Emin Sadikhov **S017026**

Introduction to Machine Learning and Artificial Neural Networks

Single and Multi-Layer Perceptron

5 December 2021

Part 1

a) Implement an SLP where the inputs are 100-dimensional flattened images and the outputs are 10-dimensional class probabilities.

b) Train the perceptron for a number of epochs (e.g., 50) using the training dataset and plot the accuracy values for each epoch on both training and test datasets.

c) Report the confusion matrices for both training and test datasets using the predictions with the highest accuracy obtained during training.

Confusion train :

[[100. 0. 1. 0. 0. 0. 0. 0. 0. 2.]

[0. 94. 0. 0. 0. 0. 0. 0. 2. 0.]

[0. 1. 82. 2. 0. 0. 0. 0. 3. 0.]

[0. 1. 8. 93. 1. 6. 0. 0. 7. 2.]

[0. 0. 1. 0. 97. 1. 0. 0. 0. 2.]

[0. 0. 0. 1. 0. 91. 0. 0. 8. 1.]

[0. 0. 2. 0. 2. 1. 100. 0. 1. 0.]

[0. 2. 4. 1. 0. 0. 0. 100. 1. 7.]

[0. 2. 2. 2. 0. 0. 0. 0. 75. 0.]

[0. 0. 0. 1. 0. 1. 0. 0. 3. 86.]]

Confusion test :

[[91. 0. 0. 0. 0. 0. 2. 0. 0. 0.]

[0. 89. 0. 0. 1. 1. 0. 0. 1. 0.]

[0. 0. 80. 0. 0. 2. 2. 2. 1. 0.]

[2. 9. 4. 91. 6. 6. 1. 6. 12. 8.]

[0. 0. 0. 1. 84. 5. 1. 2. 3. 3.]

[2. 0. 0. 6. 0. 72. 5. 1. 8. 1.]

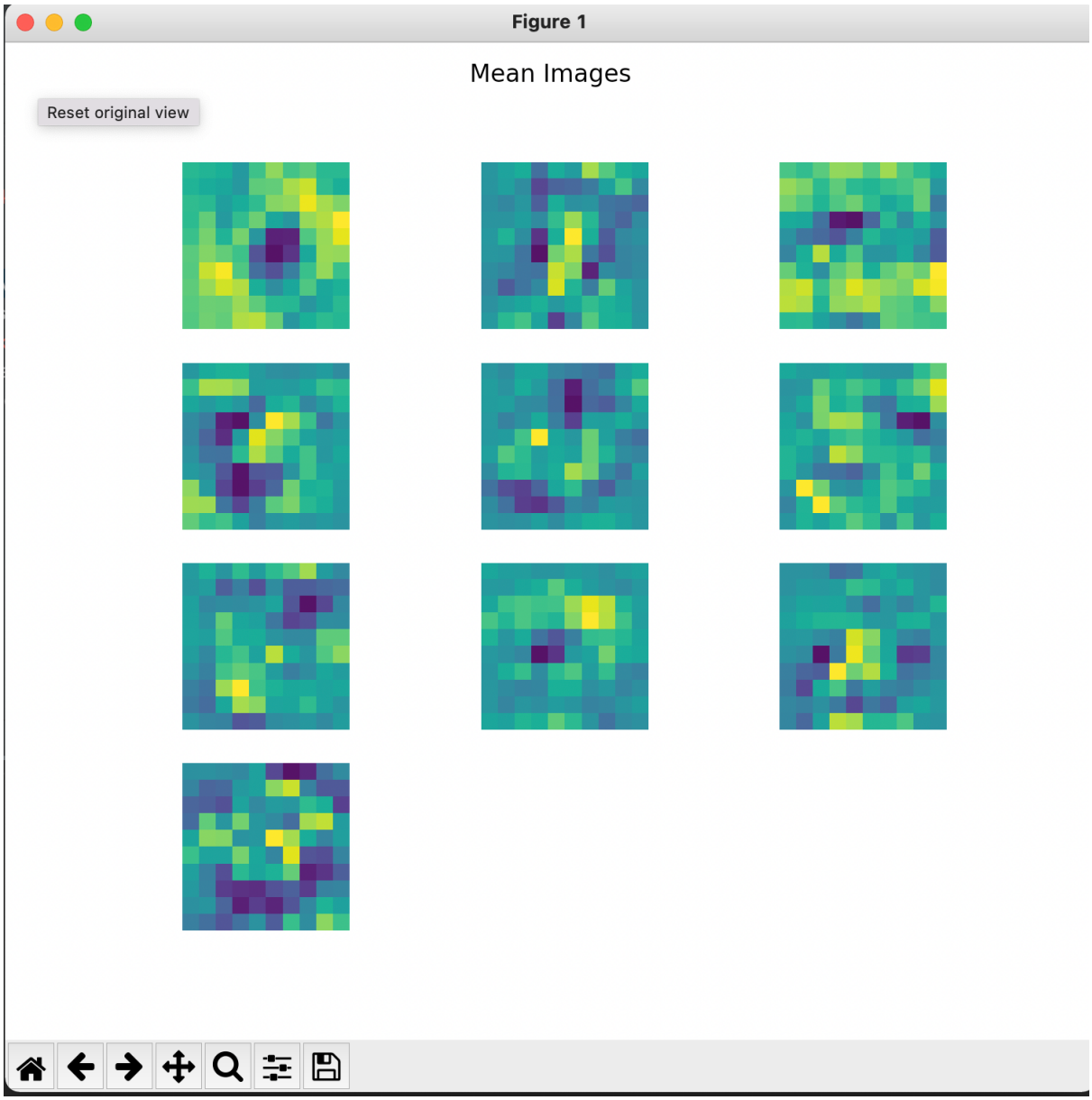
[2. 0. 3. 1. 2. 0. 88. 0. 3. 0.]

