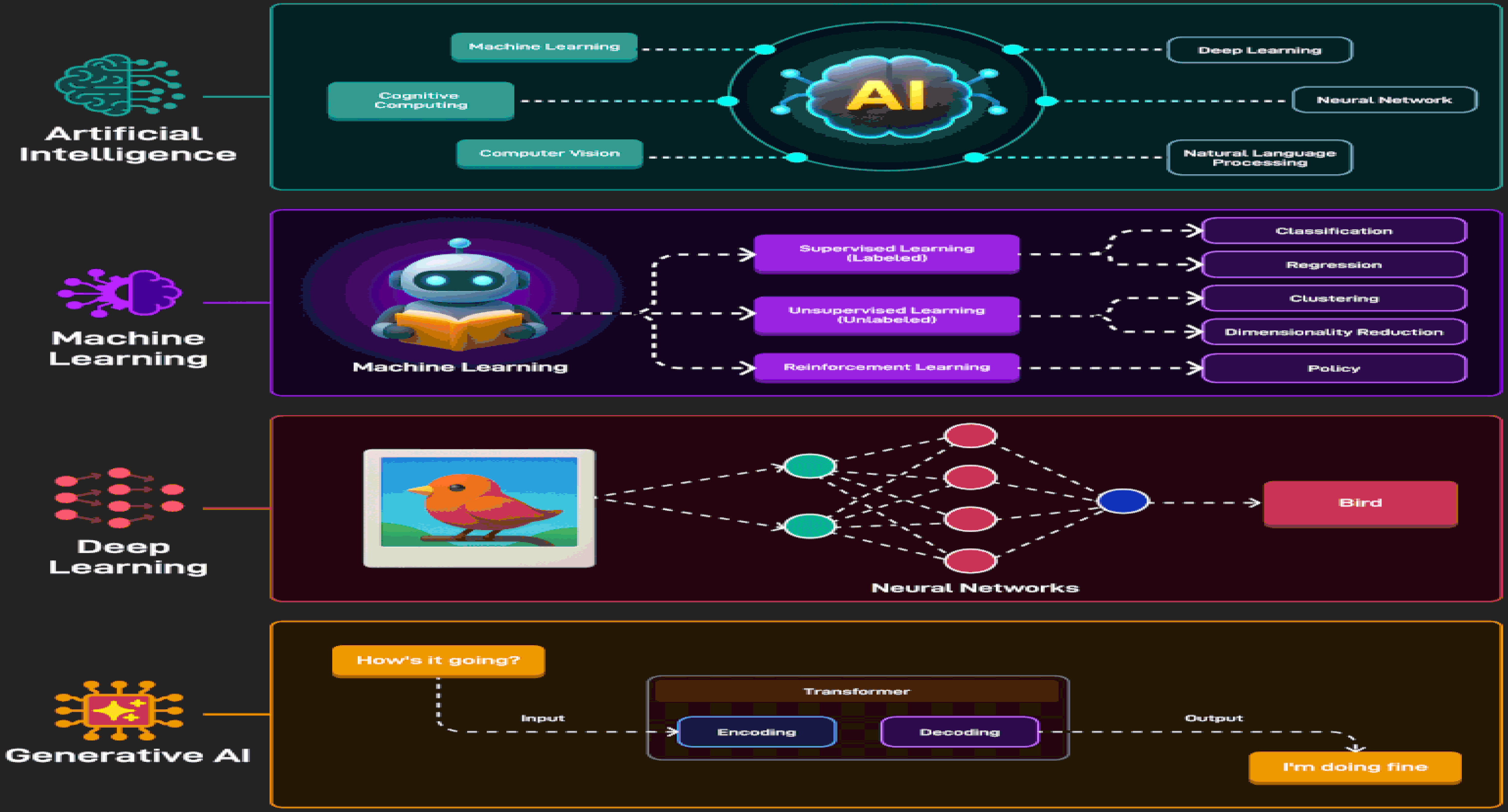
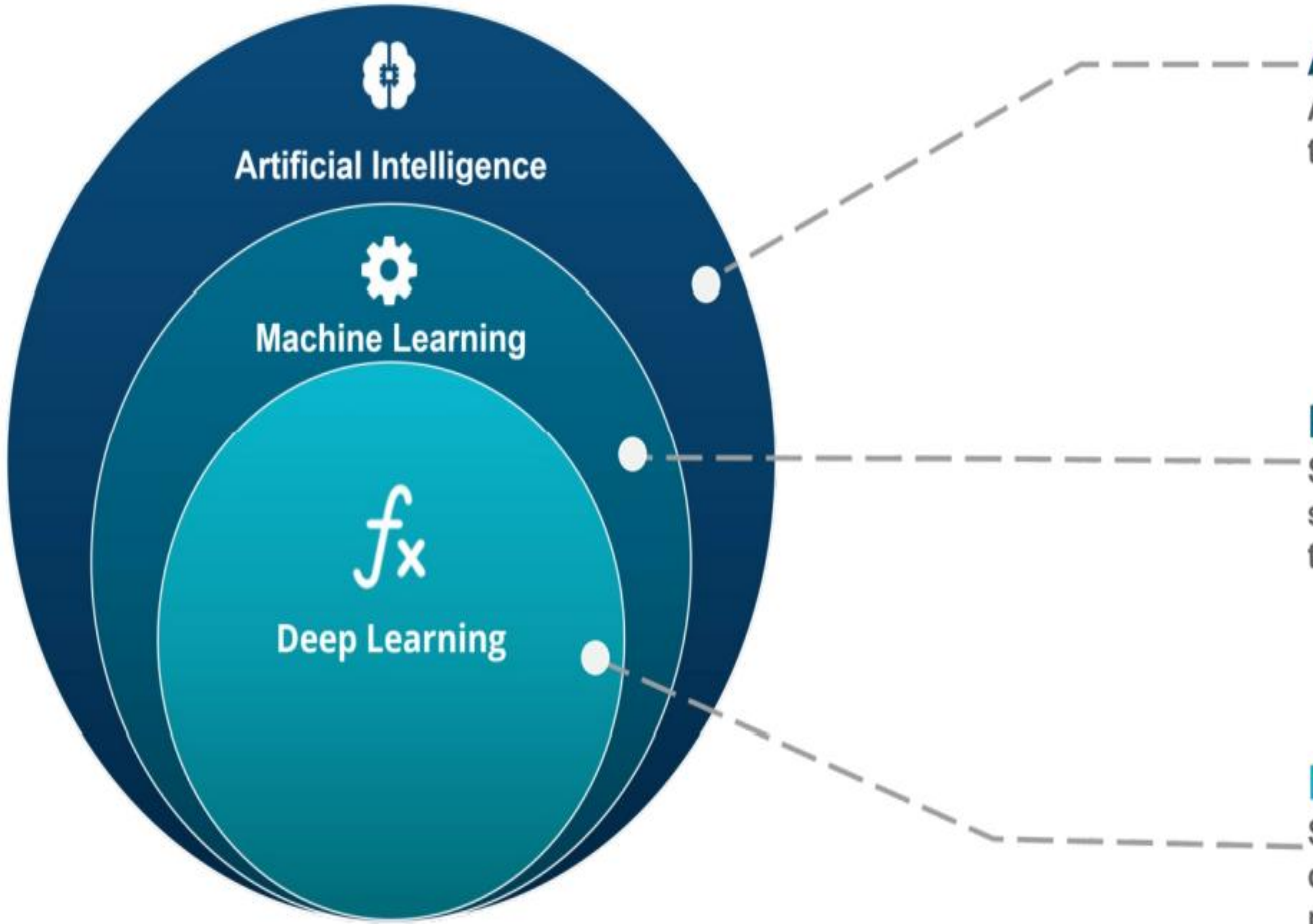
# Deep Learning

- Deep Learning is a subset of Machine Learning that uses mathematical functions to map the input to the output.
- These functions can extract non-redundant information or patterns from the data, which enables them to form a relationship between the input and the output.
- This is known as learning, and the process of learning is called training.

| Machine Learning | Deep Learning |
|---|---|
| Apply statistical algorithms to learn the hidden patterns and relationships in the dataset. | Uses artificial neural network architecture to learn the hidden patterns and relationships in the dataset. |
| Can work on the smaller amount of dataset | Requires the larger volume of dataset compared to machine learning |
| Better for the low-label task. | Better for complex task like image processing, natural language processing, etc. |
| Takes less time to train the model. | Takes more time to train the model. |
| A model is created by relevant features which are manually extracted from images to detect an object in the image. | Relevant features are automatically extracted from images. It is an end-to-end learning process. |
| Less complex and easy to interpret the result. | More complex, it works like the black box interpretations of the result are not easy. |
| It can work on the CPU or requires less computing power as compared to deep learning. | It requires a high-performance computer with GPU. |

## AI Vs Machine Learning Vs Deep Learning Vs Generative AI



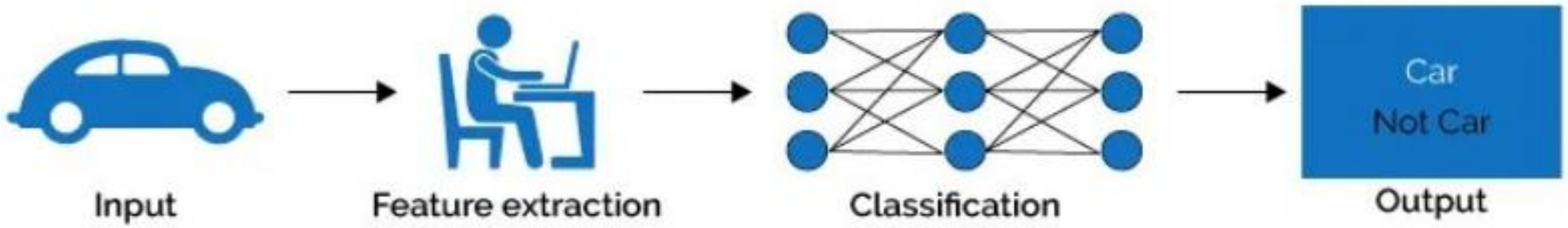


**ARTIFICIAL INTELLIGENCE**
A technique which enables machines to mimic human behaviour

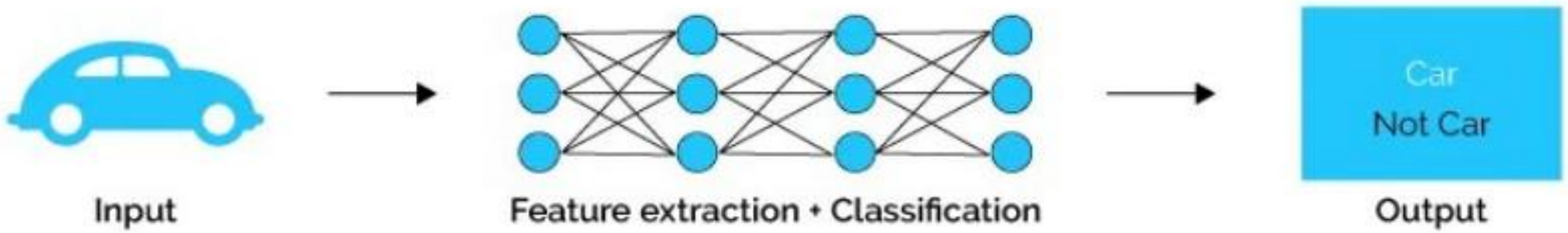
**MACHINE LEARNING**
Subset of AI technique which use statistical methods to enable machines to improve with experience

