НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра технічної кібернетики

Звіт до комп’ютерного практикуму 3 з дисципліни:
“**Програмні засоби проектування та реалізації нейромережевих систем**”

Виконав
ІП-01 Черпак А.В.

Перевірив:
Шимкович В.М.

Київ – 2023

Комп’ютерний практикум 3

Тема: Нейронні мережі прямого розповсюдження для розпізнавання зображень

Завдання:

Написати програму що реалізує нейронну мережу прямого розповсюдження для розпізнавання рукописних цифр.

Виконання:

Побудова та компіляція моделі:

```
import tensorflow as tf
from keras.engine.sequential import Sequential
from keras.optimizers.schedules.learning_rate_schedule import ExponentialDecay


def get_model(neurons_in_hidden_layers: list[int]) -> Sequential:
    model = tf.keras.Sequential()
    model.add(tf.keras.Input(shape=(28, 28)))
    model.add(tf.keras.layers.Flatten())
    for neurons in neurons_in_hidden_layers:
        model.add(tf.keras.layers.Dense(neurons, activation='relu'))
    model.add(tf.keras.layers.Dense(10, activation='softmax'))
    return model


def get_learning_rate(train_data_length: int, epochs: int, batch_size: int) ->
ExponentialDecay:
    initial_learning_rate = 10 ** (-3)
    final_learning_rate = 10 ** (-7)
    learning_rate_decay_factor = (final_learning_rate / initial_learning_rate) **
(1 / epochs)
    steps_per_epoch = int(train_data_length / batch_size)
    learning_rate = ExponentialDecay(
        initial_learning_rate=initial_learning_rate,
        decay_steps=steps_per_epoch,
        decay_rate=learning_rate_decay_factor
    )
    return learning_rate


def compile_model(model, lr):

    model.compile(loss='sparse_categorical_crossentropy', metrics=['accuracy'],
                  optimizer=tf.keras.optimizers.Adam(learning_rate=lr))
```

Тренування моделі:

```
import tensorflow as tf
from NeuralNetworkModel import get_model, get_learning_rate, compile_model


def train_model(hidden_neurons, x_train, y_train, x_test, y_test, epochs,
batch_size):
    mdl = get_model(hidden_neurons)
    lr = get_learning_rate(len(x_train), epochs, batch_size)
    compile_model(mdl, lr)
    mdl.summary()
    mdl.fit(x_train, y_train, epochs=epochs, batch_size=batch_size,
validation_data=(x_test, y_test), verbose=1)
    return mdl

if __name__ == '__main__':
    (x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

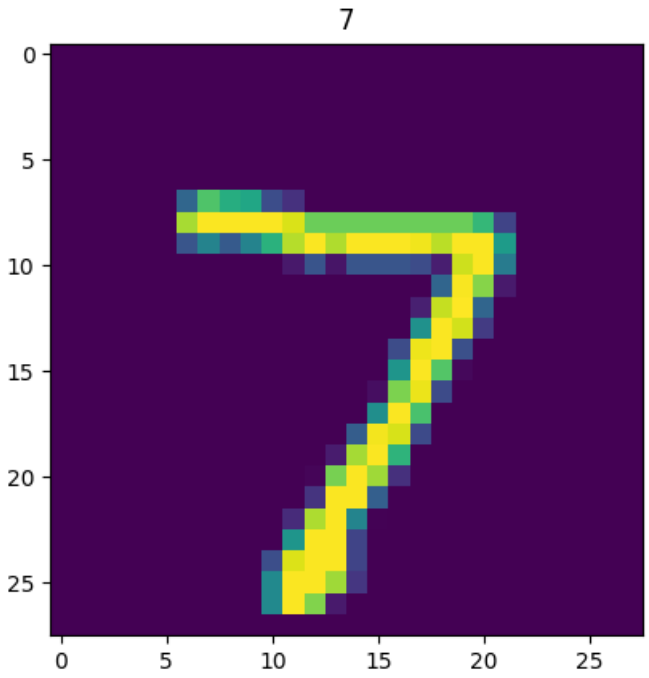
```
    model = train_model([75, 75, 75], x_train, y_train, x_test, y_test, 30, 120)
    model.evaluate(x_test, y_test)
    model.save('my_model.h5')
```

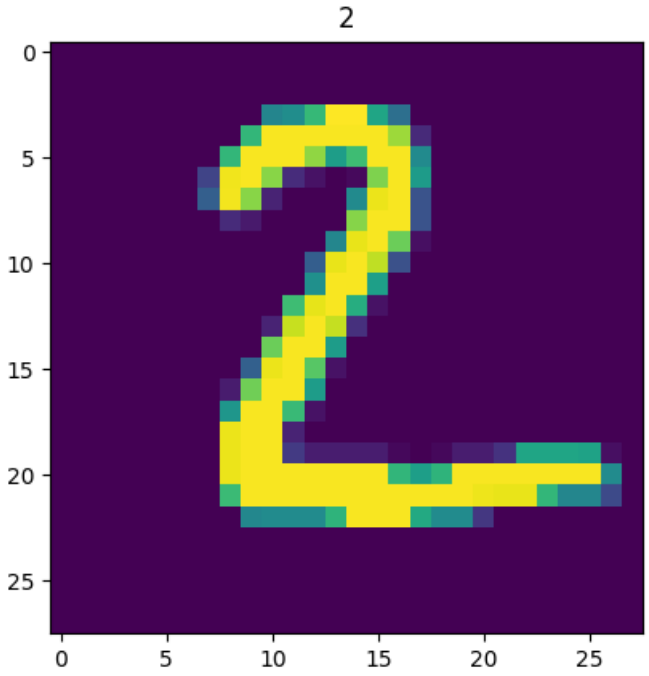
Тестування моделі:

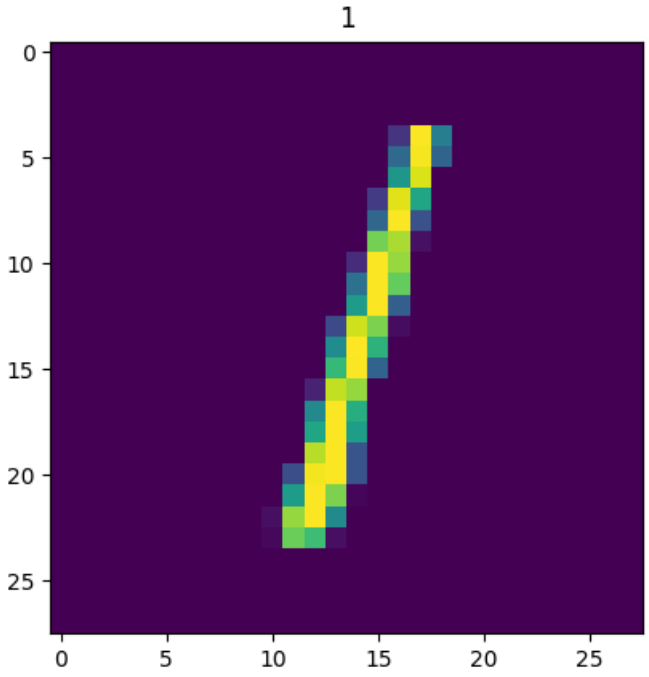
```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np


def predict_model(mdl, x, y):
    plt.imshow(x)
    predictions = mdl.predict(np.expand_dims(x, axis=0), verbose=0)
    print('Correct: ', y)
    print('Predicted: ', predictions.argmax())
    plt.title(str(predictions.argmax()))
    plt.show()


if __name__ == '__main__':
    (x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
    model = tf.keras.models.load_model('my_model.h5')
    for (x, y) in zip(x_test[:20], y_test[:20]):
        predict_model(model, x, y)
```

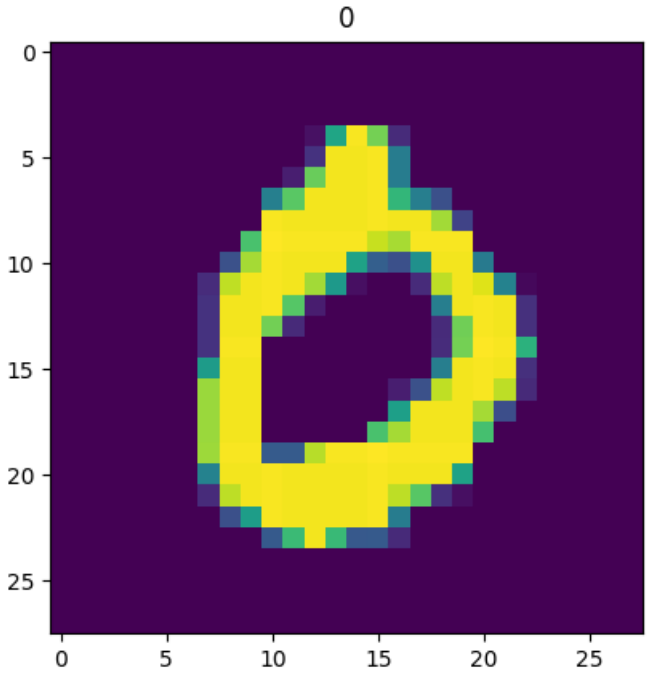
Результати тренування:

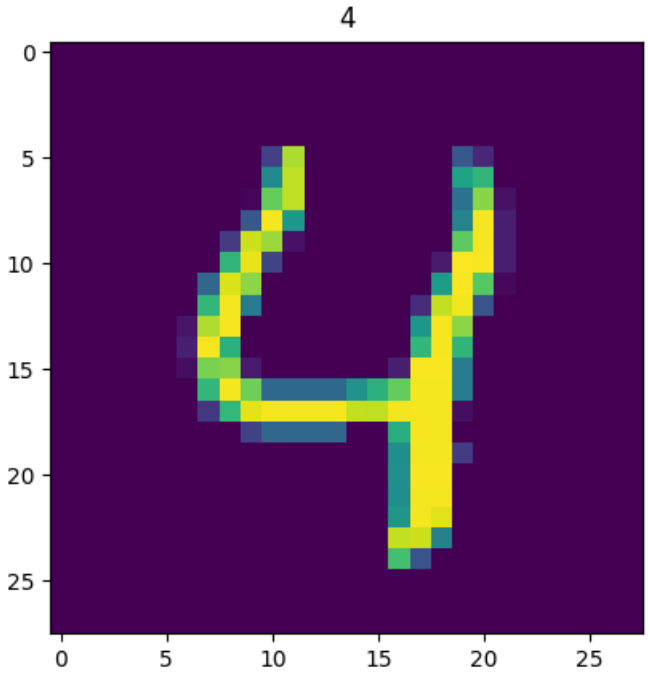
Результати тестування:

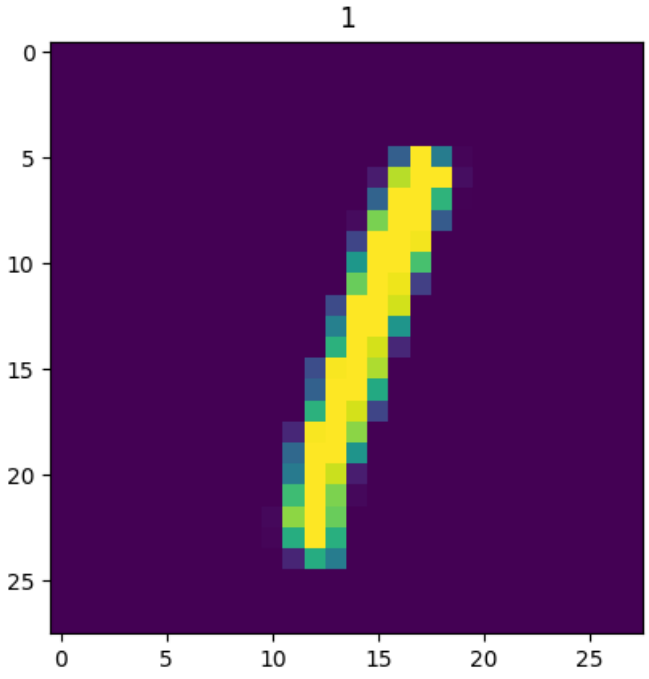


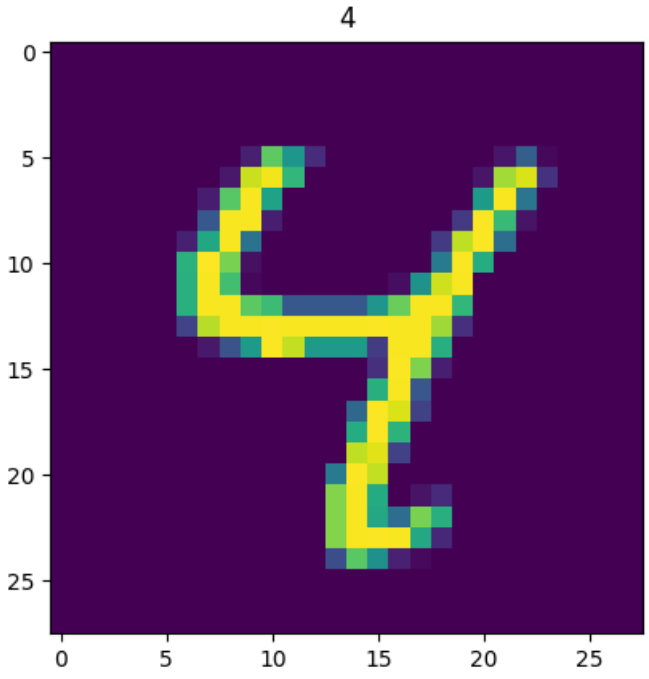

2
0
5
10
15
20
25
0
5
10
15
20
25

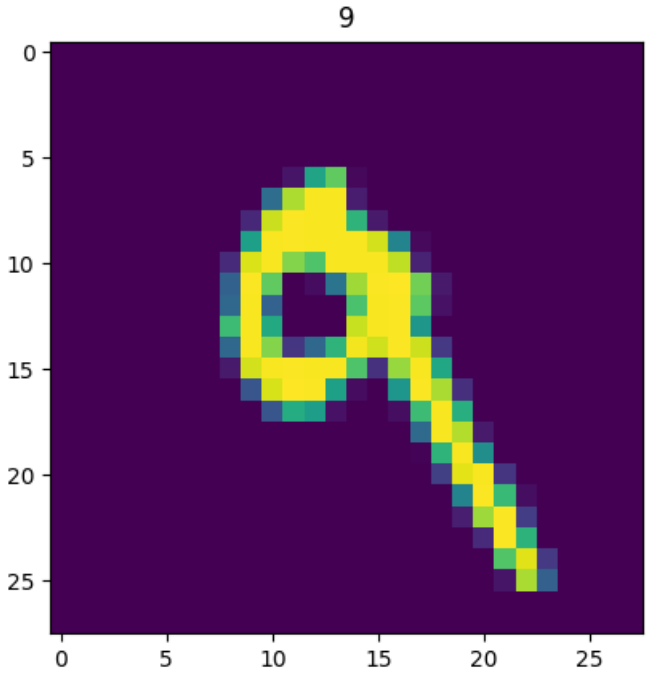

1
0
5
10
15
20
25
0
5
10
15
20
25

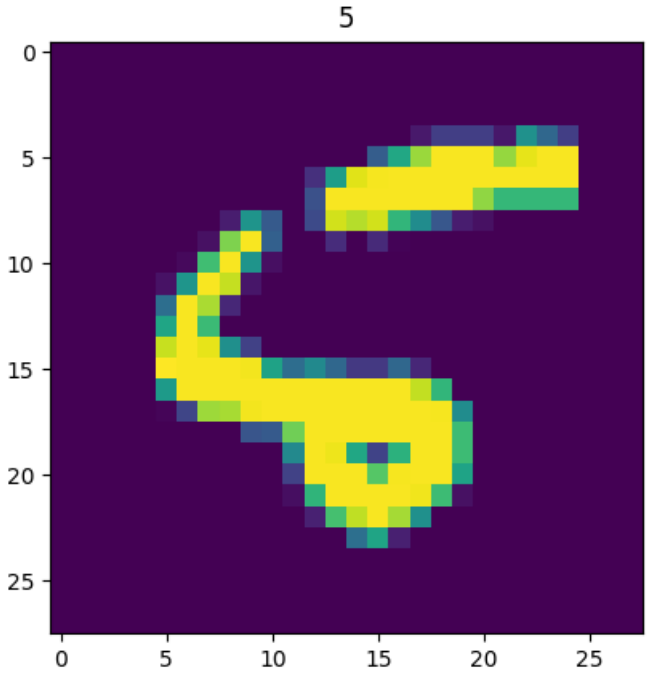

0
0
5
10
15
20
25
0
5
10
15
20
25

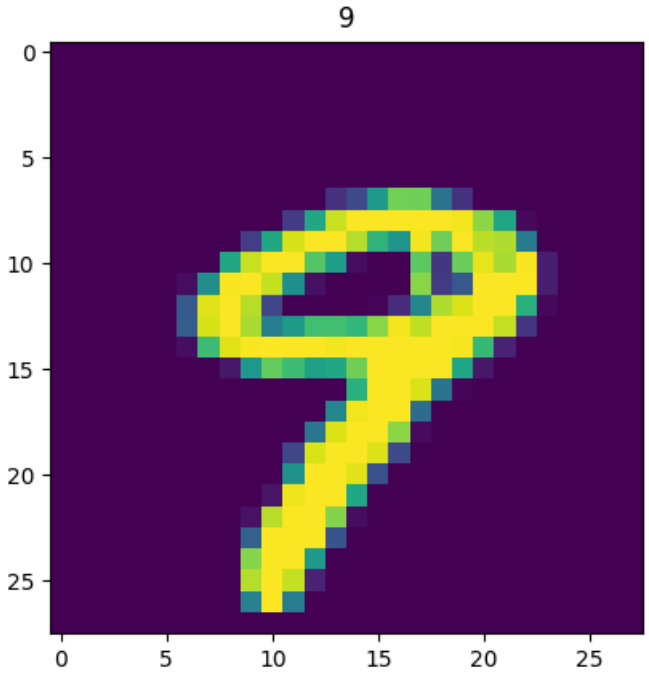

4
0
5
10
15
20
25
0
5
10
15
20
25

