המחלקה להנדסת חשמל ואלקטרוניקה

מערכות לומדות ולמידה עמוקה (31245)

Lab 6 report

פרנסיס עבוד

Prepared for: Dr. Amer Adler

Date: 18/05/2025

In this lab we will train an MLP to classify items from 10 categories in the Fashion MNIST dataset, which includes 60,000 training images (28x28 pixels) and 10,000 test images:

Ex.1:

Train the given network three times, each with a different learning rate value:0.1,0.01,0.001. Write down the achieved test set accuracy for each of the learning rates. Which one is best?

Code:

```
# -*- coding: utf-8 -*-
"""
Created on 18/05/2025

@author: Francis
"""
# TensorFlow and tf.keras
import tensorflow as tf

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

GOOGLE_COLAB=False

if GOOGLE_COLAB:
 fashion_mnist = tf.keras.datasets.fashion_mnist
 (train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
else:
 npzfile = np.load('fashion_mnist_file.npz')
 train_images = npzfile['arr_0']
 train_labels = npzfile['arr_1']
 test_images = npzfile['arr_2']
 test_labels = npzfile['arr_3']


class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
```

```
        plt.xlabel(class_names[train_labels[i]])
    plt.show()

    model = tf.keras.Sequential()
    model.add(tf.keras.layers.Flatten(input_shape=(28, 28)))
    model.add(tf.keras.layers.Dense(512, activation='sigmoid'))
    model.add(tf.keras.layers.Dense(10))
    model.add(tf.keras.layers.Softmax())
    model.summary()
    GD=tf.keras.optimizers.SGD(learning_rate=0.1)
    model.compile(optimizer=GD,
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
                  metrics=['accuracy'])

    model.fit(train_images, train_labels, batch_size=32, epochs=15)

    test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

    print('\nTest accuracy:', test_acc)
    probability_model = tf.keras.Sequential([model,
                                             tf.keras.layers.Softmax()])

    predictions = probability_model.predict(test_images)

    predictions[0]

    np.argmax(predictions[0])

    test_labels[0]
```

Output:

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense (Dense)	(None, 512)	401,920
dense_1 (Dense)	(None, 10)	5,130
softmax (Softmax)	(None, 10)	0

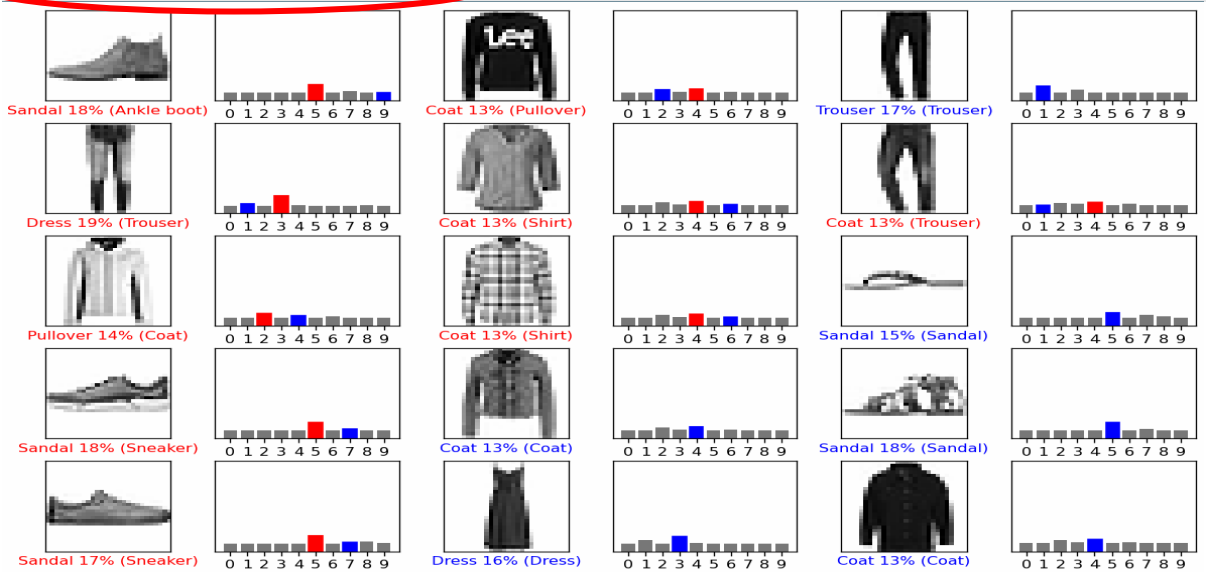
```
Total params: 407,050 (1.55 MB)
Trainable params: 407,050 (1.55 MB)
Non-trainable params: 0 (0.00 B)
```

