Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра: «Вычислительная техника»

Дисциплина: «Системы искусственного интеллекта»

Лабораторная работа №6

Сверточные сети

Вариант 20

Выполнил

Студент группы ИВТАСбд-32

Хасанов И. Н.

Проверил:

преподаватель кафедры «ВТ»

Хайруллин И. Д.

Ульяновск 2024

**Оглавление**

[Постановка задачи 3](#_Toc167468598)

[Реализация 4](#_Toc167468599)

[Вывод 5](#_Toc167468600)

[Приложение. Source.cpp 6](#_Toc167468601)

# Постановка задачи

* С использованием библиотеки Keras загрузить обучающую выборку, создать модель сверточной сети, обучить ее на обучающей выборке, сохранить модель в файл.
* Написать дополнительно программу, которая загружает модель из файла, и предоставляет возможность загрузить файл или данные любым иным способом, чтобы проверить точность классификатора.

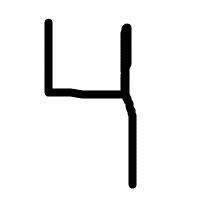
Вариант: MNIST.

# Реализация

Код написан на языке Python. Вначале загружаются данные MNIST через keras.datasets.mnist.load\_data() и выборка разделяется на обучающую и тестовую наборы. Нормализация пикселей происходит путем деления на 255 для приведения значения пикселей к 1 или 0.

Создание сверточной модели происходит путем создания массива тензоров и уменьшения выборки, затем данные приводятся к виду одномерного массива, модель компилируется и обучается.

Изображение для теста считывается через cv2.imread() в чёрно-белом виде, сжимается через cv2.resize() и приводится к виду массива.





# Вывод

Проделав данную лабораторную работу, были изучены методы работы со сверточными сетями и обработка изображений.

# Приложение. Source.cpp

|  |
| --- |
| import numpy as np  import keras  from keras import layers  from keras.models import load\_model  import cv2  # Загрузка набора данных MNIST и разделяем их на обучающие и тестовые наборы  (x\_train, y\_train), (x\_test, y\_test) = keras.datasets.mnist.load\_data()  num\_classes = 10  input\_shape = (28, 28, 1)  x\_train = x\_train.astype("float32") / 255  x\_test = x\_test.astype("float32") / 255  x\_train = np.expand\_dims(x\_train, -1)  x\_test = np.expand\_dims(x\_test, -1)  y\_train = keras.utils.to\_categorical(y\_train, num\_classes)  y\_test = keras.utils.to\_categorical(y\_test, num\_classes)  model = keras.Sequential(  [  keras.Input(shape=input\_shape),  layers.Conv2D(32, kernel\_size=(3, 3), activation="relu"),  layers.MaxPooling2D(pool\_size=(2, 2)),  layers.Conv2D(64, kernel\_size=(3, 3), activation="relu"),  layers.MaxPooling2D(pool\_size=(2, 2)),  layers.Flatten(),  layers.Dropout(0.5),  layers.Dense(num\_classes, activation="softmax"),  ]  )  model.compile(loss="categorical\_crossentropy", optimizer="adam", metrics=["accuracy"])  model.fit(x\_train, y\_train, batch\_size=128, epochs=10, validation\_data=(x\_test, y\_test), shuffle=True)  model.save("model.h5")  model = load\_model("model.h5")  score = model.evaluate(x\_test, y\_test, verbose=0)  print("Test loss:", score[0])  print("Test accuracy:", score[1])  def predict\_class(image\_path):  img = cv2.imread(image\_path)[:,:,0]  img = cv2.resize(img,(28, 28))  img = np.invert(np.array([img]))  prediction = model.predict(img)  class\_index = prediction.argmax()  return class\_index  classes = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']  image\_path = "test.png"  predicted\_class = predict\_class(image\_path)  print("Predicted class:", classes[predicted\_class]) |