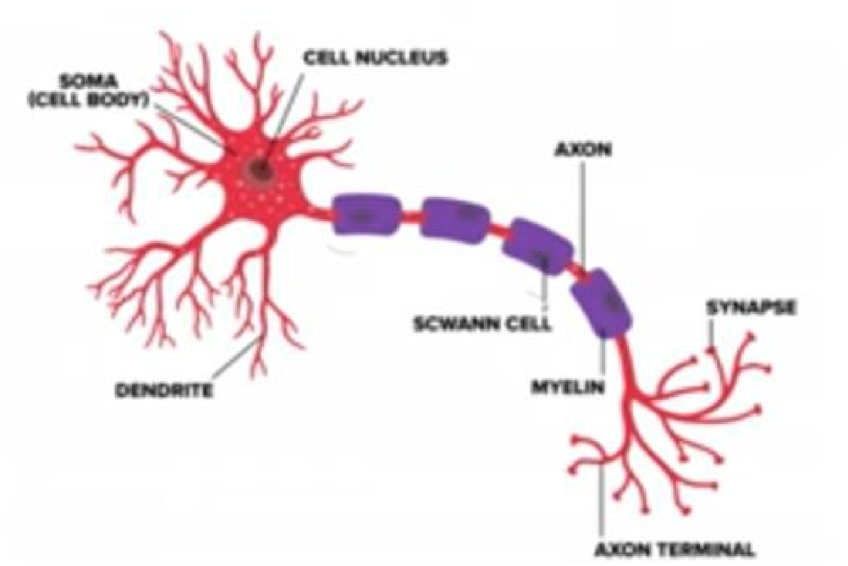
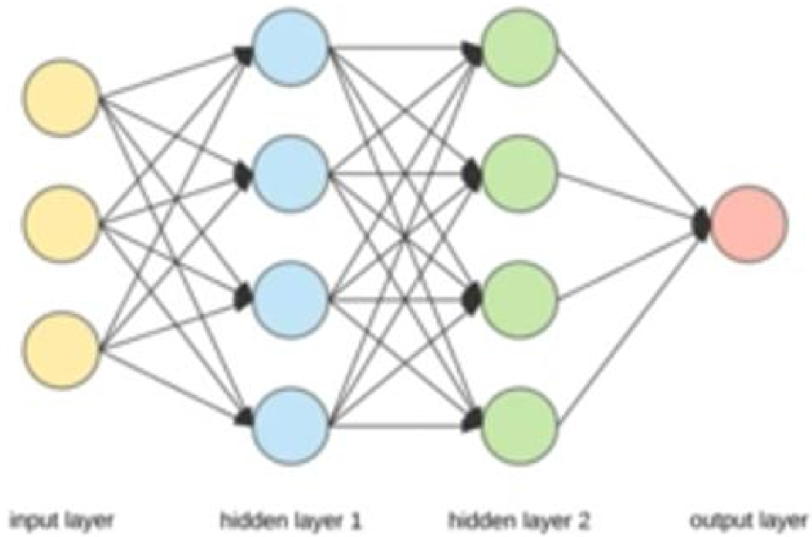
Siddhardhan

Deep Learning - Introduction

Deep Learning

Deep Learning is a subfield of Machine Learning that uses Artificial Neural Networks to learn from the data.





Deep Learning - Applications

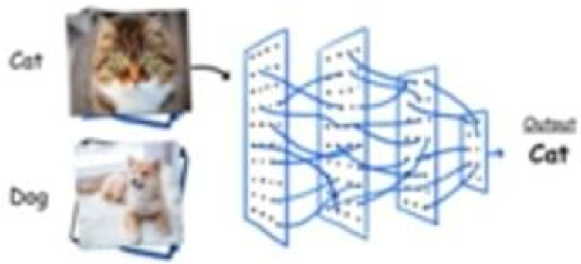
Healthcare

Autonomous cars

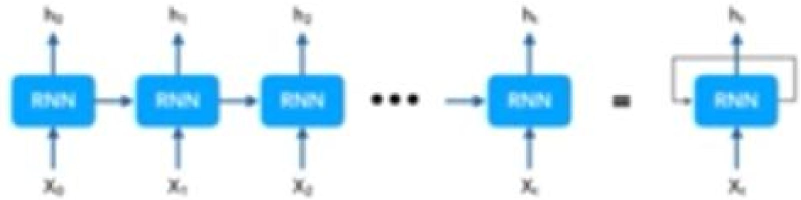
Computer Vision

Natural Language Processing

Deep Learning

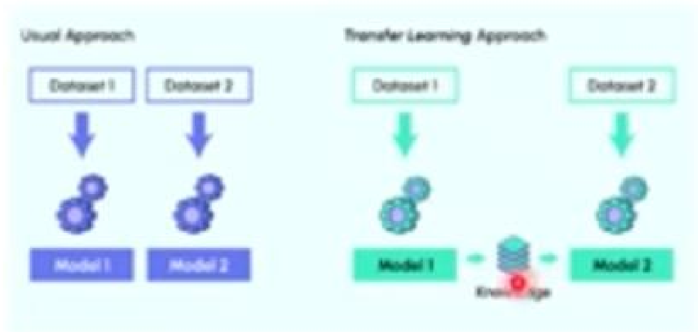


Convolutional Neural Network (CNN)



Recurrent Neural Network (RNN)

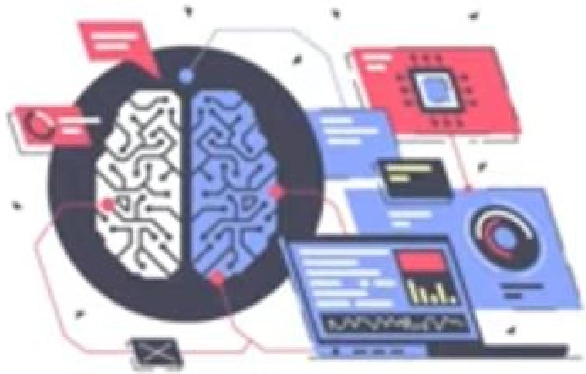
Generative Adversarial Network (GAN)



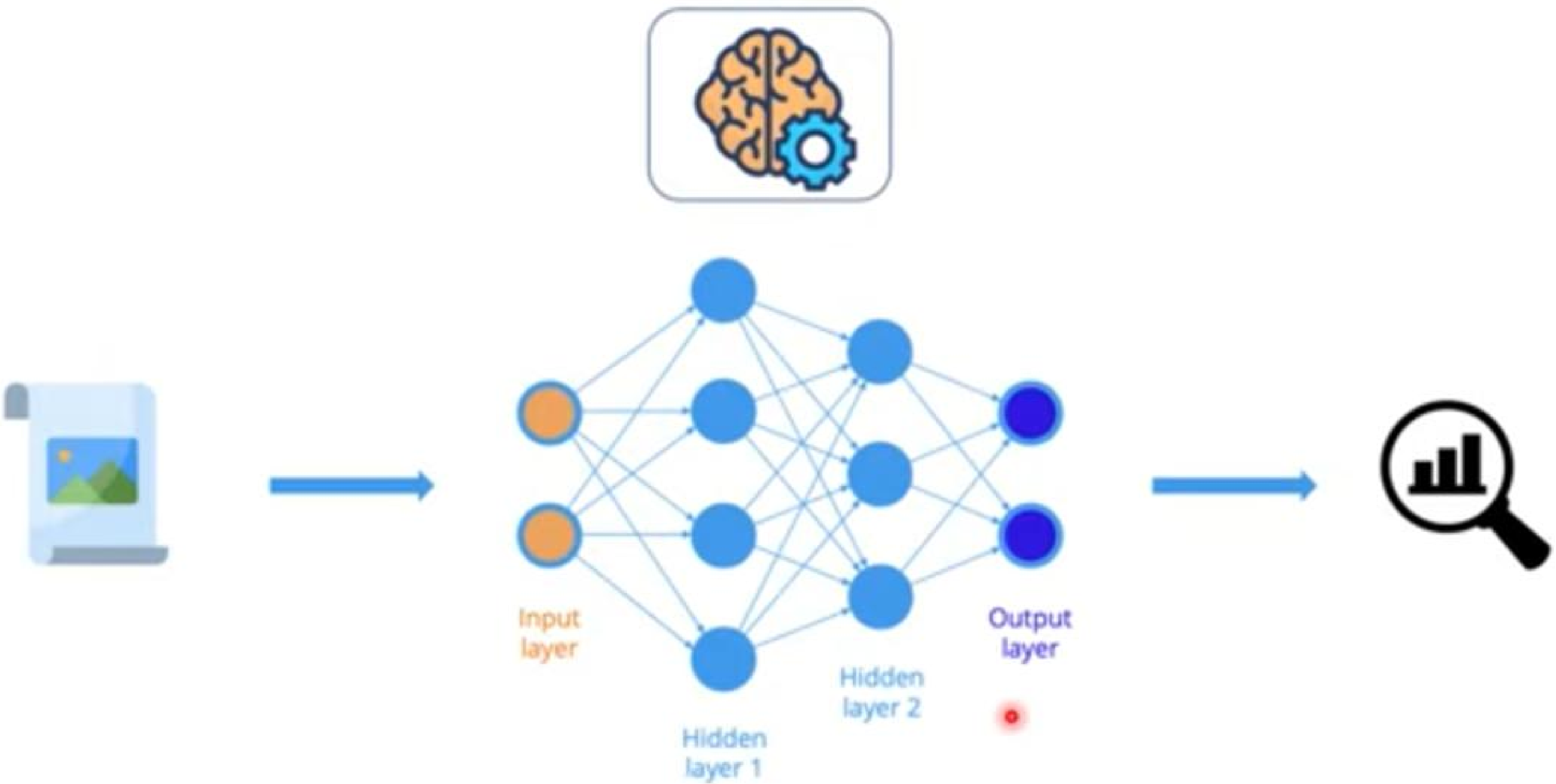
Transfer Learning

Siddhardhan

How Neural Network Works?