For 0.1:

```
GD=tf.keras.optimizers.SGD(learning_rate=0.1)
model.compile(optimizer=GD,
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
              metrics=['accuracy'])
```

```
←[1m1875/1875←[0m ←[32m━━━━━━━━━━━━━━━━━━━━←[0m←[37m←[0m ←[1m20s←[0m 11ms/step - accuracy: 0.3621 - loss: 2.2042
313/313 - 2s - 6ms/step - accuracy: 0.4150 - loss: 1.6940

Test accuracy: 0.41499999165534973
```

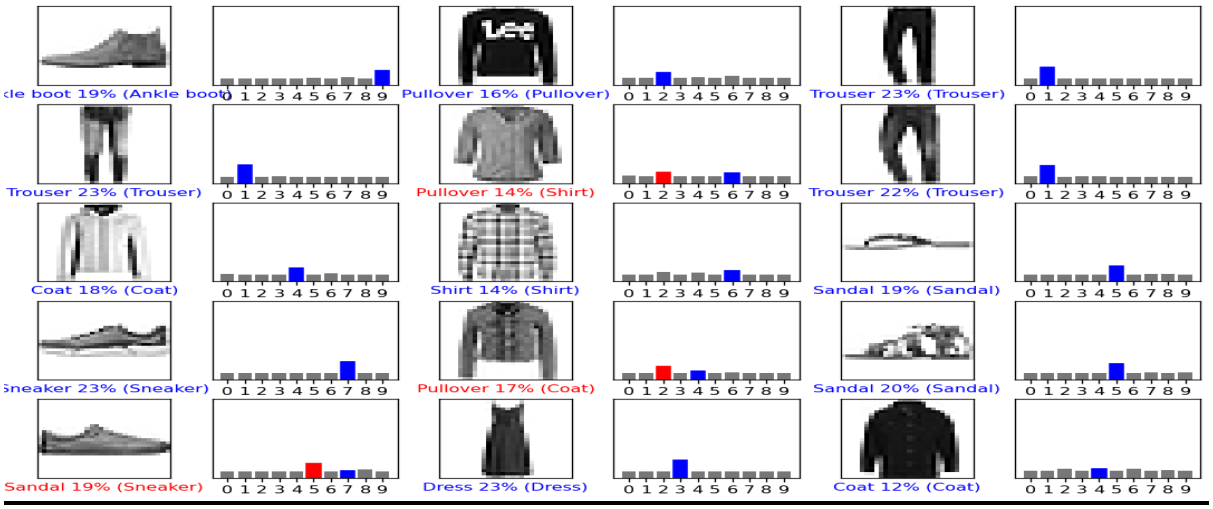


For 0.01:

```
GD=tf.keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=GD,
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
              metrics=['accuracy'])
```

```
313/313 - 4s - 12ms/step - accuracy: 0.7841 - loss: 0.5910

Test accuracy: 0.7840999960899353
```

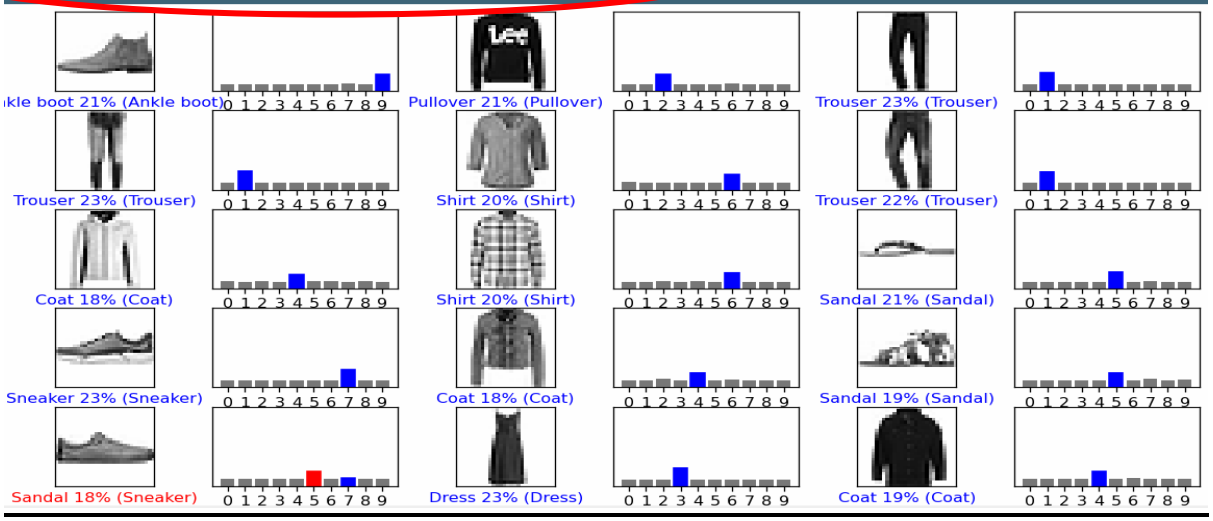


For 0.001:

```
GD=tf.keras.optimizers.SGD(learning_rate=0.001)
model.compile(optimizer=GD,
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
              metrics=['accuracy'])
```

```
313/313 - 2s - 7ms/step - accuracy: 0.8448 - loss: 0.4414

Test accuracy: 0.8447999954223633
```



