

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

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

Студент гр. 7382

Бахеров
Д.В.

Преподаватель

Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

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

Порядок выполнения работы.

1. Ознакомиться с представлением графических данных.
2. Ознакомиться с простейшим способом передачи графических данных нейронной сети.
3. Создать модель.
4. Настроить параметры обучения.
5. Написать функцию, позволяющая загружать изображение пользователя и классифицировать его.

Требования к выполнению задания.

1. Найти архитектуру сети, при которой точность классификации будет не менее 95%.
2. Исследовать влияние различных оптимизаторов, а также их параметров, на процесс обучения.
3. Написать функцию, которая позволит загружать пользовательское изображение не из датасета.

Основные теоретические положения.

MINIST – база данных рукописных цифр, имеющая подготовленный набор обучающих значений в размере 60000 примеров и тестовых значений из 10000 примеров. Это подмножество более широкого набора, доступное из NIST. Наш набор состоит из изображений размером 28x28, каждый пиксель которого представляет собой оттенок серого, цифры нормализованы по размеру и имеют фиксированный размер изображения. Таким образом, есть 10 цифр (от 0 до 9) или 10 классов для прогнозирования. Результаты сообщаются с использованием ошибки прогнозирования, которая является не чем иным, как инвертированной точностью классификации.

Ход работы.

Была создана и обучена модель искусственной нейронной сети. Код предоставлен в приложении А.

1. Оптимизатор adam.

Архитектура:

- Скорость обучения = 0.001.
- Инициализация весов – normal.
- Epochs = 5, batch_size = 128, loss = categorical_crossentropy
- Слои:

```
model.add(Flatten())
```

```
model.add(Dense(64,activation='relu'))
```

```
model.add(Dense(32, activation='relu'))
```

```
model.add(Dense(10, activation='softmax'))
```

Данная архитектура дает точность ~ 96%. Графики точности и ошибки предоставлены на рис. 1 и рис. 2 соответственно.