[2. 1. 4. 0. 0. 7. 0. 82. 2. 14.]

[1. 1. 6. 1. 0. 3. 1. 2. 61. 0.]

[0. 0. 3. 0. 7. 4. 0. 5. 9. 74.]]

d) Each output unit has 100 weights coming to them. Visualize those weights as an image for each output unit.

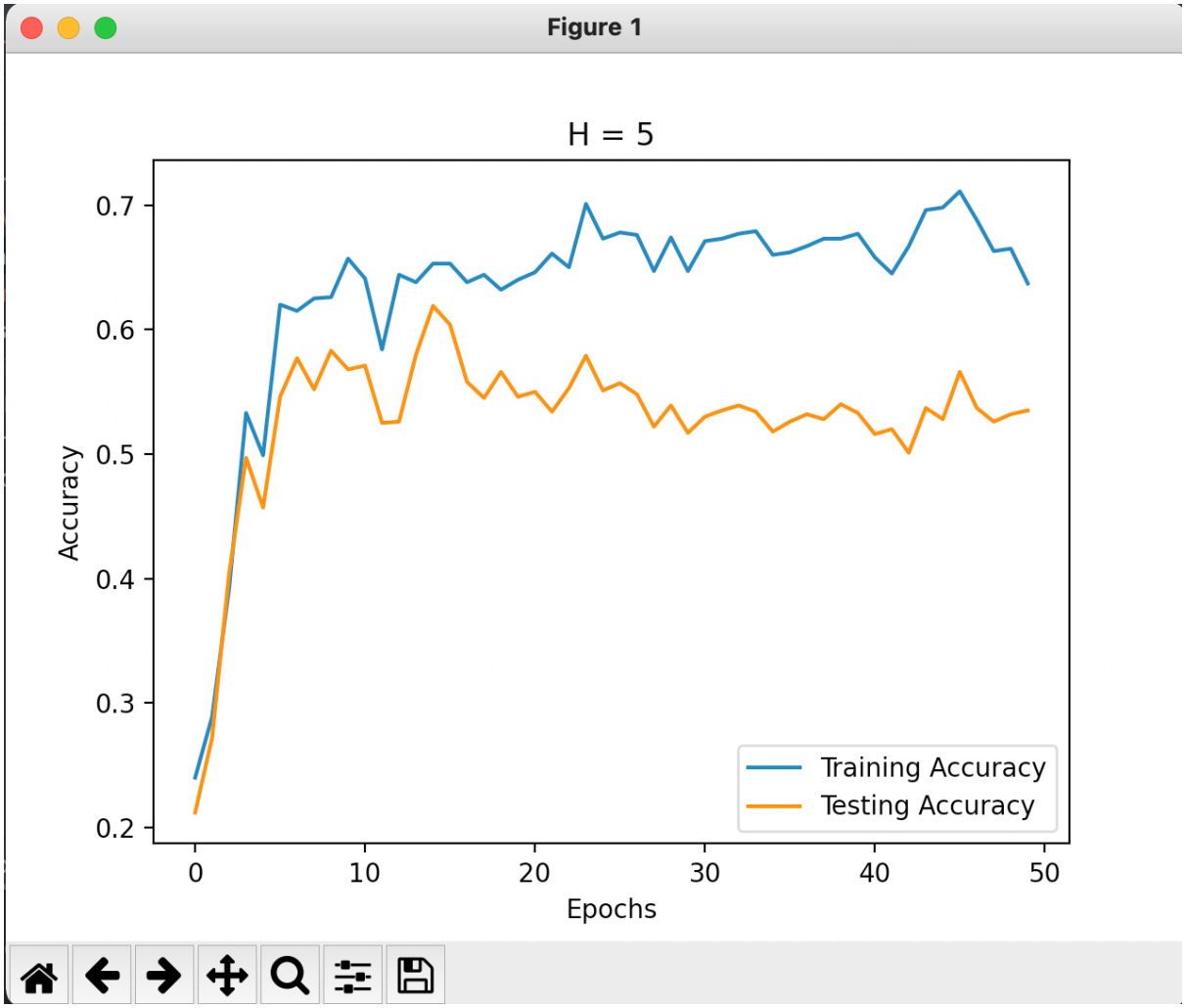


Part 2

a) Implement an MLP where the input is a 100-dimensional flattened image, the hidden layer has H units, and the outputs are 10-dimensional class probabilities.

b) Train the perceptron for a number of epochs using the training dataset and plot the accuracy values for each epoch on both training and test datasets.

