

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ
по лабораторной работе №4
по дисциплине «Искусственные нейронные сети»
Тема: Распознавание рукописных символов

Студент гр. 8383

Сосновский Д. Н.

Преподаватель

Жангиров Т. Р.

Санкт-Петербург

2021

Цель работы.

Реализовать классификацию черно-белых изображений рукописных цифр (28x28) по 10 категориям (от 0 до 9).

Задачи.

- Ознакомиться с представлением графических данных
- Ознакомиться с простейшим способом передачи графических данных нейронной сети
- Создать модель
- Настроить параметры обучения
- Написать функцию, позволяющую загружать изображение пользователи и классифицировать его

Ход работы.

Согласно методическим указаниям была составлена и обучена нейросеть. Исходный код программы приведён в приложении.

Точность полученной нейросети составляет ~ 96%, что удовлетворяет требованиям задачи.

Таким образом, архитектура полученной нейросети имеет следующий вид:

1-ый слой (входной) 28 нейронов

2-ой слой (скрытый) 256 нейронов, функция активации relu

3-ий слой (выходной) 10 нейронов, функция активации softmax

Исследование влияния оптимизаторов и их параметров на процесс обучения нейросети.

Были исследовано влияние различных оптимизаторов и их параметров на точность работы нейросети.

1) Оптимизатор **RMSprop**

Параметры оптимизатора: lr (скорость обучения), rho (коэфф. затухания скользящего среднего)

| lr | rho | test_loss | test_acc |
|------|-----|-----------|----------|
| 0.01 | 0.5 | 0.09037 | 0.97579 |
| 0.1 | 0.9 | 0.75965 | 0.89740 |

2) Оптимизатор **Adam**

Параметры оптимизатора: lr (скорость обучения), β_1 , β_2 – численные константы, стремящиеся к 1.

| lr | β_1 | β_2 | test_loss | test_acc |
|-------|-----------|-----------|-----------|----------|
| 0.001 | 0.9 | 0.9 | 0.06669 | 0.97960 |
| 0.01 | 0.99 | 0.99 | 0.13797 | 0.96960 |
| 0.01 | 0.999 | 0.999 | 0.19799 | 0.96490 |

3) Оптимизатор **Nadam**

Параметры оптимизатора: lr (скорость обучения), β_1 , β_2 аналогично Adam. Является вариацией RMSprop с функцией импульса Нестерова.

| lr | β_1 | β_2 | test_loss | test_acc |
|-------|-----------|-----------|-----------|----------|
| 0.001 | 0.9 | 0.9 | 0.07039 | 0.979499 |
| 0.01 | 0.99 | 0.99 | 0.12002 | 0.973500 |
| 0.01 | 0.999 | 0.999 | 0.12138 | 0.976400 |

4) Оптимизатор **Adadelta**

Параметры оптимизатора: lr (начальная скорость обучения), ρ – коэффициент распада доли градиента.

| lr | ρ | test_loss | test_acc |
|------|--------|-----------|----------|
| 0.5 | 0.6 | 0.13061 | 0.96139 |
| 0.9 | 0.9 | 0.08752 | 0.97390 |

5) Оптимизатор **Adamax**

Представляет собой вариацию оптимизатора Adam. Основывается на нормировании бесконечности.

Параметры оптимизатора: lr (скорость обучения), β_1 , β_2 – аналогично Adam.

| lr | β_1 | β_2 | test_loss | test_acc |
|-------|-----------|-----------|-----------|----------|
| 0.1 | 0.89 | 0.9999 | 0.13351 | 0.95990 |
| 0.02 | 0.8 | 0.8 | 0.53459 | 0.95149 |
| 0.003 | 0.99 | 0.99 | 0.07006 | 0.97780 |

Основываясь на приведённой статистике, видно средняя точность нейросети относительно всех проведенных запусков составляет около 96%, а максимальная точность составляет 97.5% с оптимизатором RMSprop и параметрами обучения $lr=0.01$, $\rho = 0.5$.

Реализация функции загрузки пользовательского изображения не из датасета.

Программа была доработана таким образом, что теперь пользователь может ввести в консоль название файла с изображением, и ему будет выведено число, которое на картинке определит обученная нейросеть.

Пример работы программы представлен на рисунках ниже.

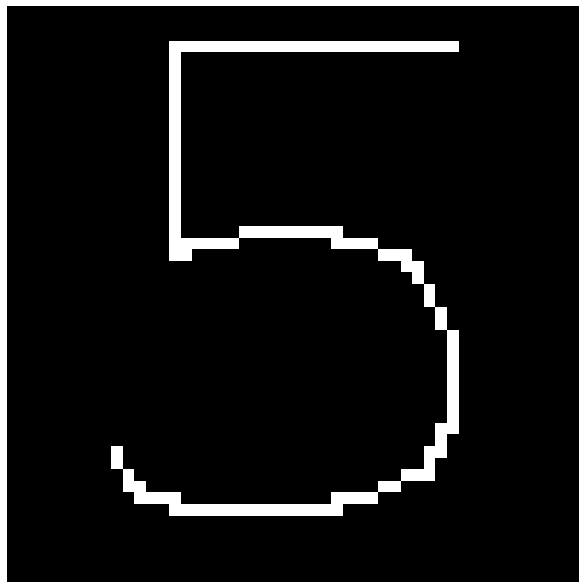


Рисунок 1 – входное изображение с цифрой «5».