**DEEP LEARNING**
Subset of ML which make the computation of multi-layer neural network feasible

## Machine Learning

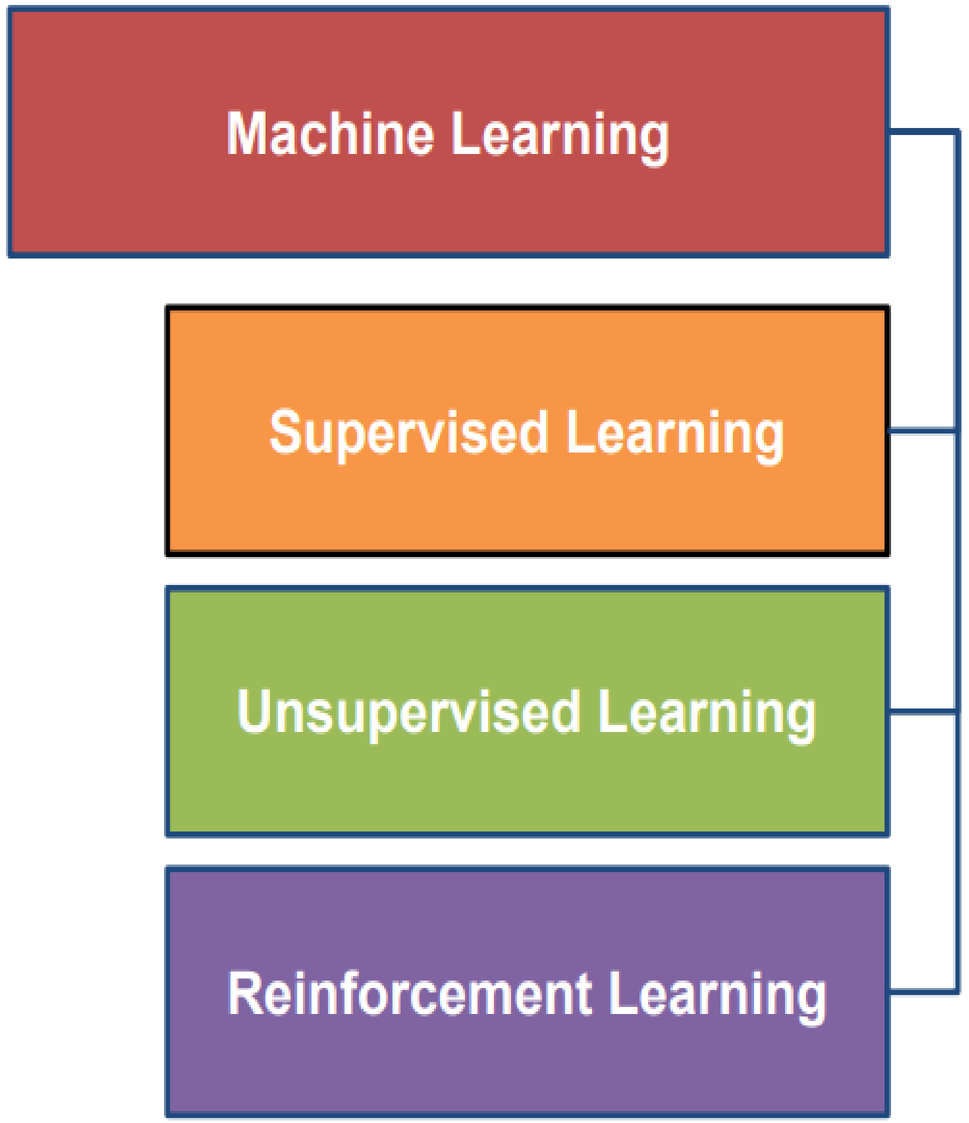


## Deep Learning

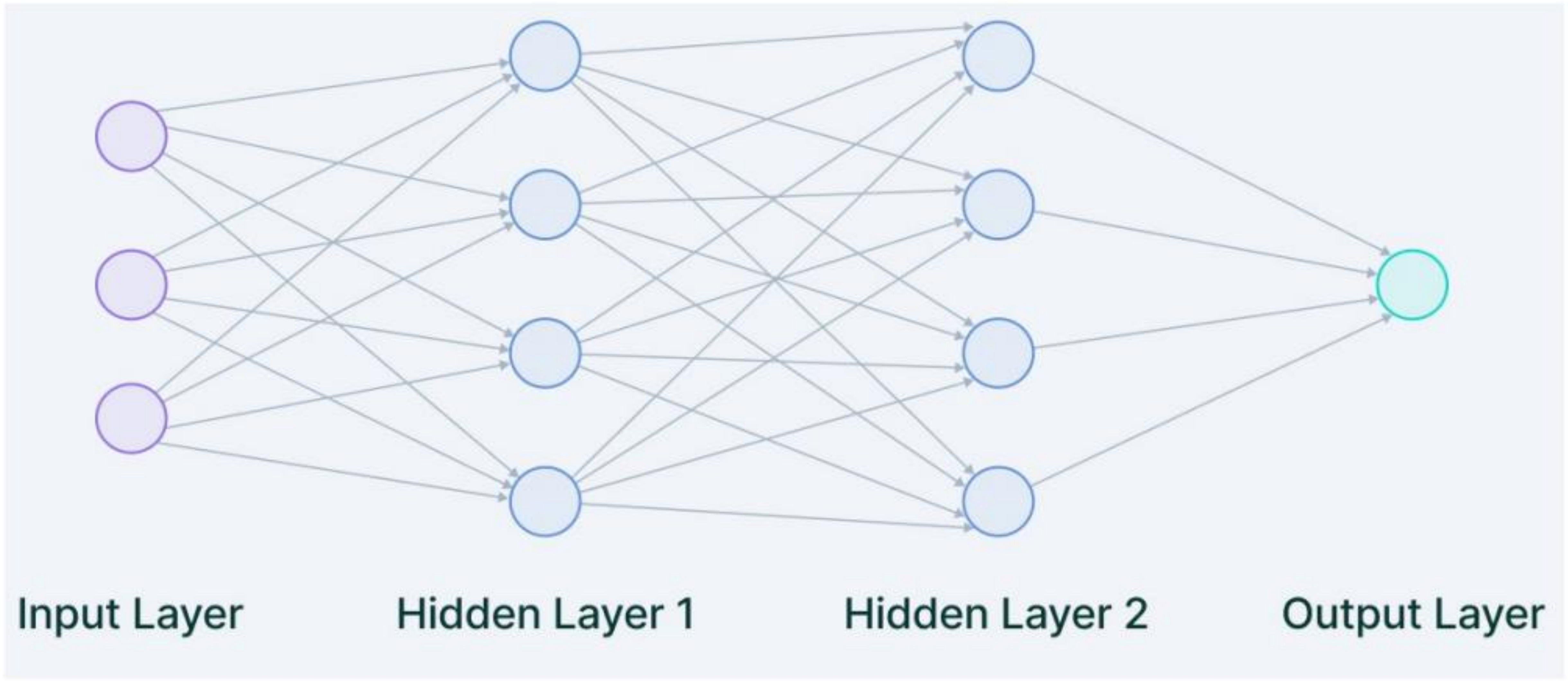


## Types of ML Algorithms

- **Supervised Learning**
  - trained with labeled data; including regression and classification problems
- **Unsupervised Learning**
  - trained with unlabeled data; clustering and association rule learning problems.
- **Reinforcement Learning**
  - no training data; stochastic Markov decision process; robotics and self-driving cars.



## How does Deep Learning work?



## Advantages of Deep Learning

**High accuracy:** Deep Learning algorithms can achieve state-of-the-art performance in various tasks such as image recognition and natural language processing.

**Automated feature engineering:** Deep Learning algorithms can automatically discover and learn relevant features from data without the need for manual feature engineering.

**Scalability:** Deep Learning models can scale to handle large and complex datasets and can learn from massive amounts of data.

**Flexibility:** Deep Learning models can be applied to a wide range of tasks and can handle various types of data such as images, text and speech.

**Continual improvement:** Deep Learning models can continually improve their performance as more data becomes available.

## Disadvantages of Deep Learning

Deep learning has made significant advancements in various fields but there are still some challenges that need to be addressed. Here are some of the main challenges in deep learning:

**1.Data availability**: It requires large amounts of data to learn from. For using deep learning it's a big concern to gather as much data for training.

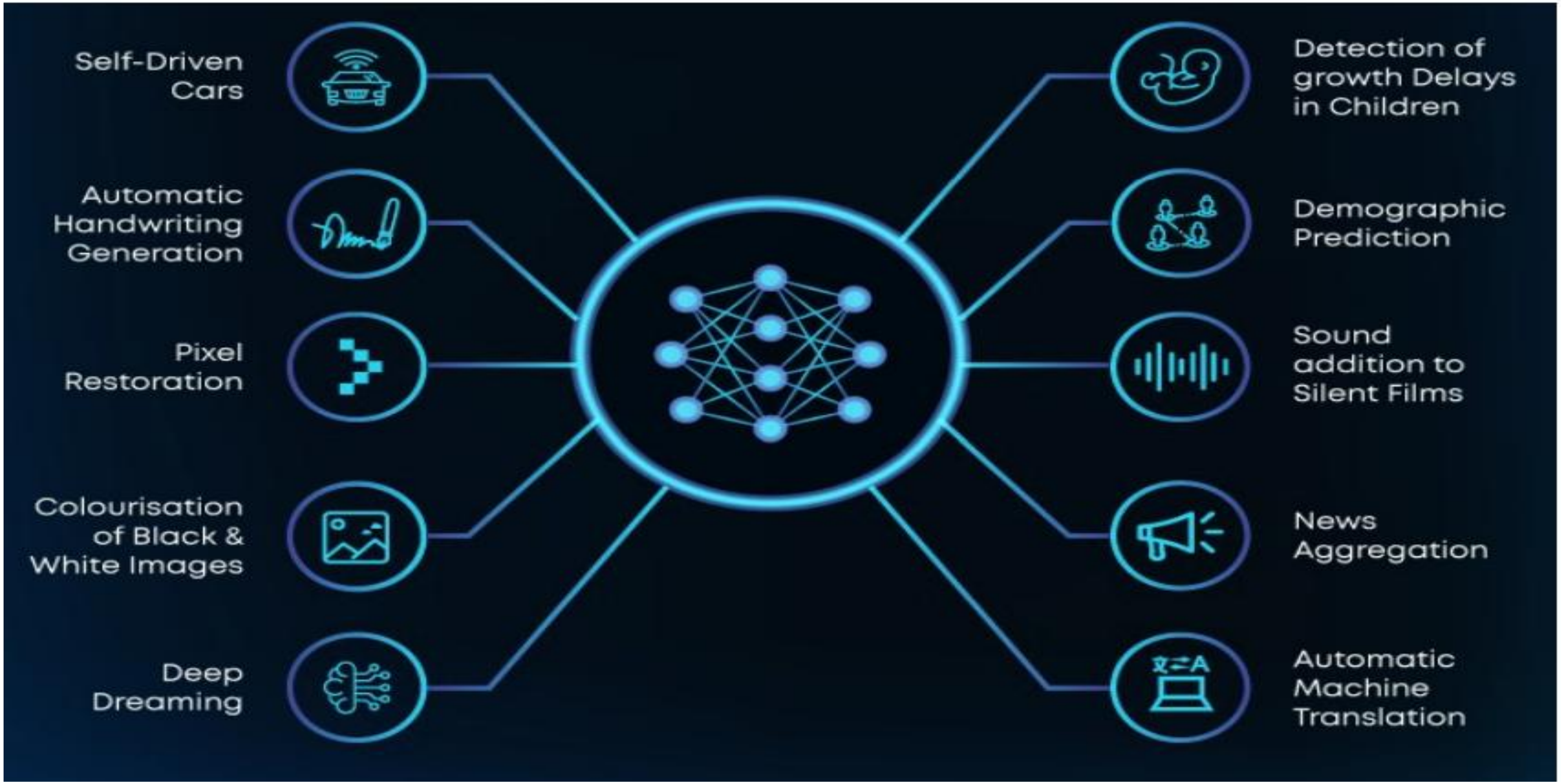
**2.Computational Resources**: For training the deep learning model, it is computationally expensive because it requires specialized hardware like GPUs and TPUs.

**3.Time-consuming:** While working on sequential data depending on the computational resource it can take very large even in days or months.

4.I**nterpretability:** Deep learning models are complex, it works like a black box. It is very difficult to interpret the result.