Plot for H = 5



Plot for H = 10

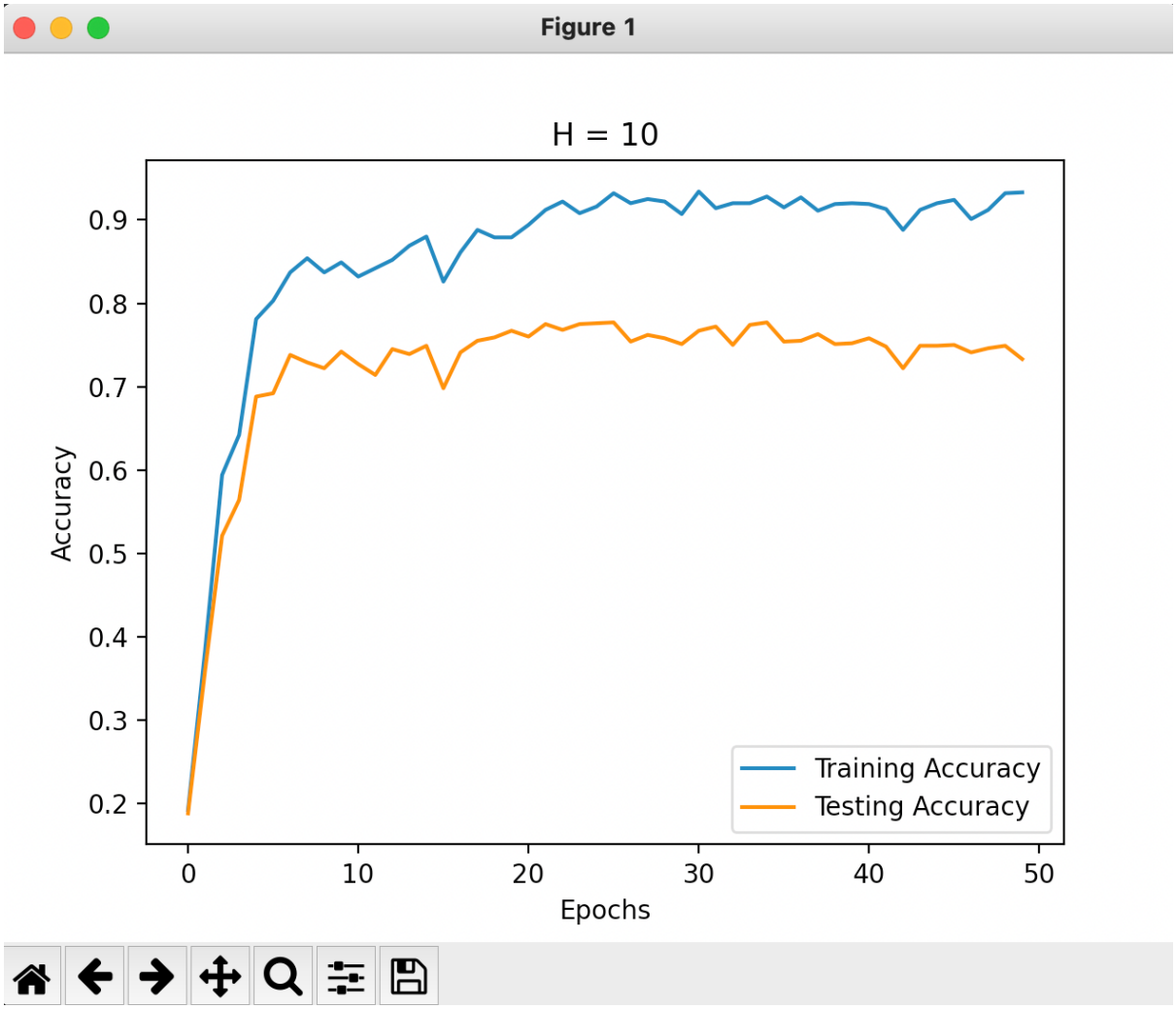


Figure 1
H = 10
Accuracy
Epochs
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50
Training Accuracy
Testing Accuracy

