# Perceptron

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

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.

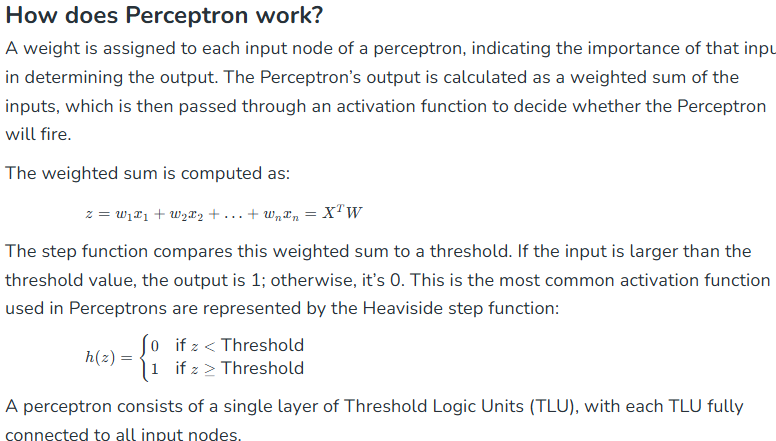
**Types of Perceptron**

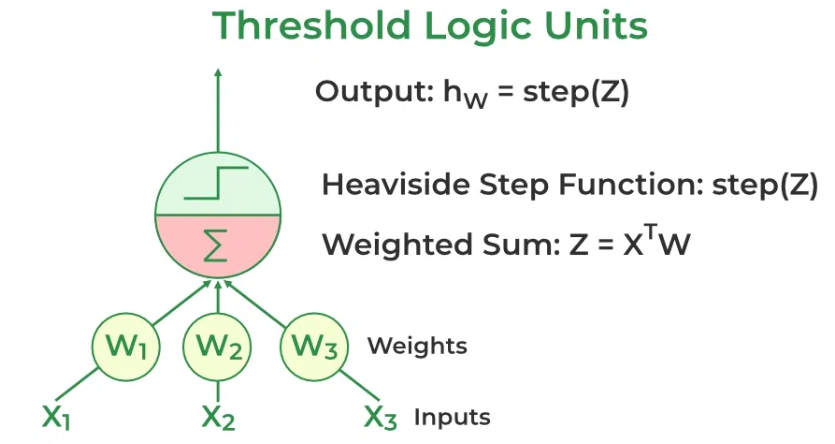
1. [**Single-Layer Perceptron**](https://www.geeksforgeeks.org/single-layer-perceptron-in-tensorflow/) is a type of perceptron is limited to learning linearly separable patterns. It is effective for tasks where the data can be divided into distinct categories through a straight line. While powerful in its simplicity, it struggles with more complex problems where the relationship between inputs and outputs is non-linear.
2. [**Multi-Layer Perceptron**](https://www.geeksforgeeks.org/multi-layer-perceptron-learning-in-tensorflow/) possess enhanced processing capabilities as they consist of two or more layers, adept at handling more complex patterns and relationships within the data.

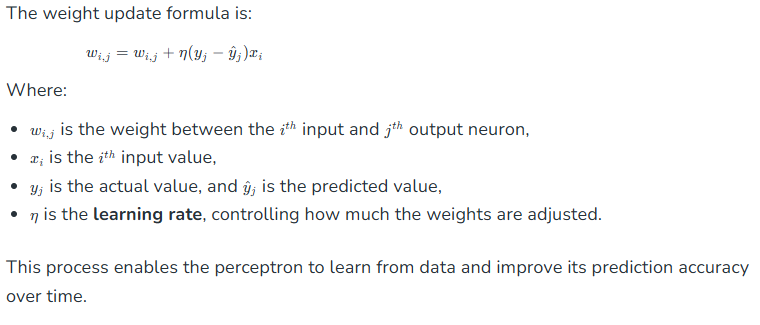
**Basic Components of Perceptron**

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**](https://www.geeksforgeeks.org/the-role-of-weights-and-bias-in-neural-networks/)**:** 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**](https://www.geeksforgeeks.org/activation-functions-neural-networks/)**:** 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**](https://www.geeksforgeeks.org/effect-of-bias-in-neural-network/)**:** 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.







**Example: Perceptron in Action**

Let’s take a simple example of classifying whether a given fruit is an apple or not based on two inputs: its weight (in grams) and its color (on a scale of 0 to 1, where 1 means red). The perceptron receives these inputs, multiplies them by their weights, adds a bias, and applies the activation function to decide whether the fruit is an apple or not.

* Input 1 (Weight): 150 grams
* Input 2 (Color): 0.9 (since the fruit is mostly red)
* Weights: [0.5, 1.0]
* Bias: 1.5

The perceptron’s weighted sum would be:

(150∗0.5)+(0.9∗1.0)+1.5=76.4(150∗0.5)+(0.9∗1.0)+1.5=76.4

*Let’s assume the activation function uses a threshold of 75. Since 76.4 > 75, the perceptron classifies the fruit as an apple (output = 1).*

## Hidden Layers

Hidden layers are intermediate layers **between input and output layers** in a neural network. Each layer consists of neurons that **extract patterns and features** from data before passing them forward.

**Why Do Hidden Layers Matter?**

* **Enable Non-Linearity**: A **Single-Layer Perceptron (SLP)** can only solve **linearly separable** problems. Hidden layers **introduce non-linearity**, allowing the network to learn complex patterns.
* **Feature Extraction**: Each hidden layer transforms inputs into **higher-level representations**.
* **Deep Learning Power**: More layers help in **deep learning** to extract deeper features.

**How Do Hidden Layers Improve a Single-Layer Model?**

A **single-layer perceptron (SLP) is limited to linear decision boundaries**. Adding hidden layers:

1. **Allows the model to learn non-linear relationships**.
2. **Improves generalization**, reducing overfitting in complex data.
3. **Enables deep feature learning**, making it more powerful for tasks like image recognition and NLP.

**How to Determine the Number of Layers & Nodes?**

🔹 **Empirical approach** (trial and tuning):

* **Simple tasks** → **1-2 hidden layers** (e.g., structured/tabular data).
* **Complex tasks** → **3+ hidden layers** (e.g., image recognition, NLP).

🔹 **Guidelines for Hidden Nodes:**

* Start with **2/3 of input features** and fine-tune.
* Try **power of two** (e.g., 32, 64, 128 neurons per layer).
* Use **grid search or hyperparameter tuning**.

**Rule of Thumb**

* **Too few neurons** → Underfitting (can’t capture complexity).
* **Too many neurons** → Overfitting (memorizes instead of generalizing).