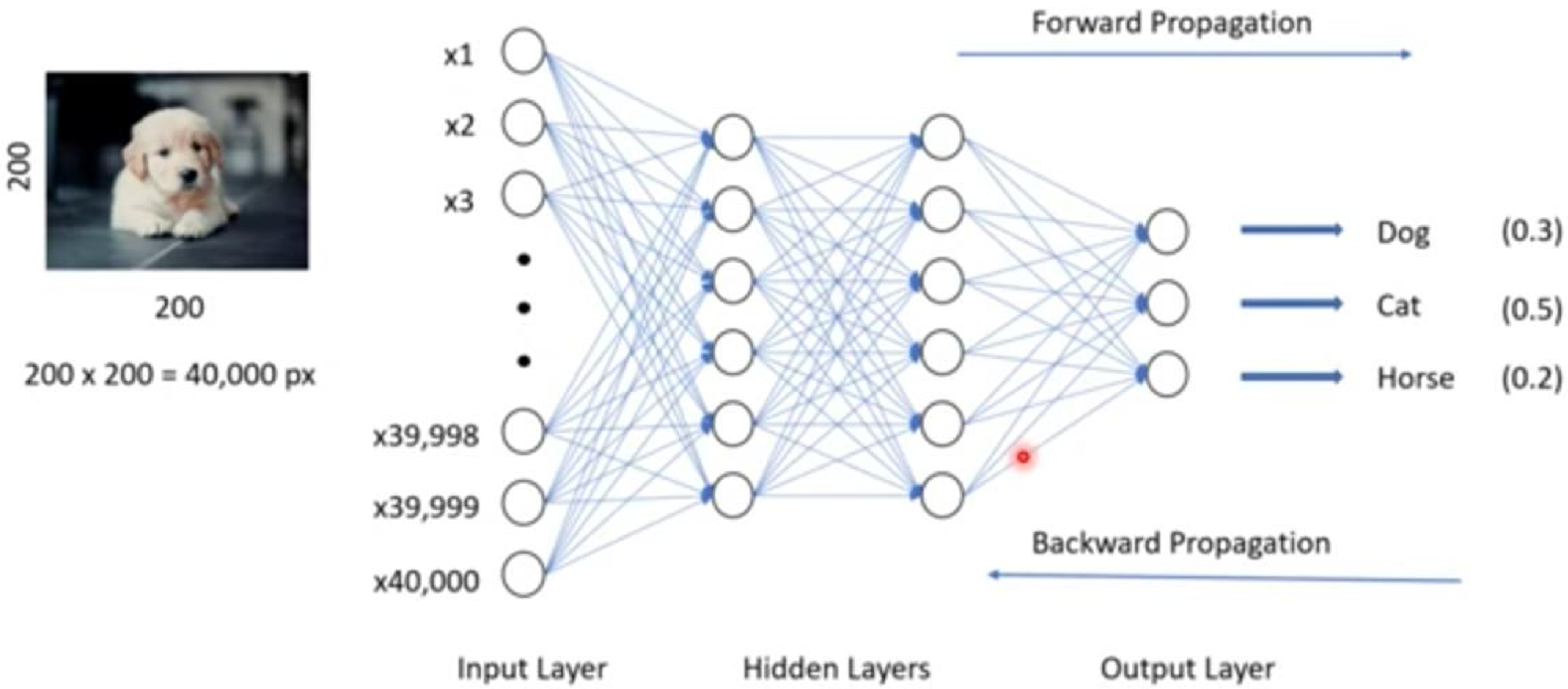


Working of a Neural Network



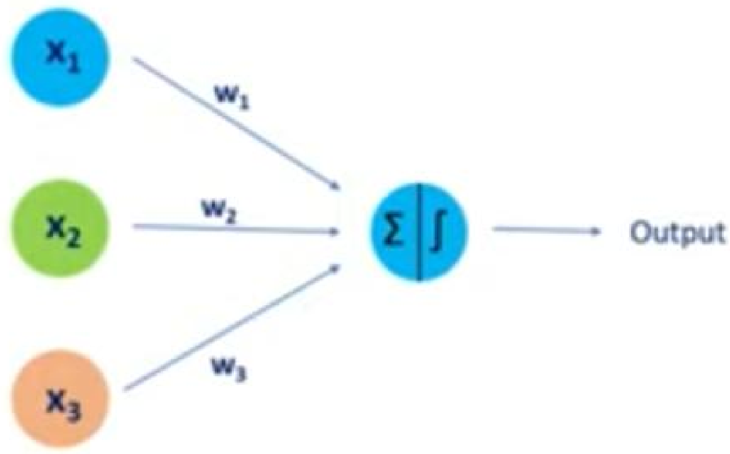
Working of a Neural Network

Activation Function (x1*w1 + x2*w2 +....+ x39,999 * w39,999 + x40,000 * w40,000 + b)



Siddhardhan

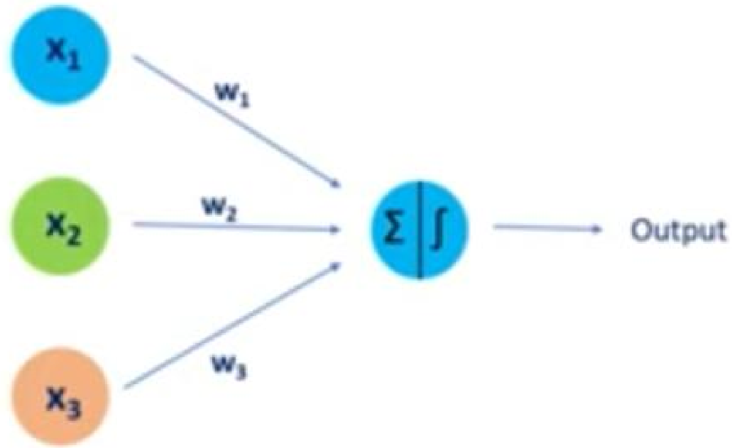
Perceptron in Deep Learning



Perceptron

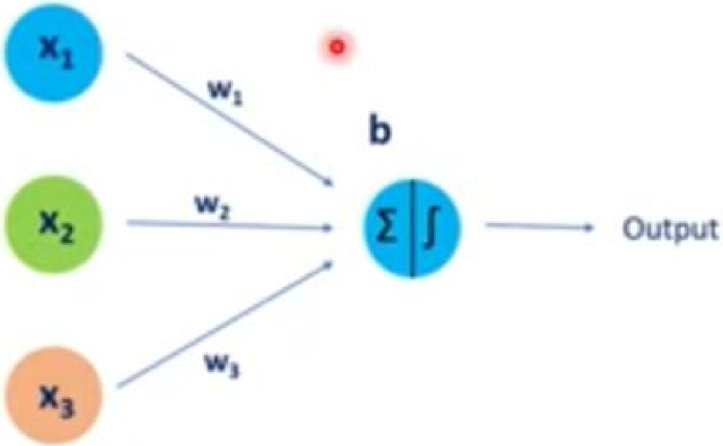
Topics:

- Deep Learning & Perceptron
- What is a Perceptron?
- Mathematical representation of a Perceptron
- Activation Functions used in a Perceptron



Perceptron

A perceptron is a basic artificial neuron that takes inputs, applies weights, combines them, and produces an output using an activation function. It is used for tasks like binary classification and is a building block of neural networks

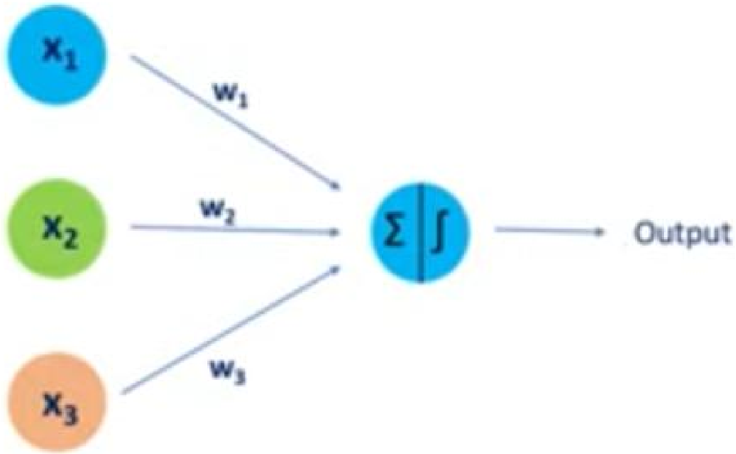


- Inputs
- Weights
- Bias
- Weighted Sum
- Activation Function
- Output

Perceptron Formula

Mathematical representation of a Perceptron:

- Input vector: $x = [x_1, x_2, \ldots, x_n]$
- Weight vector: $w = [w_1, w_2, \ldots, w_n]$
- Bias: b
- Summation function(Σ): $z = (w_1 * x_1) + (w_2 * x_2) + \ldots + (w_n * x_n) + b$
- Activation function: $\phi(z)$



Activation Functions

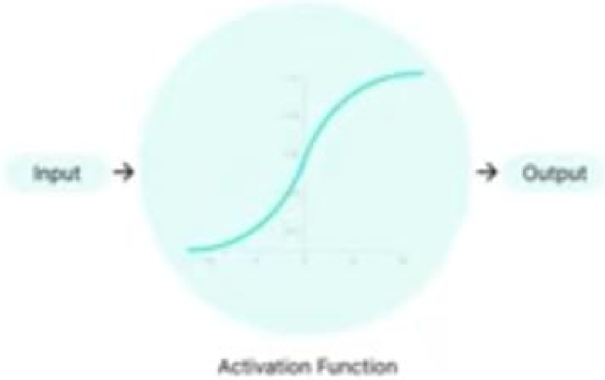
Activation Functions used in a Perceptron:

- Sigmoid function: Output is a continuous value between 0 and 1. Example: $\phi(z) = 1 / (1 + \exp(-z))$.