Plot for H = 25

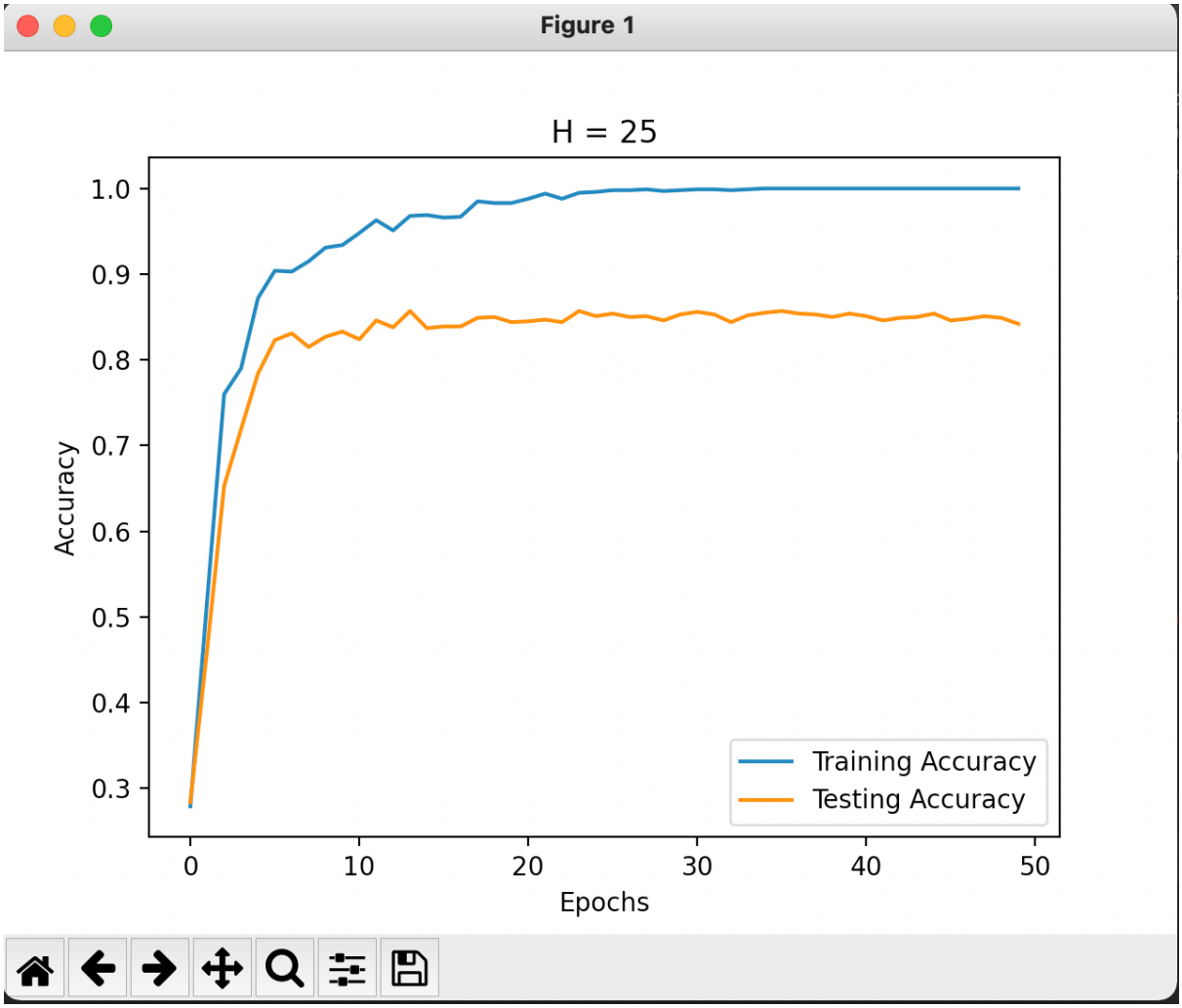


Figure 1
H = 25
Accuracy
Epochs
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
10
20
30
40
50
Training Accuracy
Testing Accuracy

Plot for H = 50

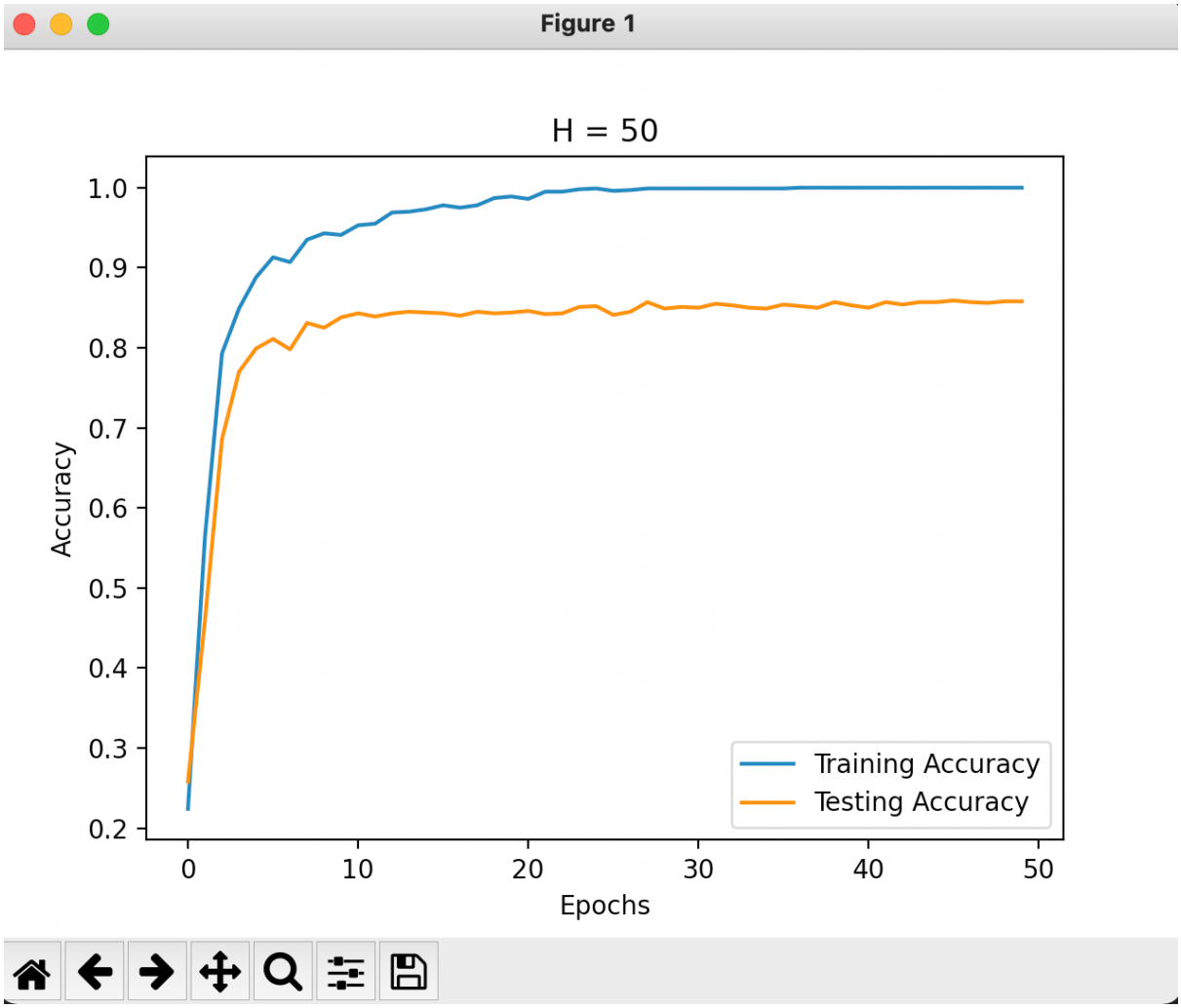


Figure 1
H = 50
Accuracy
Epochs
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50
Training Accuracy
Testing Accuracy

