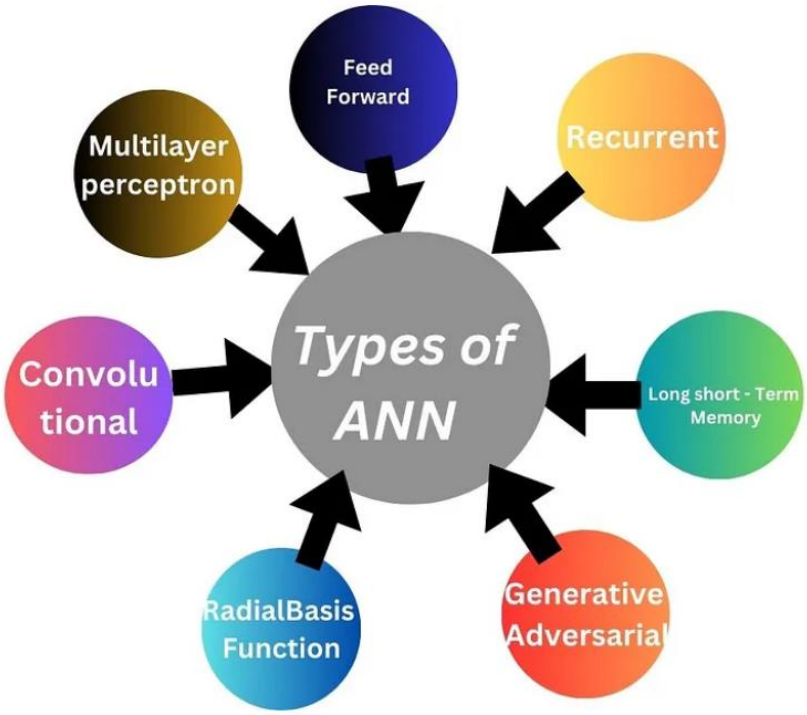
Type of artificial neural network



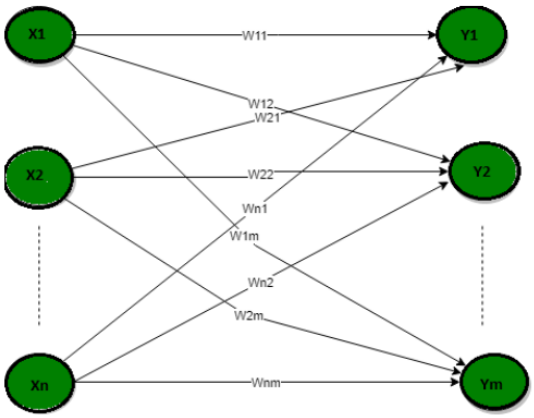
Neuron connection architecture

- **Neuron arrangement:** How processing elements (neurons) are connected determines the network's behavior.
- **Layer structure:** All ANNs share at least two layers:
- **Input layer**: Receives initial data.
- **Output layer:** Generates the network's response.
- **Hidden layers:** Optional layers between input and output. Not directly accessible, acting as a "black box."
- **Computational power:** Increasing hidden layers and neurons boosts processing power.
- **Training complexity:** More layers and neurons add complexity to the training process.

There exist five basic types of neuron connection architecture :

1. Single-layer feed-forward network
2. Multilayer feed-forward network
3. Single node with its own feedback
4. Single-layer recurrent network
5. Multilayer recurrent network

1. Single-layer feed-forward network



Structure: Only two layers - input and output.

Input layer: Passive reception of data, no computation.

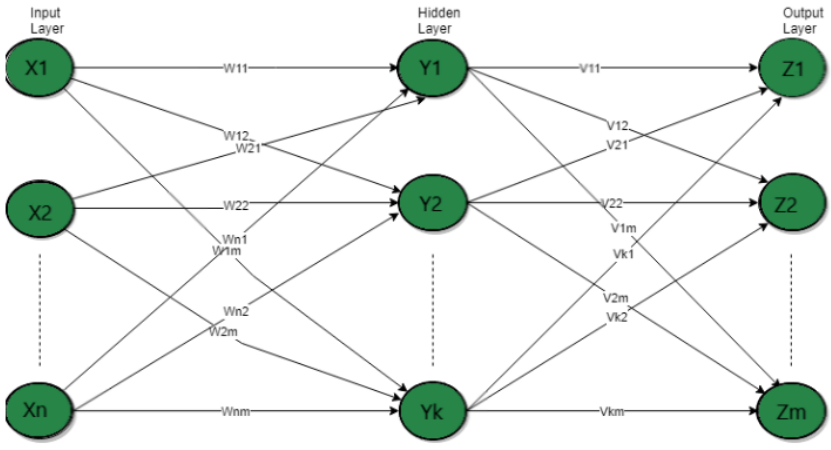
Output layer:

- Weighted summation of inputs from each input node.
- Collective activation of neurons determines the network's output.

Limitations:

- Cannot learn complex relationships or non-linearly separable data.
- Less powerful than networks with hidden layers.