- Step function: Output is binary (0 or 1) based on a threshold. Example: $\phi(z) = 1$ if $z \geq 0$, else $\phi(z) = 0$.

- Sign function: Output is binary (-1 or 1) based on the sign of the input. Example: $\phi(z) = 1$ if $z \geq 0$, else $\phi(z) = -1$.

- ReLU (Rectified Linear Unit) function: Output is the input value if it is positive, else 0. Example: $\phi(z) = \max(0, z)$.



Activation Functions

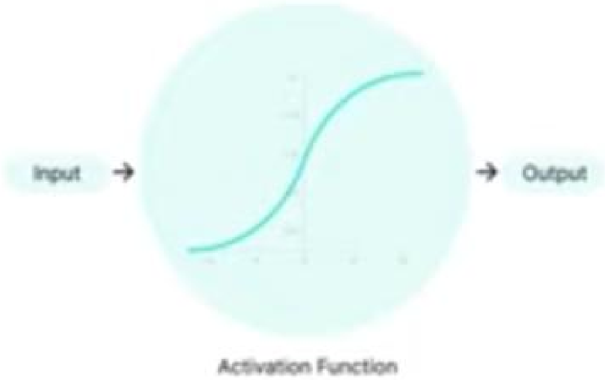
Activation Functions used in a Perceptron:

- Sigmoid function: Output is a continuous value between 0 and 1. Example: $\phi(z) = 1 / (1 + \exp(-z))$.

- Step function: Output is binary (0 or 1) based on a threshold. Example: $\phi(z) = 1$ if $z \geq 0$, else $\phi(z) = 0$.

- Sign function: Output is binary (-1 or 1) based on the sign of the input. Example: $\phi(z) = 1$ if $z \geq 0$, else $\phi(z) = -1$.

- ReLU (Rectified Linear Unit) function: Output is the input value if it is positive, else 0. Example: $\phi(z) = \max(0, z)$.



Perceptron – Example Calculation

Perceptron with Sigmoid activation function :

$x_1 = 0.2$ $w_1 = 0.1$ $b = -0.2$

$x_2 = 0.4$ $w_2 = 0.5$

$x_3 = 0.6$ $w_3 = 0.3$

$$z = w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + b$$

$$\text{Output} = \phi(z) = 1 / (1 + \exp(-z))$$

$z = (0.1 * 0.2 + 0.5 * 0.4 + 0.3 * 0.6) - 0.2$

$z = 0.02 + 0.2 + 0.18 - 0.2$

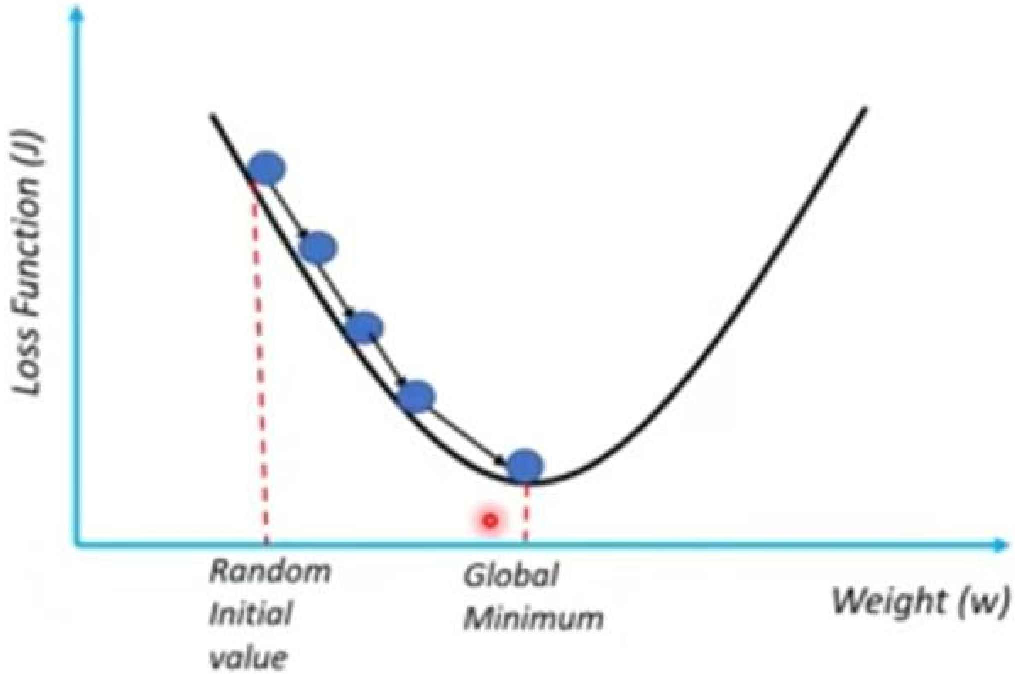
$z = 0.2$

Output = $1 / (1 + \exp(-0.2))$

Output = $1 / (1 + 0.818)$

Output = 0.5498

Gradient Descent



Gradient Descent

Gradient Descent is an optimization algorithm used for minimizing the loss function in various machine learning algorithms. It is used for updating the parameters of the learning model.

$$w = w - L*dw$$

$$b = b - L*db$$

w --> weight
b --> bias
L --> Learning Rate

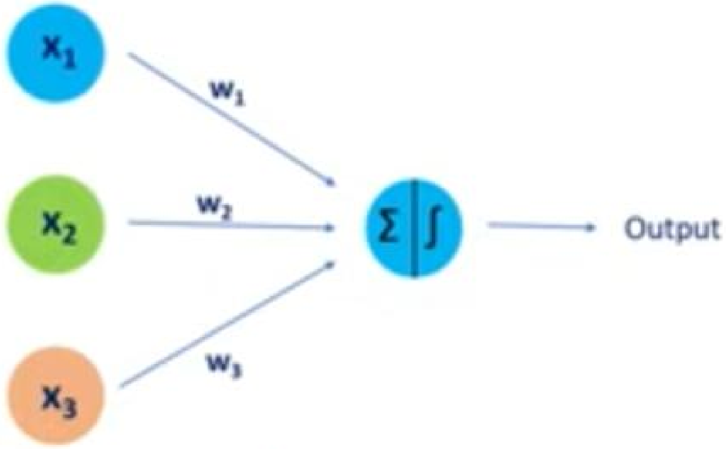
dw --> Partial Derivative of loss function with respect to w

db --> Partial Derivative of loss function with respect to b

Perceptron – Limitations

Limitations of a Perceptron:

- Linear Separability
- Binary Classification
- Lack of complexity (hidden layers complexity)
- Lack of Generalization
- Sensitivity to Initial Weights



Siddhardhan

Artificial Neural Networks

Deep Learning Course - 1.7.