Plot for H = 75

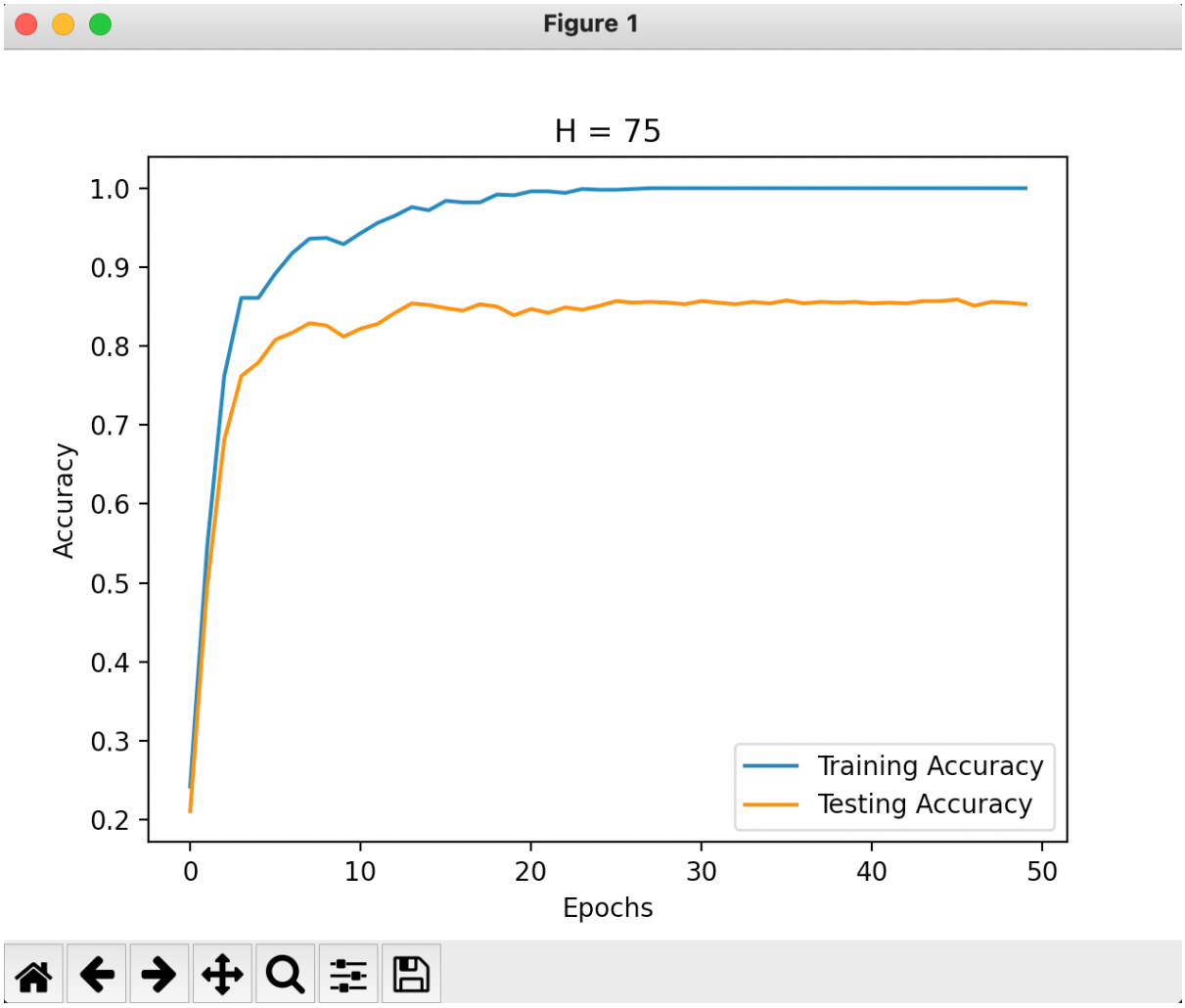


Figure 1
H = 75
Accuracy
Epochs
Training Accuracy
Testing Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50

c) Report the confusion matrices for both training and test datasets using the predictions with the highest accuracy obtained during training.

Confusion Matrixes for H = 5

Confusion train:

```
[[49.  0. 11.  5.  0. 13. 10.  0.  2.  0.]
 [ 0. 92.  4.  0.  0.  0.  0.  1.  0.  0.]
 [35.  1. 59. 18.  0. 14.  9.  0.  5.  1.]
 [ 0.  0.  0. 23.  0. 10.  0.  0.  8.  0.]
 [ 1.  0.  0.  0. 94.  8.  5.  1.  0. 13.]
 [ 3.  0.  2. 10.  0. 37.  3.  0.  2.  1.]
 [ 5.  0.  5.  0.  4.  1. 70.  0.  0.  0.]
 [ 0.  3.  3.  2.  0.  2.  0. 87.  5.  5.]
 [ 7.  4. 16. 13.  1. 10.  3.  1. 66.  4.]
 [ 0.  0.  0. 29.  1.  5.  0. 10. 12. 76.]]
```

Confusion test:

```
[[75.  0. 16.  9.  0. 15. 15.  0.  2.  0.]
 [ 0. 88.  1.  0.  0.  1.  0.  0.  2.  0.]
 [ 7.  0. 48.  3.  0.  6.  4.  2.  5.  0.]
 [ 0.  0.  0. 36.  0. 18.  0.  0. 16. 10.]
 [ 3.  0.  2.  1. 86.  3.  6.  7.  2. 17.]
 [ 4.  0.  2. 21.  3. 27.  3.  0.  7.  0.]
 [ 4.  0. 11.  0.  4.  0. 72.  0.  1.  1.]
 [ 0.  8.  4.  3.  1.  4.  0. 75.  5.  9.]
 [ 6.  4. 15. 24.  3. 14.  0.  5. 54.  5.]
 [ 1.  0.  1.  3.  3. 12.  0. 11.  6. 58.]]
```

