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

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине «Искусственные нейронные сети»

Тема: «Распознавание рукописных символов»

Студент гр. 7383 _____ Ханова Ю. А.

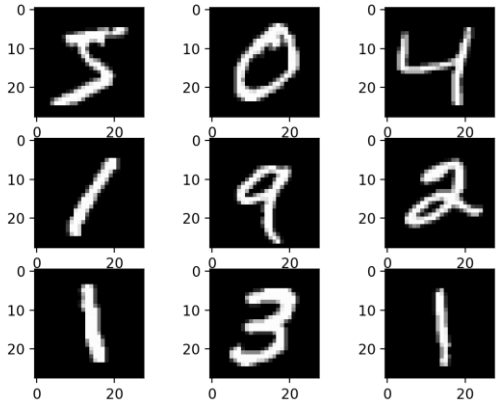
Преподаватель _____ Жукова Н. А.

Санкт-Петербург

2020

Цели.

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



Набор данных содержит 60,000 изображений для обучения и 10,000 изображений для тестирования.

Задачи.

- Ознакомиться с представлением графических данных
- Ознакомиться с простейшим способом передачи графических данных нейронной сети
- Создать модель
- Настроить параметры обучения
- Написать функцию, позволяющая загружать изображение пользователи и классифицировать его
- Найти архитектуру сети, при которой точность классификации будет не менее 95%
- Исследовать влияние различных оптимизаторов, а также их параметров, на процесс обучения
- Написать функцию, которая позволит загружать пользовательское изображение не из датасета

Ход работы.

Задаем архитектуру сети, настраиваем параметры для этапа компиляции:

• функцию потерь, которая определяет, как сеть должна оценивать качество своей работы на обучающих данных и, соответственно, как корректировать ее в правильном направлении

• оптимизатор — механизм, с помощью которого сеть будет обновлять себя, опираясь на наблюдаемые данные и функцию потерь

• метрики для мониторинга на этапах обучения и тестирования — здесь нас будет интересовать только точность (доля правильно классифицированных изображений).

Сравним работу сети под влиянием различных оптимизаторов и их параметров. На рис. 1-6 представлены графики ошибки и точности для оптимизаторов, а в табл.1-2 представлены значения ошибки и точности:

Таблица 1 – значения ошибки

	Adam	Adagrad	SGD
0.001	0.07203216347778216	0.5431112371921539	0.9706998738288879
0.01	0.1336180197651818	0.24055980096161367	0.32691451192498205

