Deep Learning

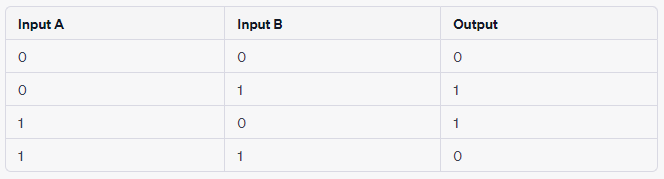
Experiment 02

IMPLEMENTING EXOR

AIM: Implementing a basic XOR neural network using TensorFlow.

The XOR problem can be defined as follows:

Given two binary input variables (A and B), the task is to output 1 (positive class) if the inputs are different (one is 0 and the other is 1), and output 0 (negative class) if the inputs are the same (both 0 or both 1). The truth table for the XOR problem is the same as the one I provided earlier:



Implementing a simple XOR neural network using the popular deep learning library TensorFlow. For this example, we will use a multi-layer perceptron (MLP) with one hidden layer.

**Code:**

Step 1: Import the necessary libraries

import tensorflow as tf

import numpy as np

Step 2: Prepare the input data and output labels for the XOR problem

# Input data for XOR problem

x\_train = np.array([[0, 0],

[0, 1],

[1, 0],

[1, 1]])

# Output labels for XOR problem

y\_train = np.array([[0], [1], [1], [0]])

Step 3: Define the neural network model

model = tf.keras.Sequential([

tf.keras.layers.Dense(units=2, activation='relu', input\_shape=(2,)), # Hidden layer with 2 neurons and ReLU activation

tf.keras.layers.Dense(units=1, activation='sigmoid') # Output layer with 1 neuron and sigmoid activation

])

Step 4: Compile the model

model.compile(optimizer='adam',loss='binary\_crossentropy', metrics=['accuracy'])

Step 5: Train the model on the XOR data

model.fit(x\_train, y\_train, epochs=1000, batch\_size=1)

Step 6: Test the model and make predictions

predictions = model.predict(x\_train)

print(predictions.round()) # Round predictions to 0 or 1

Output:

[[0]

[1]

[1]

[0]]