Confusion Matrixes for H = 10

Confusion train:

```
[[98.  0.  0.  1.  0.  1.  0.  0.  0.  0.]
 [ 0. 99.  0.  0.  0.  1.  1.  2.  0.  0.]
 [ 0.  1. 97.  2.  0.  0.  3.  0.  1.  0.]
 [ 0.  0.  0. 89.  0.  5.  0.  0.  1.  0.]
 [ 0.  0.  0.  0. 90.  1.  0.  0.  0.  3.]
 [ 2.  0.  0.  3.  2. 82.  0.  1.  3.  2.]
 [ 0.  0.  1.  0.  5.  4. 96.  0.  0.  0.]
 [ 0.  0.  2.  2.  1.  0.  0. 95.  0.  3.]
 [ 0.  0.  0.  0.  1.  2.  0.  0. 95.  1.]
 [ 0.  0.  0.  3.  1.  4.  0.  2.  0. 91.]]
```

Confusion test:

```
[[86.  0.  1.  1.  1.  1.  3.  0.  3.  3.]
 [ 0. 99.  1.  0.  0.  1.  0.  2.  0.  0.]
 [ 6.  1. 88.  8.  0.  2.  3.  4.  1.  0.]
 [ 1.  0.  0. 70.  0.  8.  0.  3.  2.  5.]
 [ 3.  0.  1.  0. 70.  3.  3.  2.  2.  5.]
 [ 4.  0.  1. 14.  2. 53.  4.  0. 11.  2.]
 [ 0.  0.  5.  1.  5.  6. 84.  0.  4.  0.]
 [ 0.  0.  1.  1.  0.  4.  0. 83.  0.  9.]
 [ 0.  0.  2.  3.  2. 11.  3.  1. 72.  4.]
 [ 0.  0.  0.  2. 20. 11.  0.  5.  5. 72.]]
```

Confusion Matrixes for H = 25

Confusion train:

```
[[100.   0.   0.   0.   0.   0.   1.   0.   0.   1.]
 [ 0.  98.   1.   0.   0.   0.   0.   0.   0.   0.]
```

[0. 1. 97. 3. 0. 1. 1. 0. 0. 0.]
[0. 0. 0. 92. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 96. 1. 0. 0. 0. 0.]
[0. 0. 0. 3. 0. 97. 0. 0. 1. 1.]
[0. 0. 0. 0. 3. 1. 98. 0. 1. 0.]
[0. 0. 0. 2. 0. 0. 0. 99. 0. 4.]
[0. 1. 1. 0. 0. 0. 0. 0. 98. 1.]
[0. 0. 1. 0. 1. 0. 0. 1. 0. 93.]]

Confusion test

[[95. 0. 0. 0. 0. 0. 2. 0. 2. 0.]
 [0. 99. 0. 1. 1. 1. 0. 0. 0. 0.]
 [1. 0. 86. 2. 0. 0. 2. 8. 3. 0.]
 [0. 0. 0. 73. 0. 3. 0. 0. 2. 2.]
 [0. 0. 2. 0. 80. 0. 0. 3. 1. 0.]
 [1. 0. 0. 18. 1. 78. 5. 1. 5. 0.]
 [2. 0. 4. 1. 4. 7. 91. 0. 3. 0.]
 [0. 0. 3. 1. 0. 5. 0. 84. 1. 6.]
 [0. 1. 5. 3. 0. 1. 0. 0. 80. 1.]
 [1. 0. 0. 1. 14. 5. 0. 4. 3. 91.]]

Confusion Matrixes for H = 50

Confusion train:

[[100. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 100. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 100. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 100. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 100. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 100. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 100. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 100. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 100. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 100.]]

Confusion test:

```
[[ 96.  0.  0.  0.  0.  2.  2.  0.  3.  0.]
 [ 0. 100.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  0. 85.  1.  0.  1.  0.  5.  2.  0.]
 [ 0.  0.  2. 80.  1.  1.  0.  9.  3.  4.]
 [ 0.  0.  2.  0. 83.  0.  0.  2.  3.  1.]
 [ 0.  0.  0. 14.  0. 78.  6.  1.  4.  2.]
 [ 3.  0.  3.  2.  2.  5. 91.  0.  1.  0.]
 [ 0.  0.  4.  0.  0.  7.  0. 80.  1.  6.]
 [ 0.  0.  3.  2.  2.  2.  1.  0. 81.  2.]
 [ 0.  0.  1.  1. 12.  4.  0.  3.  2. 85.]]
```

Confusion Matrixes for H =7 5

Confusion train:

```
[[100.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. 100.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. 100.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. 100.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0. 100.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0. 100.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0. 100.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0. 100.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0. 100.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0. 100.]]
```

Confusion test:

```
[[ 97.  0.  1.  0.  0.  1.  2.  1.  3.  0.]
 [ 0. 100.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. 86.  2.  0.  2.  3.  5.  3.  0.]
 [ 0.  0.  1. 79.  1.  2.  0.  8.  3.  4.]
 [ 0.  0.  1.  0. 84.  0.  0.  1.  3.  0.]
 [ 0.  0.  0. 14.  0. 80.  5.  1.  3.  1.]
 [ 1.  0.  3.  2.  4.  5. 89.  0.  2.  0.]
 [ 0.  0.  3.  0.  0.  6.  0. 80.  2.  7.]
 [ 0.  0.  4.  3.  1.  2.  1.  0. 79.  3.]
 [ 2.  0.  1.  0. 10.  2.  0.  4.  2. 85.]]
```

