```
import nn
import numpy as np
import matplotlib.pyplot as plt
%matplotlib qt

# Rarely changed parameters
shuffle = True # Shuffle the training data for stichastic gradient
descent
activation_function = 'sigmoid' # Activation function for hidden layer
('sigmoid', 'relu')
initialization = 'HeNormal' # Weight initialization method
('HeNormal', 'Normal')
save = False # Save the trained model parameters
show_each_plot = False # Show each plot during training
file_num = 2 # The dataset to use

## Experiments with hidden layer size

# Hyperparameters
learning_rate = 0.1
epochs = 100
hidden_layer_size = np.arange(5, 51, 5)

# Accuracy arrays
train_acc = np.zeros((len(hidden_layer_size), epochs))
test_acc = np.zeros((len(hidden_layer_size), 1))

# Call neural network for each hidden layer size
for i in range(len(hidden_layer_size)):
    train, test = nn.main(hidden_layer_size[i], learning_rate, epochs,
                          activation_function, initialization,
                          save, shuffle, show_each_plot, file_num)
    train_acc[i] = train
    test_acc[i] = test
    print(f"Hidden layer size: {hidden_layer_size[i]} | Final Train
accuracy : {train[-1]} | Test accuracy: {test}")

# Plot training accuracy vs. epochs
plt.figure("Training Accuracy")
for i in range(len(hidden_layer_size)):
    plt.plot(np.arange(1, epochs+1), train_acc[i], label=f"Hidden
units: {hidden_layer_size[i]}")

plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.title("Training Accuracy vs. Epochs for differnet hidden layer
sizes")
plt.legend()

# Plot test accuracy vs. hidden layer size
```

```
plt.figure("Test Accuracy")
plt.plot(hidden_layer_size, test_acc)
plt.xlabel("Hidden layer size")
plt.ylabel("Accuracy")
plt.ylim(50,100)
plt.title("Test Accuracy vs. Hidden Layer Size")

plt.show()
```

```
Hidden layer size: 5 | Final Train accuracy : 85.25 | Test accuracy:
82.0
Hidden layer size: 10 | Final Train accuracy : 70.25 | Test accuracy:
61.25000000000001
Hidden layer size: 15 | Final Train accuracy : 92.0 | Test accuracy:
90.5
Hidden layer size: 20 | Final Train accuracy : 90.75 | Test accuracy:
84.5
Hidden layer size: 25 | Final Train accuracy : 79.0 | Test accuracy:
69.25
Hidden layer size: 30 | Final Train accuracy : 70.0 | Test accuracy:
56.49999999999999
Hidden layer size: 35 | Final Train accuracy : 70.25 | Test accuracy:
60.75000000000001
Hidden layer size: 40 | Final Train accuracy : 81.25 | Test accuracy:
71.0
Hidden layer size: 45 | Final Train accuracy : 81.75 | Test accuracy:
70.5
Hidden layer size: 50 | Final Train accuracy : 74.25 | Test accuracy:
64.25
```

```
## Experiments with epochs

# Hyperparameters
learning_rate = 0.1
hidden_layer_size = 15
epochs = np.arange(60, 250, 20)

# Accuracy arrays
train_acc = np.zeros((len(epochs), epochs[-1]))
test_acc = np.zeros((len(epochs), 1))

# Call neural network for each number of epochs
for i in range(len(epochs)):
    train, test = nn.main(hidden_layer_size, learning_rate, epochs[i],
                          activation_function, initialization,
                          save, shuffle, show_each_plot, file_num)
    train_acc[i,:len(train)] = train
    test_acc[i] = test
    # Pad with NaNs to not have zero values in plot
    train_acc[i,len(train):] = np.nan
```

```
    print(f"Epochs: {epochs[i]} | Final Train accuracy : {train[-1]} | 
Test accuracy: {test}")

# Plot training accuracy vs. epochs
plt.figure("Training Accuracy")
for i in range(len(epochs)):
    plt.plot(np.arange(1, epochs[-1]+1), train_acc[i], label=f"Epochs: 
{epochs[i]}")

plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.title("Training Accuracy vs. Epochs for differnet number of epochs 
stopped at")
plt.legend()

# Plot test accuracy vs. hidden layer size
plt.figure("Test Accuracy")
plt.plot(epochs, test_acc)
plt.xlabel("Epochs (stopped at)")
plt.ylabel("Accuracy")
plt.ylim(50,100)
plt.title("Test Accuracy vs. Epochs (stopped at)")

plt.show()

Epochs: 60 | Final Train accuracy : 65.75 | Test accuracy: 56.75
Epochs: 80 | Final Train accuracy : 75.25 | Test accuracy: 70.5
Epochs: 100 | Final Train accuracy : 77.25 | Test accuracy: 
63.24999999999999
Epochs: 120 | Final Train accuracy : 90.25 | Test accuracy: 83.25
Epochs: 140 | Final Train accuracy : 90.0 | Test accuracy: 85.0
Epochs: 160 | Final Train accuracy : 95.0 | Test accuracy: 91.0
Epochs: 180 | Final Train accuracy : 97.75 | Test accuracy: 93.75
Epochs: 200 | Final Train accuracy : 91.5 | Test accuracy: 82.25
Epochs: 220 | Final Train accuracy : 99.75 | Test accuracy: 94.75