2. Multilayer feed-forward network



- ❖ Internal layer not directly interacting with external input/output.
- ❖ Increases network's computational power for complex tasks.

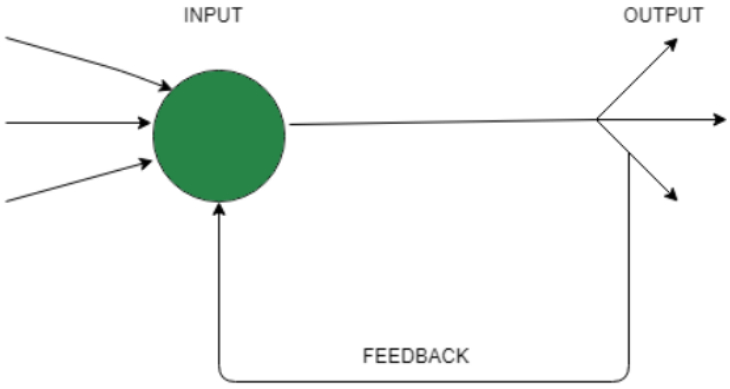
Benefits:

- ❖ Enables learning non-linear relationships and complex data patterns.
- ❖ Creates a "feed-forward" network due to information flow:

- Input function processes data.
- Intermediate computations through hidden layers refine the data.
- Final output (Z) determined by these computations.

Key characteristic: No feedback loops - outputs don't directly influence network inputs.

3. Single node with its own feedback



Single Node with own Feedback

- Outputs can be fed back as inputs to the same layer or previous layers.
- Introduces internal memory allowing the network to remember past information.

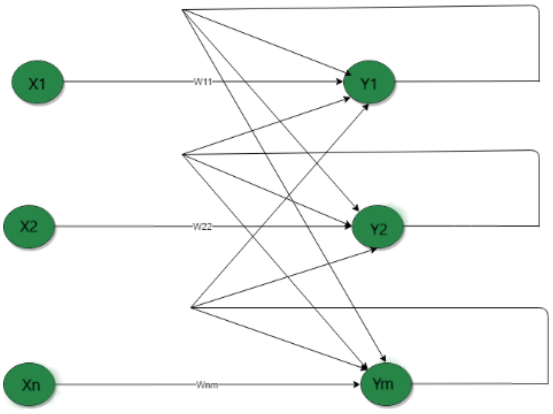
Recurrent networks:

Specific type of feedback network with closed loops, where outputs loop back to influence future inputs.

Example:

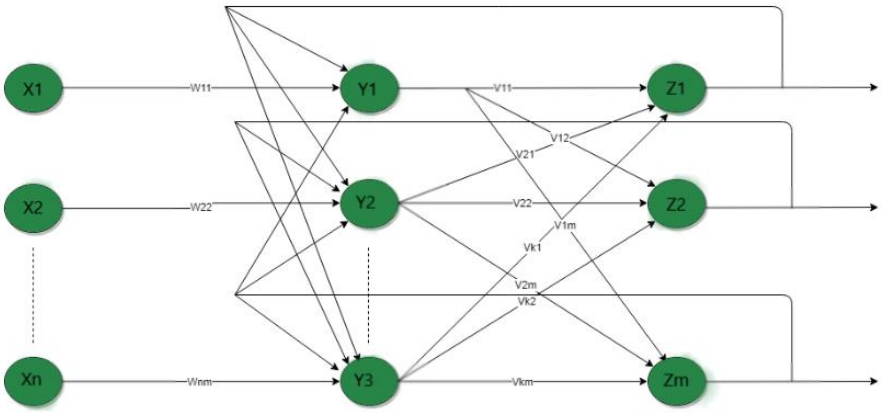
The provided figure shows a single recurrent network with one neuron and a feedback loop to itself. This basic structure can be expanded to involve multiple neurons and complex feedback configurations.

4. Single-layer recurrent network



- **Structure:** Single hidden layer with feedback loops.
- **Information flow:** Directed, forming a sequence like a graph.
- **Key feature:** Internal memory through feedback - outputs influence future inputs.
- **Benefit:** Captures temporal dynamics in sequential data (e.g., time series, sentences).
- **Difference from feedforward networks:** RNNs remember past information, impacting future outputs.

5. Multilayer recurrent network



- **Structure:** Multiple hidden layers with complex feedback connections.
- **Processing:** Applies the same task to each element in a sequence.
- **Memory:** Hidden state captures information about the sequence throughout processing.
- **Output:** Dependent on previous computations, not requiring new inputs at every step.
- **Key point:** Internal memory (hidden state) allows RNNs to learn from and utilize context while processing sequential data.

Perceptron

- A **Perceptron** is an **Artificial Neuron**
- It is the simplest possible **Neural Network**
- **Neural Networks** are the building blocks of **Machine Learning**.