Ex.2:

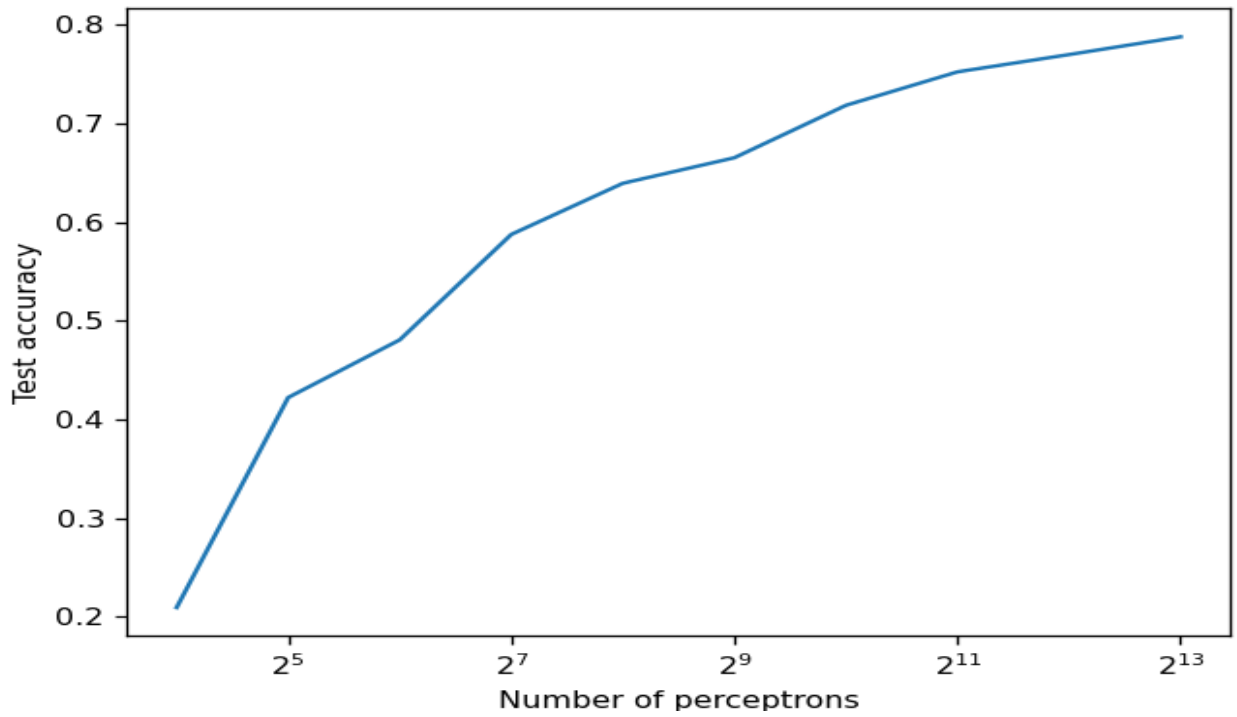
Use the best learning rate from section (1) and plot a graph of the test set accuracy as a function of the hidden layer size (number of perceptrons) in the range of 2^k, with k=4,5,6,7,8,9,10,11,12,13,14. Which value of 2^k provided the best results?

Code:

```
#ex2
hidden_sizes = [2**k for k in range(4, 14)]
test_accs = []
print(hidden_sizes)
for size in hidden_sizes:
    print(size)
    model = tf.keras.Sequential([
        tf.keras.layers.Flatten(input_shape=(28, 28)),
        tf.keras.layers.Dense(size, activation='sigmoid'),
        tf.keras.layers.Dense(size, activation='sigmoid'),
        tf.keras.layers.Dense(10)
    ])
    model.summary()
    GD = tf.keras.optimizers.SGD(learning_rate=0.001)
    model.compile(optimizer=GD,
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])
    model.fit(train_images, train_labels, batch_size=32, epochs=1)

    test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
    test_accs.append(test_acc)
    print('\nTest accuracy:', test_acc)
plt.plot(hidden_sizes, test_accs)
plt.xscale('log', base=2)
plt.xlabel('Number of perceptrons')
plt.ylabel('Test accuracy')
plt.show()
```