Epochs: 240 | Final Train accuracy : 99.5 | Test accuracy: 97.25

## Experiments with learning rate

# Hyperparameters
learning_rate = np.arange(0.6, 1.51, 0.1).round(2)
epochs = 80
hidden_layer_size = 15

# Accuracy arrays
train_acc = np.zeros((len(learning_rate), epochs))
test_acc = np.zeros((len(learning_rate), 1))

# Call neural network for each learning rate
for i in range(len(learning_rate)):
```

```python
    train, test = nn.main(hidden_layer_size, learning_rate[i], epochs,
                          activation_function, initialization,
                          save, shuffle, show_each_plot, file_num)
    train_acc[i] = train
    test_acc[i] = test
    print(f"Learning rate: {learning_rate[i]} | Final Train accuracy :
{train[-1]} | Test accuracy: {test}")

# Plot training accuracy vs. epochs
plt.figure("Training Accuracy")
for i in range(len(learning_rate)):
    plt.plot(np.arange(1, epochs+1), train_acc[i], label=f"Learning
rate: {learning_rate[i]}")

plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.title("Training Accuracy vs. Epochs for differnet learning rates")
plt.legend()

# Plot test accuracy vs. hidden layer size
plt.figure("Test Accuracy")
plt.plot(learning_rate, test_acc)
plt.xlabel("Learning Rate")
plt.ylabel("Accuracy")
plt.ylim(60,100)
plt.title("Test Accuracy vs. Learning Rate")

plt.show()
```

```
Learning rate: 0.6 | Final Train accuracy : 99.25 | Test accuracy:
98.25
Learning rate: 0.7 | Final Train accuracy : 98.25 | Test accuracy:
86.25
Learning rate: 0.8 | Final Train accuracy : 88.25 | Test accuracy:
88.0
Learning rate: 0.9 | Final Train accuracy : 100.0 | Test accuracy:
99.0
Learning rate: 1.0 | Final Train accuracy : 93.0 | Test accuracy:
91.25
Learning rate: 1.1 | Final Train accuracy : 89.0 | Test accuracy: 71.0
Learning rate: 1.2 | Final Train accuracy : 90.25 | Test accuracy:
88.75
Learning rate: 1.3 | Final Train accuracy : 98.25 | Test accuracy:
95.25
Learning rate: 1.4 | Final Train accuracy : 95.0 | Test accuracy: 88.0
Learning rate: 1.5 | Final Train accuracy : 98.5 | Test accuracy: 98.0
```

```python
## Experiments with other parameters

# Hyperparameters
```

```
learning_rate = 0.5
epochs = 80
hidden_layer_size = 15

## Experiments with activation function for hidden layer
activation_function = ['sigmoid', 'relu']
initialization = 'HeNormal'

# Accuracy arrays
train_acc = np.zeros((len(activation_function), epochs))

# Call neural network for each activation function
for i in range(len(activation_function)):
    train, test = nn.main(hidden_layer_size, learning_rate, epochs,
                          activation_function[i], initialization,
                          save, shuffle, show_each_plot, file_num)
    train_acc[i] = train
    print(f"Activation function: {activation_function[i]} | Final
Train accuracy : {train[-1]} | Test accuracy: {test}")

# Plot training accuracy vs. epochs
plt.figure("Activation Function")
for i in range(len(activation_function)):
    plt.plot(np.arange(1, epochs+1), train_acc[i], label=f"Activation
function: {activation_function[i]}")

plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.title("Training Accuracy vs. Epochs for differnet activation
functions")
plt.legend()

## Experiments with weight initialization
initialization = ['HeNormal', 'Normal']
activation_function = 'sigmoid'

# Accuracy arrays
train_acc = np.zeros((len(initialization), epochs))

# Call neural network for each weight initialization method
for i in range(len(initialization)):
    train, test = nn.main(hidden_layer_size, learning_rate, epochs,
                          activation_function, initialization[i],
                          save, shuffle, show_each_plot, file_num)
    train_acc[i] = train
    print(f"Weight initialization: {initialization[i]} | Final Train
accuracy : {train[-1]} | Test accuracy: {test}")

# Plot training accuracy vs. epochs
plt.figure("Weight Initialization")
```

```python
for i in range(len(initialization)):
    plt.plot(np.arange(1, epochs+1), train_acc[i], label=f"Weight initialization: {initialization[i]}")

plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.title("Training Accuracy vs. Epochs for differnet weight initialization methods")
plt.legend()

plt.show()
```

```
Activation function: sigmoid | Final Train accuracy : 96.0 | Test accuracy: 92.5
Activation function: relu | Final Train accuracy : 74.0 | Test accuracy: 71.5
Weight initialization: HeNormal | Final Train accuracy : 95.75 | Test accuracy: 92.5
Weight initialization: Normal | Final Train accuracy : 98.5 | Test accuracy: 95.75
```