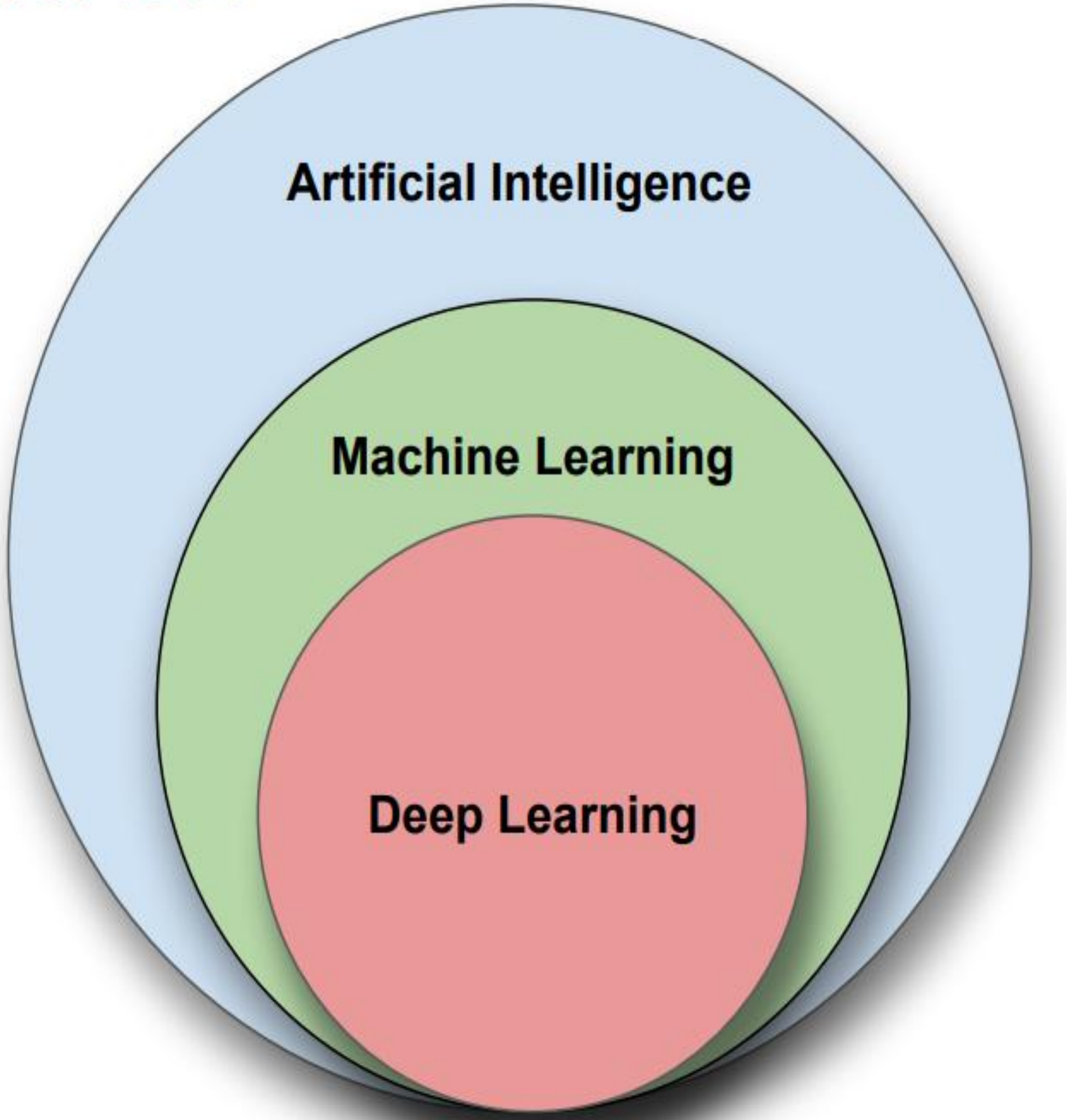
**5.Overfitting:** when the model is trained again and again it becomes too specialized for the training data leading to overfitting and poor performance on new data.

# Deep Learning: Applications


Self-Driven Cars
Automatic Handwriting Generation
Pixel Restoration
Colourisation of Black & White Images
Deep Dreaming
Detection of growth Delays in Children
Demographic Prediction
Sound addition to Silent Films
News Aggregation
Automatic Machine Translation

# Relationship of AI, ML and DL

- **Artificial Intelligence (AI)** is anything about man-made intelligence exhibited by machines.
- **Machine Learning (ML)** is an approach to achieve **AI**.
- **Deep Learning (DL)** is one technique to implement **ML**.

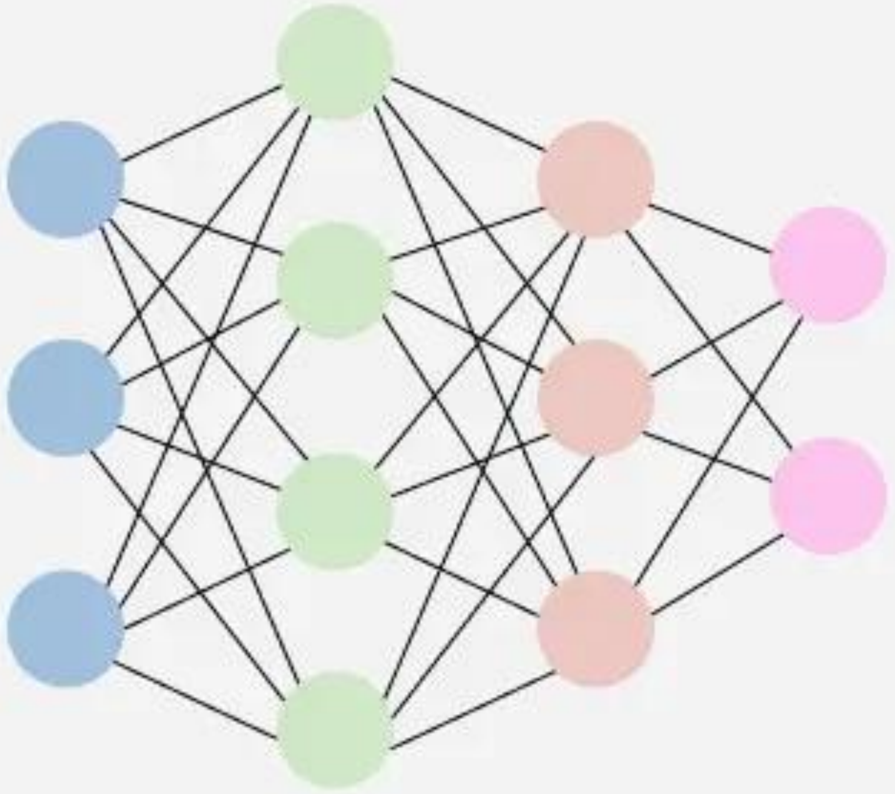


# Deep Learning: Limitations

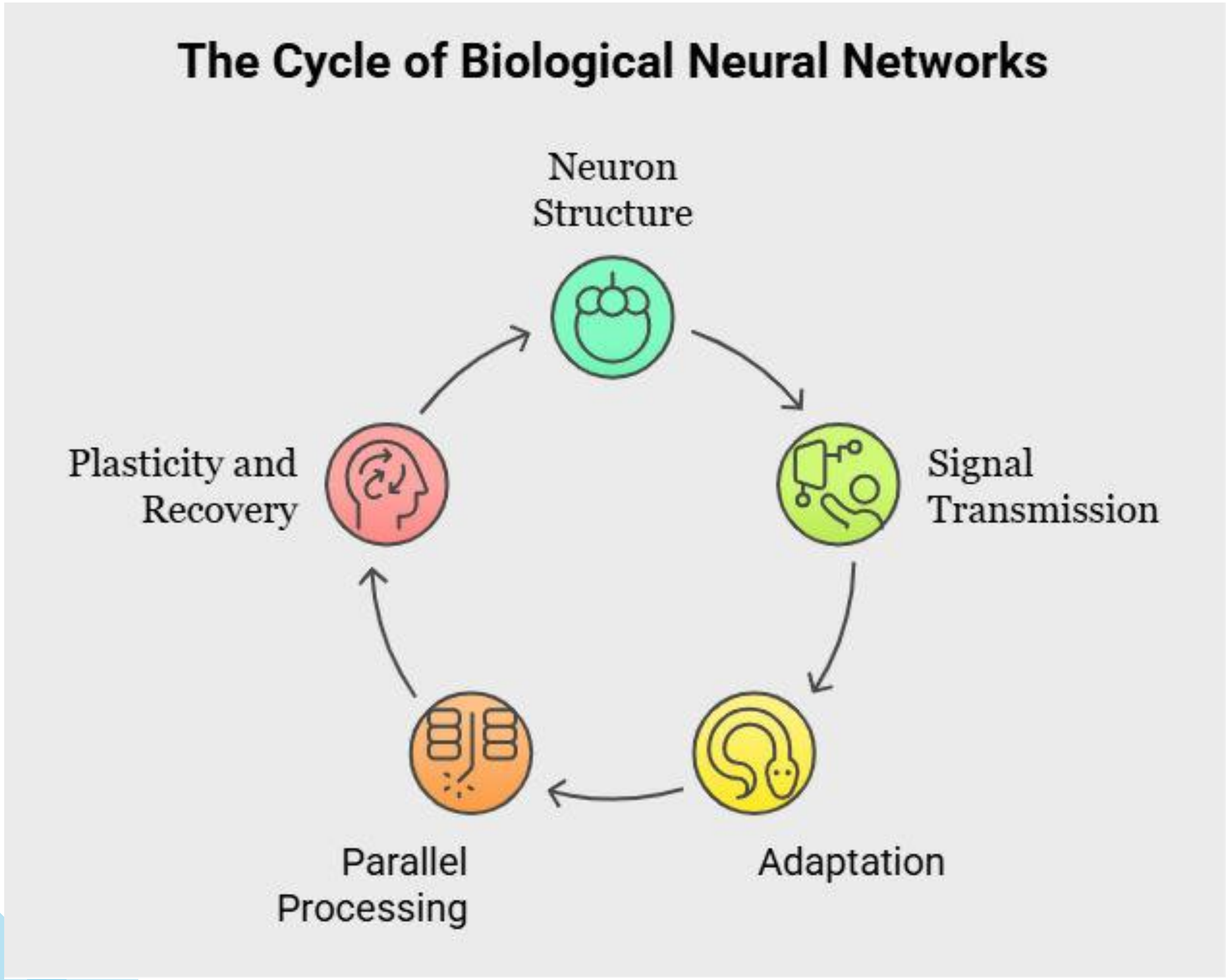
- Data availability
- The complexity of the model
- Lacks global generalization
- Incapable of Multitasking
- Hardware dependence

## What is Deep Learning?

1. Deep learning uses multi-layered neural networks.
2. Learns from large datasets to make predictions automatically.



- **Biological Neural Network?**
- A Biological Neural Network is a network of nerve cells, known as neurons, found in all living organisms. Biological neurons connect through specialized contact points called synapses. Signals that pass from one neuron to another carry the instructions for actions, thoughts, and learning.
- Unlike computer-based networks, Biological Neural Networks rely on both electrical impulses and chemical messengers to function. This setup makes learning possible over a lifetime, whether it is picking up a new language or grasping a new concept in science.


The Cycle of Biological Neural Networks
Neuron Structure
Signal Transmission
Adaptation
Parallel Processing
Plasticity and Recovery

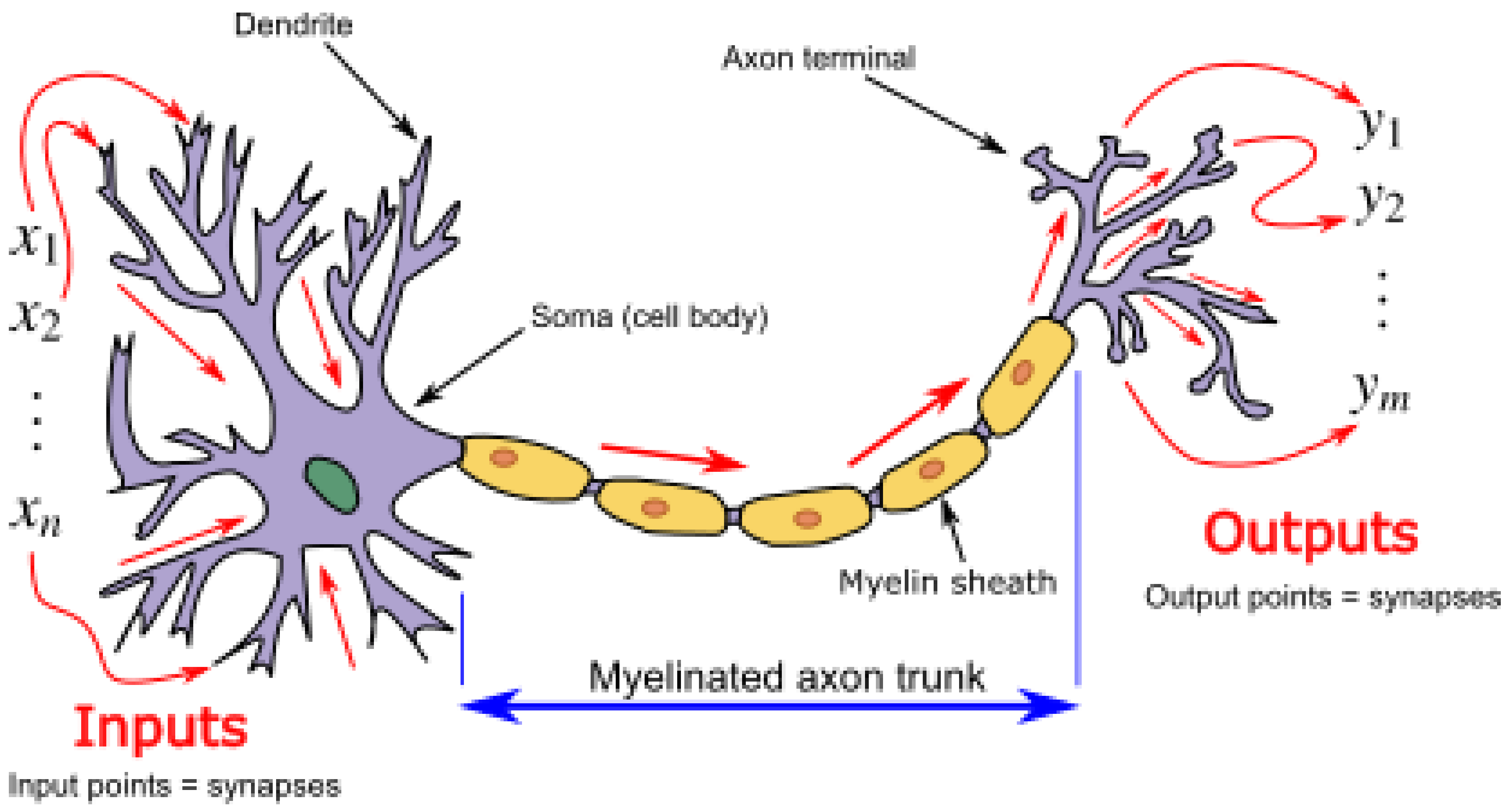
•**Biological Neuron Structure:** Each biological neuron has three main parts: the cell body, dendrites, and an axon. Dendrites receive incoming signals, and the axon sends signals out. When the cell body sums up enough input, it generates an electrical impulse.

•**Synaptic Connections:** Biological neurons connect at synapses, where chemical messengers (neurotransmitters) jump between cells. Stronger connections make it easier for signals to move from neuron to neuron.

•**Adaptation:** Biological Neural Networks adjust their links through the growth or removal of synapses. If you keep revising your notes, the neurons behind that process build a stronger bond. When a skill goes unused, the related synapses may weaken.

•**Parallel Processing:** Thousands of neurons can fire signals at once. This helps your brain handle many tasks, such as focusing on a conversation while you walk to another room.

•**Plasticity and Recovery:** Biological Neural Networks can reorganize after injury or during skill-building. This flexibility is why you can still learn something new at any age. It is also why some recovery is possible if part of the brain is damaged.


Dendrite
Axon terminal
$x_1$
$x_2$
$\vdots$
$x_n$
Soma (cell body)
$y_1$
$y_2$
$\vdots$
$y_m$
Outputs
Output points = synapses
Myelin sheath
Myelinated axon trunk
Inputs
Input points = synapses

## What is an Artificial Neural Network?

An Artificial Neural Network (ANN) is a computer-based model inspired by how neurons in the human brain pass signals. However, it relies on math and data instead of chemical and electrical impulses to recognize patterns or make predictions.

You can picture an ANN as a set of digital ‘units’ aka ‘artificial neurons’ organized in layers. Each artificial neuron takes input, multiplies it by a weight, and then sends the result forward.

With enough layers and examples, an ANN can pick up on subtle patterns — whether it’s sorting emails or suggesting new songs.

## How Does an ANN Learn?

Learning in an ANN involves adjusting the weights between units to make the output more accurate.

•**Data Input:** You provide training examples, like images of handwritten digits or snippets of speech.

•**Forward Pass:** The ANN processes each example and makes an initial guess (for example, identifying a digit as “7”).

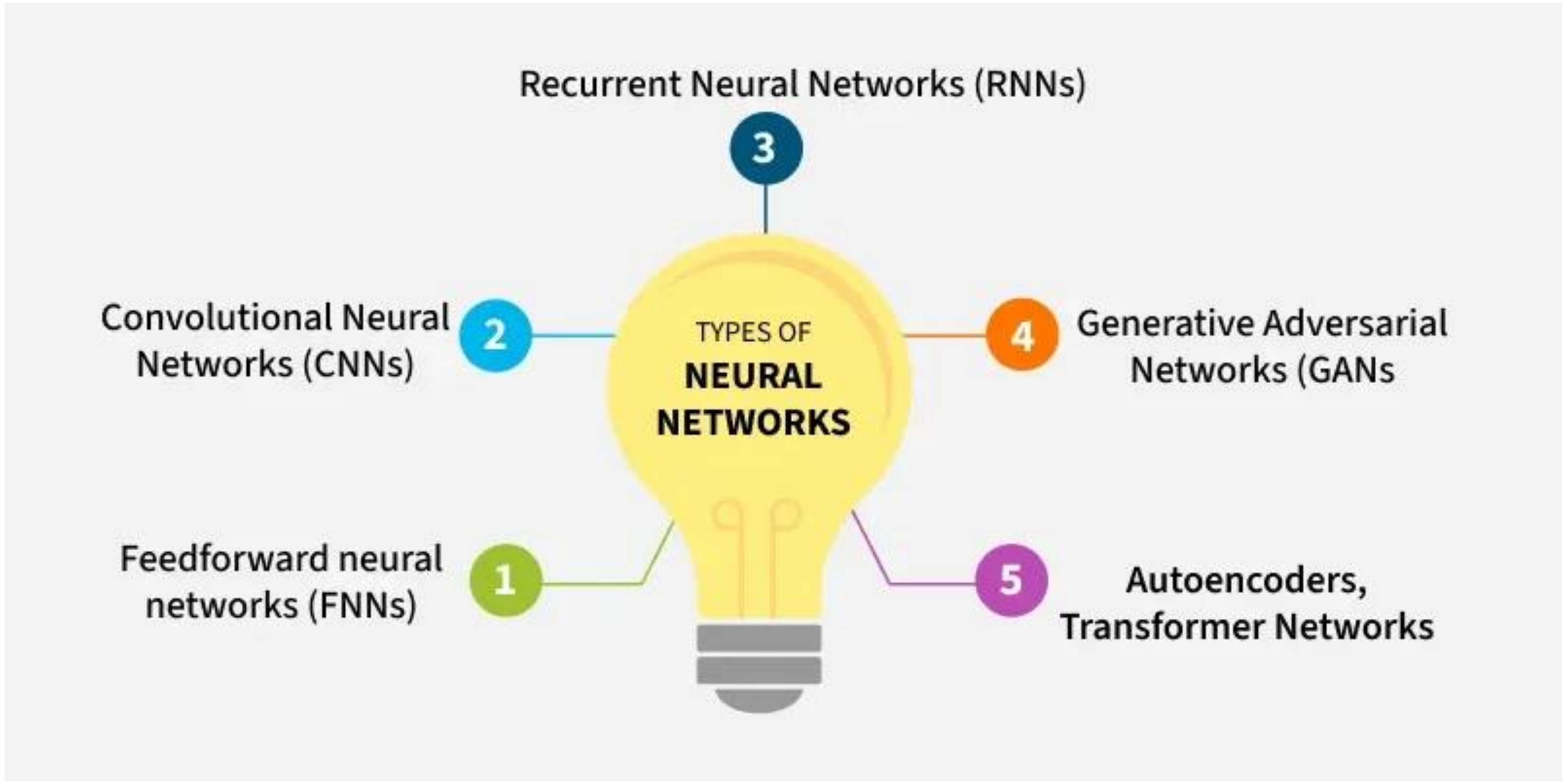
•**Error Calculation:** It compares its guess with the correct answer. The difference is its “error.”

•**Weight Adjustment:** The network tweaks the weights that led to the error. This fine-tuning is guided by algorithms such as backpropagation.

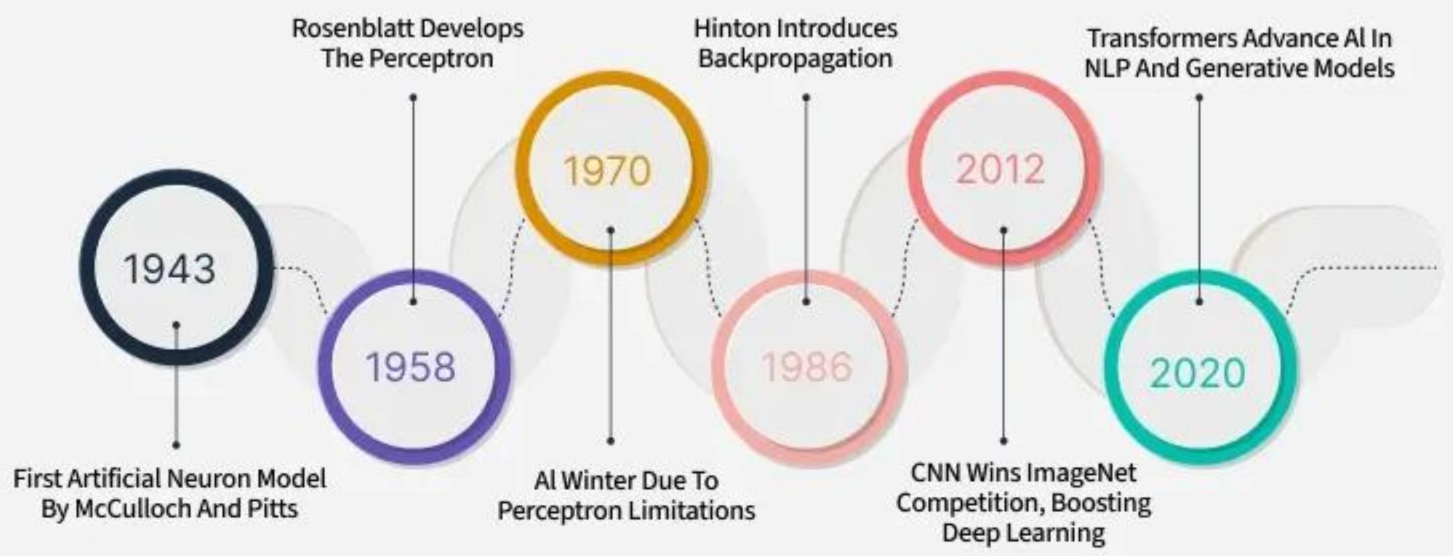
•**Iteration:** By repeating these steps with many examples, the ANN gradually improves its predictions.

## Difference Between ANN and BNN

| FEATURE | BIOLOGICAL NEURAL NETWORK (BNN) | ARTIFICIAL NEURAL NETWORK (ANN) |
|---|---|---|
| Building Block | Real neurons with dendrites, cell bodies, and axons. | Digital units (nodes) that handle numeric weights and activations. |
| Signal Type | Electrical impulses (action potentials) + chemical transmitters. | Numeric values passed between layers. |
| Learning Process | Local changes in synapse strength (e.g., neurons that fire together wire together). | Data-driven adjustments of weights guided by error calculations. |
| Adaptation | Continuous remodeling of connections through growth or pruning (plasticity). | Structured training phases that depend on large datasets and backpropagation. |
| Energy Use | About 20 watts to run a vast parallel system. | Often requires powerful processors, which can draw far more energy. |
| Fault Tolerance | Can reroute signals if neurons or areas are damaged. | Sensitive to broken nodes unless special redundancies are designed. |
| Memory Storage | Distributed across connected neurons; shaped by daily experiences. | Encoded in numeric weights; may forget old tasks when retrained. |
| Speed | Slower signal transfer but extensive parallelism. | Can compute very quickly on specialized hardware, but often in a more linear sequence. |
| Interpretability | Hard to measure individual neuron contributions in complex thoughts. | Sometimes seen as a “black box” of weights and biases. |


Recurrent Neural Networks (RNNs)
3
Convolutional Neural Networks (CNNs)
2
TYPES OF NEURAL NETWORKS
4
Generative Adversarial Networks (GANs
Feedforward neural networks (FNNs)
1
5
Autoencoders, Transformer Networks

# History And Evolution Of Neural Networks


Rosenblatt Develops The Perceptron
Hinton Introduces Backpropagation
Transformers Advance AI In NLP And Generative Models
1943
1958
1970
1986
2012
2020
First Artificial Neuron Model By McCulloch And Pitts
AI Winter Due To Perceptron Limitations
CNN Wins ImageNet Competition, Boosting Deep Learning

## Understanding Neural Networks in Deep Learning

Neural networks are capable of learning and identifying patterns directly from data without pre-defined rules. These networks are built from several key components:

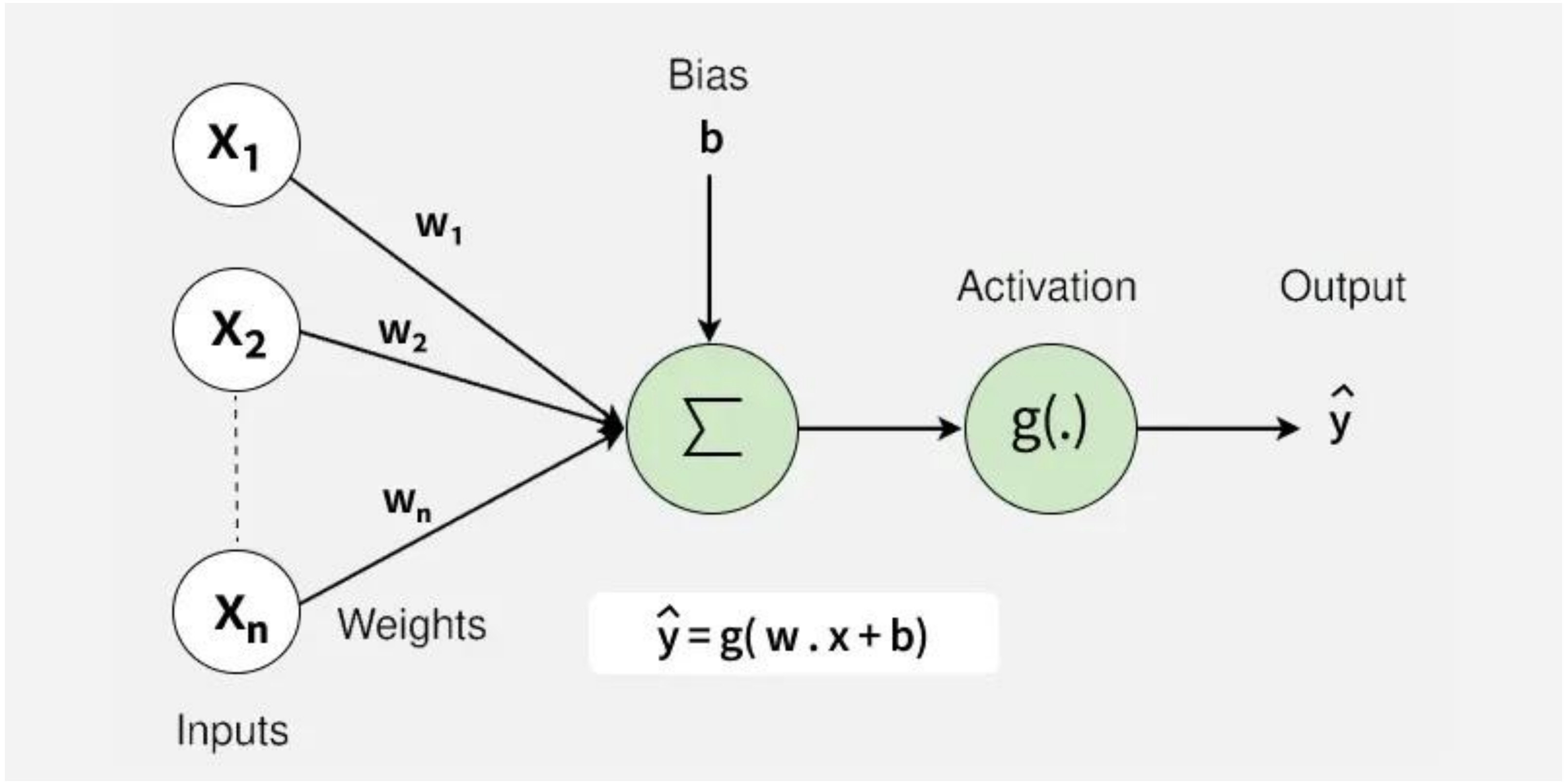
1. **Neurons**: The basic units that receive inputs, each neuron is governed by a threshold and an activation function.

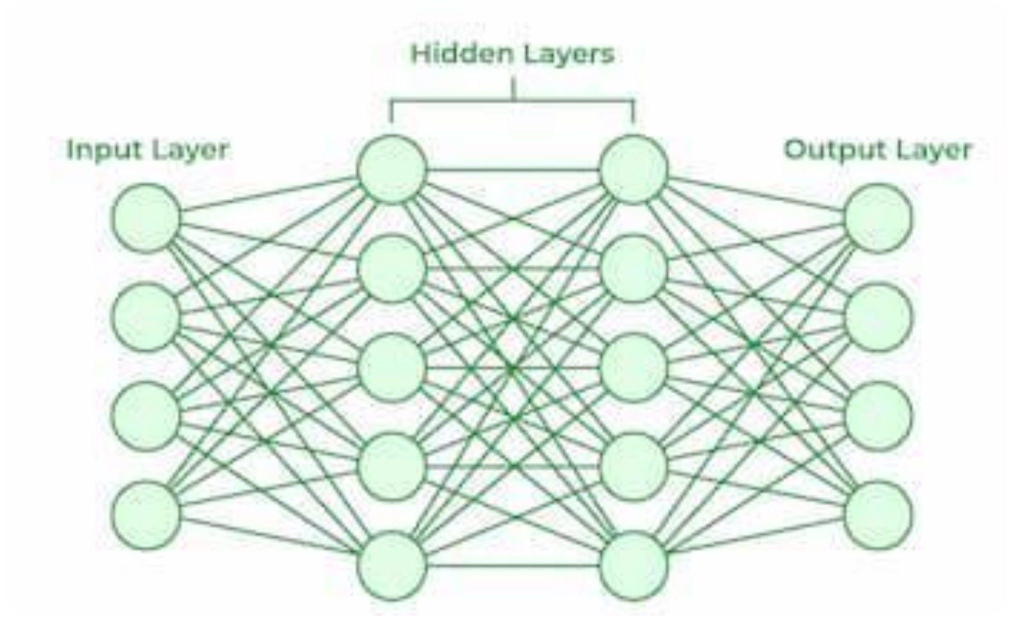
2. **Connections**: Links between neurons that carry information, regulated by weights and biases.

3. **Weights and Biases**: These parameters determine the strength and influence of connections.

4. **Propagation Functions**: Mechanisms that help process and transfer data across layers of neurons.

5. **Learning Rule**: The method that adjusts weights and biases over time to improve accuracy.


Bias
b
$X_1$
$X_2$
$X_n$
$w_1$
$w_2$
$w_n$
Weights
Inputs
$\sum$
Activation
$g(.)$
Output
$\hat{y}$
$\hat{y} = g(w \cdot x + b)$



- **Layers in Neural Network Architecture**
- **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.

- **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.

- **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.

## Types of Neural Network

- Artificial Neural Network
- Convolutional Neural Network
- Recurrent Neural Network
- Generative Adversarial Network

**Implementing Models of Artificial Neural Network**

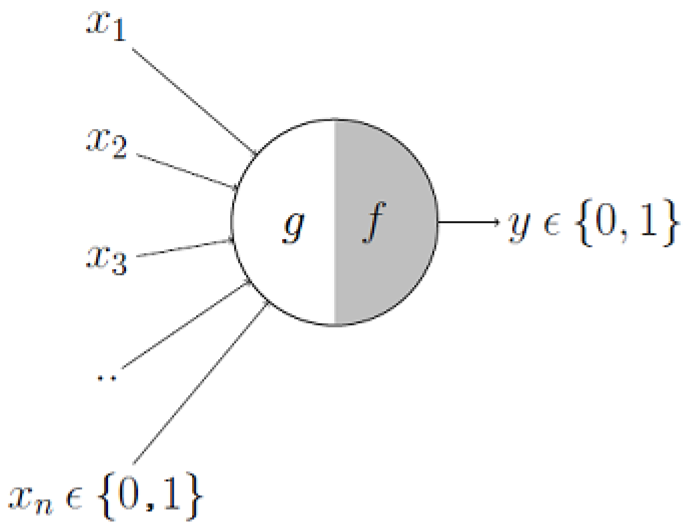
## 1. McCulloch-Pitts Model of Neuron

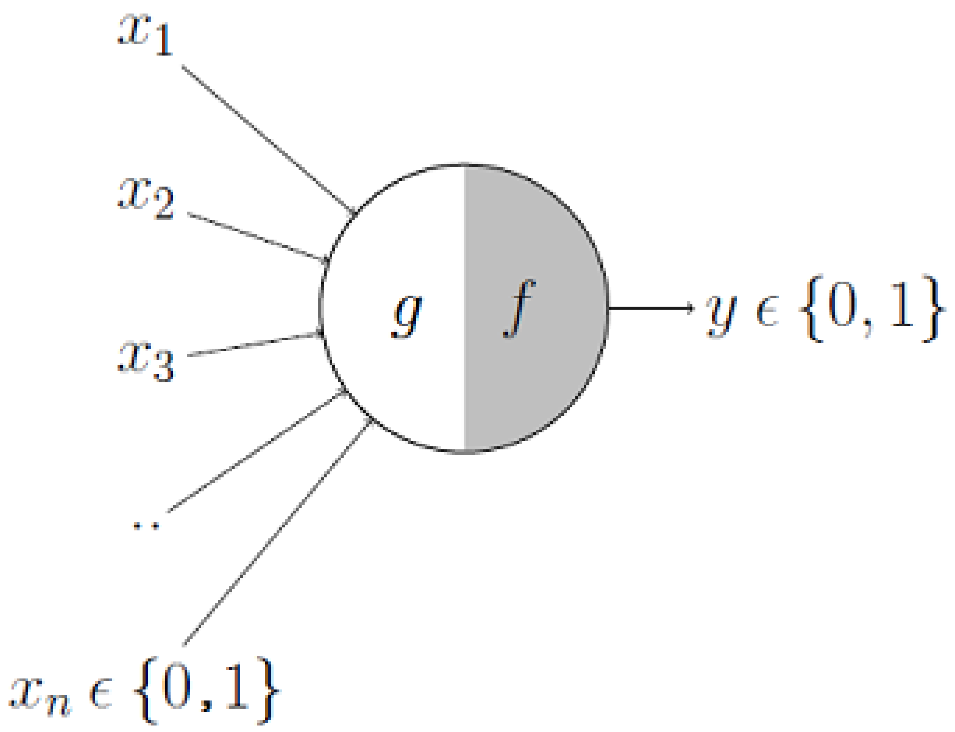
The first computational model of a neuron was proposed by Warren MuCulloch (neuroscientist) and Walter Pitts (logician) in 1943.

It may be divided into 2 parts. The first part, ***g*** takes an input (ahem dendrite ahem), performs an aggregation and based on the aggregated value the second part, ***f*** makes a decision.

It has only two types of inputs — **Excitatory and Inhibitory.** The excitatory inputs have weights of positive magnitude and the inhibitory weights have weights of negative magnitude. The inputs of the McCulloch-Pitts neuron could be either 0 or 1. It has a threshold function as an activation function. So, the output signal $y_{out}$ is 1 if the input $y_{sum}$ is greater than or equal to a given threshold value, else 0

The McCulloch-Pitts Neuron is the first computational model of a neuron. It can be divided into two parts:
**Aggregation:** The neuron aggregates multiple boolean inputs (0 or 1).
**Threshold Decision:** Based on the aggregated value, the neuron makes a decision using a threshold function.


$x_1$
$x_2$
$x_3$
..
$x_n \epsilon \{0,1\}$
$g$
$f$
$y \epsilon \{0,1\}$


$x_1$
$x_2$
$x_3$
..
$x_n \epsilon \{0,1\}$
$g$
$f$
$y \epsilon \{0,1\}$

Lets suppose that I want to predict my own decision, whether to watch a random football game or not on TV. The inputs are all boolean i.e., {0,1} and my output variable is also boolean {0: Will watch it, 1: Won't watch it}.

- So, ***x_1*** could be *isPremierLeagueOn* (I like Premier League more)

- ***x_2*** could be *isItAFriendlyGame* (I tend to care less about the friendlies)

- ***x_3*** could be *isNotHome* (Can't watch it when I'm running errands. Can I?)

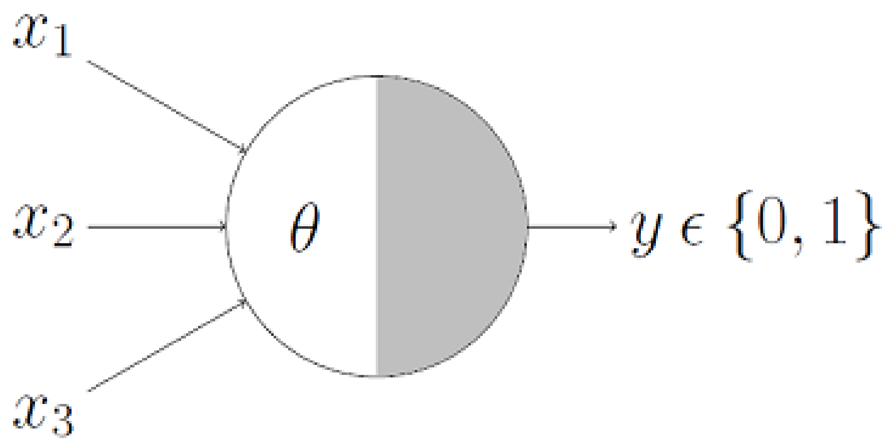
- ***x_4*** could be *isManUnitedPlaying* (I am a big Man United fan. GGMU!) and so on

These inputs can either be *excitatory* or *inhibitory*. Inhibitory inputs are those that have maximum effect on the decision making irrespective of other inputs i.e., if ***x_3*** is 1 (not home) then my output will always be 0 i.e., the neuron will never fire, so ***x_3*** is an inhibitory input. Excitatory inputs are NOT the ones that will make the neuron fire on their own but they might fire it when combined together. Formally, this is what is going on:

$$g(x_1, x_2, x_3, ..., x_n) = g(\mathbf{x}) = \sum_{i=1}^{n} x_i$$

$$\begin{aligned} y = f(g(\mathbf{x})) &= 1 \quad if \quad g(\mathbf{x}) \geq \theta \\ &= 0 \quad if \quad g(\mathbf{x}) < \theta \end{aligned}$$

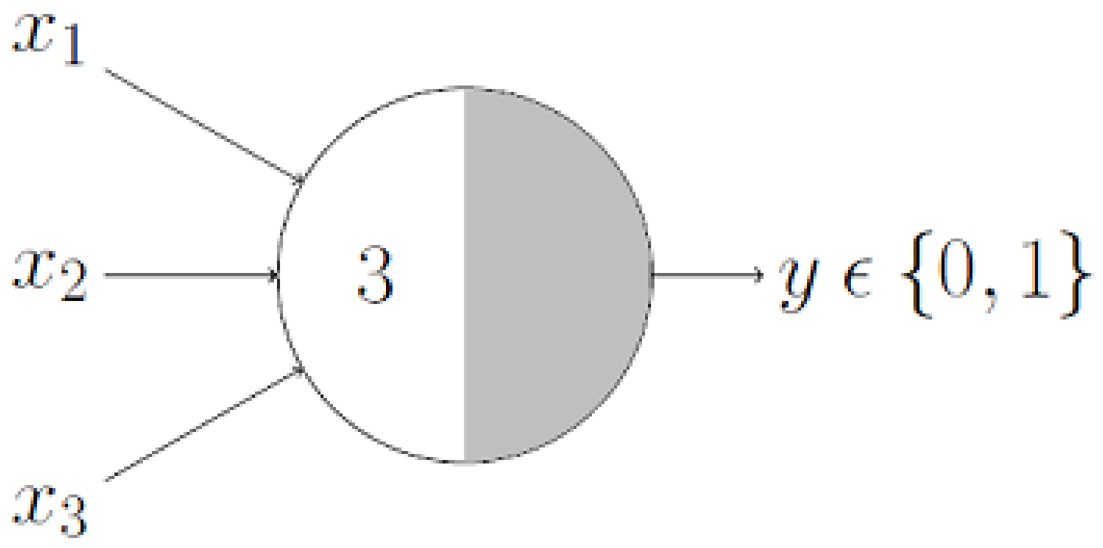
## Boolean Functions Using M-P Neuron



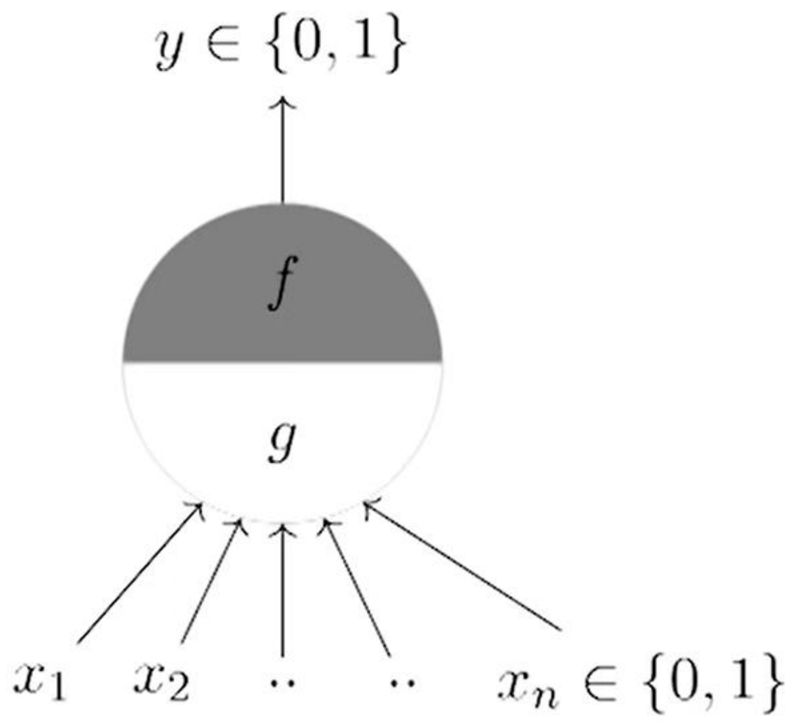
This representation just denotes that, for the boolean inputs ***x_1***, ***x_2*** and ***x_3*** if the ***g*(x)** i.e., **sum ≥ theta**, the neuron will fire otherwise, it won't.

- **AND Function:** Fires when all inputs are ON (( $x1 + x2 + x3 >= 3$ )).
- **OR Function:** Fires when any input is ON (( $x1 + x2 + x3 >= 1$ )).
- **Inhibitory Input Function:** Fires only when specific conditions are met (e.g., ( $x1$ ) AND NOT ( $x2$ )).
- **NOR Function:** Fires when all inputs are OFF.
- **NOT Function:** Inverts the input.

## AND Function



An AND function neuron would only fire when ALL the inputs are ON i.e., ***g*(x)** ≥ 3 here.



- McCulloch (neuroscientist) and Pitts (logician) proposed a highly simplified computational model of the neuron (1943)
- $g$ aggregates the inputs and the function $f$ takes a decision based on this aggregation
- The inputs can be excitatory or inhibitory
- $y = 0$ if any $x_i$ is inhibitory, else

$$g(x_1, x_2, ..., x_n) = g(\mathbf{x}) = \sum_{i=1}^{n} x_i$$

$$\begin{aligned} y = f(g(\mathbf{x})) &= 1 \quad if \quad g(\mathbf{x}) \geq \theta \\ &= 0 \quad if \quad g(\mathbf{x}) < \theta \end{aligned}$$

- $\theta$ is called the thresholding parameter

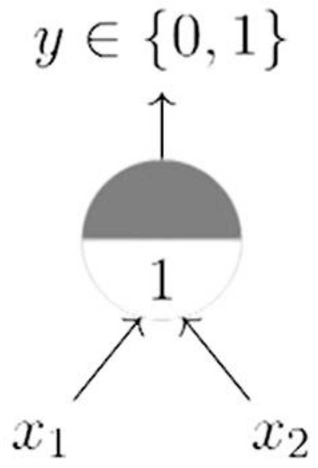
## Geometric Interpretation

The McCulloch-Pitts Neuron can be visualized geometrically by plotting inputs in a multi-dimensional space and drawing a decision boundary:

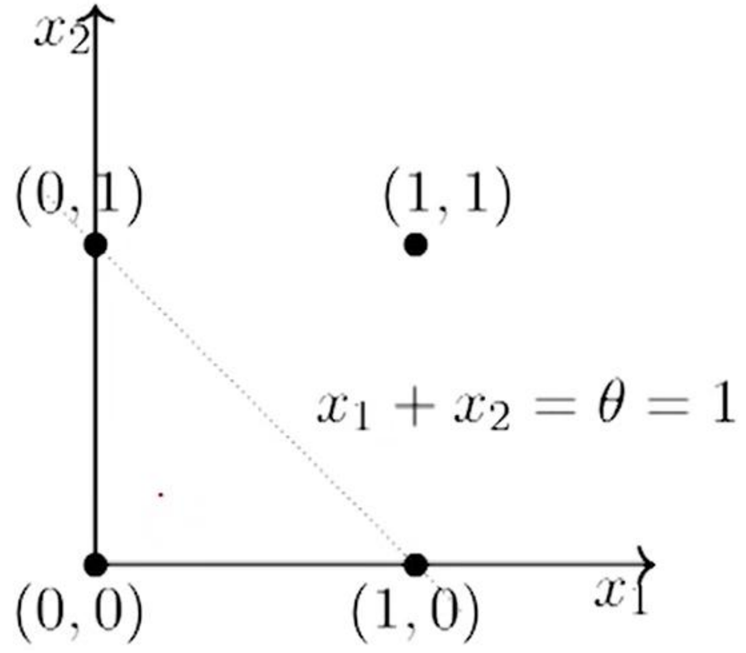
- **OR Function:** In 2D, the decision boundary is a line (( x1 + x2 = 1 )).

- **AND Function:** The decision boundary is a line (( x1 + x2 = 2 )).

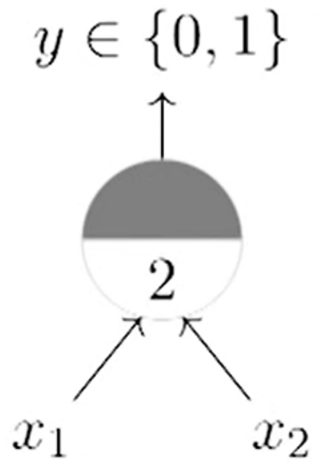
- **Generalization:** The decision boundary becomes a plane in higher dimensions for more inputs.



OR function
$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 1$

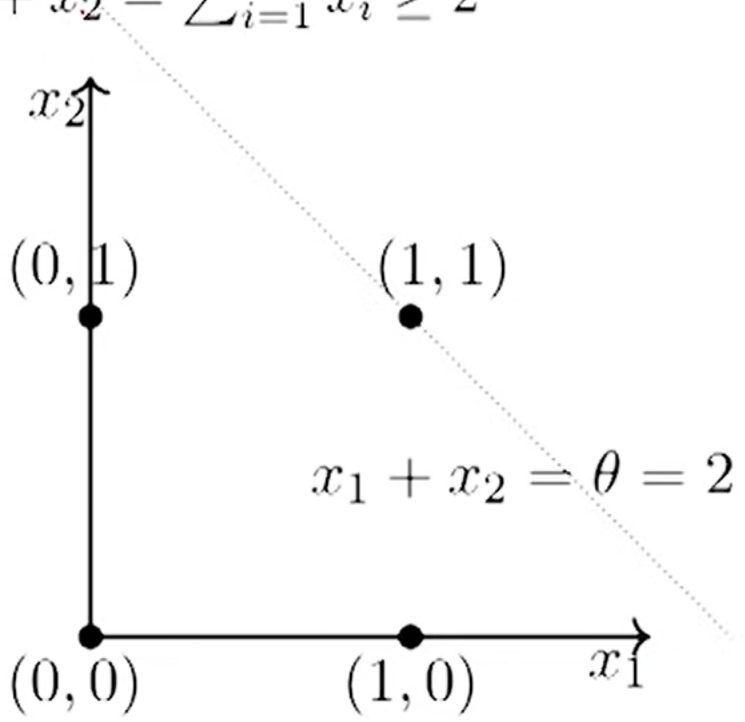


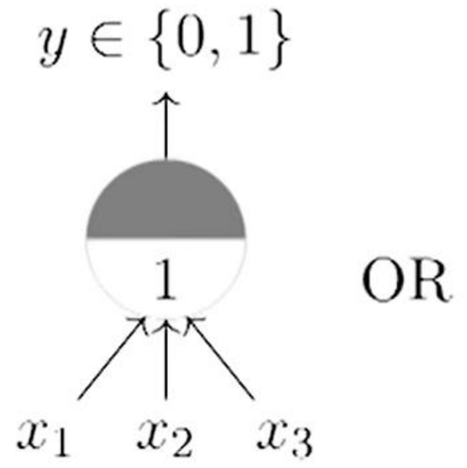
- A single MP neuron splits the input points (4 points for 2 binary inputs) into two halves
- Points lying on or above the line $\sum_{i=1}^{n} x_i - \theta = 0$ and points lying below this line
- In other words, all inputs which produce an output 0 will be on one side ($\sum_{i=1}^{n} x_i < \theta$) of the line and all inputs which produce an output 1 will lie on the other side ($\sum_{i=1}^{n} x_i \geq \theta$) of this line
- Let us convince ourselves about this with a few more examples (if it is not already clear from the math)