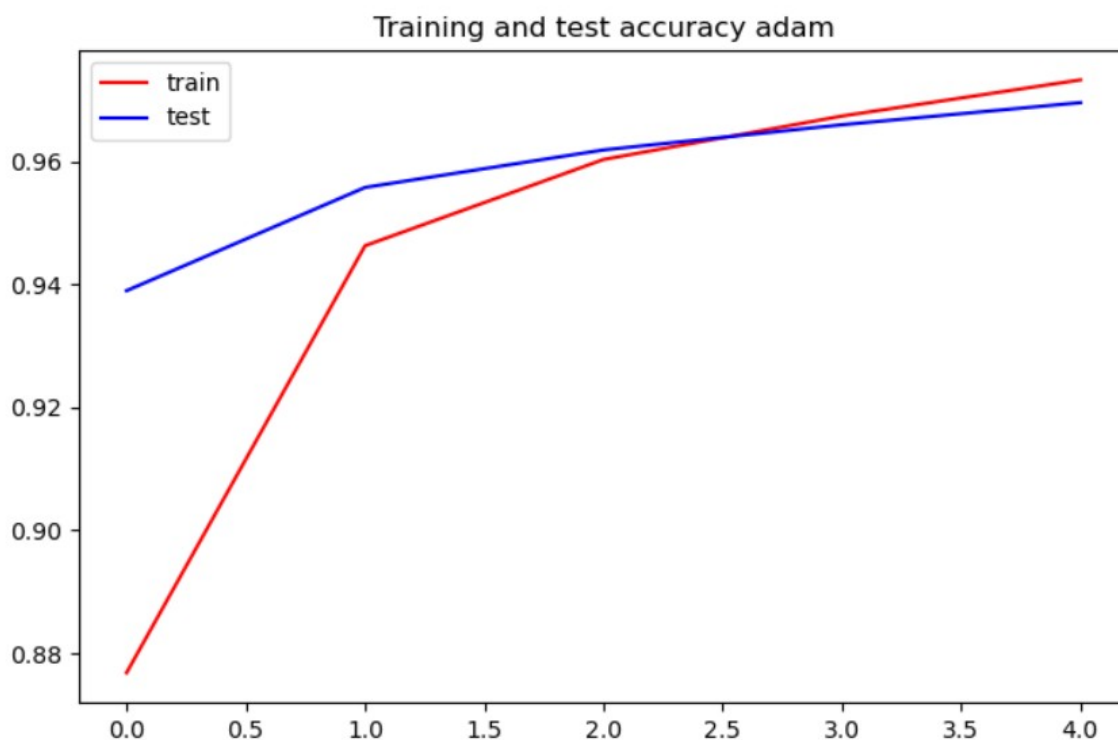


Рисунок 1 – График точности для оптимизатора adam

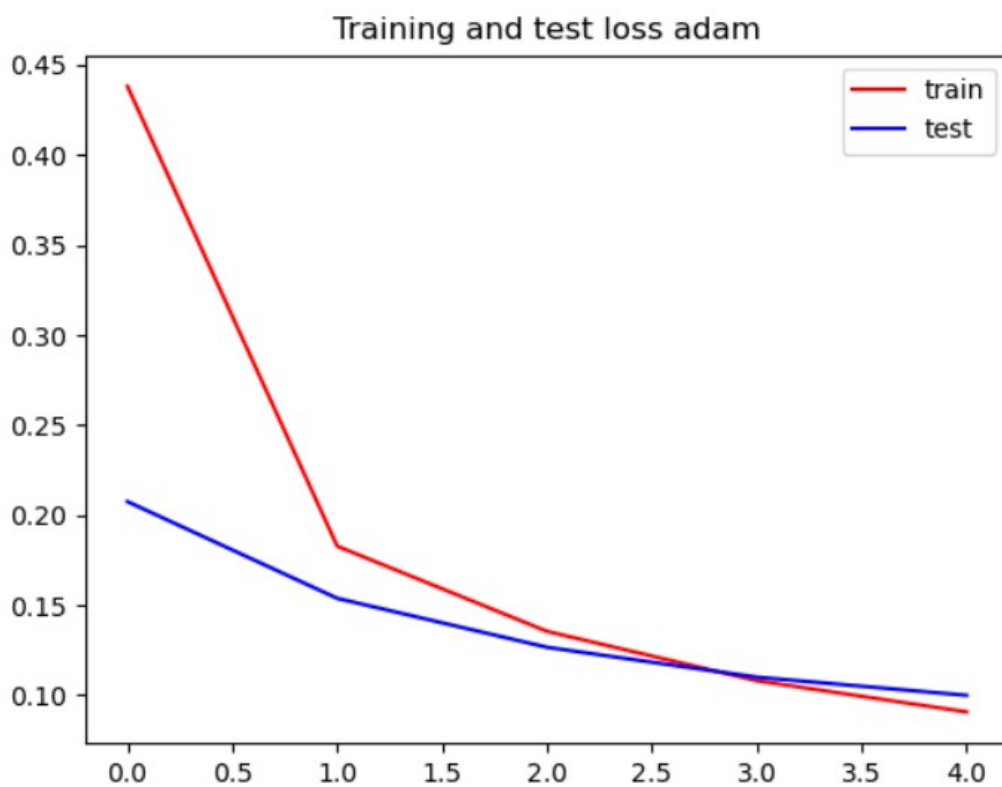


Рисунок 2 – График потерь для оптимизатора adam

Напишем функцию, которая позволит загружать пользовательское изображение не из датасета:

```
def openImage(path):  
    return numpy.asarray(Image.open(path))
```

Выводы.

В ходе работы была изучена задача классификации рукописных цифр с помощью азы данных MNIST. Подобрана архитектура, дающая точность выше 95%, таковой оказалась adam. Также была написана функция загрузки изображения в память программы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
from keras.layers import Dense, Flatten
from keras.models import Sequential
from keras.datasets import mnist
from keras.utils import to_categorical
from keras import optimizers
from PIL import Image
import numpy
import matplotlib.pyplot as plt

def openImage(path):
    return numpy.asarray(Image.open(path))

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

train_images = train_images / 255.0
test_images = test_images / 255.0

model = Sequential()
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))

def compile_fit_print(optimizer, name):
    model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
    H = model.fit(train_images, train_labels, epochs=5, batch_size=128, validation_data=(test_images,
test_labels))

    plt.figure(1, figsize=(8, 5))
    plt.title('Training and test accuracy ' + name)
    plt.plot(H.history['accuracy'], 'r', label='train')
    plt.plot(H.history['val_accuracy'], 'b', label='test')
    plt.legend()
    plt.show()
    plt.clf()

    plt.figure(1, figsize=(8, 5))
    plt.title('Training and test loss ' + name)
    plt.plot(H.history['loss'], 'r', label='train')
    plt.plot(H.history['val_loss'], 'b', label='test')
    plt.legend()
    plt.show()
    plt.clf()

compile_fit_print(optimizers.Adam(), 'adam')
```