Components of Perceptron:

- Input Layer: Receives data (one or more neurons).
- Weights: Strengths of connections between input and output neurons.
- Bias: Influences overall output regardless of input.
- Activation Function: Converts weighted sum to final output (e.g., 0/1).
- Output: Single binary value (class/category).
- Training Algorithm: Adjusts weights and biases to minimize error.

Types of Perceptron:

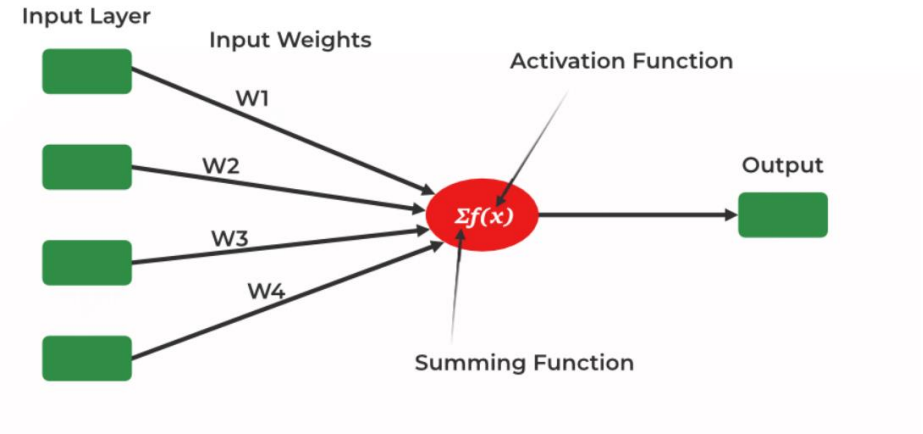
1) **Single layer Perceptron**: Simplest form, limited to linearly separable data.
2) **Multilayer Perceptron:** More complex, capable of learning intricate patterns.

Single Layer Perceptron:

- One of the first neural networks (1958, Frank Rosenblatt).
- Also known as "artificial neural network" for binary classification tasks.
- Computes simple logic gates like AND, OR, and NOR.

Main Functionality:

- ✓ Input: Receives data from the input layer.
- ✓ Weighted Sum: Applies weights to each input and sums them.
- ✓ Nonlinear Activation: Passes the sum through a nonlinear function (e.g., Heaviside step function) to determine the output.

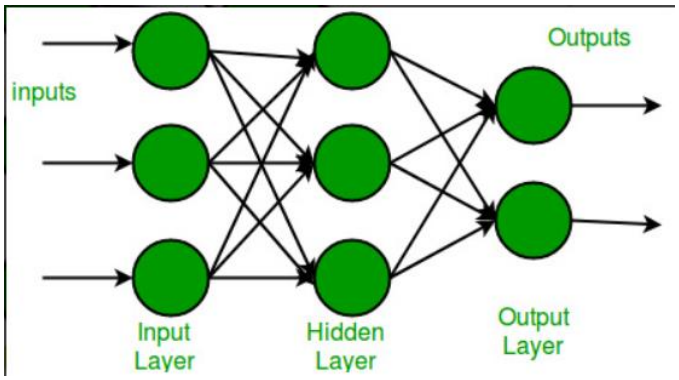


Multi-Layer Perceptron (MLP): A Powerful Neural Network Architecture

- ❖ An artificial neural network with multiple layers of interconnected neurons.
- ❖ Also known as MLP.
- ❖ Can transform any input dimension to the desired output dimension.

Key features:

- ✓ Multiple layers: Composed of an input layer, one or more hidden layers, and an output layer.
- ✓ Fully connected: Neurons in each layer are connected to all neurons in the next layer.
- ✓ Activation functions: Each neuron applies an activation function (e.g., sigmoid) to its weighted sum of inputs.



Perceptron Example

Imagine a perceptron (in your brain).

The perceptron tries to decide if you should go to a concert.

Is the artist good? Is the weather good?

What weights should these facts have?

Criteria	Input	Weight
Artists is Good	**x1** = 0 or 1	**w1** = 0.7
Weather is Good	**x2** = 0 or 1	**w2** = 0.6
Friend will Come	**x3** = 0 or 1	**w3** = 0.5
Food is Served	**x4** = 0 or 1	**w4** = 0.3
Alcohol is Served	**x5** = 0 or 1	**w5** = 0.4

The Perceptron Algorithm

Frank Rosenblatt suggested this algorithm:

1. Set a threshold value
2. Multiply all inputs with its weights
3. Sum all the results
4. Activate the output

1. Set a threshold value:

- Threshold = 1.5

2. Multiply all inputs with its weights:

- x1 * w1 = 1 * 0.7 = 0.7
- x2 * w2 = 0 * 0.6 = 0
- x3 * w3 = 1 * 0.5 = 0.5
- x4 * w4 = 0 * 0.3 = 0
- x5 * w5 = 1 * 0.4 = 0.4

3. Sum all the results:

- 0.7 + 0 + 0.5 + 0 + 0.4 = 1.6 (The Weighted Sum)

4. Activate the Output:

- Return true if the sum > 1.5 ("Yes I will go to the Concert")