LISTA 3 – WSI

Piotr Maciejończyk

Zadanie 1.

```python
import tensorflow as tf

# Step 1: Load MNIST dataset
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Step 2: Preprocess the data
x_train, x_test = x_train / 255.0, x_test / 255.0

# Step 3: Define the neural network architecture
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

# Step 4: Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Step 5: Train the model
model.fit(x_train, y_train, epochs=5)

# Step 6: Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"Test accuracy: {test_acc}")
```

Opis procesu działania programu krok po kroku:

1. Pobranie zbioru danych z The MNIST Database.

2. Przeskalowanie wartości pikseli i ich normalizacja.

3. Utworzenie architektury sieci neuronowej.

4. Skompilowanie modelu.

5. Wytrenowanie sieci neuronowej.

6. Przetestowanie wyćwiczonego modelu na danych testowych.

Otrzymana dokładność testu:

```
Epoch 1/5
1875/1875 [==============================] - 24s 13ms/step - loss: 0.1423 - accuracy: 0.9568
Epoch 2/5
1875/1875 [==============================] - 23s 12ms/step - loss: 0.0463 - accuracy: 0.9860
Epoch 3/5
1875/1875 [==============================] - 24s 13ms/step - loss: 0.0323 - accuracy: 0.9896
Epoch 4/5
1875/1875 [==============================] - 23s 13ms/step - loss: 0.0239 - accuracy: 0.9924
Epoch 5/5
1875/1875 [==============================] - 24s 13ms/step - loss: 0.0171 - accuracy: 0.9947
313/313 [==============================] - 2s 4ms/step - loss: 0.0333 - accuracy: 0.9907
Test accuracy: 0.9907000064849854
```

Test accuracy: 0.9907000064849854

Zadanie 2.

Mój zbiór testowy składa się łącznie ze 100 obrazów 28 x 28 pikseli (po 10 egzemplarzy każdej cyfry). Cyfry rysowane były białym kolorem na czarnym tle. Mój zbiór utworzyłem w taki sposób:

```python
def gen_my_test_data():
    directory = 'TensorsImages'
    y = [i for i in range(10) for _ in range(10)]
    x = []
    for filename in os.scandir(directory):
        if filename.is_file():
            img = cv2.imread(filename.path, 0)
            pic = []
            for i in range(img.shape[0]): #traverses through height of the image
                row = []
                for j in range(img.shape[1]): #traverses through width of the image
                    row.append(img[i][j]/255)
                pic.append(row)
            x.append(pic)
    return np.array(x), np.asarray(y)


my_test_x, my_test_y = gen_my_test_data()
```

Sama architektura sieci neuronowej jest taka sama jak w Zadaniu 1. Na koniec działania programu obliczałem precyzję testu dla zbioru danych MNIST oraz dla mojego zbioru cyfr:

```python
# Step 6: Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"Standard Test accuracy: {test_acc}")

my_test_loss, my_test_acc = model.evaluate(my_test_x, my_test_y)
print(f"My Test accuracy: {my_test_acc}")
```

Przykładowe wywołanie programu:

```
Epoch 1/5
1875/1875 [==============================] - 26s 13ms/step - loss: 0.1425 - accuracy: 0.9563
Epoch 2/5
1875/1875 [==============================] - 25s 13ms/step - loss: 0.0476 - accuracy: 0.9851
Epoch 3/5
1875/1875 [==============================] - 24s 13ms/step - loss: 0.0327 - accuracy: 0.9892
Epoch 4/5
1875/1875 [==============================] - 24s 13ms/step - loss: 0.0240 - accuracy: 0.9921
Epoch 5/5
1875/1875 [==============================] - 24s 13ms/step - loss: 0.0191 - accuracy: 0.9937
313/313 [==============================] - 2s 4ms/step - loss: 0.0326 - accuracy: 0.9902
Standard Test accuracy: 0.9901999831199646
4/4 [==============================] - 0s 4ms/step - loss: 0.3591 - accuracy: 0.8700
My Test accuracy: 0.8700000047683716
```

Współczynnik rozpoznawalności dla zbioru danych MNIST:

```
Standard Test accuracy: 0.9901999831199646
```

Współczynnik rozpoznawalności dla mojego zbioru danych:

```
My Test accuracy: 0.8700000047683716
```

Zatem dla mojego zbioru danych precyzja rozpoznawalności danych jest bardzo wysoka, choć nie aż tak duża jak dla zbioru danych MNIST. Oprócz tego, dla moich danych wartość funkcji kosztu jest ponad 10-krotnie większa. Obniżony współczynnik rozpoznawalności oraz podwyższona wartość funkcji straty mogą wynikać np. z mojego sposobu zapisu cyfry „1” oraz „7”:

W zbiorze danych MNIST (na podstawie, którego wyćwiczyłem sieć neuronową) cyfra „1” pisana jest zazwyczaj jako pionowa kreska, a cyfra „7” występuje głównie bez poziomej linii. Innym powodem może być również słaba rozdzielczość moich narysowanych cyfr.