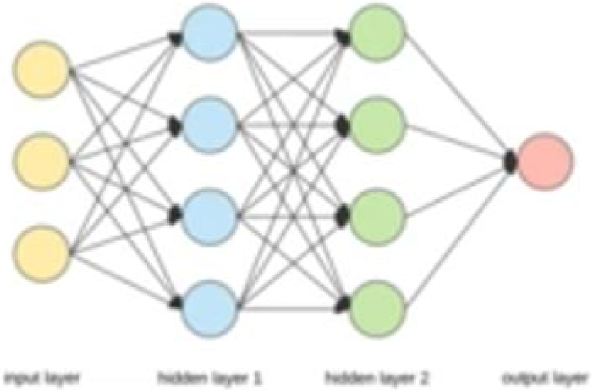
Artificial Neural Network

An Artificial Neural Network (ANN) is a computational model inspired by the human brain's network of neurons. ANNs consist of layers of interconnected nodes or neurons, each processing inputs and passing their outputs to the next layer.

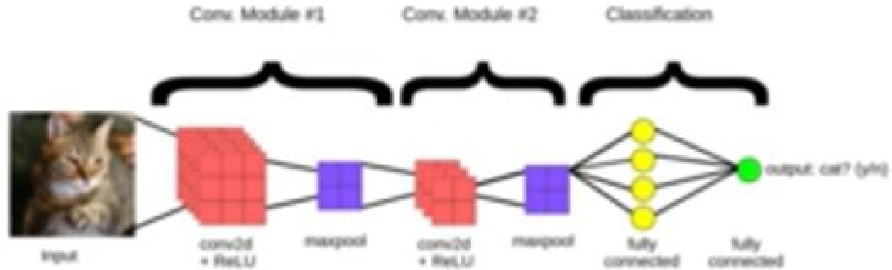
Few Specialized Types of ANN:

- Convolutional Neural Network (CNN)
- Recurrent Neural Network
- GAN

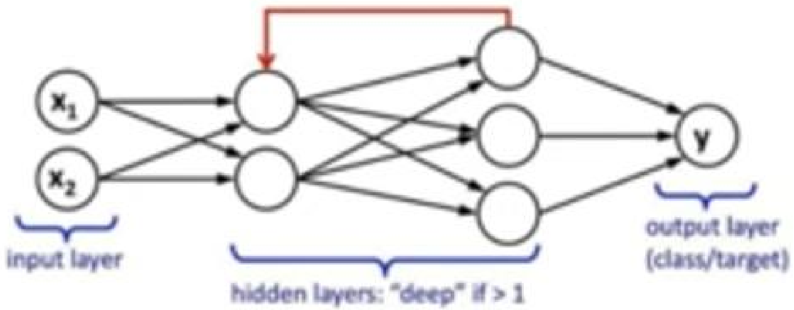
Artificial Neural Networks



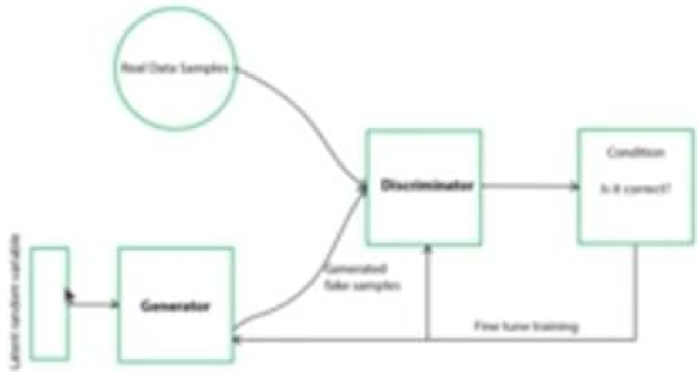
Multi Layer Perceptron (MLP)



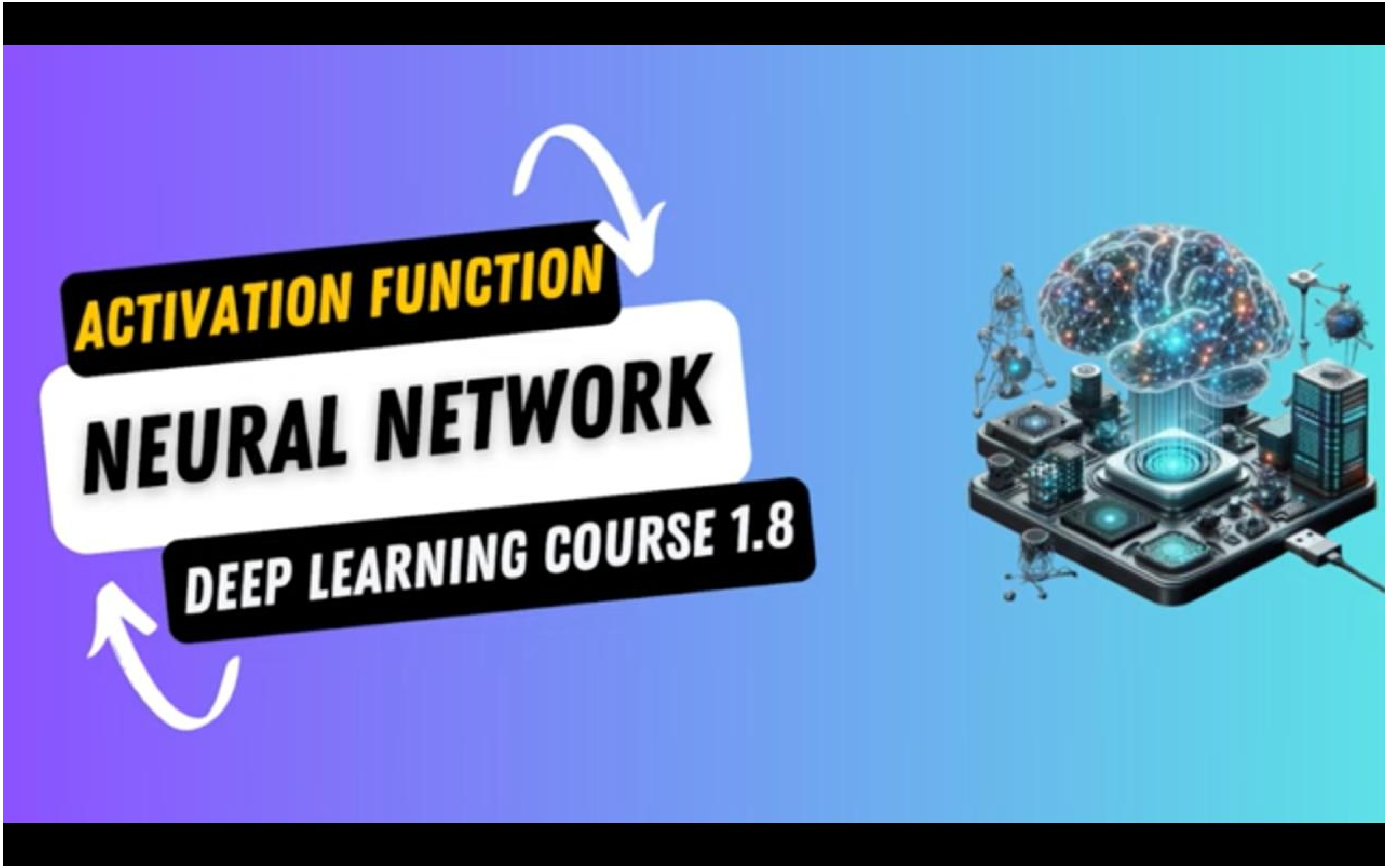
Convolutional Neural Network (CNN)