GitHub Link for the code: https://github.com/SadikhovEmin/CS454

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

LEARNING_RATE = 0.1
CLASS_SIZE = 10
FEATURES_SIZE = 100
training_matrix = pd.read_csv('training.csv', header=None, skiprows=1)
testing_matrix = pd.read_csv('testing.csv', header=None, skiprows=1)

weight_matrix = np.random.uniform(low=-0.01, high=0.01,
                                  size=(FEATURES_SIZE + 1, CLASS_SIZE))  # Random ma
trix for weights
weight_matrix_best = np.zeros((FEATURES_SIZE + 1, CLASS_SIZE))  # Best possible valu
es for matrix for weights


def softmax(x):
    value = np.exp(x - np.max(x))
    return value / value.sum()


def encode(y_value):
    return np.transpose(np.eye(10)[y_value])


def accuracy_confusion_matrix(x):
    correct = 0
    total = 0
    for i in range(len(x)):
        for j in range(len(x[i])):
            total += x[i][j]
            if i == j:
                correct += x[i][j]
    return correct / total


def forward(slp_object):
    for dataset_row_count in range(slp_object.number_of_rows):  # might need to make
 slp_object SLP_TRAIN
        row = slp_object.x.iloc[dataset_row_count, :]
        row = np.append(row, 1)
        row = np.transpose(row)

        o = np.zeros(CLASS_SIZE)

        for i in range(CLASS_SIZE):
            o[i] = np.dot(weight_matrix[:, i], row)

        slp_object.y_matrix[dataset_row_count] = softmax(o)


class SLP(object):
    def __init__(self, matrix):
        self.labels = matrix.iloc[:, 0]  # Gets only the first column (Just class na
me)
        self.x = matrix.iloc[:, 1:] / 255  # Gets all rows and columns except the fi
rst one
        self.number_of_rows = matrix.shape[0]
        self.y_matrix = np.zeros((self.number_of_rows, CLASS_SIZE))


def train(SLP_Train, SLP_Test):
    global weight_matrix_best
    accuracy_best = 0
    best_label_train = []
    best_label_test = []
    confusion_matrix_train = np.zeros((CLASS_SIZE, CLASS_SIZE))  # Confusion matrix
for training set with zeroes
    confusion_matrix_test = np.zeros((CLASS_SIZE, CLASS_SIZE))  # Confusion matrix f
or testing set with zeroes

    for epoch in range(50):
```

```
        temp_weight = np.zeros(weight_matrix.shape)  # Copy w_matrix to the temp mat
rix

        for dataset_row_count in range(SLP_Train.number_of_rows):
            row = SLP_Train.x.iloc[dataset_row_count, :]
            row = np.append(row, 1)
            row = np.transpose(row)

            o = np.zeros(CLASS_SIZE)

            for i in range(CLASS_SIZE):
                o[i] = np.dot(weight_matrix[:, i], row)

            SLP_Train.y_matrix[dataset_row_count] = softmax(o)

            encode_matrix = encode(SLP_Train.labels.iloc[dataset_row_count])

            for i in range(CLASS_SIZE):
                for j in range(FEATURES_SIZE + 1):
                    temp_weight[j, i] += (encode_matrix[i] - SLP_Train.y_matrix[data
set_row_count][i]) * row[j]

            for i in range(CLASS_SIZE):
                for j in range(FEATURES_SIZE + 1):
                    weight_matrix[j, i] += temp_weight[j, i] * LEARNING_RATE

        forward(SLP_Train)
        forward(SLP_Test)

        label_train = []
        label_test = []

        correct_train = 0
        correct_test = 0
        for dataset_row_count in range(SLP_Train.number_of_rows):
            if SLP_Train.labels.iloc[dataset_row_count] == np.argmax(SLP_Train.y_mat
rix[dataset_row_count]):
                correct_train += 1
            if SLP_Test.labels.iloc[dataset_row_count] == np.argmax(SLP_Test.y_matri
x[dataset_row_count]):
                correct_test += 1

            label_train.append(np.argmax(SLP_Train.y_matrix[dataset_row_count]))
            label_test.append(np.argmax(SLP_Test.y_matrix[dataset_row_count]))

        accuracy_training = correct_train / SLP_Train.number_of_rows
        accuracy_test = correct_test / SLP_Test.number_of_rows

        print("Epoch: ", epoch + 1, "Training Accuracy: ", accuracy_training, "Testi
ng Accuracy: ", accuracy_test)

        if accuracy_test > accuracy_best:
            accuracy_best = accuracy_test
            weight_matrix_best = weight_matrix
            best_label_train = label_train
            best_label_test = label_test

    print('weight matrix best : ', weight_matrix_best.shape)

    for i in range(SLP_Train.number_of_rows):
        confusion_matrix_train[best_label_train[i], SLP_Train.labels[i]] += 1

    for i in range(SLP_Test.number_of_rows):
        confusion_matrix_test[best_label_test[i], SLP_Test.labels[i]] += 1

    print('Confusion train : ', confusion_matrix_train)
    print('Confusion test : ', confusion_matrix_test)


def plot():
    fig, axes = plt.subplots(4, 3)
    fig.set_figheight(8)
```

```
    fig.set_figwidth(8)
    fig.suptitle('Mean Images')
    for label in range(12):
        if label < 10:
            axes[label // 3][label % 3].imshow(weight_matrix_best[:-1, :][:, label].
reshape(10, 10), )
        axes[label // 3][label % 3].axis('off')
    plt.show()


if __name__ == '__main__':
    SLP_Train = SLP(training_matrix)
    SLP_Test = SLP(testing_matrix)

    train(SLP_Train, SLP_Test)
    plot()
```

```
import numpy as np
import pandas as pd
import random
import matplotlib.pyplot as plt


training_matrix = pd.read_csv('training.csv', header=None, skiprows=1)
testing_matrix = pd.read_csv('testing.csv', header=None, skiprows=1)
LEARNING_RATE = 0.1
CLASS_SIZE = 10
FEATURES_SIZE = 100


def accuracy_confusion_matrix(x):
    correct = 0
    total = 0
    for i in range(len(x)):
        for j in range(len(x[i])):
            total += x[i][j]
            if i == j:
                correct += x[i][j]
    return correct / total


def encode(y):
    return np.transpose(np.eye(10)[y])


def sigmoid(x):
    return 1. / (1. + np.exp(-x))


