

| **Title:** Implementation of OR function with bipolar inputs and targets using Adaline network. Assume the required parameters for training of the network. |
| --- |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** To learn Adaline network.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

CO2: To understand the features of neural networks and different learning methods.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**Pre Lab/ Prior Concepts:**

**Adaptive Linear Neuron (Adaline):**

Adaline which stands for Adaptive Linear Neuron, is a network having a single linear unit.

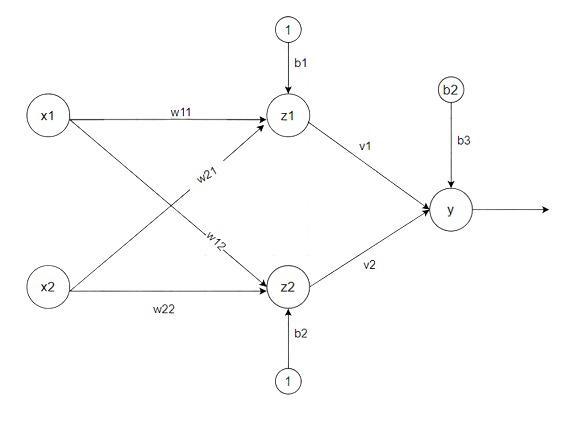
It was developed by Widrow and Hoff in 1960.

Some important points about Adaline are as follows −

* It uses bipolar activation function.
* It tries to minimize the Mean-Squared Error (MSE) between the actual output and the desired/target output.

The weights and the bias are adjustable

**Architecture:**

****

**Algorithm:**

Step 1: Initialize the following to start the training −Weights, Bias, Learning rate *α*

Step 2: While the stopping condition is False do steps 3 to 7.

Step 3: for each training set perform steps 4 to 6.

Step 4: Set activation of input unit xi = si for (i=1 to n).

Step 5: compute net input to output unit

Here, b is the bias and n is the total number of neurons.

Step 6: Update the weights and bias for i=1 to n 



and calculate

Step 7: Test the stopping condition. The stopping condition may be when the weight changes at a low rate or no change.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Implementation Details:**

Implementation of OR function with bipolar inputs and targets using Adaline network.



import numpy as np

def adaline(X, t, learning\_rate=0.1, initial\_weights=0.1, mean\_squared\_error=2, max\_epochs=100):

# Initialize weights and bias

weights = np.array([initial\_weights] \* X.shape[1])

bias = np.random.rand(1)

for epoch in range(max\_epochs):

total\_error = 0

for i in range(len(X)):

net = np.dot(X[i], weights) + bias

y = net

error = t[i] - y

total\_error += error\*\*2

weights += learning\_rate \* error \* X[i]

bias += learning\_rate \* error

if total\_error < mean\_squared\_error:

print(f"Mean squared error: {total\_error}")

break

return weights, bias, epoch + 1

# Input data and targets

X = np.array([[1, 1], [1, -1], [-1, 1], [-1, -1]])

t = np.array([1, 1, 1, -1])

# Train the Adaline network

weights, bias, epochs = adaline(X, t)

total\_error = 0

# Test the trained network

for x in X:

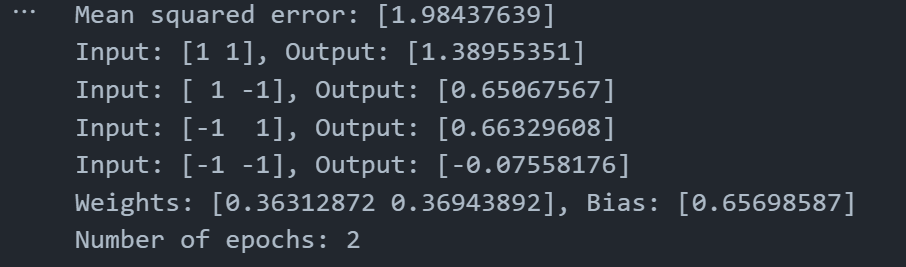
net = np.dot(x, weights) + bias

y = net

print(f"Input: {x}, Output: {y}")

print(f"Weights: {weights}, Bias: {bias}")