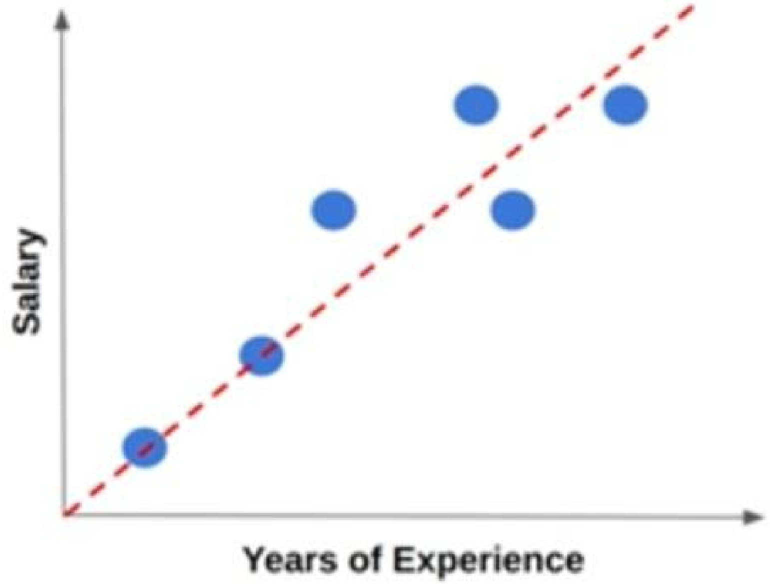
Recurrent Neural Network (RNN)



Generative Adversarial Network (GAN)


ACTIVATION FUNCTION
NEURAL NETWORK
DEEP LEARNING COURSE 1.8

Working of ML Models

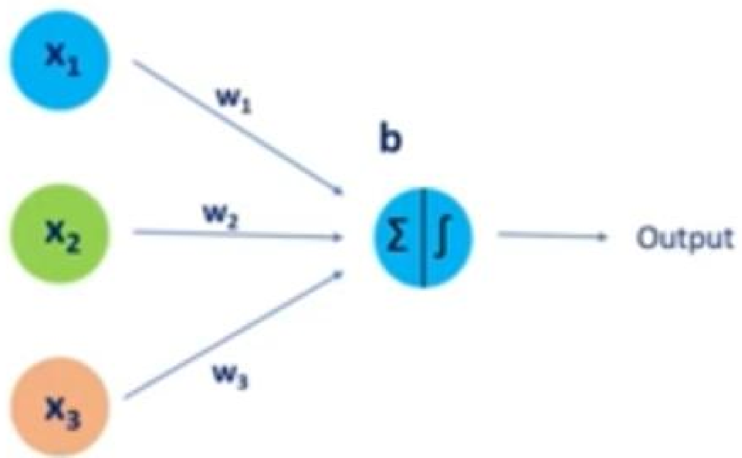


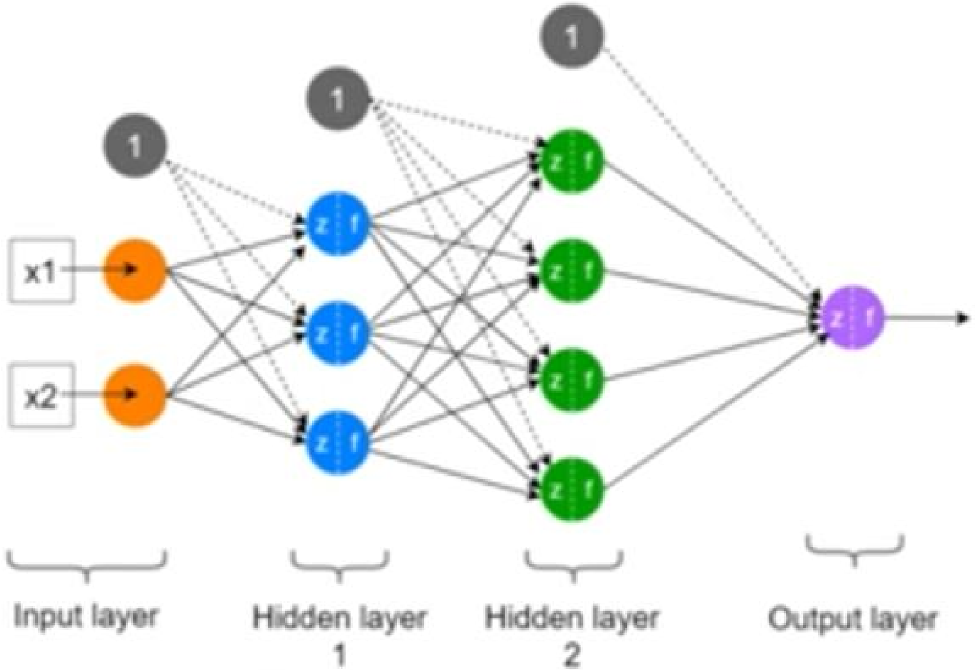
$Y = mx + c$

Linear Relationship

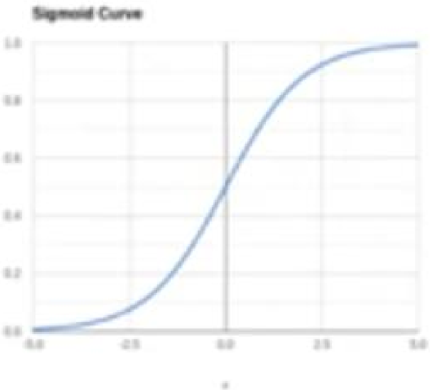
Activation Function

An activation function is a mathematical function applied to a neuron's processed input, transforming the neuron's output to the next layer in the network. The primary purpose of an activation function is to introduce non-linearity into the output of a neuron, which enables the network to learn complex patterns during training.

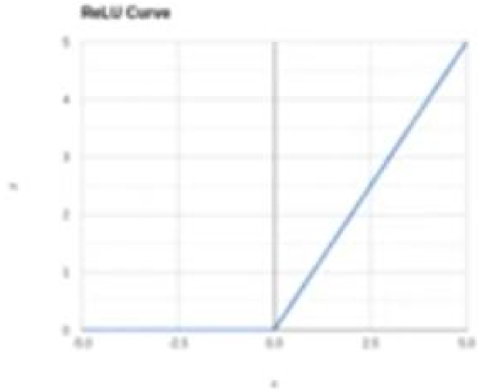




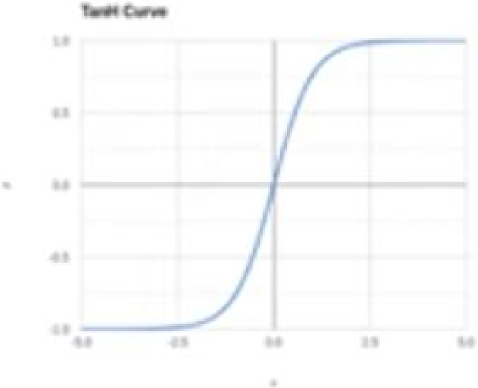
Activation Function - Types



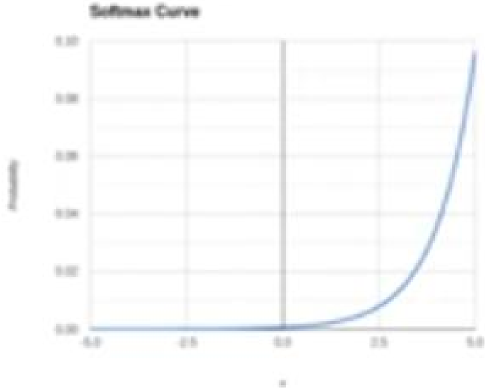
- Range: 0 to 1
- Useful for binary classification
- Susceptible to the vanishing gradient problem



- Range: 0 to ∞
- Computational efficiency
- Mitigates vanishing gradient problem



- Range: -1 to 1
- Used in Hidden layers
- Zero-centered making it effective in some cases



- Range: Probability
- Output layer in Multi-class classification
- Raw values to probability

Activation Function - Types

Function	Output Range	Effective for	Selection
Sigmoid	0-1	Binary classification	Use with caution
ReLU	0-∞	Hidden layers	Popular choice
Tanh	-1-1	Zero-centered outputs	Less common
Softmax	0-1 (sum to 1)	Use in output layer	Multi-class classification

Siddhardhan

ACTIVATION FUNCTION

MATHEMATICAL UNDERSTANDING

DEEP LEARNING COURSE 1.9

Activation Function

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

$$\mathrm{ReLU}(x) = \max(0, x)$$

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

$$\mathrm{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Sigmoid

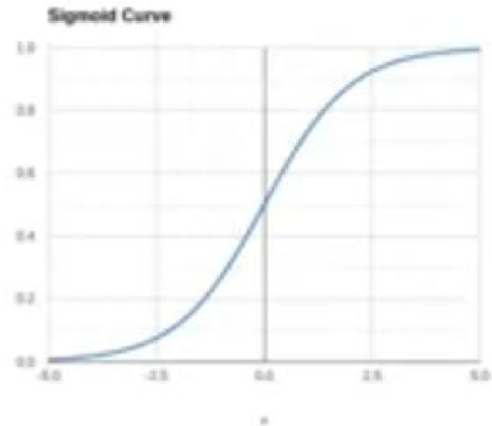
$$Sigmoid\ Function\ Calculation\ for\ x = 2$$

$$Given\ the\ sigmoid\ function:$$

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

$$For\ x = 2:$$

$$\sigma(2) = \frac{1}{1+e^{-2}} \approx 0.881$$



ReLU

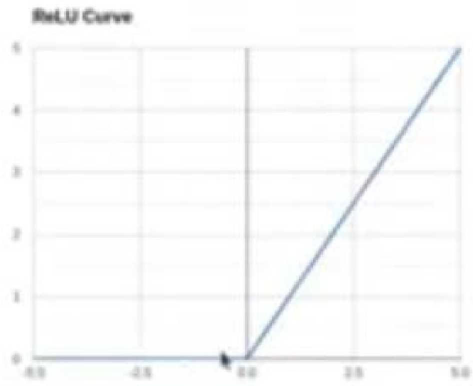
Rectified Linear Unit (ReLU) Function Calculation for $x = -3$