Таблица 2 – значения точности

	Adam	Adagrad	SGD
0.001	0.9781	0.8762	0.8015
0.01	0.9695	0.9328	0.9106

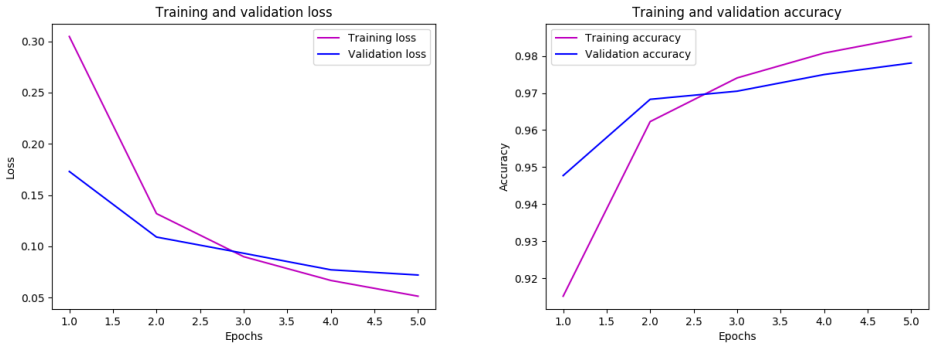


Рисунок 1 – Adam, learning_rate = 0.001

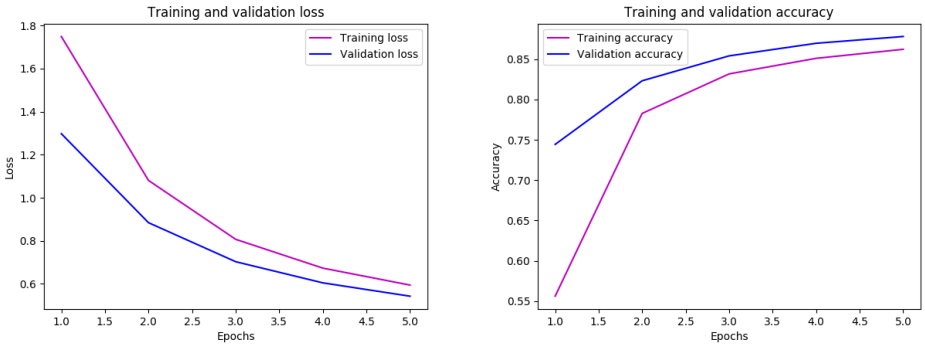


Рисунок 2– Adagrad, learning_rate = 0.001

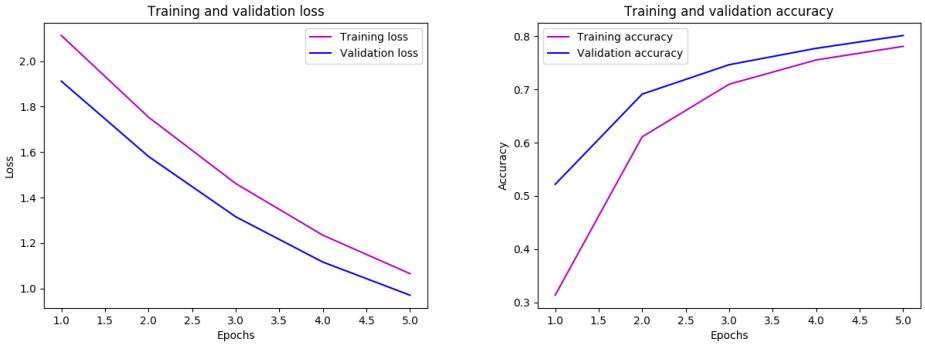


Рисунок 3 – SDG, learning_rate = 0.001

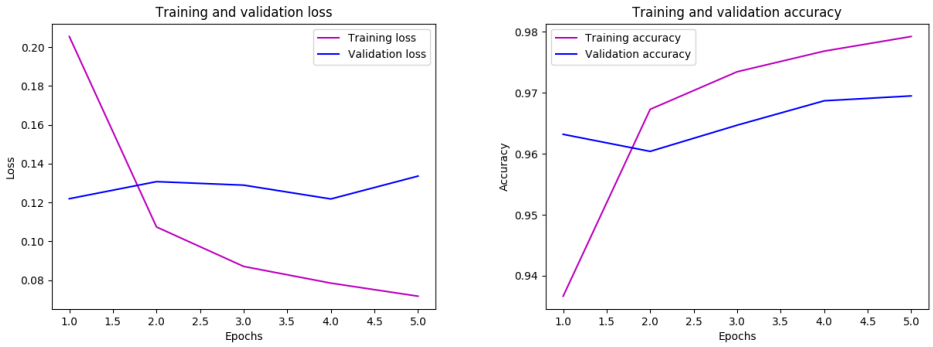


Рисунок 4 – Adam, learning_rate = 0.001

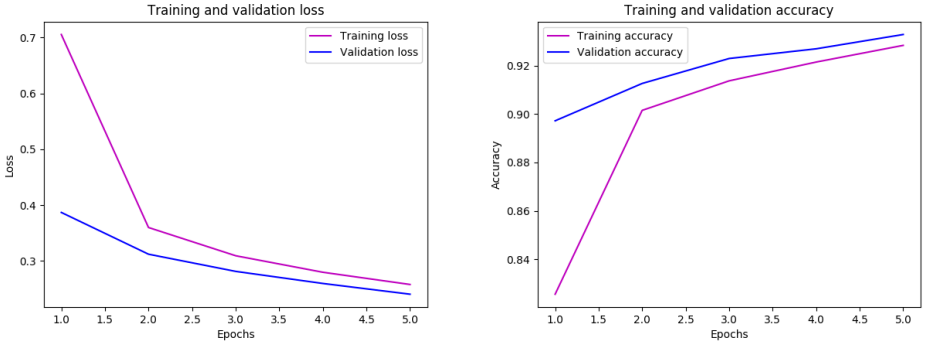


Рисунок 5 – Adagrad, learning_rate = 0.01

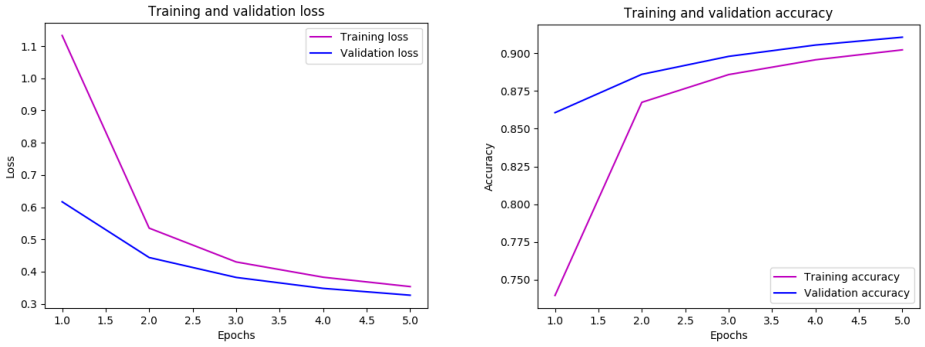


Рисунок 6 – SDG, learning_rate = 0.01

Можно заметить, что показатели улучшились, при увелечении скорости обучения для Adagrad и SGD. Для оптимизатора Adam, при увеличении скорости обучения, точность уменьшилась, а ошибка увеличилась.

Из графиков и значений следует, что наилучшая работа сети наблюдается при заданном оптимизаторе Adam с параметром learning_rate = 0.001

Вывод.

В ходе выполнения лабораторной работы была определена архитектура сети, при которой точность классификации будет не менее 95%. Было исследовано влияние различных оптимизаторов, а также их параметров, на процесс обучения. Написана функция, которая позволит загружать пользовательское изображение не из датасета. Код программы представлен в приложении А.

Приложения

Приложение А

```
from PIL import Image

import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.optimizers import *
from keras.utils import to_categorical
from tensorflow.keras.layers import Dense, Activation, Flatten
from tensorflow.keras.models import Sequential

def predictions_single(path, model):
    image = Image.open(path)
    width = image.size[0]
    height = image.size[1]
    if width != 28 and height != 28:
        image = image.resize((28, 28))
    image = image.convert('L')
    pix = np.asarray(image, dtype='uint8')
    pix = pix / 255
    one = (np.expand_dims(pix, 0))
    predict = model.predict(one)
    num = np.argmax(predict[0])
    return num

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'))
model.compile(optimizer=
SGD(learning_rate=0.01),loss='categorical_crossentropy',
metrics=['accuracy'])
history = model.fit(train_images, train_labels, epochs=5,
batch_size=128, validation_data=(test_images, test_labels))
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'm', label='Training loss')
```

```
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
plt.clf()

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
plt.plot(epochs, acc, 'm', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

test_loss, test_acc = model.evaluate(test_images, test_labels)
print('test_loss:', test_loss)
print('test_acc:', test_acc)
print(predictions_single("five.png", model))
```