Листинг 1 – лог консолей тестирования изображения на рис. 1.

```
=====
Введите имя файла с изображением:
five.png
5
=====
```

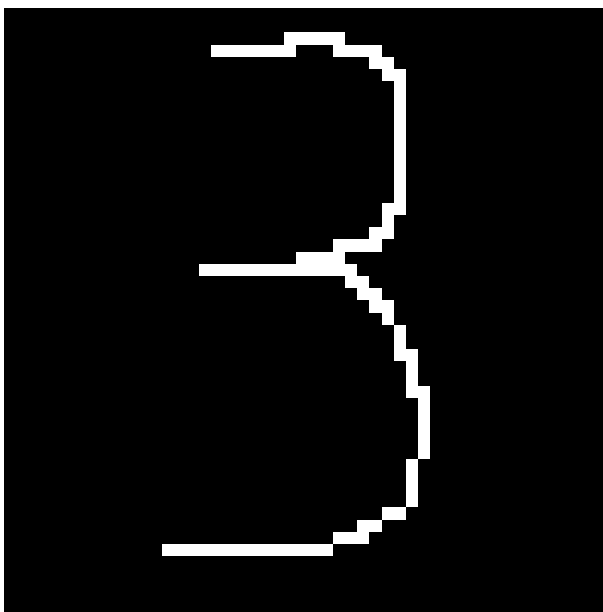


Рисунок 2 – входное изображение с цифрой «2».

Листинг 2 – лог консолей тестирования изображения на рис. 2.

```
=====
Введите имя файла с изображением:
three.png
3
=====
```

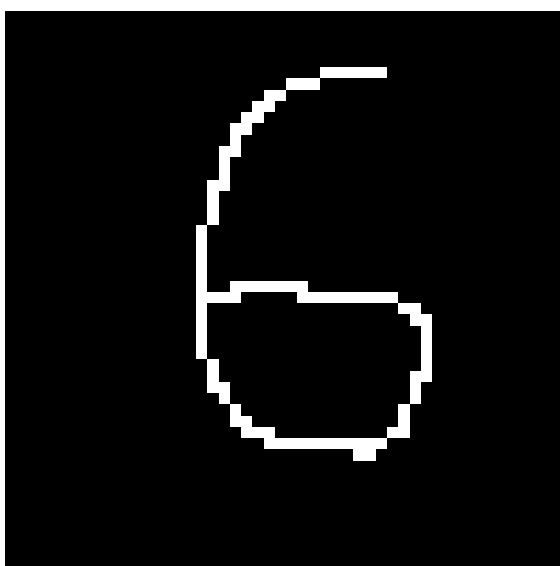


Рисунок 2 – входное изображение с цифрой «2».

Листинг 3 – лог консолей тестирования изображения на рис. 3.

```
=====
Введите имя файла с изображением:
```

```
six.png
```

```
6
=====
```

Вывод.

В ходе выполнения лабораторной работы была создана нейронная сеть, которая классифицирует черно-белые изображения рукописных цифр по 10 категориям. Была найдена оптимальная конфигурация обучения сети, при которой сеть показывала максимальную точность.

ПРИЛОЖЕНИЕ

В данном приложении приведен исходный код программы.

```
from tensorflow.keras.layers import Dense, Activation, Flatten
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
import tensorflow.keras.optimizers as optimizers
import tensorflow as tf
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
import matplotlib.pyplot as plt
import numpy as np

mnist = tf.keras.datasets.mnist
(train_images, train_labels), (test_images, test_labels) =
mnist.load_data()

train_images = train_images / 255.0
test_images = test_images / 255.0
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

model = Sequential()
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(10, activation='softmax'))

optimizer = optimizers.RMSprop(learning_rate=0.01, rho=0.5)
# optimizer = optimizers.Adam(learning_rate=0.01, beta_1=0.999,
# beta_2=0.999)
# optimizer = optimizers.Nadam(learning_rate=0.01, beta_1=0.9999,
# beta_2=0.9999)
# optimizer = optimizers.Adadelta(learning_rate=0.9, rho=0.9)
# optimizer = optimizers.Adamax(learning_rate=0.003, beta_1=0.99,
# beta_2=0.99)

model.compile(optimizer=optimizer, loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, batch_size=128)

test_loss, test_acc = model.evaluate(test_images, test_labels)
print('test_loss: ', np.around(test_loss, decimals=5))
print('test_acc: ', test_acc)

while True:
    print('Введите имя файла с изображением: ')
    filename = input()
    img = load_img(filename, color_mode='grayscale', target_size=(28,
28))
    img = img_to_array(img)
```

```
img = img.reshape(1, 28, 28, 1)
img = img.astype('float32') / 255.0
digit = np.argmax(model.predict(img), axis=-1)

print(digit[0])
```