Given the ReLU function :

$$\mathrm{ReLU}(x) = \max(0, x)$$

$$For\ x = -3 :$$

$$\mathrm{ReLU}(-3) = \max(0, -3) = 0$$



Tan h

$$Hyperbolic\ Tangent\ (Tanh)\ Function\ Calculation\ for\ x = 1$$

$$Given\ the\ tanh\ function:$$

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

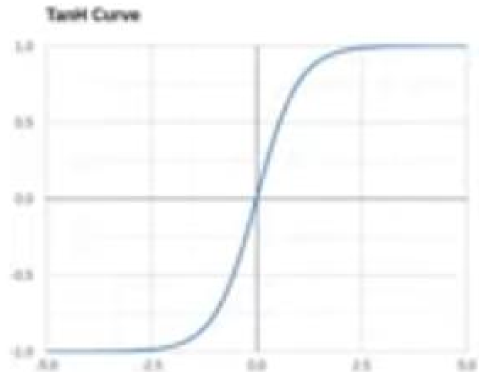
e^{-1000} −

$$\frac{0 - 1000}{0 + 1000}$$

$$For\ x = 1:$$

$$\tanh(1) = \frac{e^1 - e^{-1}}{e^1 + e^{-1}} \approx 0.762$$

−1



Softmax

Softmax calculation for $x_1 = 2,\ x_2 = 1,$ *and* $x_3 = -1,$

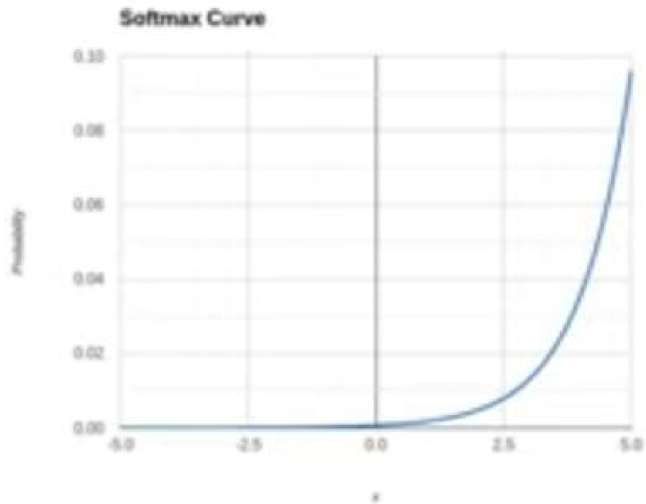
Given the Softmax function :

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

$$\text{Softmax}(x_1) = \frac{e^2}{e^2+e^1+e^{-1}} \approx 0.705$$

$$\text{Softmax}(x_2) = \frac{e^1}{e^2+e^1+e^{-1}} \approx 0.259$$

$$\text{Softmax}(x_3) = \frac{e^{-1}}{e^2+e^1+e^{-1}} \approx 0.035$$

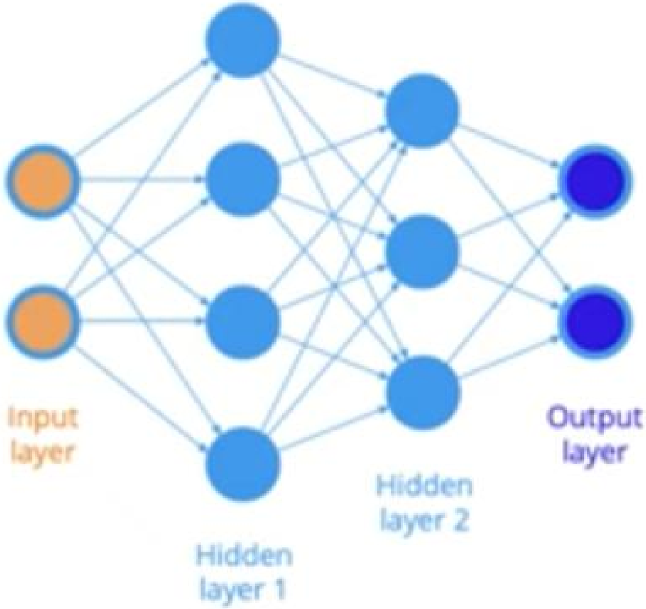


Siddhardhan

Loss Function, Optimizer, Forward & Backward Propagation

Deep Learning Course - 1.10

Neural Network - Working



Cat / No Cat

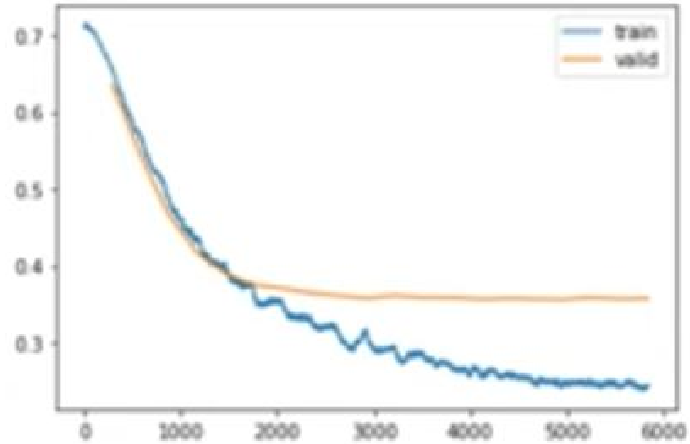
Loss Function

A **Loss Function** quantifies how well a model's predictions match the actual target values in the training data.

The goal of training a deep learning model is to **minimize** this loss function, which helps the model learn the underlying patterns in the data and make accurate predictions on new, unseen data.

Examples:

- Mean Squared Error (MSE)
- Mean Absolute Error (MAE)
- Cross-Entropy Loss

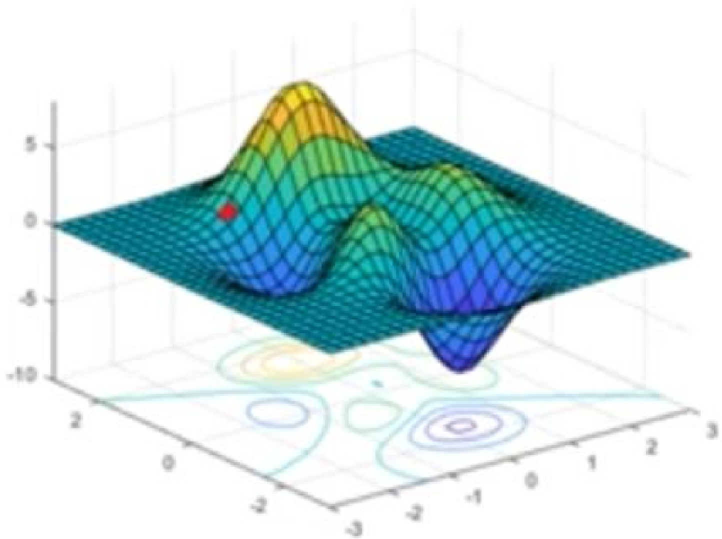


Optimizer

In neural networks, **optimizers** are algorithms used to minimize the error function or loss function by adjusting the **weights** and **biases** of the network during the training process.
They play a crucial role in updating the model parameters in a way that helps the model **converge** to the optimal solution.

Examples:

- Stochastic Gradient Descent (SGD)
- Momentum
- Adam (Adaptive Moment Estimation)



Forward & Backward Propagation