Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №3**

з дисципліни «Програмні засоби проектування та реалізаціїї нейромережевих систем»

Тема: " Нейронної мережі прямого розповсюдження для розпізнавання зображення "

Виконав: Перевірив:

студент групи ІП-93 Шимкович Володимир

Домінський Валентин Миколайович

Олексійович

Київ 2022

Зміст:

[Мета: 3](#_Toc103010390)

[Вихідний код 3](#_Toc103010391)

[Результат роботи: 4](#_Toc103010392)

[Висновки: 4](#_Toc103010393)

## Мета:

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

## Вихідний код

import tensorflow as tf

import keras

import matplotlib.pyplot as plt

import cv2

image\_size = 28

training\_text = "Training -"

test\_text = "Testing -"

relu\_activation = 'relu'

softmax\_activation = 'softmax'

optimizer\_type = 'rmsprop'

loss\_type = 'categorical\_crossentropy'

metrics\_type = 'accuracy'

batch\_size = 256

epochs = 20

(x\_train\_origin, y\_train\_origin), (x\_test\_origin, y\_test\_origin) = tf.keras.datasets.mnist.load\_data()

print(training\_text, len(x\_train\_origin))

print(test\_text, len(x\_test\_origin))

print(y\_train\_origin[1])

plt.imshow(x\_train\_origin[1])

x\_train = x\_train\_origin.reshape((len(x\_train\_origin), image\_size \* image\_size))

x\_test = x\_test\_origin.reshape((len(x\_test\_origin), image\_size \* image\_size))

# Normalizes images: `uint8` -> `float32`

# TFDS provide images of type tf.uint8, while the model expects

# tf.float32. Therefore, you need to normalize images

x\_train = x\_train.astype('float32') / 255

x\_test = x\_test.astype('float32') / 255

# vector which has integers that represent different categories,

# can be converted into a numpy array (or) a matrix which has binary

# values and has columns equal to the number of categories in the data

y\_train = tf.keras.utils.to\_categorical(y\_train\_origin)

y\_test = tf.keras.utils.to\_categorical(y\_test\_origin)

model = keras.models.Sequential([

keras.layers.Dense(512, activation = relu\_activation, input\_shape = (image\_size \* image\_size,)),

keras.layers.Dense(10, activation = softmax\_activation)

])

model.compile(optimizer = optimizer\_type, loss = loss\_type, metrics = [metrics\_type])

model.fit(x\_train, y\_train, epochs = epochs, batch\_size = batch\_size)

test\_loss, test\_acc = model.evaluate(x\_test, y\_test)

def get\_images(image\_name):

image = 255-cv2.imread(image\_name, 0)

image\_small = cv2.resize(image, (image\_size, image\_size))

image\_to\_predict = image\_small.reshape((1, image\_size \* image\_size))

return (image\_small, image\_to\_predict)

def predict(image\_to\_predict):

pred = model.predict(image\_to\_predict)[0]

for x in range(10):

if (pred[x] == 1.0):

return x

x = int(x) + 1

(first\_image, image\_to\_predict) = get\_images('3.png')

first\_image\_prediction = predict(image\_to\_predict)

(second\_image, image\_to\_predict) = get\_images('2.png')

second\_image\_prediction = predict(image\_to\_predict)

(third\_image, image\_to\_predict) = get\_images('0.png')

third\_image\_prediction = predict(image\_to\_predict)

# the way We want to put Our image in model

fig = plt.figure()

fig.set\_size\_inches(100, 100)

imgplot = plt.imshow(image\_to\_predict)

fig = plt.figure()

ax = fig.add\_subplot(1, 3, 1)

imgplot = plt.imshow(third\_image)

ax.set\_title(third\_image\_prediction)

ax = fig.add\_subplot(1, 3, 2)

imgplot = plt.imshow(second\_image)

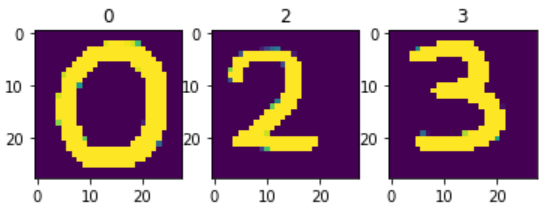
ax.set\_title(second\_image\_prediction)

ax = fig.add\_subplot(1, 3, 3)

imgplot = plt.imshow(first\_image)

ax.set\_title(first\_image\_prediction)

## Результат роботи:



## Висновки:

Я дізнався більше інформації про feedforward neural network (FFD) мережі - немає з'єднань зворотного зв'язку, в яких виходи моделі повертаються в саму себе