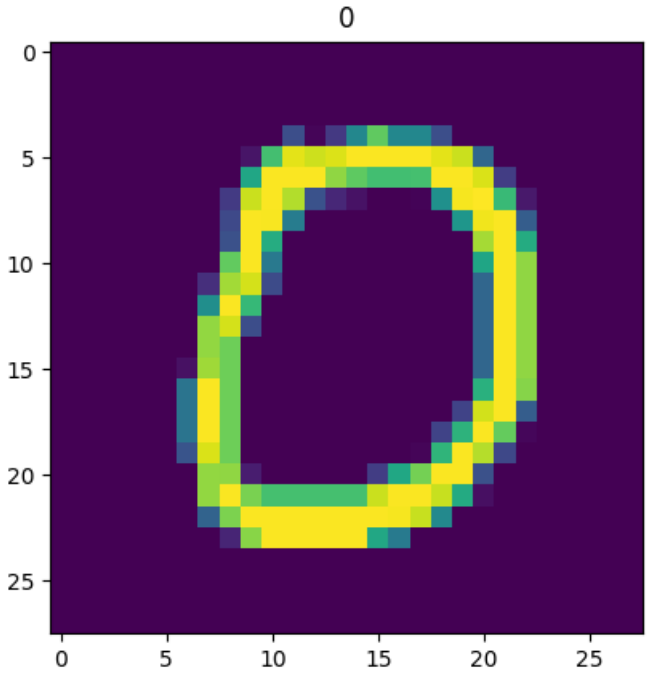

1
0
5
10
15
20
25
0
5
10
15
20
25

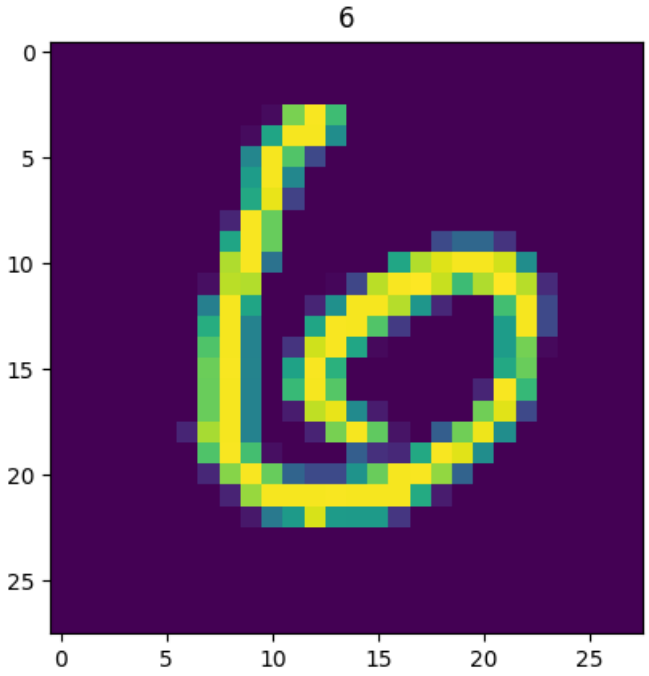

4
0
5
10
15
20
25
0
5
10
15
20
25

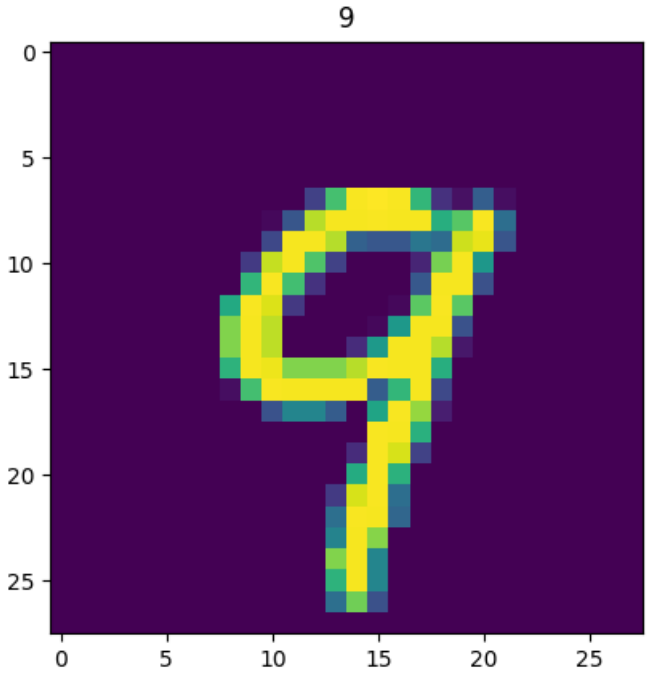

9
0
5
10
15
20
25
0
5
10
15
20
25

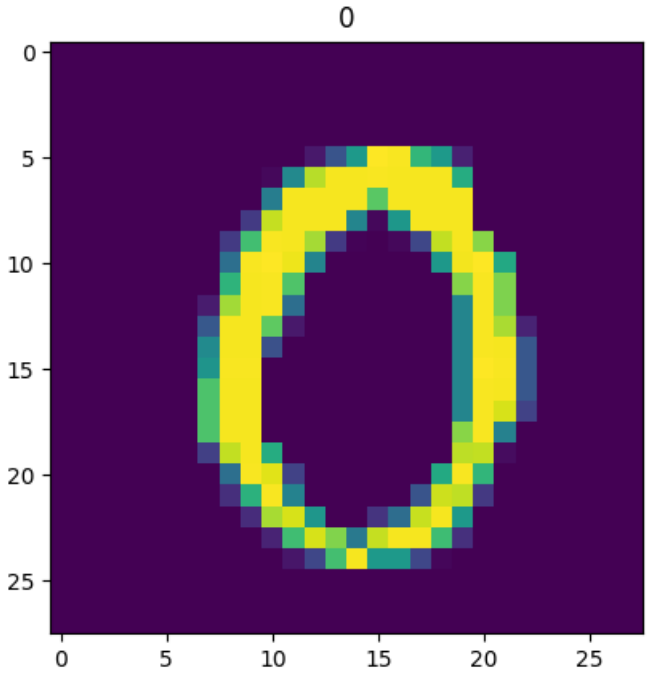

5
0
5
10
15
20
25
0
5
10
15
20
25

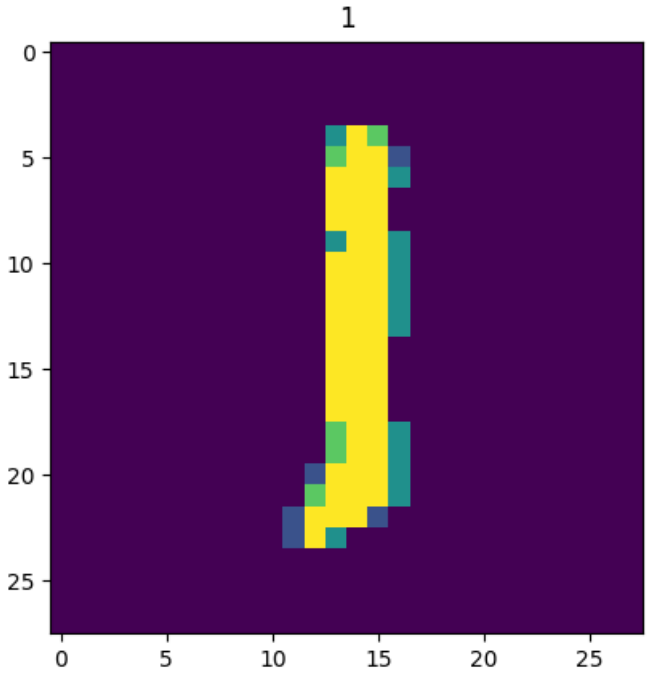

9
0
5
10
15
20
25

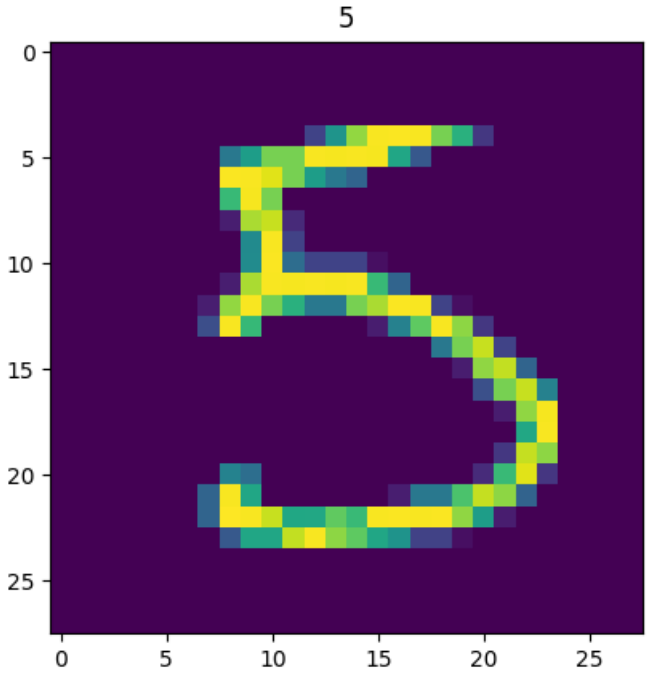

0
5
10
15
20
25

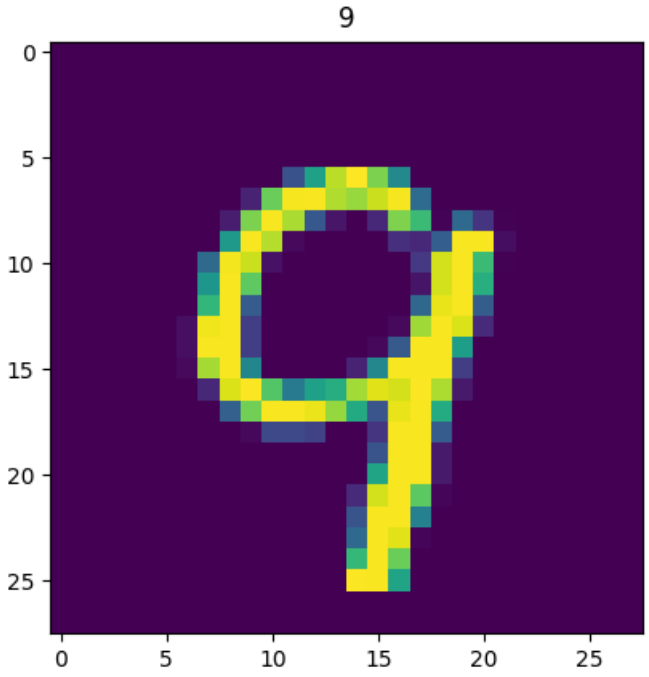

6
0
5
10
15
20
25
0
5
10
15
20
25

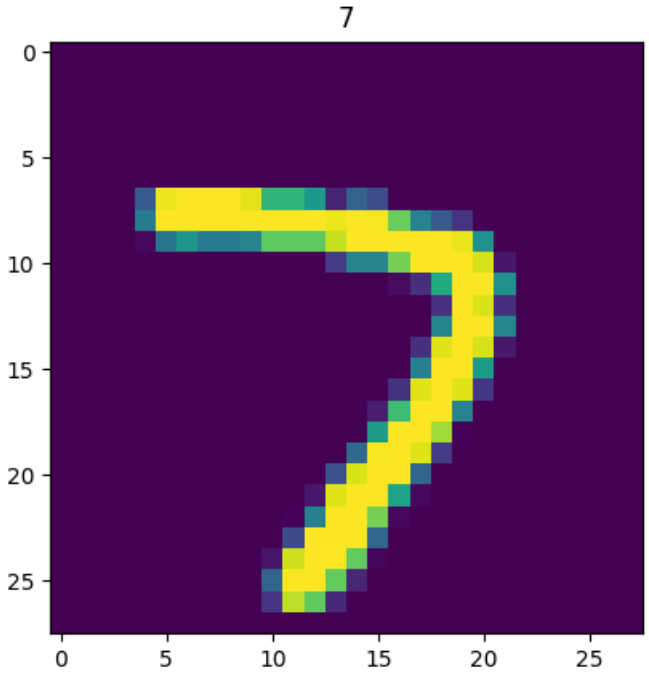

9
0
5
10
15
20
25
0
5
10
15
20
25

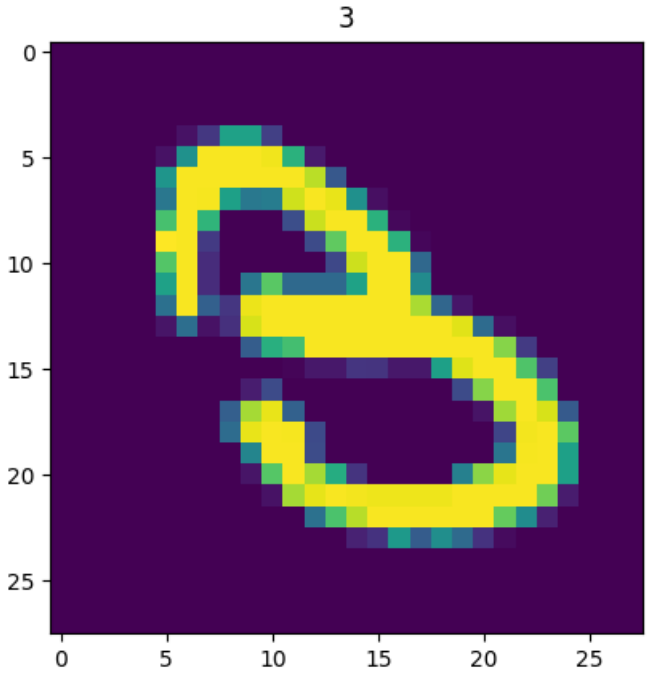

0
0
5
10
15
20
25
0
5
10
15
20
25


1
0
5
10
15
20
25
0
5
10
15
20
25


5
0
5
10
15
20
25
0
5
10
15
20
25


9
0
5
10
15
20
25
0
5
10
15
20
25





Висновок:

Під час виконання комп’ютерного практикуму ми реалізували нейромережі трьох різних типів зі змінною кількістю шарів та нейронів у них. Потім ці нейромережі були протестовані шляхом моделювання функції двох змінних $f(x, y) = x^2+y^2$. Найкращі результати показала мережа cascade-forward-backprop з одним прихованим шаром у 20 нейронів. Втім, варто визнати, що з кожним запуском результати дуже відрізняються, а інколи певна мережа може взагалі відмовитися тренуватися.