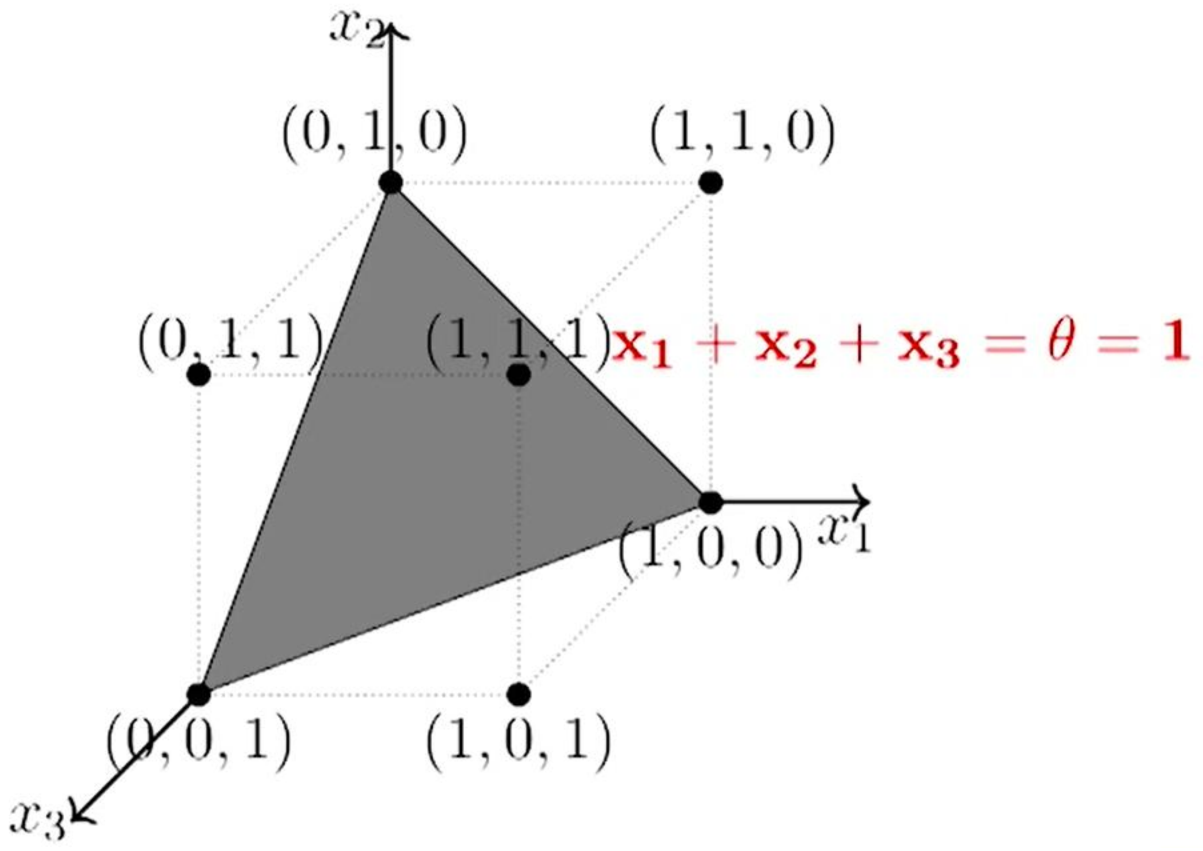


AND function

$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 2$







- What if we have more than 2 inputs?
- Well, instead of a line we will have a plane
- For the OR function, we want a plane such that the point (0,0,0) lies on one side and the remaining 7 points lie on the other side of the plane

## Limitations Of M-P Neuron

- Despite its pioneering role, the McCulloch-Pitts Neuron has limitations:
- Inability to handle non-boolean inputs.
- The requirement to manually set thresholds.
- All inputs are treated equally; no weighting mechanism.
- Cannot handle functions that are not linearly separable like XOR.

## The story ahead ...

- What about non-boolean (say, real) inputs ?
- Do we always need to hand code the threshold ?
- Are all inputs equal ? What if we want to assign more weight (importance) to some inputs ?
- What about functions which are not linearly separable ?

## What is Perceptron?

A perceptron is the smallest element of a neural network. Perceptron is a single-layer neural network linear or a Machine Learning algorithm used for supervised learning of various binary classifiers. It works as an artificial neuron to perform computations by learning elements and processing them for detecting the business intelligence and capabilities of the input data. A perceptron network is a group of simple logical statements that come together to create an array of complex logical statements, known as the neural network.

It is a type of neural network that performs binary classification that maps input features to an output decision, usually classifying data into one of two categories, such as 0 or 1.

Perceptron consists of a single layer of input nodes that are fully connected to a layer of output nodes. It is particularly good at learning **linearly separable patterns**. It utilizes a variation of artificial neurons called **Threshold Logic Units (TLU)**, which were first introduced by McCulloch and Walter Pitts in the 1940s. This foundational model has played a crucial role in the development of more advanced neural networks and machine learning algorithms.

The **Perceptron** is one of the simplest **artificial neural network** architectures, introduced by Frank Rosenblatt in 1957. It is primarily used for **binary classification**.

## Types of Perceptron

**1.Single-Layer Perceptron** is a type of perceptron is limited to learning linearly separable patterns. It is effective for tasks where the data can be divided into distinct categories through a straight line. While powerful in its simplicity, it struggles with more complex problems where the relationship between inputs and outputs is non-linear.

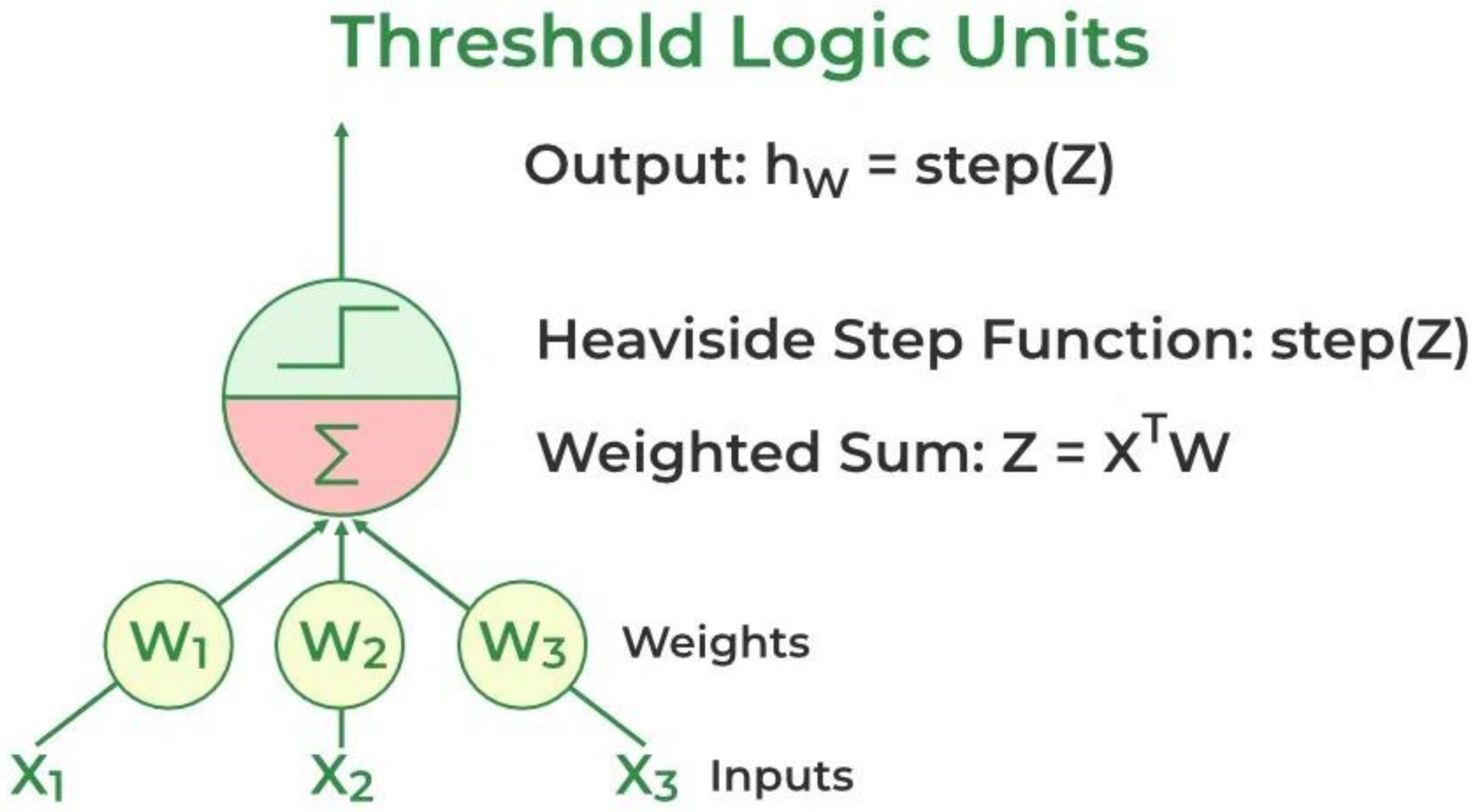
**2.Multi-Layer Perceptron** possess enhanced processing capabilities as they consist of two or more layers, adept at handling more complex patterns and relationships within the data.

## Basic Components of Perceptron

A Perceptron is composed of key components that work together to process information and make predictions.

- **Input Features:** The perceptron takes multiple input features, each representing a characteristic of the input data.
- **Weights:** Each input feature is assigned a weight that determines its influence on the output. These weights are adjusted during training to find the optimal values.
- **Summation Function:** The perceptron calculates the weighted sum of its inputs, combining them with their respective weights.
- **Activation Function:** The weighted sum is passed through the **Heaviside step function**, comparing it to a threshold to produce a binary output (0 or 1).
- **Output:** The final output is determined by the activation function, often used for **binary classification** tasks.
- **Bias:** The bias term helps the perceptron make adjustments independent of the input, improving its flexibility in learning.
- **Learning Algorithm:** The perceptron adjusts its weights and bias using a learning algorithm, such as the Perceptron Learning Rule, to minimize prediction errors.

## How does Perceptron work?



**How does Perceptron work?**
A weight is assigned to each input node of a perceptron, indicating the importance of that input in determining the output. The Perceptron’s output is calculated as a weighted sum of the inputs, which is then passed through an activation function to decide whether the Perceptron will fire.

The weighted sum is computed as:

z=w1x1+w2x2+…+wnxn=XTW$z=w1x1+w2x2+…+wnxn=XTW$

The step function compares this weighted sum to a threshold. If the input is larger than the threshold value, the output is 1; otherwise, it's 0. This is the most common activation function used in Perceptrons are represented by the Heaviside step function:

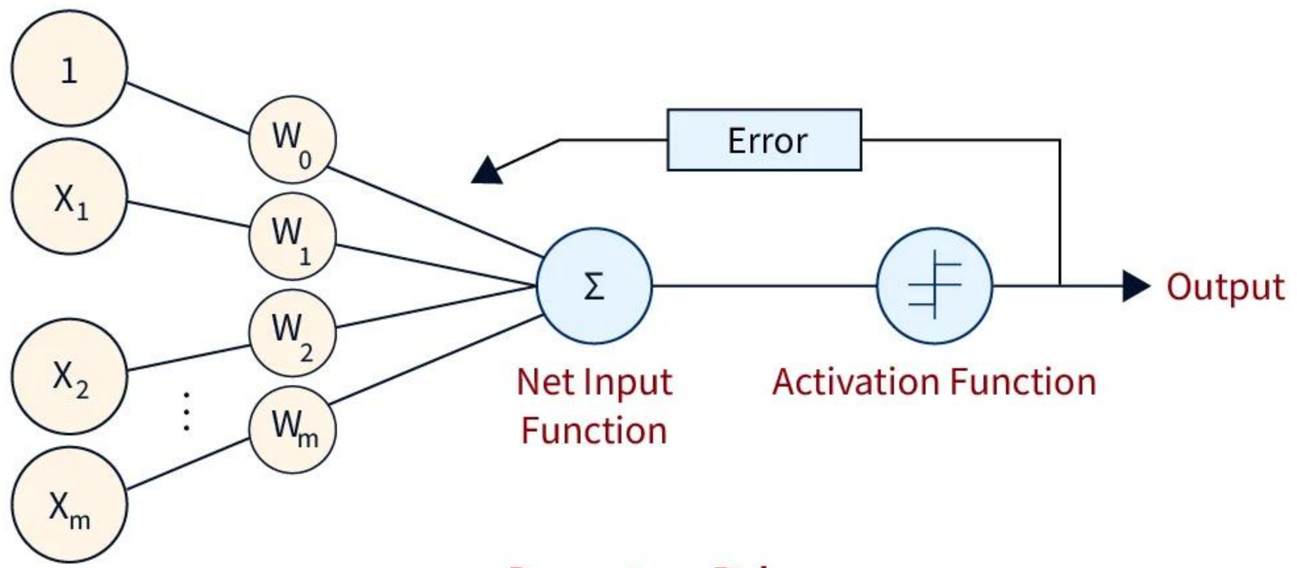
h(z)={0if z<Threshold1if z≥Threshold$h(z)$={01if $z$<Thresholdif $z$≥Threshold

A perceptron consists of a single layer of Threshold Logic Units (TLU), with each TLU fully connected to all input nodes.

**What is the Perceptron Learning Algorithm?**

**There are four significant steps in a perceptron learning algorithm:**

1. First, multiply all input values with corresponding weight values and then add them to determine the weighted sum. Mathematically, we can calculate the weighted sum as follows: $\sum \mathrm{wi}*\mathrm{xi}=\mathrm{x1}*\mathrm{w1}+\mathrm{x2}*\mathrm{w2}+\ldots+\mathrm{wn}*\mathrm{xn}\sum wi*xi=x1*w1+x2*w2+\ldots+wn*xn$. Add another essential term called bias 'b' to the weighted sum to improve the model performance. $\sum \mathrm{wi}*\mathrm{xi}+\mathrm{b}\sum wi*xi+b$.
2. Next, an activation function is applied to this weighed sum, producing a binary or a continuous-valued output. $\mathrm{Y}=\mathrm{f}(\sum \mathrm{wi}*\mathrm{xi}+\mathrm{b})Y=f(\sum wi*xi+b)$
3. Next, the difference between this output and the actual target value is computed to get the error term, E, generally in terms of mean squared error. The steps up to this form the forward propagation part of the algorithm. $\mathrm{E}=(\mathrm{Y}-\mathrm{Yactual})2E=(Y-Yactual)2$
4. We optimize this error (loss function) using an optimization algorithm. Generally, some form of gradient descent algorithm is used to find the optimal values of the hyperparameters like learning rate, weight, Bias, etc. This step forms the backward propagation part of the algorithm.



**Perceptron Rule**

## Feedforward Neural Networks (FFN)

- Consist of input, hidden, and output layers
- Each layer applies transformations and passes data forward
- Used for image classification, NLP, and more

## Loss Functions and Output

- Loss measures prediction error: MSE, Cross-Entropy
- Network learns by minimizing loss during training

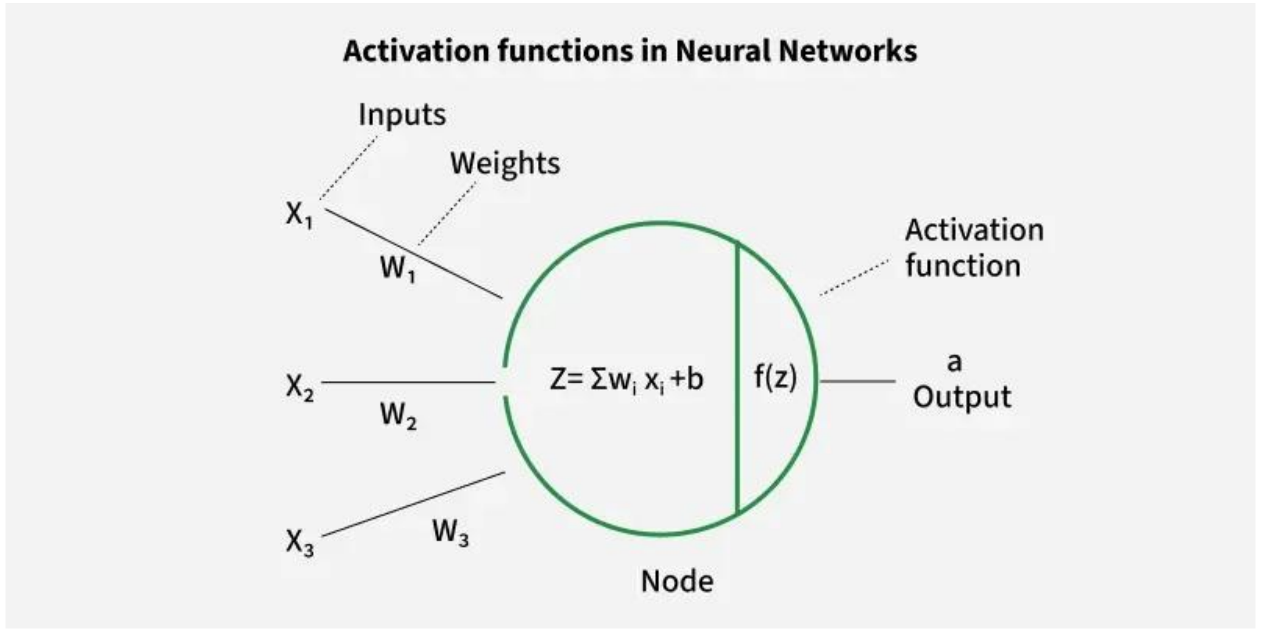
## Limitations of Perceptron

- Cannot solve non-linearly separable problems (e.g., XOR)
- Requires multilayer networks for complex patterns

## Activation functions in Neural Networks

Activation Function is a mathematical function applied to the output of a neuron. It introduces non-linearity into the model, allowing the network to learn and represent complex patterns in the data. Without this non-linearity feature a neural network would behave like a linear regression model no matter how many layers it has.

Activation function decides whether a neuron should be activated by calculating the weighted sum of inputs and adding a bias term. This helps the model make complex decisions and predictions by introducing non-linearities to the output of each neuron.



## Introducing Non-Linearity in Neural Network

Non-linearity means that the relationship between input and output is not a straight line. In simple terms the output does not change proportionally with the input. A common choice is the ReLU function defined as σ(x)=max(0,x)$\sigma(x)=\max(0,x)$.

Imagine you want to classify apples and bananas based on their shape and color.

•If we use a linear function it can only separate them using a straight line.

•But real-world data is often more complex like overlapping colors, different lighting, etc.

•By adding a non-linear activation function like ReLU, Sigmoid or Tanh the network can create curved decision boundaries to separate them correctly.

## Types of Activation Functions in Deep Learning

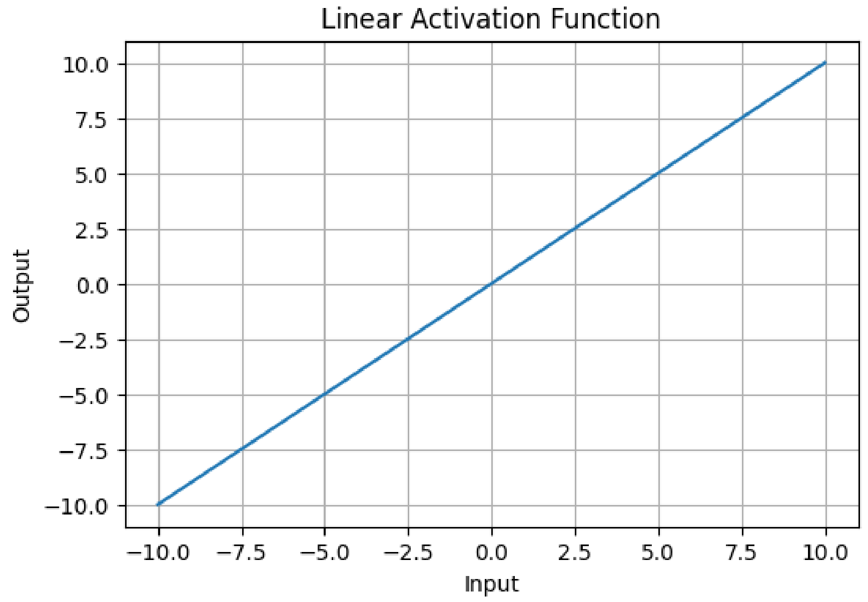
### 1. Linear Activation Function

Linear Activation Function resembles straight line define by y=x. No matter how many layers the neural network contains if they all use linear activation functions the output is a linear combination of the input.

•The range of the output spans from (−∞ to +∞)(−∞ to +∞).

•Linear activation function is used at just one place i.e. output layer.

•Using linear activation across all layers makes the network's ability to learn complex patterns limited.



## 2. Non-Linear Activation Functions

### 1. Sigmoid Function

**Sigmoid Activation Function** is characterized by 'S' shape. It is mathematically defined asA=11+e−x*A*=1+*e*−*x*1. This formula ensures a smooth and continuous output that is essential for gradient-based optimization methods.

•It allows neural networks to handle and model complex patterns that linear equations cannot.

•The output ranges between 0 and 1, hence useful for binary classification.

•The function exhibits a steep gradient when x values are between -2 and 2. This sensitivity means that small changes in input x can cause significant changes in output y which is critical during the training process.

## 2. Tanh Activation Function

**Tanh function** (hyperbolic tangent function) is a shifted version of the sigmoid, allowing it to stretch across the y-axis. It is defined as:

f(x)=tanh(x)=21+e−2x−1.*f*(*x*)=tanh(*x*)=1+*e*−2*x*2−1.

Alternatively, it can be expressed using the sigmoid function:

tanh(x)=2×sigmoid(2x)−1tanh(*x*)=2×sigmoid(2*x*)−1

•**Value Range**: Outputs values from -1 to +1.

•**Non-linear**: Enables modeling of complex data patterns.

•**Use in Hidden Layers**: Commonly used in hidden layers due to its zero-centered output, facilitating easier learning for subsequent layers.

## 3. ReLU (Rectified Linear Unit) Function

**ReLU activation** is defined by $A(x)=\max(0,x)A(x)=\max(0,x)$, this means that if the input x is positive, ReLU returns x, if the input is negative, it returns 0.

•**Value Range**: $[0,\infty)[0,\infty)$, meaning the function only outputs non-negative values.

•**Nature**: It is a non-linear activation function, allowing neural networks to learn complex patterns and making backpropagation more efficient.

•**Advantage over other Activation:** ReLU is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation.

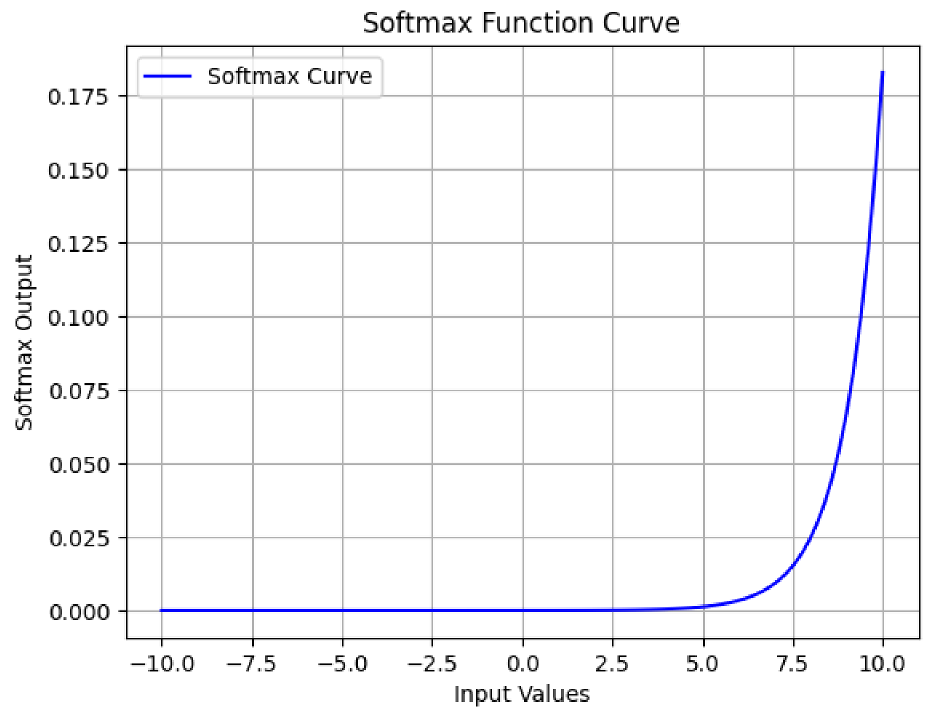
## Exponential Linear Units

### 1. Softmax Function

**Softmax function** is designed to handle multi-class classification problems. It transforms raw output scores from a neural network into probabilities. It works by squashing the output values of each class into the range of 0 to 1 while ensuring that the sum of all probabilities equals 1.

•Softmax is a non-linear activation function.

•The Softmax function ensures that each class is assigned a probability, helping to identify which class the input belongs to.


Softmax Function Curve
Softmax Curve
Softmax Output
Input Values