Date of the Session:__/__/___ **Time of the Session:____to________**

SKILLING -4:

Implement a feedforward neural network and write the backpropagation code for training the network. Use numpy for all matrix/vector operations. You are not allowed to use any automatic differentiation packages. This network will be trained and tested using the XOR input with one output And also with Fashion-MNIST dataset with each image size as 28 x 28. Train the MNIST model to classify the images into one of 10 classes.

```
import numpy as np
from tensorflow.keras.datasets import fashion_mnist

def sigmoid(x): return 1 / (1 + np.exp(-x))
def sigmoid_derivative(x): return x * (1 - x)
def softmax(x): exp_x = np.exp(x - np.max(x, axis=1, keepdims=True)); return exp_x /
np.sum(exp_x, axis=1, keepdims=True)
def one_hot(y, num_classes): return np.eye(num_classes)[y]

X_xor, y_xor = np.array([[0, 0], [0, 1], [1, 0], [1, 1]]), np.array([[0], [1], [1], [0]])

class NeuralNetwork:
    def __init__(self, input_size, hidden_size, output_size):
        self.W1, self.b1 = np.random.randn(input_size, hidden_size) * 0.1, np.zeros((1,
hidden_size))
        self.W2, self.b2 = np.random.randn(hidden_size, output_size) * 0.1, np.zeros((1,
output_size))

    def forward(self, X):
        self.a1 = sigmoid(np.dot(X, self.W1) + self.b1)
        self.a2 = sigmoid(np.dot(self.a1, self.W2) + self.b2)
        return self.a2

    def backward(self, X, y, lr):
        dz2, dz1 = self.a2 - y, np.dot((self.a2 - y), self.W2.T) * sigmoid_derivative(self.a1)
        self.W2 -= lr * np.dot(self.a1.T, dz2) / X.shape[0]
        self.b2 -= lr * np.sum(dz2, axis=0, keepdims=True) / X.shape[0]
```

```
        self.W1 -= lr * np.dot(X.T, dz1) / X.shape[0]
    self.b1 -= lr * np.sum(dz1, axis=0, keepdims=True) / X.shape[0]

    def train(self, X, y, epochs=10000, lr=0.1):
        for i in range(epochs):
            self.forward(X); self.backward(X, y, lr)
            if i % 1000 == 0: print(f"Epoch {i}, Loss: {np.mean(np.square(y - self.a2))}")

NeuralNetwork(2, 2, 1).train(X_xor, y_xor, epochs=10000, lr=0.1)

(train_X, train_y), (test_X, test_y) = fashion_mnist.load_data()
train_X, test_X = train_X.reshape(-1, 28*28) / 255.0, test_X.reshape(-1, 28*28) / 255.0
train_y, test_y = one_hot(train_y, 10), one_hot(test_y, 10)

class FashionMNIST_NN:
    def __init__(self, input_size=784, hidden_size=128, output_size=10):
        self.W1, self.b1 = np.random.randn(input_size, hidden_size) * 0.1, np.zeros((1,
hidden_size))
        self.W2, self.b2 = np.random.randn(hidden_size, output_size) * 0.1, np.zeros((1,
output_size))

    def forward(self, X):
        self.a1 = sigmoid(np.dot(X, self.W1) + self.b1)
        self.a2 = softmax(np.dot(self.a1, self.W2) + self.b2)

    def backward(self, X, y, lr):
        dz2, dz1 = self.a2 - y, np.dot((self.a2 - y), self.W2.T) * sigmoid_derivative(self.a1)
        self.W2 -= lr * np.dot(self.a1.T, dz2) / X.shape[0]
        self.b2 -= lr * np.sum(dz2, axis=0, keepdims=True) / X.shape[0]
        self.W1 -= lr * np.dot(X.T, dz1) / X.shape[0]
        self.b1 -= lr * np.sum(dz1, axis=0, keepdims=True) / X.shape[0]

    def train(self, X, y, epochs=10, lr=0.1, batch_size=128):
        for epoch in range(epochs):
            indices = np.random.permutation(X.shape[0])
            for i in range(0, X.shape[0], batch_size):
                batch_X, batch_y = X[indices[i:i+batch_size]], y[indices[i:i+batch_size]]
                self.forward(batch_X); self.backward(batch_X, batch_y, lr)
```

```
        print(f"Epoch {epoch+1}, Loss: {-np.mean(batch_y * np.log(self.a2 + 1e-8))}")
FashionMNIST_NN().train(train_X, train_y, epochs=10, lr=0.1)
```

Output:

```
Epoch 0, Loss: 0.2502720778643092
Epoch 1000, Loss: 0.2500000092971527
Epoch 2000, Loss: 0.25000000912835374
Epoch 3000, Loss: 0.2500000089617544
Epoch 4000, Loss: 0.25000000879730505
Epoch 5000, Loss: 0.25000000863495725
Epoch 6000, Loss: 0.2500000084746638
Epoch 7000, Loss: 0.2500000083163781
Epoch 8000, Loss: 0.2500000081600551
Epoch 9000, Loss: 0.25000000800565025
Epoch 1, Loss: 0.06790259371130945
Epoch 2, Loss: 0.04973718426920947
Epoch 3, Loss: 0.05829247369689424
Epoch 4, Loss: 0.04573250149950783
Epoch 5, Loss: 0.04940333995359443
Epoch 6, Loss: 0.038010312983734924
Epoch 7, Loss: 0.0378802080111726
Epoch 8, Loss: 0.042812868550955445
Epoch 9, Loss: 0.056805648805318874
Epoch 10, Loss: 0.04287070101739048
```

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured ______ out of 50 Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation: