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:

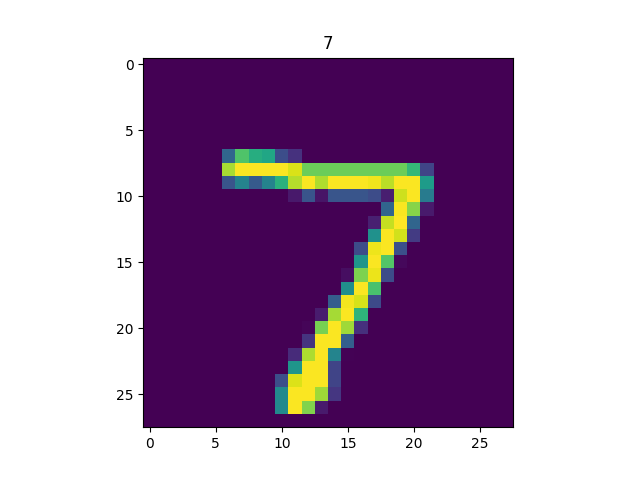
import tensorflow as tf  
from NeuralNetworkModel import get\_model, get\_learning\_rate, compile\_model  
  
  
def train\_model(hidden\_neurons, x\_train, y\_train, x\_test, y\_test, epochs, batch\_size):  
 mdl = get\_model(hidden\_neurons)  
 lr = get\_learning\_rate(len(x\_train), epochs, batch\_size)  
 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')

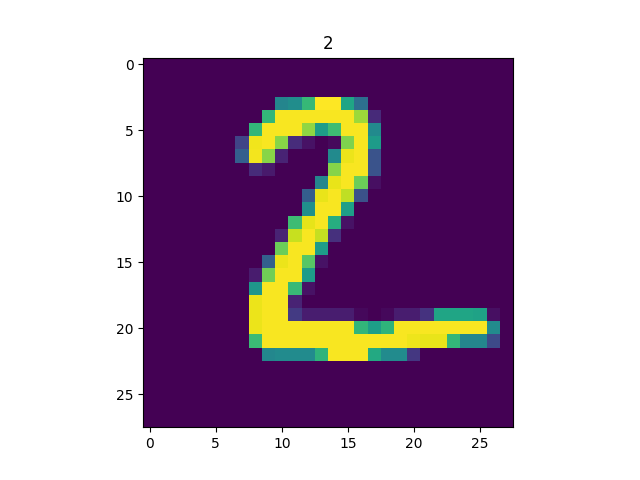
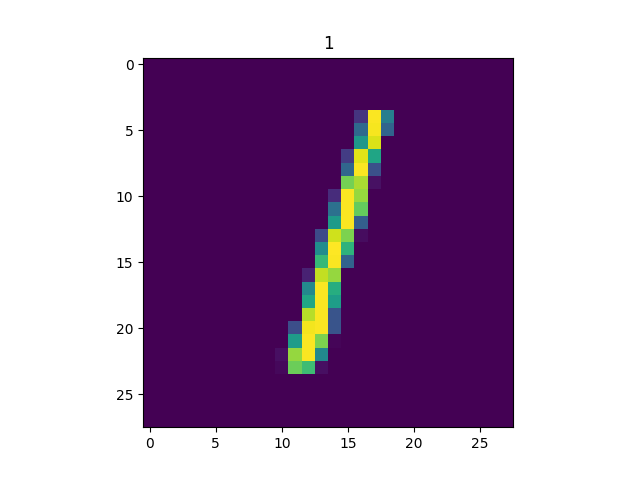
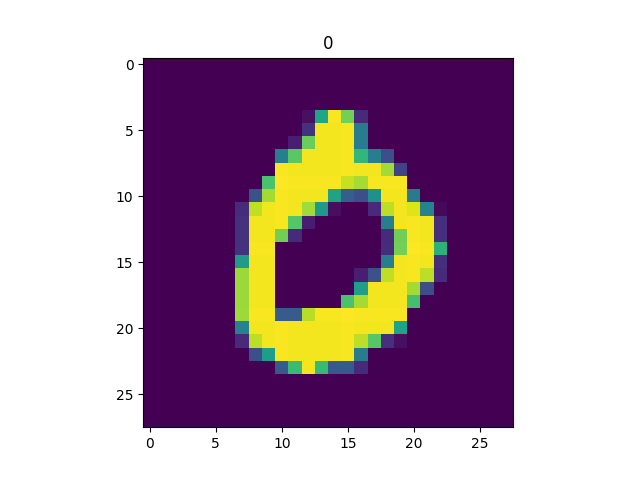
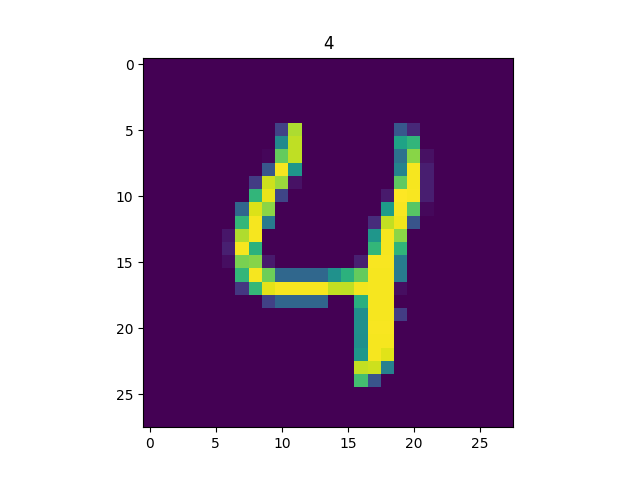
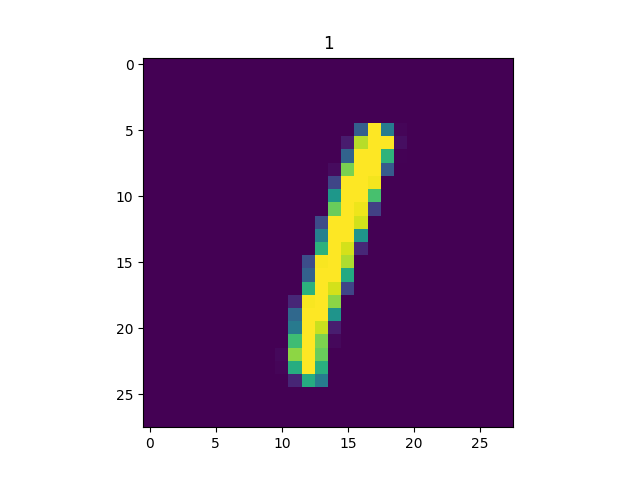
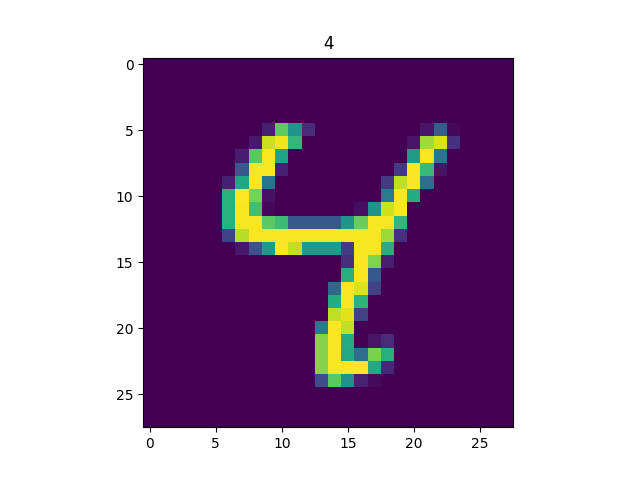
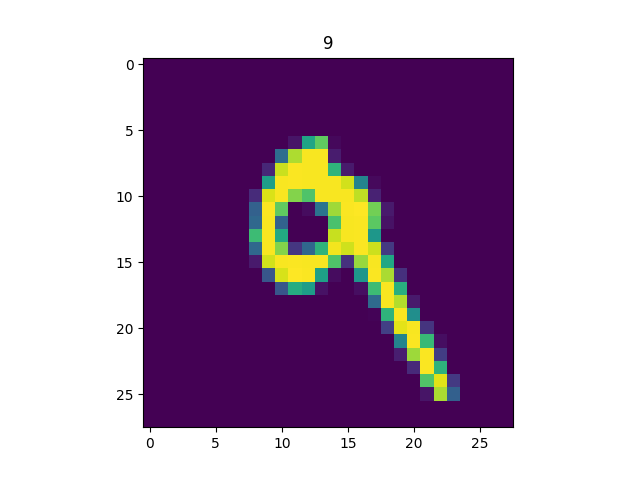
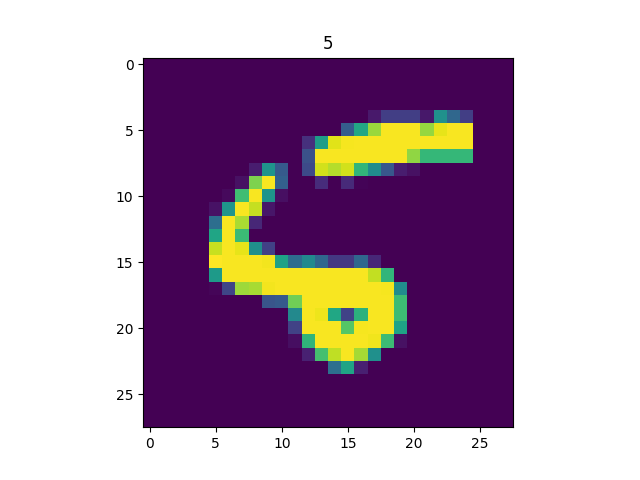
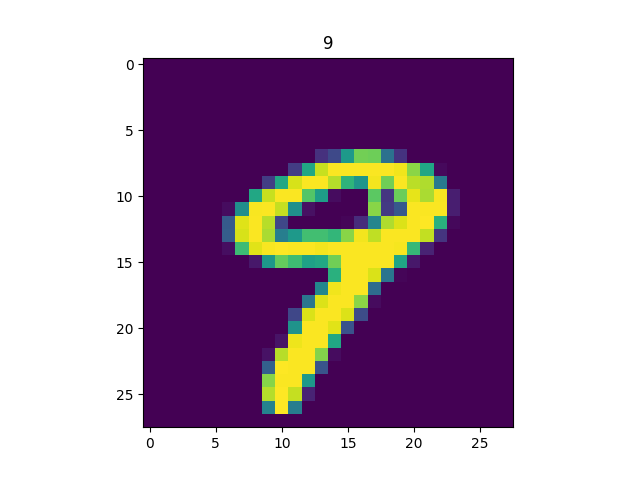
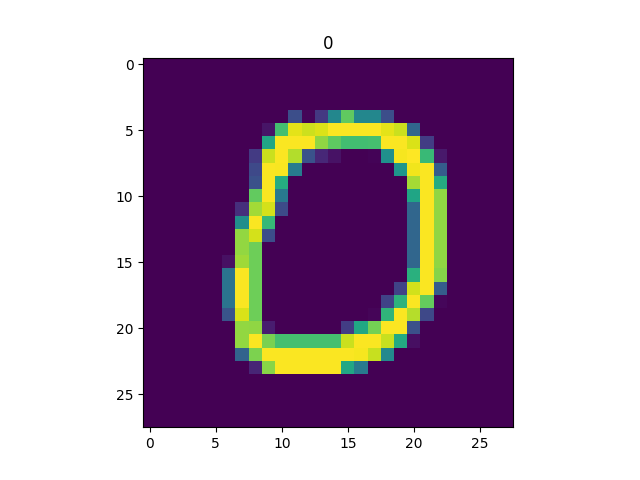