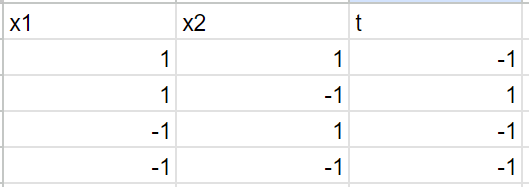
print(f"Number of epochs: {epochs}")

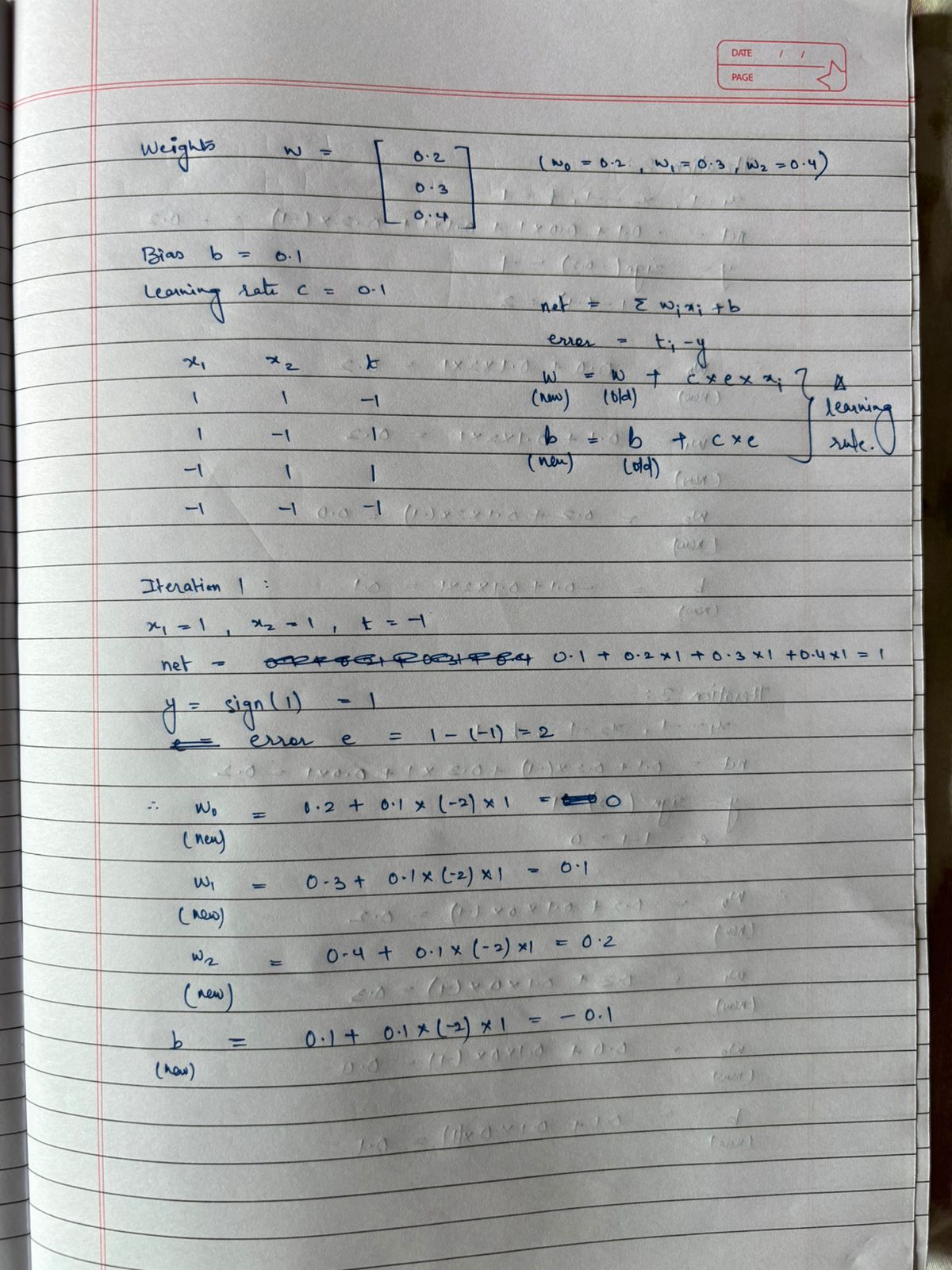