Output:



Ex.3:

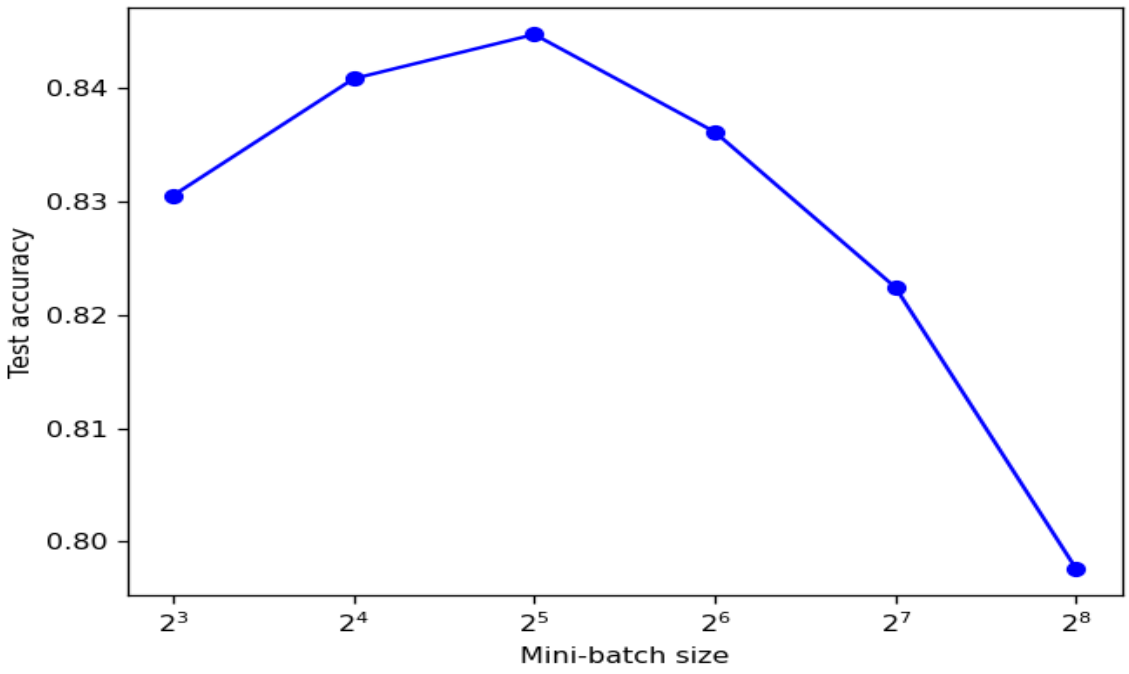
Use the best learning rate from section (1) and number of perceptrons = 512. Plot a graph of the test set accuracy as a function of the mini-batch size (number of samples per minibatch) in the range of 2^k, k=3,4,5,6,7,8. Which value of 2^k provided the best results?

Code:

```
#ex3
batch_sizes = [2**k for k in range(3, 9)]
test_accs = []
for batch in batch_sizes:
    model = tf.keras.Sequential([
        tf.keras.layers.Flatten(input_shape=(28, 28)),
        tf.keras.layers.Dense(512, activation='sigmoid'),
        tf.keras.layers.Dense(10)
    ])
    model.summary()
    GD = tf.keras.optimizers.SGD(learning_rate=0.001)
    model.compile(optimizer=GD,
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])

    model.fit(train_images, train_labels, batch_size=batch, epochs=15)
    test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
    test_accs.append(test_acc)
    print('\nTest accuracy:', test_acc)
plt.plot(batch_sizes, test_accs, 'bo-')
plt.xscale('log', base=2)
plt.xlabel('Mini-batch size')
plt.ylabel('Test accuracy')
plt.show()
```

Output:



Ex.4:

Use the best learning rate from section (1), number of perceptrons = 512, and mini-batch-size=32. Plot a graph of the test set accuracy as a function of the number of epochs, in the range of 10,11,…,29,30.

Code:

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(512, activation='sigmoid'),
    tf.keras.layers.Dense(10)
])
model.summary()
GD = tf.keras.optimizers.SGD(learning_rate=0.001)
model.compile(optimizer=GD,
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
test_accs = []
epochs = range(10, 31)
for epoch in epochs:
    print(epoch)
    model.fit(train_images, train_labels, batch_size=32, epochs=epoch)
    test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
    test_accs.append(test_acc)
    print('\nTest accuracy:', test_acc)

plt.plot(epochs, test_accs, 'bo-')
plt.xlabel('Epochs')
plt.ylabel('Test accuracy')
plt.show()
```

Output: