Neural Network

Neural networks are machine learning models that mimic the complex functions of the human brain. These models consist of interconnected nodes or neurons that process data, learn patterns and enable tasks such as pattern recognition and decision-making.

Key Components of a Neural Network:

- Neurons: The basic units of computation that receive and process inputs using an activation function.
- Connections: Links between neurons, where data flows, regulated by weights and biases.
- Weights & Biases: Parameters that influence how much impact an input has on a neuron's output.
- Propagation Function: Processes and transmits data across the network layers.
- Learning Rule: Adjusts the weights and biases during training to improve performance.

Learning in neural networks follows a structured, three-stage process:

1. Input Computation: Data is fed into the network.
2. Output Generation: Based on the current parameters, the network generates an output.
3. Iterative Refinement: The network refines its output by adjusting weights and biases, gradually improving its performance on diverse tasks.

Importance of Neural Networks

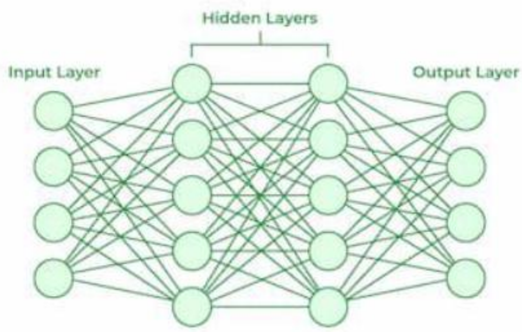
Neural networks are powerful for solving complex problems and discovering patterns in large datasets. They are essential to technologies such as:

- Natural Language Processing
- Autonomous Vehicles
- Medical Diagnosis

They enhance efficiency, support automation, and are a foundational technology behind modern artificial intelligence.

Layers in Neural Network Architecture

1. **Input Layer:** This is where the network receives its input data. Each input neuron in the layer corresponds to a feature in the input data.
2. **Hidden Layers:** These layers perform most of the computational heavy lifting. A neural network can have one or multiple hidden layers. Each layer consists of units (neurons) that transform the inputs into something that the output layer can use.
3. **Output Layer:** The final layer produces the output of the model. The format of these outputs varies depending on the specific task like classification, regression.



Forward propagation

Forward propagation is the process through which input data moves through a neural network to produce an output. It starts at the input layer, where each neuron receives inputs that are multiplied by associated weights, added to a bias, and then passed through a mathematical operation called a linear transformation. This result is then fed into an activation function (like ReLU, sigmoid, or tanh), which introduces non-linearity and allows the network to learn complex patterns. This process continues layer by layer until the final output is generated. The output may be a probability (for classification) or a continuous value (for regression), depending on the task.

1. Linear Transformation: Each neuron in a layer receives inputs which are multiplied by the weights associated with the connections. These products are summed together and a bias is added to the sum.

2. Activation: The result of the linear transformation (denoted as z) is then passed through an activation function. The activation function is crucial because it introduces non-linearity into the system, enabling the network to learn more complex patterns. Popular activation functions include ReLU, sigmoid and tanh.

$$z = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b$$

where

- w represents the weights
- x represents the inputs
- b is the bias

Backward propagation

Backward propagation, or backpropagation, is the learning phase where the network improves its predictions. After forward propagation, the network calculates the loss—the difference between the predicted and actual output—using a loss function such as mean squared error or cross-entropy. Then, using calculus and the chain rule, the network computes the gradients—how much each weight and bias contributed to the error. These gradients guide how the parameters should be adjusted. The network then updates its weights and biases using an optimization algorithm like stochastic gradient descent (SGD), moving in the direction that minimizes the loss. This forward and backward cycle repeats for many iterations, gradually improving the model's accuracy.