Experiment 01

Aim: Implement Mc-Culloch Pitts model for binary logic functions.

Apparatus: Laptop / Desktop with Internet Connection, Google Colab

Theory:

The McCulloch-Pitts (MCP) model, proposed in 1943 by Warren McCulloch and Walter Pitts, is a simplified mathematical model of a biological neuron. It's considered the first neural network model and forms the foundation for artificial neural networks. The MCP neuron receives binary inputs (0 or 1), applies weighted connections, and produces a binary output (0 or 1) based on a threshold.

Key Characteristics of the McCulloch-Pitts Neuron:

1) Binary Inputs and Outputs:

Each input to the neuron, as well as its output, is either 0 or 1.

2) Weighted Connections:

Each input is associated with a weight, representing the strength of the connection.

3) Excitatory and Inhibitory Connections:

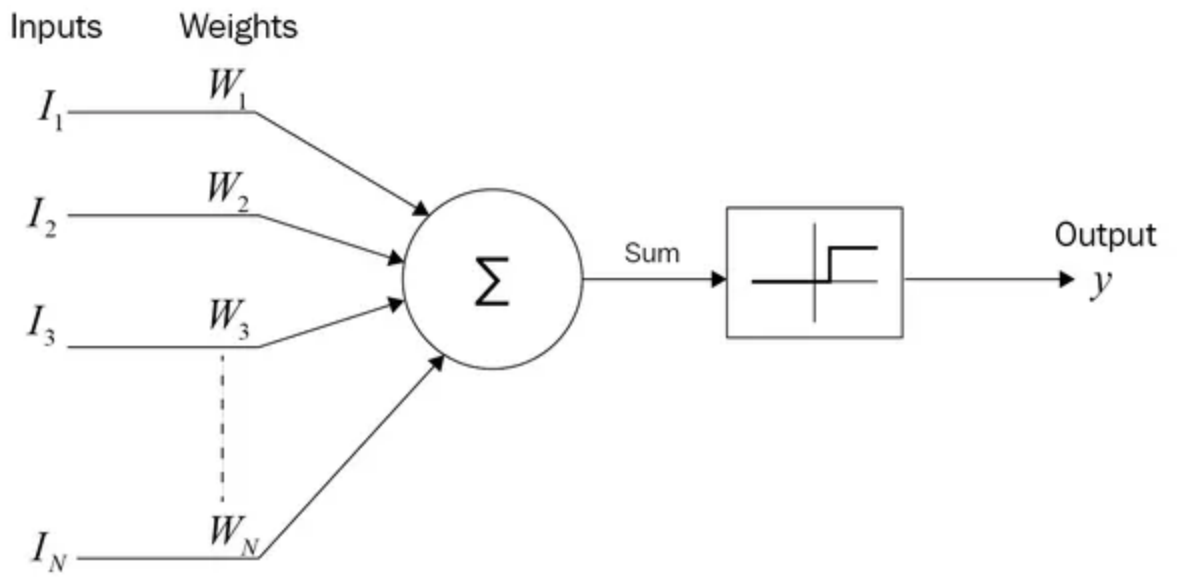
Positive weights represent excitatory connections (increase the likelihood of firing), while negative weights represent inhibitory connections (decrease the likelihood of firing).

4) Threshold Activation:

The neuron fires (outputs 1) if the sum of weighted inputs exceeds a predefined threshold; otherwise, it remains inactive (outputs 0).

5) Logical Operations:

By adjusting the weights and thresholds, MCP neurons can be configured to implement various logical operations like AND, OR, and NOT.



An AND gate is a fundamental digital logic gate that outputs a high signal (1) only when all of its inputs are also high (1). If any input is low (0), the output will be low (0). It essentially performs a logical conjunction, similar to the "and" operator in Boolean algebra.

Key Characteristics:

- **Inputs:** Typically has two or more inputs.
- **Output:** A single output.
- **Logic:** The output is 1 only when all inputs are 1.

Truth Table:

Input A	**Input B**	**Output (A AND B)**
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate

An OR gate is a basic digital logic gate that outputs a high signal (1) if any of its inputs are high (1). It performs a logical disjunction, similar to the “or” operator in Boolean algebra.

Key Characteristics:

- Inputs: Typically has two or more inputs.
- Output: A single output.
- Logic: The output is 1 if any input is 1.

Truth Table:

Input A	Input B	Output (A OR B)
0	0	0
0	1	1
1	0	1
1	1	1

NOR Gate

A NOR gate is a universal gate that combines an OR gate followed by a NOT gate. It outputs a high signal (1) only when all inputs are low (0). It performs the negation of logical disjunction.

Key Characteristics:

- Inputs: Typically has two or more inputs.
- Output: A single output.
- Logic: The output is 1 only when all inputs are 0; otherwise, it's 0.

Truth Table:

Input A	Input B	Output (A NOR B)
0	0	1
0	1	0
1	0	0
1	1	0

Implementation Code:

```
class McCullochPittsNeuron:
    def __init__(self, excitatory=None, inhibitory=None, threshold=1):
        """
        excitatory: list of indices for excitatory inputs
        inhibitory: list of indices for inhibitory inputs
        threshold: firing threshold for sum of excitatory inputs
        """
        self.excitatory = excitatory or []
        self.inhibitory = inhibitory or []
        self.threshold = threshold
    def activate(self, inputs):
        """
        inputs: list or tuple of binary values (0 or 1)
        returns: 1 if neuron fires, else 0
        """
```

```
        # If any inhibitory input is active, neuron does not fire
        for i in self.inhibitory:
            if inputs[i] == 1:
                return 0
        # Sum excitatory inputs
        s = sum(inputs[i] for i in self.excitatory)
        return 1 if s >= self.threshold else 0
# Logic gates using McCulloch-Pitts neurons
neurons = {
    "AND": McCullochPittsNeuron(excitatory=[0, 1], inhibitory=[], threshold=2),
    "OR":  McCullochPittsNeuron(excitatory=[0, 1], inhibitory=[], threshold=1),
    "NOR": McCullochPittsNeuron(excitatory=[], inhibitory=[0, 1], threshold=1),
}
# Test truth table
inputs = [(0, 0), (0, 1), (1, 0), (1, 1)]
for name, neuron in neurons.items():
    print(f"--- {name} gate ---")
    for inp in inputs:
        print(f"{inp} -> {neuron.activate(inp)}")
    print()
```

Output:

```
--- AND gate ---
(0, 0) -> 0
(0, 1) -> 0
(1, 0) -> 0
(1, 1) -> 1

--- OR gate ---
(0, 0) -> 0
(0, 1) -> 1
(1, 0) -> 1
(1, 1) -> 1

--- NOR gate ---
(0, 0) -> 0
(0, 1) -> 0
(1, 0) -> 0
(1, 1) -> 0
```

Conclusion:

The McCulloch-Pitts neuron model was successfully implemented to simulate basic binary logic functions such as AND, OR, and NOR gates.