def softmax(x):
    e_x = np.exp(x - np.max(x))
    return e_x / e_x.sum()


class MLP(object):
    def __init__(self, matrix):
        self.labels = matrix.iloc[:, 0]  # Gets only the first column (Just class na
me)
        self.x = matrix.iloc[:, 1:] / 255  # Gets all rows and columns except the fi
rst one
        self.number_of_rows = matrix.shape[0]
        self.y_matrix = np.zeros((self.number_of_rows, CLASS_SIZE))


def forward(mlp_object, hidden_layer, weight_matrix, field_matrix):
    for instance in range(mlp_object.number_of_rows):
        row = mlp_object.x.iloc[instance, :]
        row = np.append(1, row)
        row = np.transpose(row)

        logit = np.zeros(hidden_layer)
        for h in range(hidden_layer):
            w_h_T = np.transpose(weight_matrix[:, h])
            logit[h] = np.dot(w_h_T, row)

        logit = sigmoid(logit)
        logit = np.append(1, logit)
        logit = np.transpose(logit)

        activation = np.zeros(CLASS_SIZE)
        for i in range(CLASS_SIZE):
            activation[i] = np.dot(np.transpose(field_matrix[:, i]), logit)

        mlp_object.y_matrix[instance] = softmax(activation)


def train(MLP_Train, MLP_Test):
    for H in [5, 10, 25, 50, 75]:
        accuracy_best = 0
        best_label_train = []
```

```
        best_label_test = []

        training_accuracies = []
        test_accuracies = []

        field_matrix = np.random.uniform(low=-0.01, high=0.01, size=(H + 1, CLASS_SI
ZE))
        weight_matrix = np.random.uniform(low=-0.01, high=0.01, size=(FEATURES_SIZE
+ 1, H))
        for epochs in range(50):
            random_list = list(range(MLP_Train.number_of_rows))
            random.shuffle(random_list)

            for element in random_list:
                row = MLP_Train.x.iloc[element, :]
                row = np.append(1, row)
                row = np.transpose(row)

                logit = np.zeros(H)

                for h in range(H):
                    w_h_T = np.transpose(weight_matrix[:, h])
                    logit[h] = np.dot(w_h_T, row)

                logit = sigmoid(logit)

                logit = np.append(1, logit)
                logit = np.transpose(logit)

                activation = np.zeros(CLASS_SIZE)
                for i in range(CLASS_SIZE):
                    activation[i] = np.dot(np.transpose(field_matrix[:, i]), logit)

                MLP_Train.y_matrix[element] = softmax(activation)

                change_field_matrix = np.zeros(field_matrix.shape)
                change_weight_matrix = np.zeros(weight_matrix.shape)

                r_t = encode(MLP_Train.labels.iloc[element])

                for i in range(CLASS_SIZE):
                    change_field_matrix[:, i] = LEARNING_RATE * (r_t[i] - MLP_Train.
y_matrix[element][i]) * logit

                for h in range(H):
                    total = 0.0
                    for i in range(CLASS_SIZE):
                        total += (r_t[i] - MLP_Train.y_matrix[element][i]) * field_m
atrix[h, i]
                    change_weight_matrix[:, h] = LEARNING_RATE * total * logit[h + 1
] * (1 - logit[h + 1]) * row

                for i in range(CLASS_SIZE):
                    field_matrix[:, i] += change_field_matrix[:, i]

                for h in range(H):
                    weight_matrix[:, h] += change_weight_matrix[:, h]

            forward(MLP_Train, H, weight_matrix, field_matrix)
            forward(MLP_Test, H, weight_matrix, field_matrix)

            label_train = []
            label_test = []

            correct_train = 0
            correct_test = 0
            for t in range(MLP_Train.number_of_rows):
                if MLP_Train.labels.iloc[t] == np.argmax(MLP_Train.y_matrix[t]):
                    correct_train += 1
                if MLP_Test.labels.iloc[t] == np.argmax(MLP_Test.y_matrix[t]):
                    correct_test += 1

                label_train.append(np.argmax(MLP_Train.y_matrix[t]))
```

```
                label_test.append(np.argmax(MLP_Test.y_matrix[t]))

            accuracy_training = correct_train / MLP_Train.number_of_rows
            accuracy_test = correct_test / MLP_Test.number_of_rows

            training_accuracies.append(accuracy_training)
            test_accuracies.append(accuracy_test)

            print("H: ", H, " Epoch: ", epochs + 1, "Training Accuracy: ", accuracy_
training, "Testing Accuracy: ",
                  accuracy_test)

            if accuracy_test > accuracy_best:
                accuracy_best = accuracy_test
                best_label_train = label_train
                best_label_test = label_test

        confusion_matrix_train = np.zeros((CLASS_SIZE, CLASS_SIZE))
        confusion_matrix_test = np.zeros((CLASS_SIZE, CLASS_SIZE))

        for i in range(MLP_Train.number_of_rows):
            confusion_matrix_train[best_label_train[i], MLP_Train.labels[i]] += 1

        for i in range(MLP_Test.number_of_rows):
            confusion_matrix_test[best_label_test[i], MLP_Test.labels[i]] += 1

        print('H = ', H)
        print('Confusion train : ', confusion_matrix_train)
        print('Confusion test : ', confusion_matrix_test)

        plt.title("H = " + str(H))
        plt.plot(training_accuracies, label='Training Accuracy')
        plt.plot(test_accuracies, label='Testing Accuracy')
        plt.xlabel("Epochs")
        plt.ylabel("Accuracy")
        plt.legend(loc='lower right')
        plt.show()


if __name__ == '__main__':
    MLP_Train = MLP(training_matrix)
    MLP_Test = MLP(testing_matrix)

    train(MLP_Train, MLP_Test)
```