Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧЕТ**

по лабораторной работе №2

по дисциплине «Программные средства кибертехнических систем»

на тему «Работа с нейронной сетью»

Выполнили:

студенты группы 22ВВП1

Хоссейни Нежад С.А.С.М.

Амиров И.Р.

Сергунов М. Р.

Приняли:

Зинкин С.А.

Карамышева Н.С.

Пенза 2025

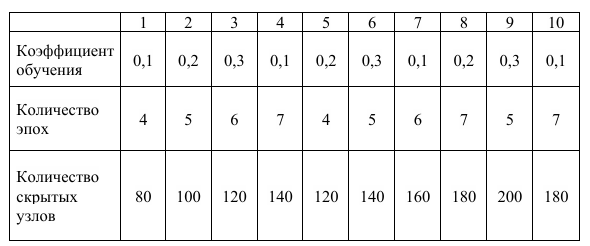
**Цель работы**

изучение программы по распознаванию рукописных цифр при помощи трехслойной нейронной сети.

**Задание**

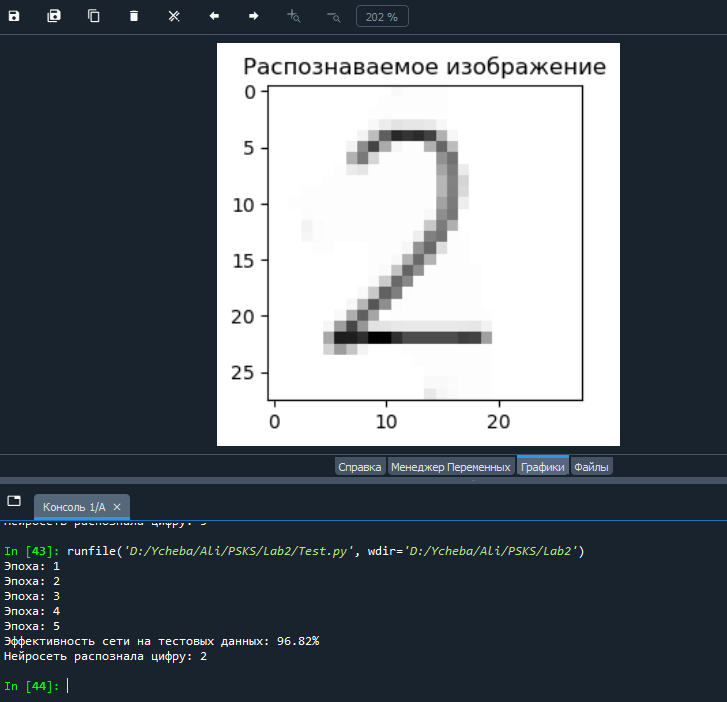
Изучить теоретическую информацию и программу по распознаванию рукописных цифр на языке Python. Запустить программу в среде Spyder(Anaconda), проанализировать полученные результаты. Затем внести правки в код в соответствии с вариантом и сравнить результаты с полученными ранее. Опросить сеть с использованием тестовой базы данных (собственноручно написанных цифр).

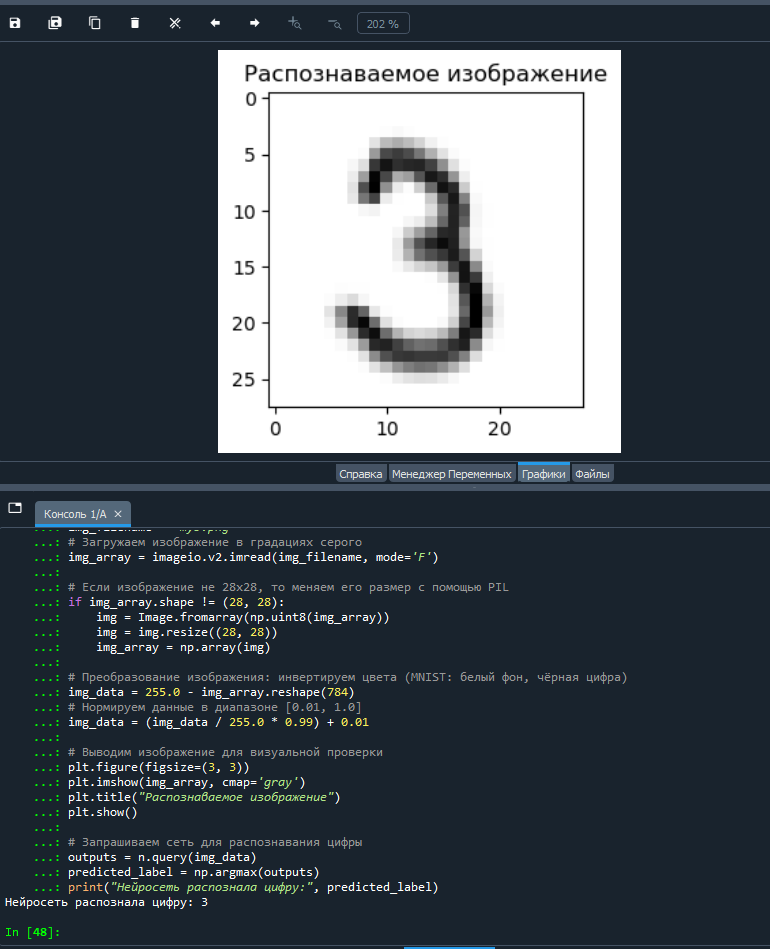
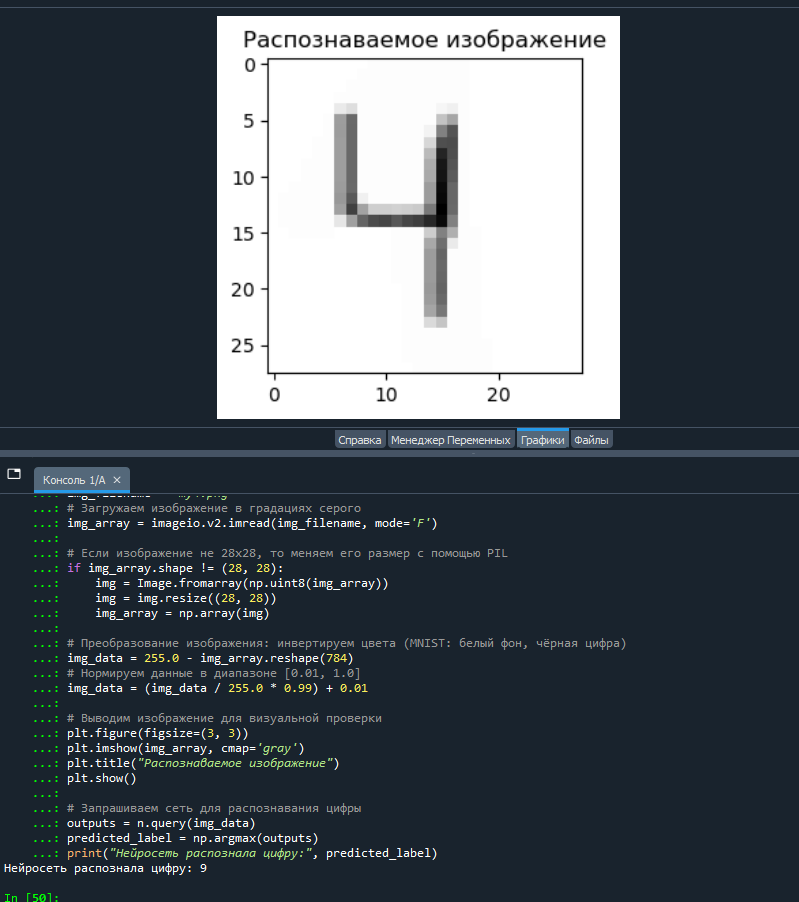
Вариант 7

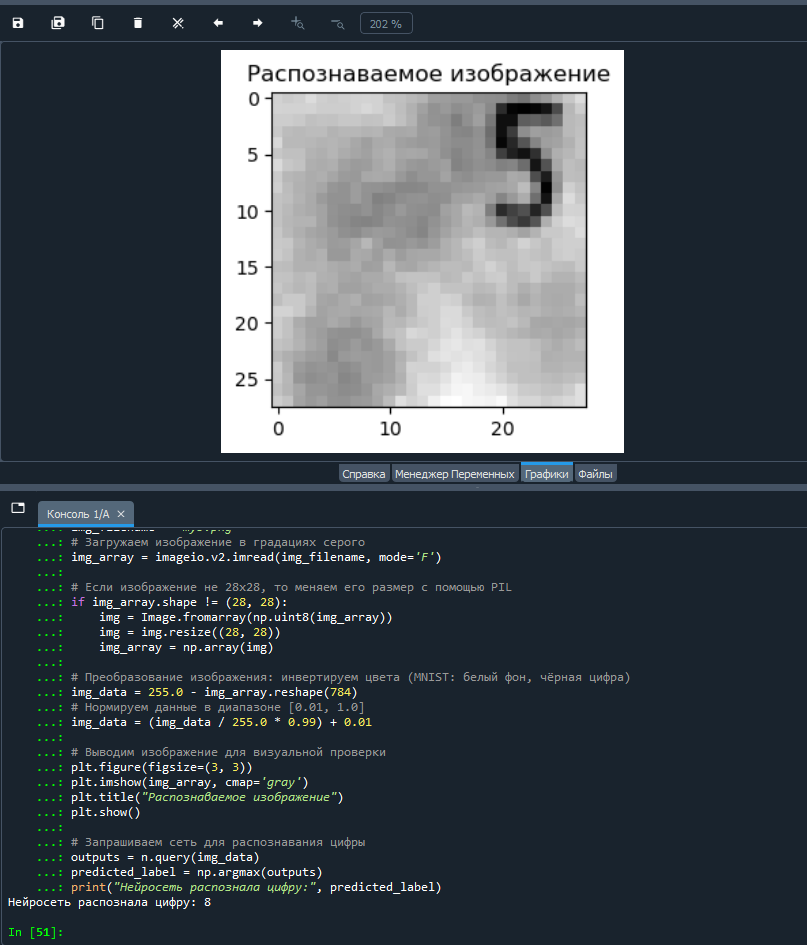


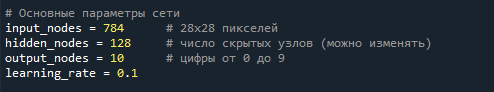
**Ход работы**

**Запустили нейронную сеть со стандартными значениями.\**

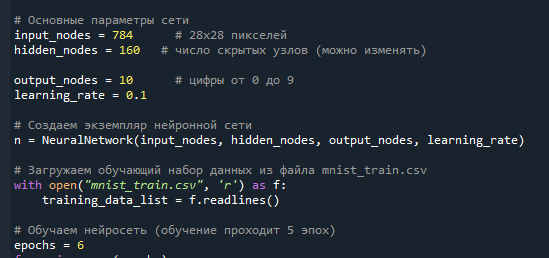
****

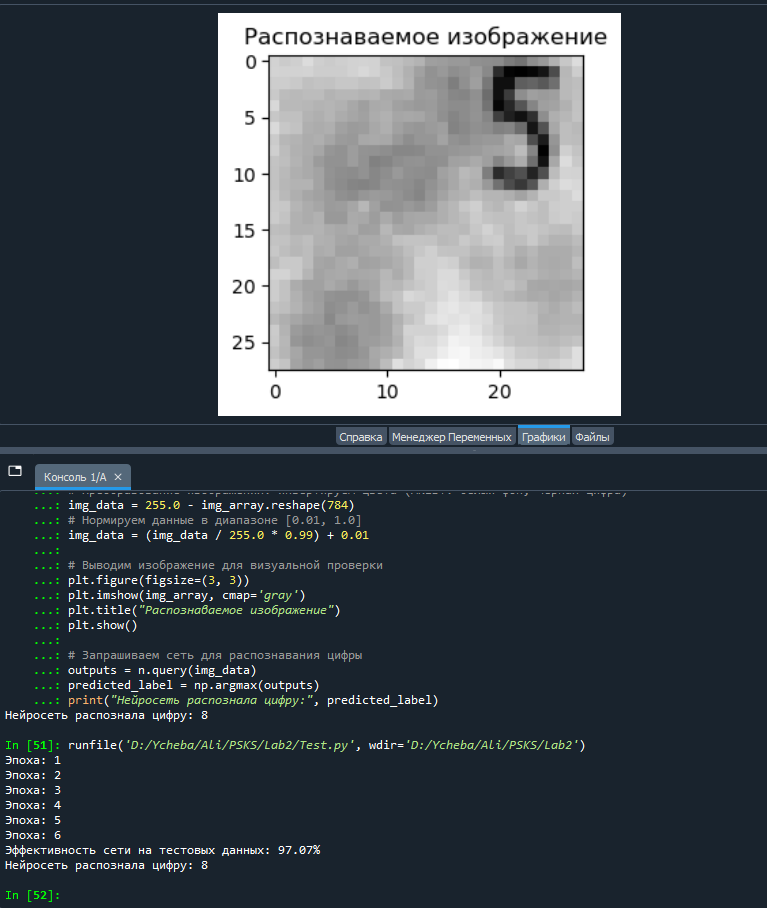
** **

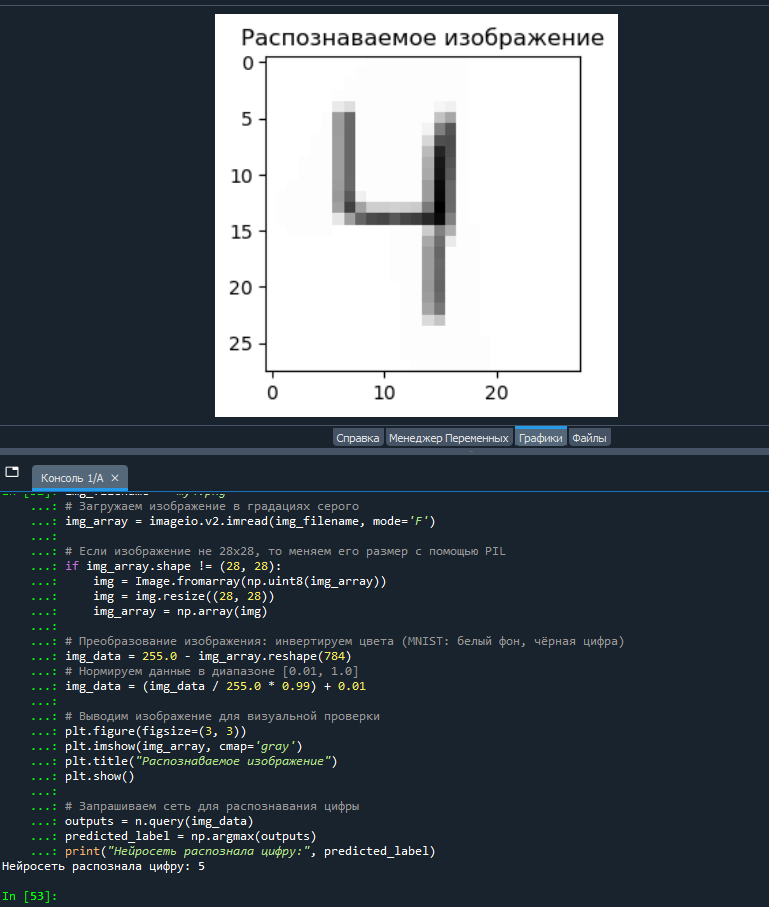
****

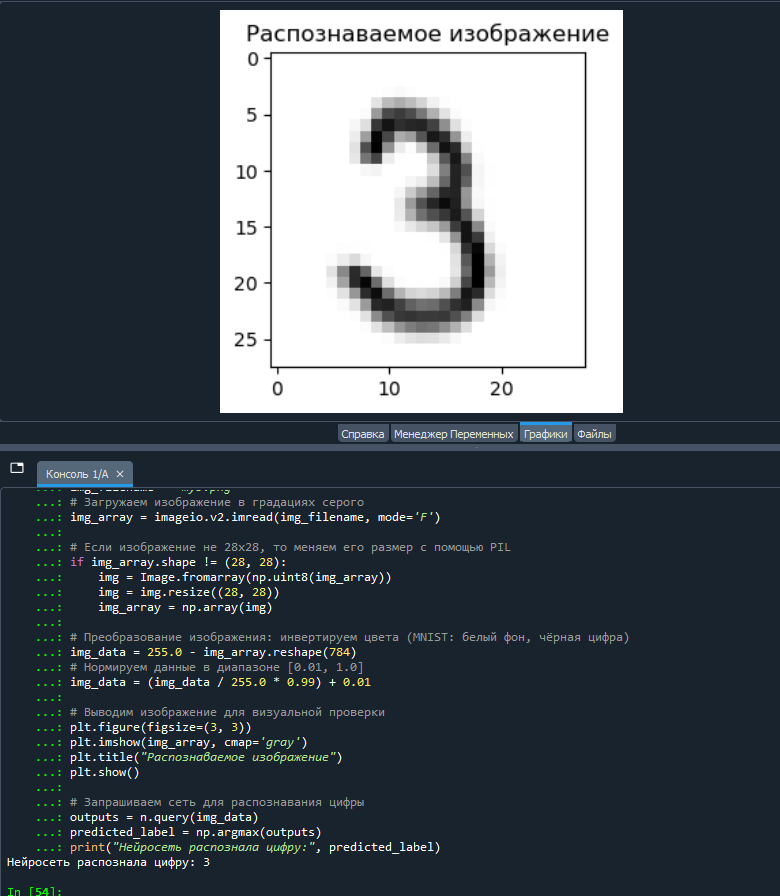
****

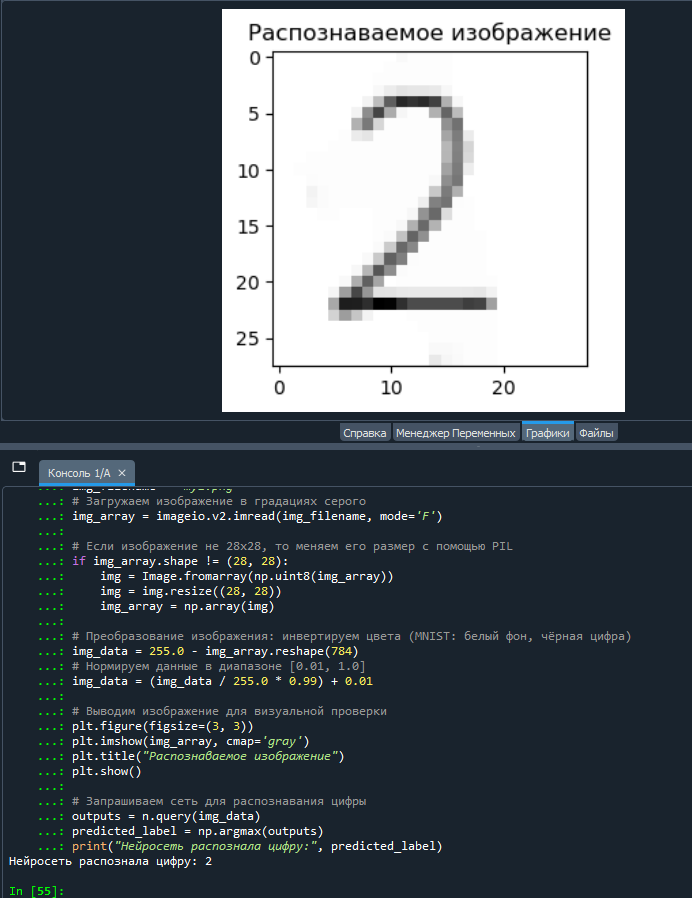
**Поменяли данные указанные в варианте.**

****

****

****

****

****

**Эффективность со стандартными значениями: 96,82 %**

**Эффективность со значениями из вариантов: 97.07 %**

**Листинг**

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

import numpy as np

import scipy.special

import imageio

import matplotlib.pyplot as plt

import os

from PIL import Image # для изменения размера изображения (если необходимо)

# Определение класса нейронной сети

class NeuralNetwork:

def \_\_init\_\_(self, inputnodes, hiddennodes, outputnodes, learningrate):

self.inodes = inputnodes

self.hnodes = hiddennodes

self.onodes = outputnodes

# Инициализация весов: нормальное распределение со средним 0 и стандартным отклонением,

# равным обратному квадратному корню от количества входных (или скрытых) узлов.

self.wih = np.random.normal(0.0, pow(self.inodes, -0.5), (self.hnodes, self.inodes))

self.who = np.random.normal(0.0, pow(self.hnodes, -0.5), (self.onodes, self.hnodes))

self.lr = learningrate

# Функция активации – сигмоида (expit)

self.activation\_function = lambda x: scipy.special.expit(x)

def train(self, inputs\_list, targets\_list):

# Передаем входные данные в виде столбца (вектор-столбец)

inputs = np.array(inputs\_list, ndmin=2).T

targets = np.array(targets\_list, ndmin=2).T

# Вычисляем входные данные скрытого слоя

hidden\_inputs = np.dot(self.wih, inputs)

hidden\_outputs = self.activation\_function(hidden\_inputs)

# Вычисляем входные данные итогового слоя

final\_inputs = np.dot(self.who, hidden\_outputs)

final\_outputs = self.activation\_function(final\_inputs)

# Вычисляем ошибки итогового слоя (таргеты - полученный ответ)

output\_errors = targets - final\_outputs

# Ошибка скрытого слоя – передается обратно через веса

hidden\_errors = np.dot(self.who.T, output\_errors)

# Корректировка весов между скрытым и итоговым слоями

self.who += self.lr \* np.dot((output\_errors \* final\_outputs \* (1.0 - final\_outputs)),

hidden\_outputs.T)

# Корректировка весов между входным и скрытым слоями

self.wih += self.lr \* np.dot((hidden\_errors \* hidden\_outputs \* (1.0 - hidden\_outputs)),

inputs.T)

def query(self, inputs\_list):

inputs = np.array(inputs\_list, ndmin=2).T

# Вычисляем входные данные скрытого слоя

hidden\_inputs = np.dot(self.wih, inputs)

hidden\_outputs = self.activation\_function(hidden\_inputs)

# Вычисляем входные данные итогового слоя

final\_inputs = np.dot(self.who, hidden\_outputs)

final\_outputs = self.activation\_function(final\_inputs)

return final\_outputs

# Основные параметры сети

input\_nodes = 784 # 28x28 пикселей

hidden\_nodes = 160 # число скрытых узлов (можно изменять)

output\_nodes = 10 # цифры от 0 до 9

learning\_rate = 0.1

# Создаем экземпляр нейронной сети

n = NeuralNetwork(input\_nodes, hidden\_nodes, output\_nodes, learning\_rate)

# Загружаем обучающий набор данных из файла mnist\_train.csv

with open("mnist\_train.csv", 'r') as f:

training\_data\_list = f.readlines()

# Обучаем нейросеть (обучение проходит 5 эпох)

epochs = 6

for e in range(epochs):

print("Эпоха:", e + 1)

for record in training\_data\_list:

all\_values = record.strip().split(',')

# Первый элемент – метка (правильная цифра)

targets = np.zeros(output\_nodes) + 0.01

targets[int(all\_values[0])] = 0.99

# Нормируем входные данные (значения пикселей от 0 до 255 преобразуем в диапазон 0.01 – 1.0)

inputs = (np.asfarray(all\_values[1:]) / 255.0 \* 0.99) + 0.01

n.train(inputs, targets)

# Тестирование нейросети на наборе данных из файла mnist\_test.csv

with open("mnist\_test.csv", 'r') as f:

test\_data\_list = f.readlines()

# Сохраняем результаты тестирования (правильный ответ или ошибка)

scorecard = []

for record in test\_data\_list:

all\_values = record.strip().split(',')

correct\_label = int(all\_values[0])

inputs = (np.asfarray(all\_values[1:]) / 255.0 \* 0.99) + 0.01

outputs = n.query(inputs)

label = np.argmax(outputs)

if label == correct\_label:

scorecard.append(1)

else:

scorecard.append(0)

scorecard\_array = np.asarray(scorecard)

accuracy = scorecard\_array.sum() / scorecard\_array.size \* 100

print("Эффективность сети на тестовых данных: {:.2f}%".format(accuracy))

# Распознавание собственного изображения (my1.png)

img\_filename = "my2.png"

# Загружаем изображение в градациях серого

img\_array = imageio.v2.imread(img\_filename, mode='F')

# Если изображение не 28x28, то меняем его размер с помощью PIL

if img\_array.shape != (28, 28):

img = Image.fromarray(np.uint8(img\_array))

img = img.resize((28, 28))

img\_array = np.array(img)

# Преобразование изображения: инвертируем цвета (MNIST: белый фон, чёрная цифра)

img\_data = 255.0 - img\_array.reshape(784)

# Нормируем данные в диапазоне [0.01, 1.0]

img\_data = (img\_data / 255.0 \* 0.99) + 0.01

# Выводим изображение для визуальной проверки

plt.figure(figsize=(3, 3))

plt.imshow(img\_array, cmap='gray')

plt.title("Распознаваемое изображение")

plt.show()

# Запрашиваем сеть для распознавания цифры

outputs = n.query(img\_data)

predicted\_label = np.argmax(outputs)

print("Нейросеть распознала цифру:", predicted\_label)

**Вывод:** в результате выполнения лабораторной работы изучили программы по распознаванию рукописных цифр при помощи трехслойной нейронной сети.