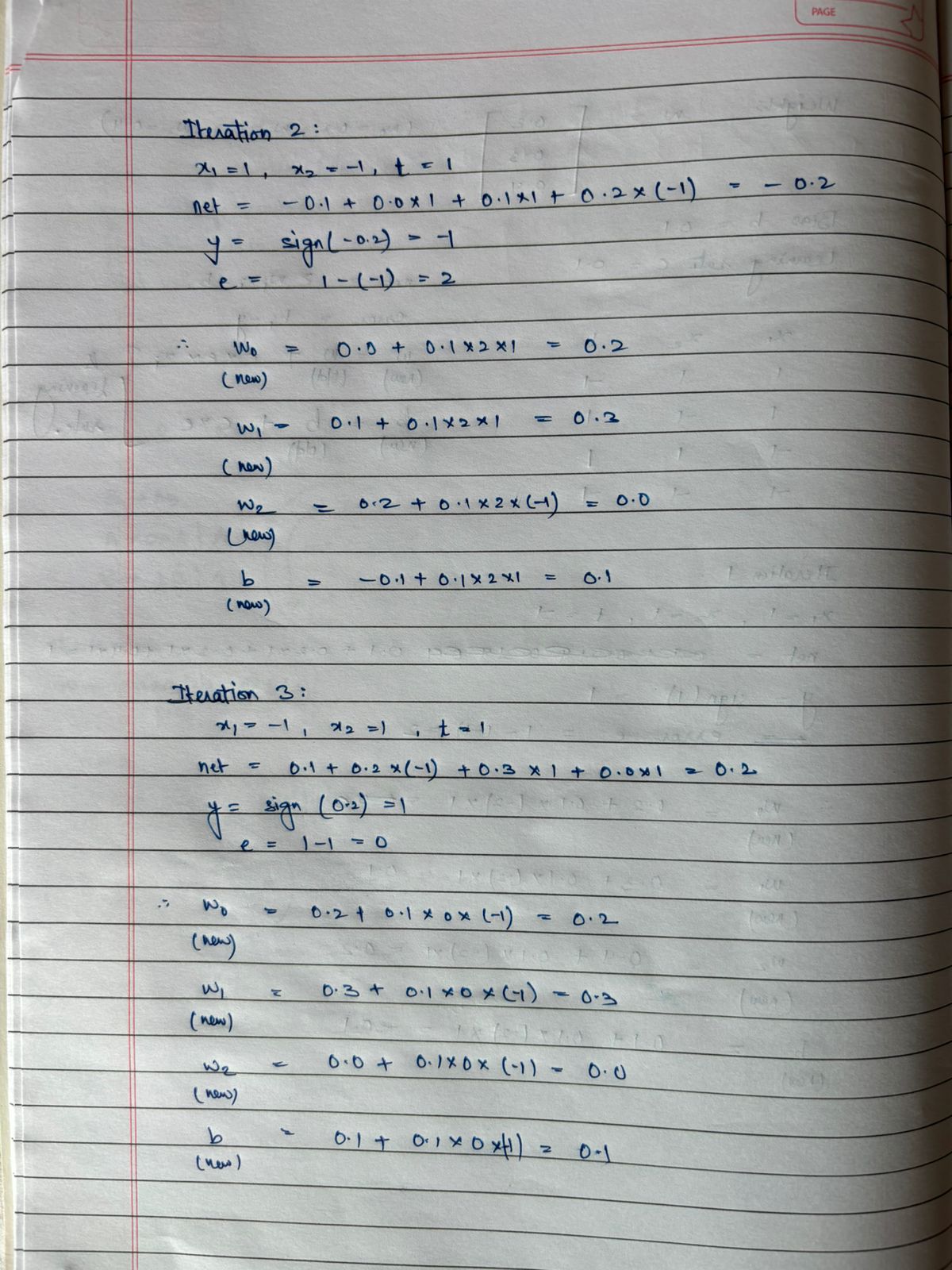
**Conclusion:** Learnt how to implement Adeline network for OR function for bipolar inputs.

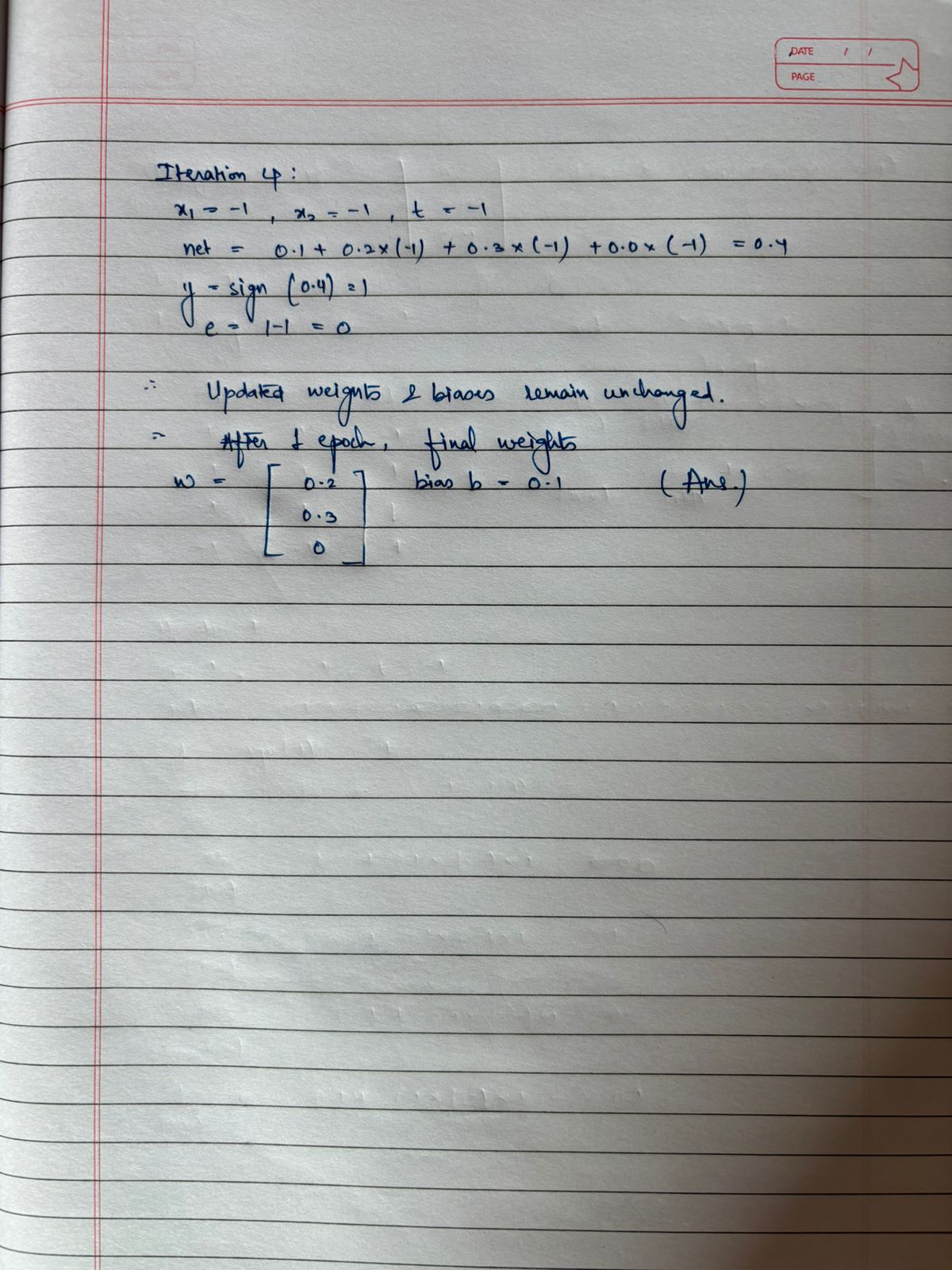
**Post Lab Descriptive Questions:**

Use Adaline network to train AND NOT function with bipolar inputs and targets. Perform 1 epoch of training









**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**