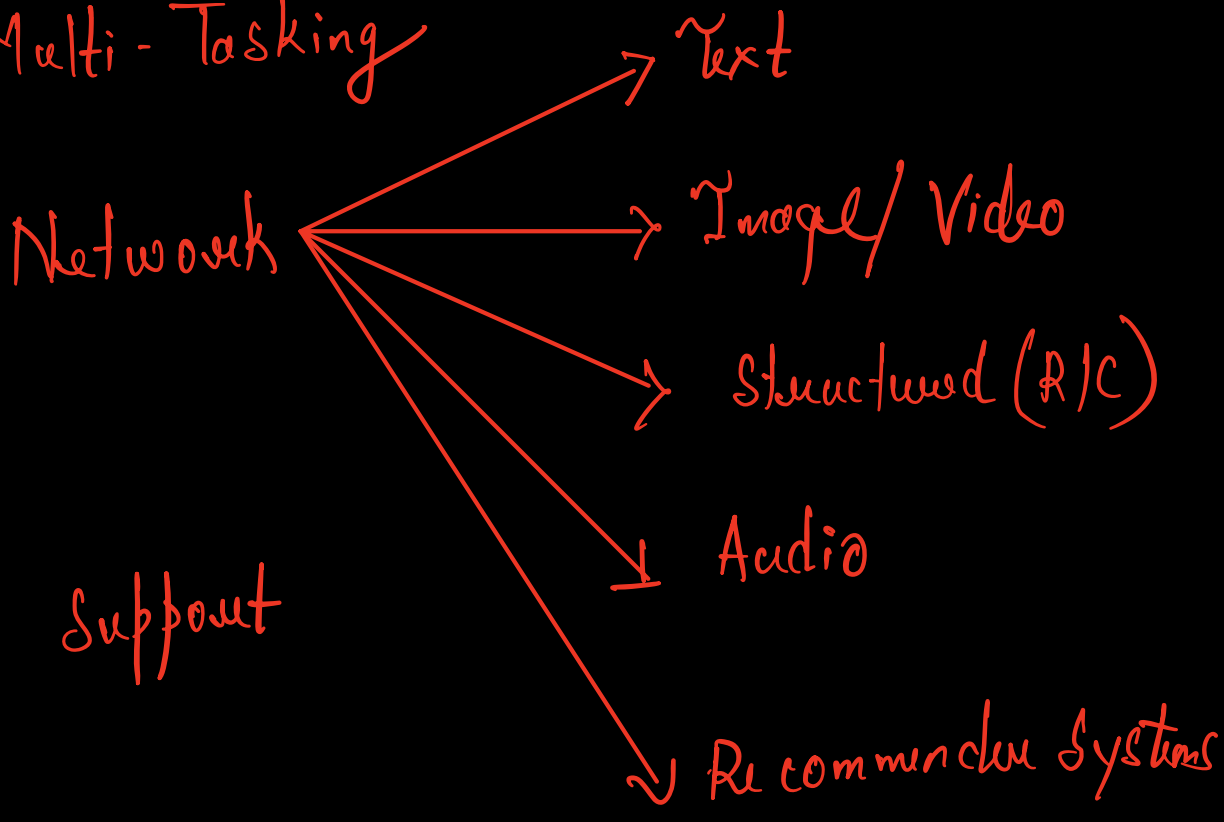
Train	Validation	Test
80%	10%	10%
70%	15%	15%

1) Lacks of Global Generalization

Explainability, Interpretability

I → B.B → 0

2) Multi-Tasking



LLMs
Multimodal Support
↓
Text, Images

Applications

1) Speech Recognition

2) Pattern Recognition
   ↳ Finding Tumour cells in CT Scan

3) NLP → Transformers
   GPT 3

4) Recommender System :- Netflix, YT

Healthcare

1) Medical Image Analysis

Transportation

Self - Driving Cars

Agriculture

Plant Disease

Crop Monitoring

Soil Monitoring

Live Stock Monitoring

RTOS → Real Time Operating Systems.

Rockets, Missiles

Drones :-