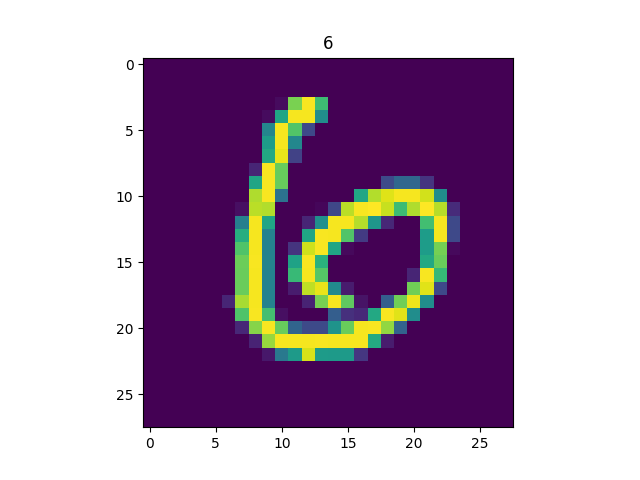
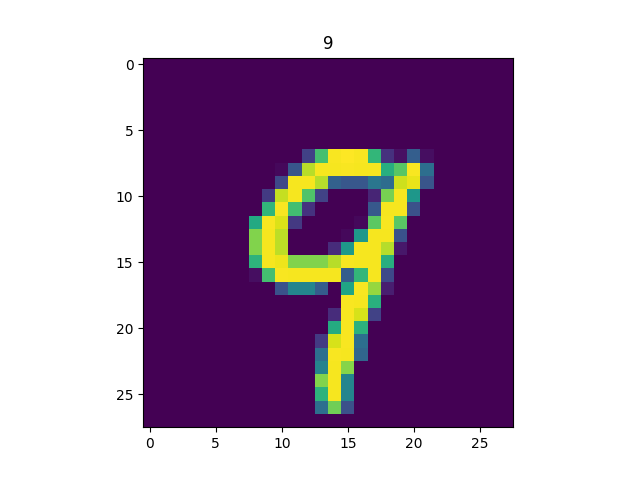
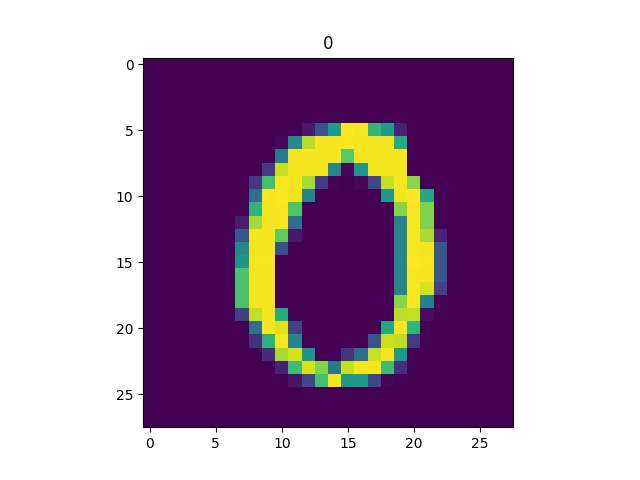
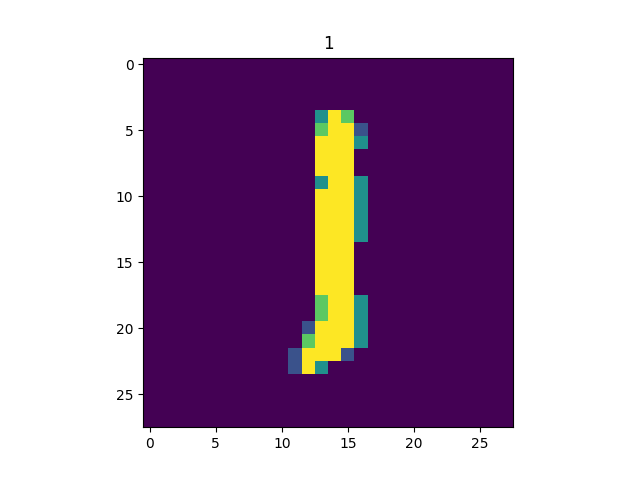
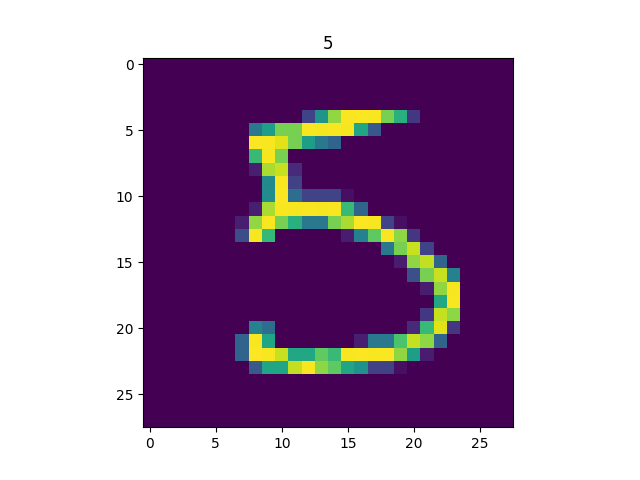
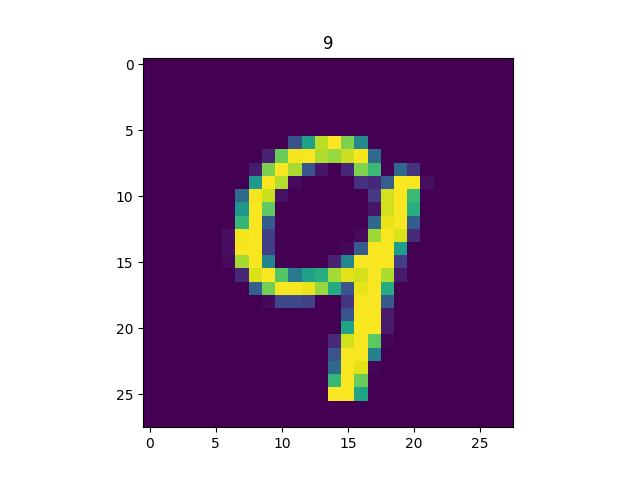
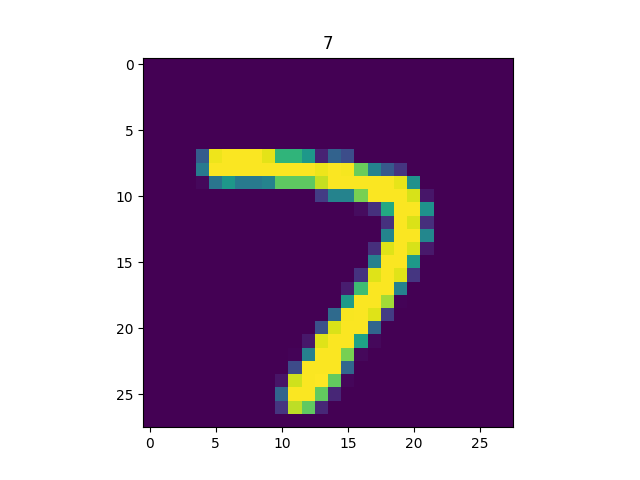
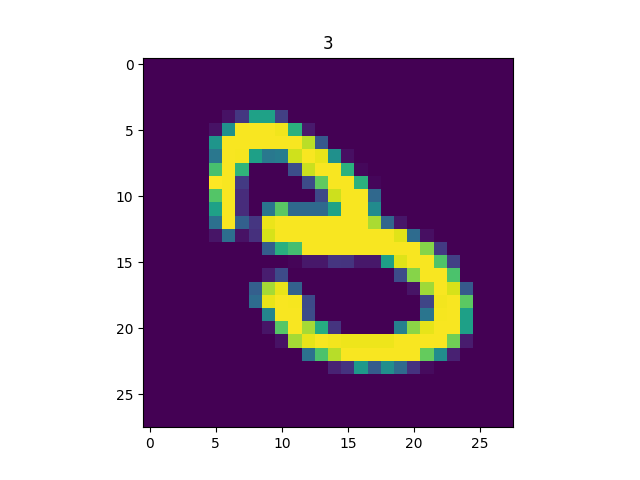
Тестування моделі:

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:

****

**** ****                

**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) = x2+y2. 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.