Котолевский Максим Николаевич, группа 19

Лабораторная работа № 1

**Вариант № 1**

**Цель работы**

Необходимо реализовать классификацию черно-белых изображений, которые представляют из себя рукописные цифры. Эти рукописные цифры представлены изображениями размерностью 28х28 пикселей по 10-ти категориям.

**Задание**

1. Разработать ПО для реализации задачи распознавания рукописных цифр.
2. Создать примеры, показывающую работу сети (сеть распознает верно).
3. Создать примеры, показывающую работу сети (сеть распознает не верно).

**Код программы (внесённые изменения в шаблон кода выделены)**

import cv2

import numpy as np

from keras.datasets import mnist

(train\_images, train\_labels), (test\_images, test\_labels) = mnist.load\_data()

train\_images.shape  
len(train\_labels)  
train\_labels  
test\_images.shape  
len(test\_labels)  
test\_labels

from keras import models

from keras import layers

network = models.Sequential() //последовательная модель данных

network.add(layers.Dense(512, activation='relu', input\_shape=(28 \* 28,)))

// первый слой, модуль обработки данных

network.add(layers.Dense(10, activation='softmax')) //второй слой

network.compile(optimizer='rmsprop',

 loss='categorical\_crossentropy',

 metrics=['accuracy'])

//компилируем сеть (оптимизатор, функция потерь и метрика)

train\_images = train\_images.reshape((60000, 28 \* 28))

train\_images = train\_images.astype('float32') / 255

test\_images = test\_images.reshape((10000, 28 \* 28))

test\_images = test\_images.astype('float32') / 255

//масштабируем в нужный диапазон от 0 до 1

from tensorflow.keras.utils import to\_categorical

train\_labels = to\_categorical(train\_labels)

test\_labels = to\_categorical(test\_labels)

//кодируем метки категорий

network.fit(train\_images, train\_labels, epochs=5, batch\_size=128)

test\_loss, test\_acc = network.evaluate(test\_images, test\_labels)

print('test\_acc:', test\_acc)

//подаем обучающие данные

digit = train\_images[100]

import matplotlib.pyplot as plt

digit = digit.reshape(28, 28)

plt.imshow(digit, cmap=plt.cm.binary)

plt.show()

pred = network.predict(digit.reshape(1,28\*28))

print('Digit: ', pred.argmax())

digit = cv2.imread('1\_1.png', cv2.IMREAD\_GRAYSCALE)

digit = cv2.bitwise\_not(digit)

import matplotlib.pyplot as plt

digit = digit.reshape(28, 28)

plt.imshow(digit, cmap=plt.cm.binary)

plt.show()

pred = network.predict(digit.reshape(1,28\*28))

print('Digit: ', pred.argmax())

digit = cv2.imread('1\_2.png', cv2.IMREAD\_GRAYSCALE)

digit = cv2.bitwise\_not(digit)

import matplotlib.pyplot as plt

digit = digit.reshape(28, 28)

plt.imshow(digit, cmap=plt.cm.binary)

plt.show()

pred = network.predict(digit.reshape(1,28\*28))

print('Digit: ', pred.argmax())

digit = cv2.imread('3\_2.png', cv2.IMREAD\_GRAYSCALE)

digit = cv2.bitwise\_not(digit)

import matplotlib.pyplot as plt

digit = digit.reshape(28, 28)

plt.imshow(digit, cmap=plt.cm.binary)

plt.show()

pred = network.predict(digit.reshape(1,28\*28))

print('Digit: ', pred.argmax())

digit = cv2.imread('3\_1.png', cv2.IMREAD\_GRAYSCALE)

digit = cv2.bitwise\_not(digit)

import matplotlib.pyplot as plt

digit = digit.reshape(28, 28)

plt.imshow(digit, cmap=plt.cm.binary)

plt.show()

pred = network.predict(digit.reshape(1,28\*28))

print('Digit: ', pred.argmax())

digit = cv2.imread('4\_1.png', cv2.IMREAD\_GRAYSCALE)

digit = cv2.bitwise\_not(digit)

import matplotlib.pyplot as plt

digit = digit.reshape(28, 28)

plt.imshow(digit, cmap=plt.cm.binary)

plt.show()

pred = network.predict(digit.reshape(1,28\*28))

print('Digit: ', pred.argmax())

digit = cv2.imread('4\_2.png', cv2.IMREAD\_GRAYSCALE)

digit = cv2.bitwise\_not(digit)

import matplotlib.pyplot as plt

digit = digit.reshape(28, 28)

plt.imshow(digit, cmap=plt.cm.binary)

plt.show()

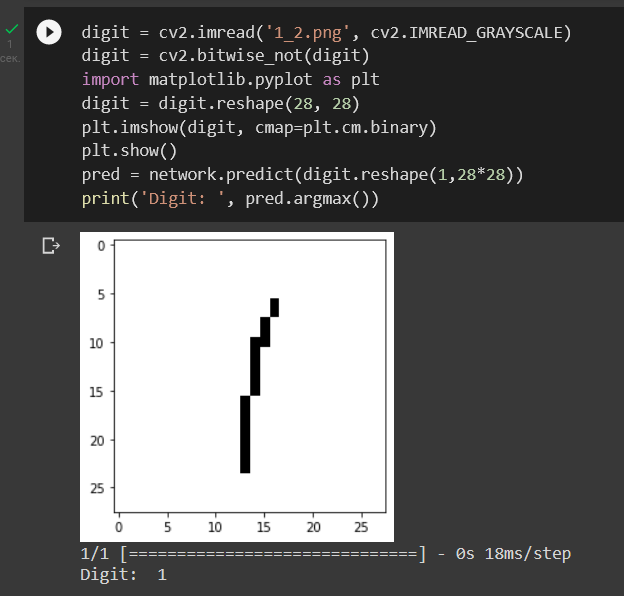
pred = network.predict(digit.reshape(1,28\*28))

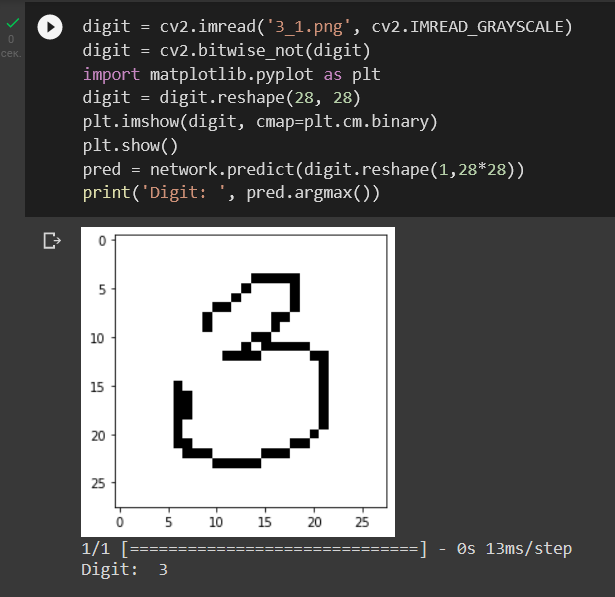
print('Digit: ', pred.argmax())

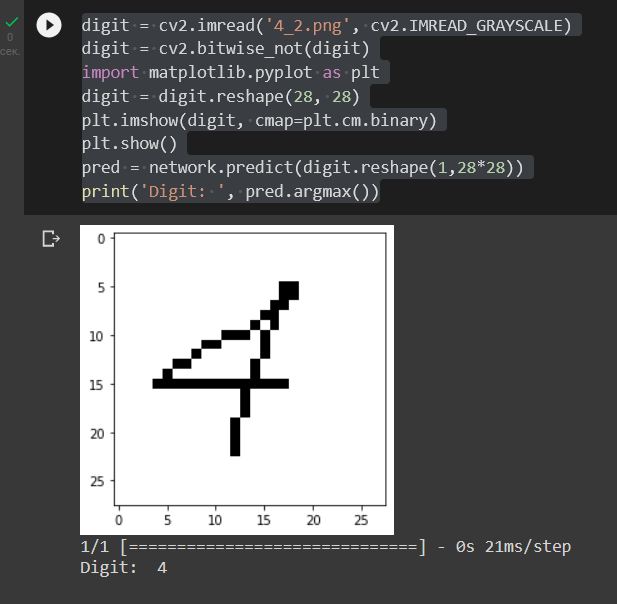
**Результаты выполнения задания**

Было разработано ПО для реализации задачи распознования рукописных цифр. Были созданы примеры, показывающие работу сети.

Сеть распознает верно:







Сеть распознает неверно:

