

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»

Студентка гр. 7381

Машина Ю.Д.

Преподаватель

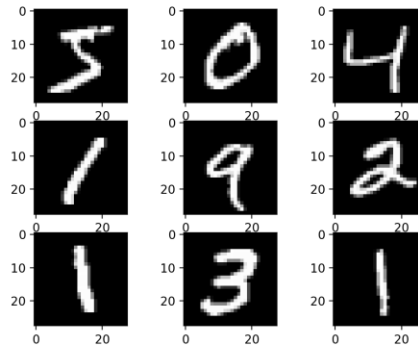
Жукова Н. А.

Санкт-Петербург

2020

Цели.

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



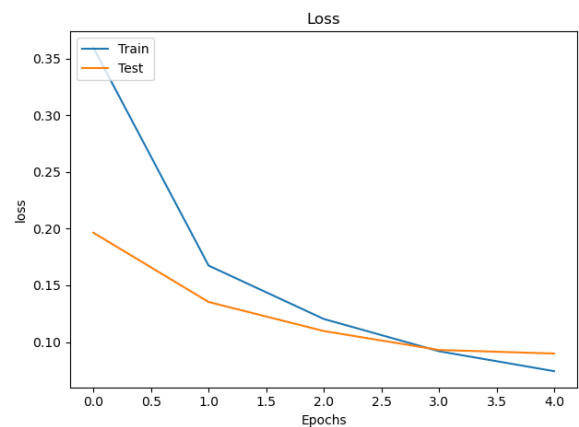
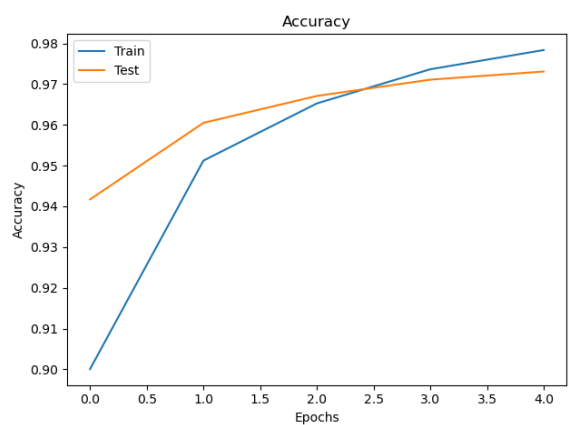
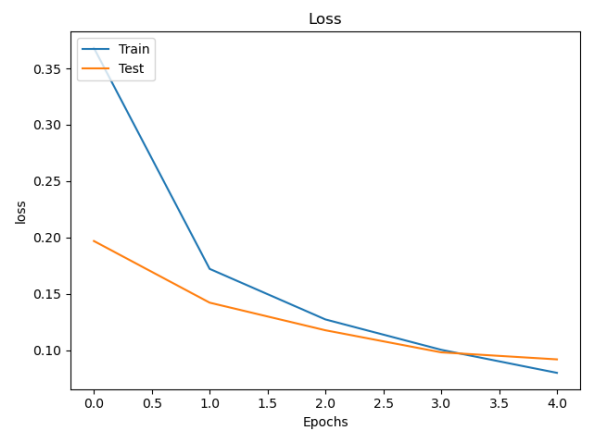
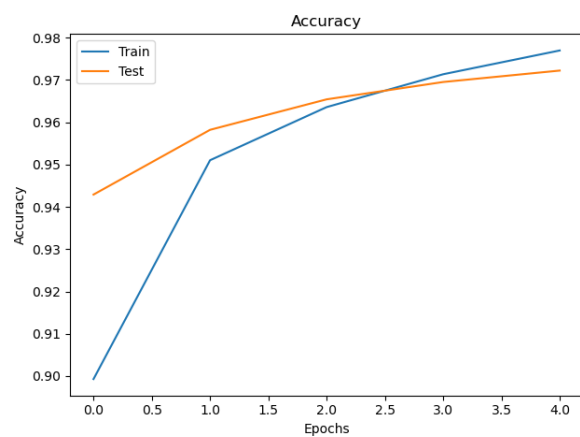
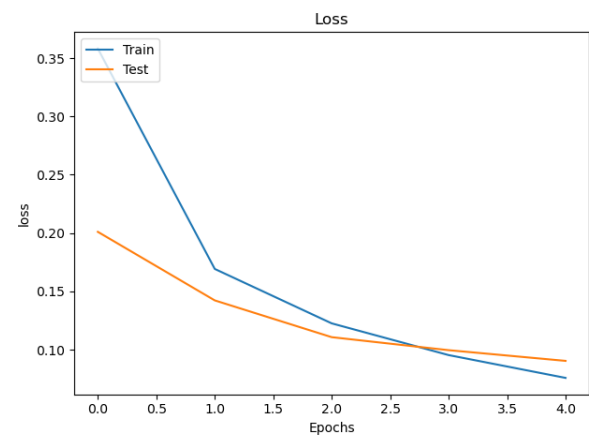
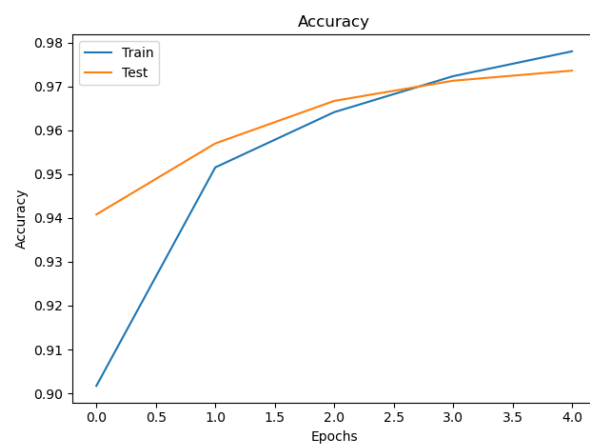
Задачи.

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

Выполнение работы.

1) Найти архитектуру сети, при которой точность классификации будет не менее 95%.

Подходит архитектура сети с двумя добавленными слоями, где на первом из них 128 нейронов с функцией активации `relu`, а на втором из них 10 с функцией активации `softmax`, он выходной. Оптимизатор `adam`. `batch_size=128`. Через 5 эпох точность классификации ни разу не упала ниже 95%.



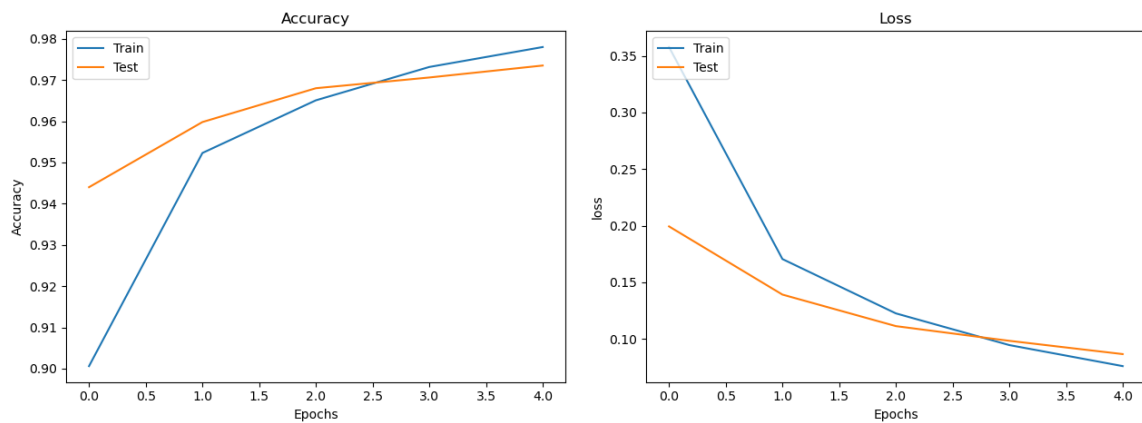


Рисунок 1 – Так в среднем выглядят графики обучения при такой модели.

2) Исследовать влияние различных оптимизаторов, а также их параметров, на процесс обучения.

Оптимизаторы при их дефолтных параметрах (

```
Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999,
amsgrad=False),
RMSprop(learning_rate=0.001, rho=0.9),
SGD(learning_rate=0.01, momentum=0.0, nesterov=False),
Nadam(learning_rate=0.002, beta_1=0.9, beta_2=0.999),
Adagrad(learning_rate=0.01),
Adadelat(learning_rate=1.0, rho=0.95),
Adamax(learning_rate=0.002, beta_1=0.9, beta_2=0.999)
):
```

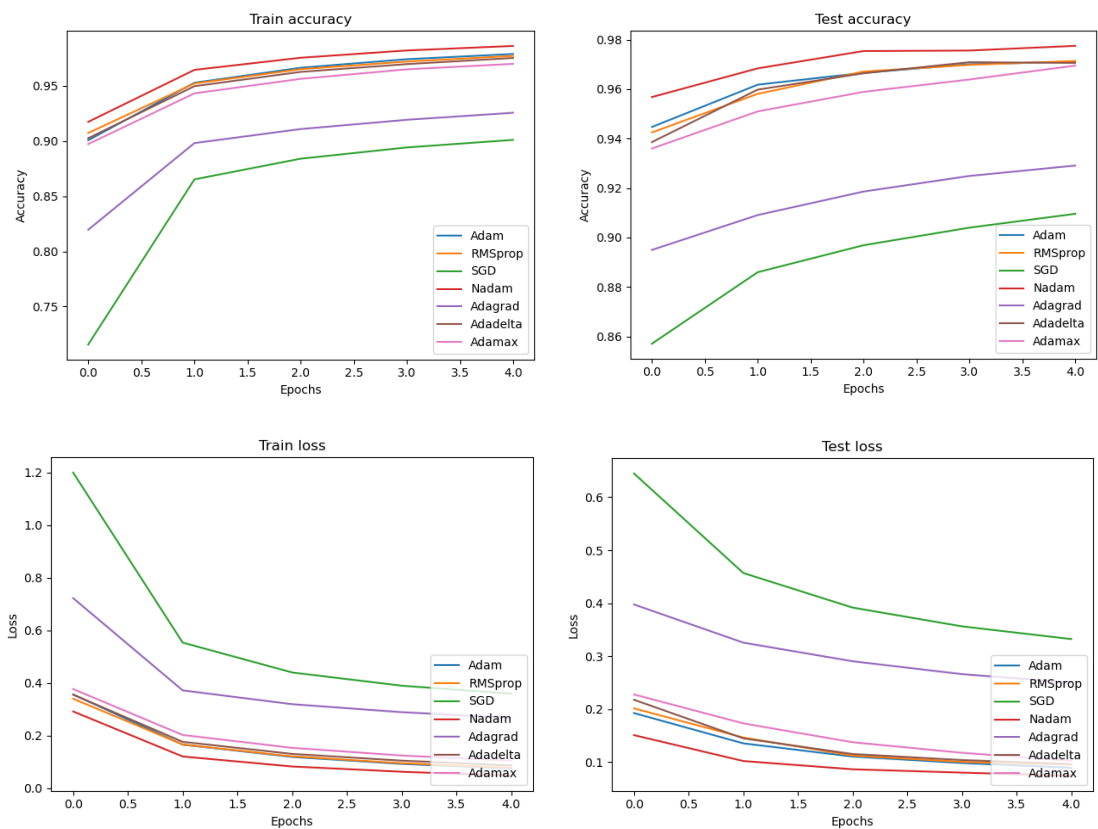


Рисунок 1 – Оптимизаторы при их дефолтных параметрах для данной модели.

Наилучший результат показывает Nadam. Единственный общий параметр у всех оптимизаторов – `learning_rate`. Можно посмотреть, как отличаются результаты оптимизаторов при одной скорости обучения.

Результаты при `learning_rate = 0.00001`:

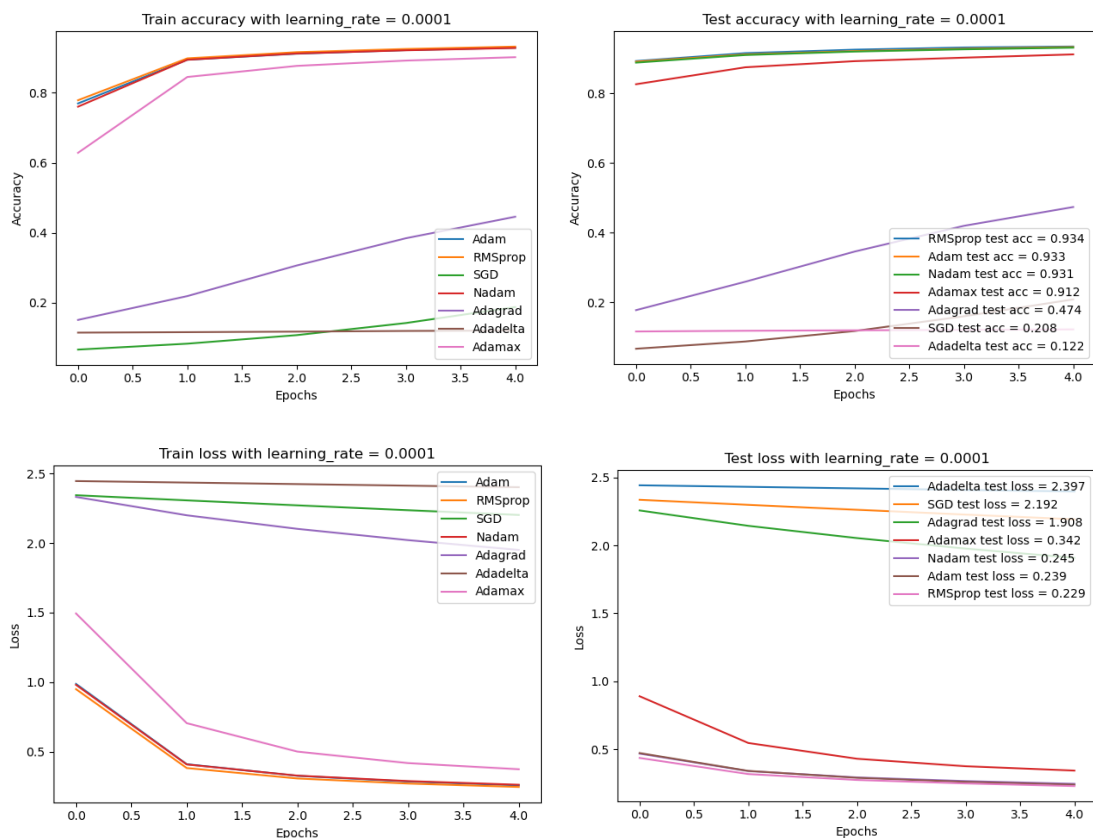
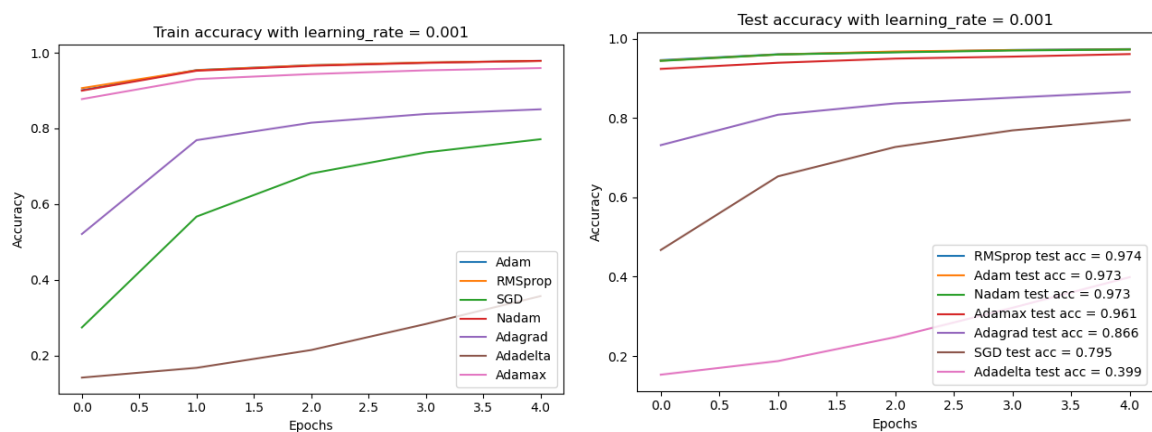


Рисунок 2 – Результаты при learning_rate = 0.00001.

Наилучший результат принадлежит RMSProp.

Результаты при learning_rate = 0.0001:



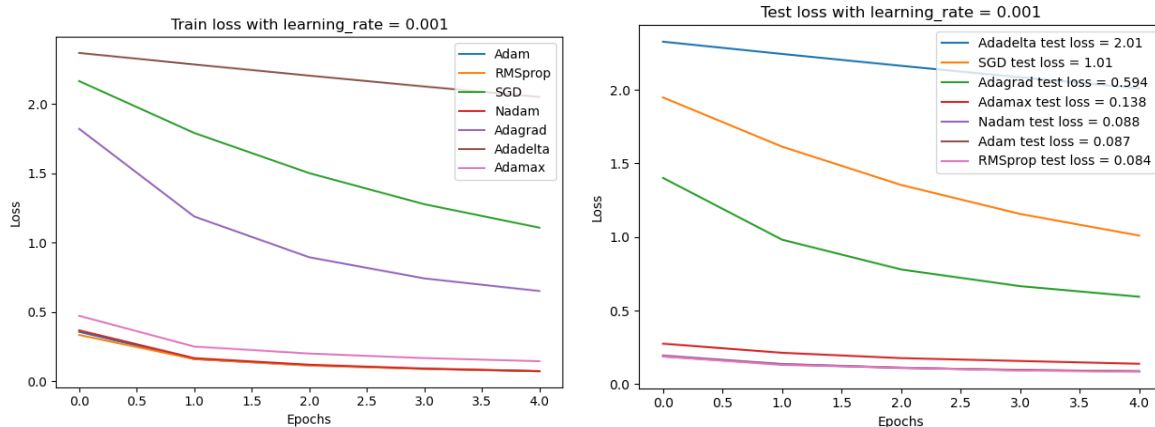


Рисунок 3 – Результаты при learning_rate = 0.0001.

Наилучший результат принадлежит RMSProp.

Результаты при learning_rate = 0.001:

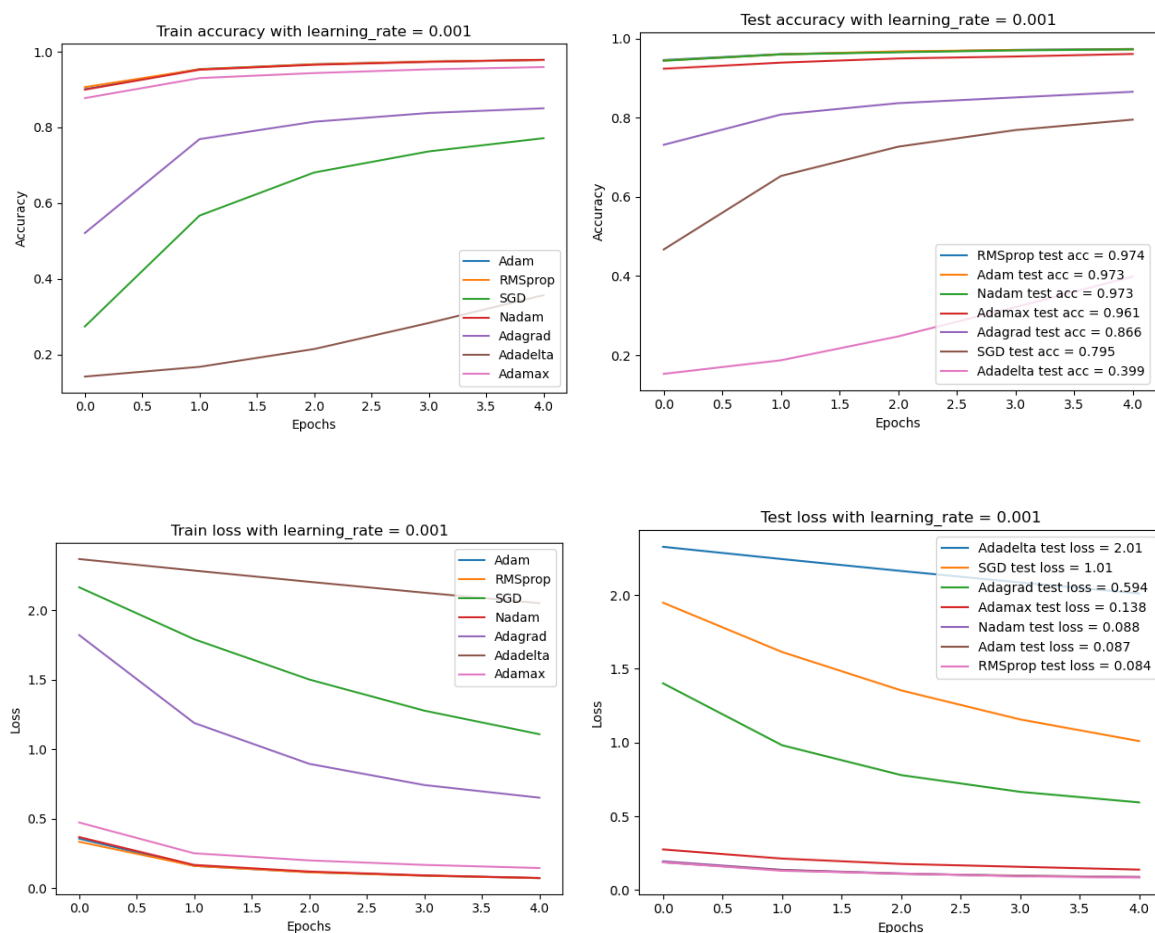


Рисунок 4 – Результаты при learning_rate = 0.001.

Наилучший результат принадлежит RMSProp.

Результаты при learning rate = 0.01:

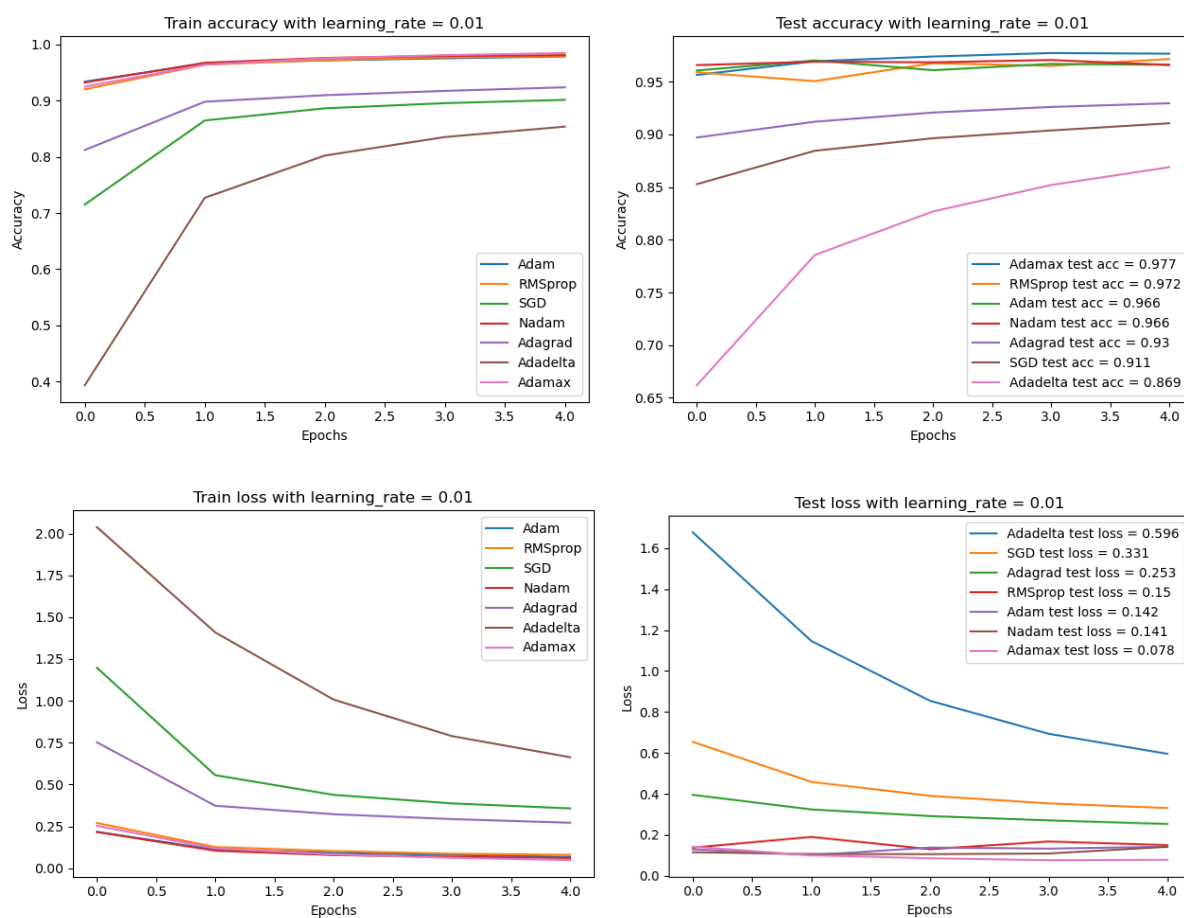


Рисунок 5 – Результаты при learning rate = 0.01.

Наилучший результат принадлежит Adamax.

Результаты при learning rate = 0.1:

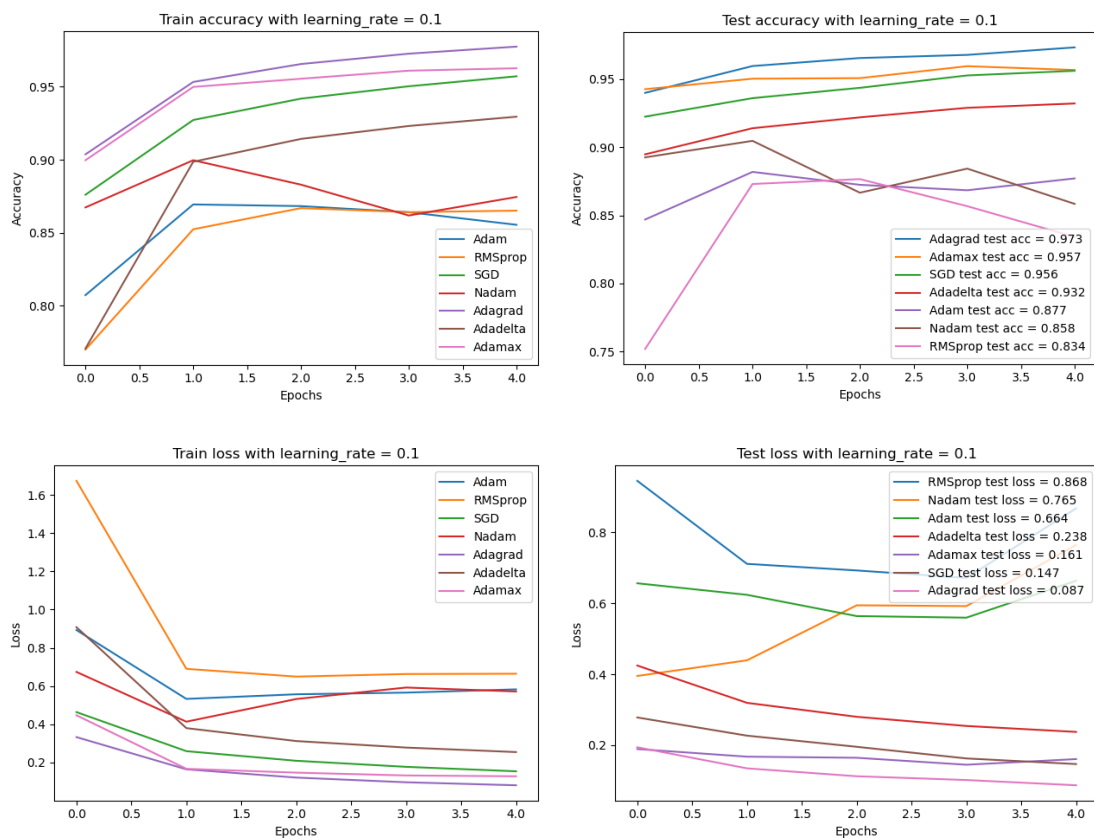


Рисунок 6 – Результаты при learning rate = 0.1.

Наилучший результат принадлежит Adagrad.

Результаты при learning rate = 1:

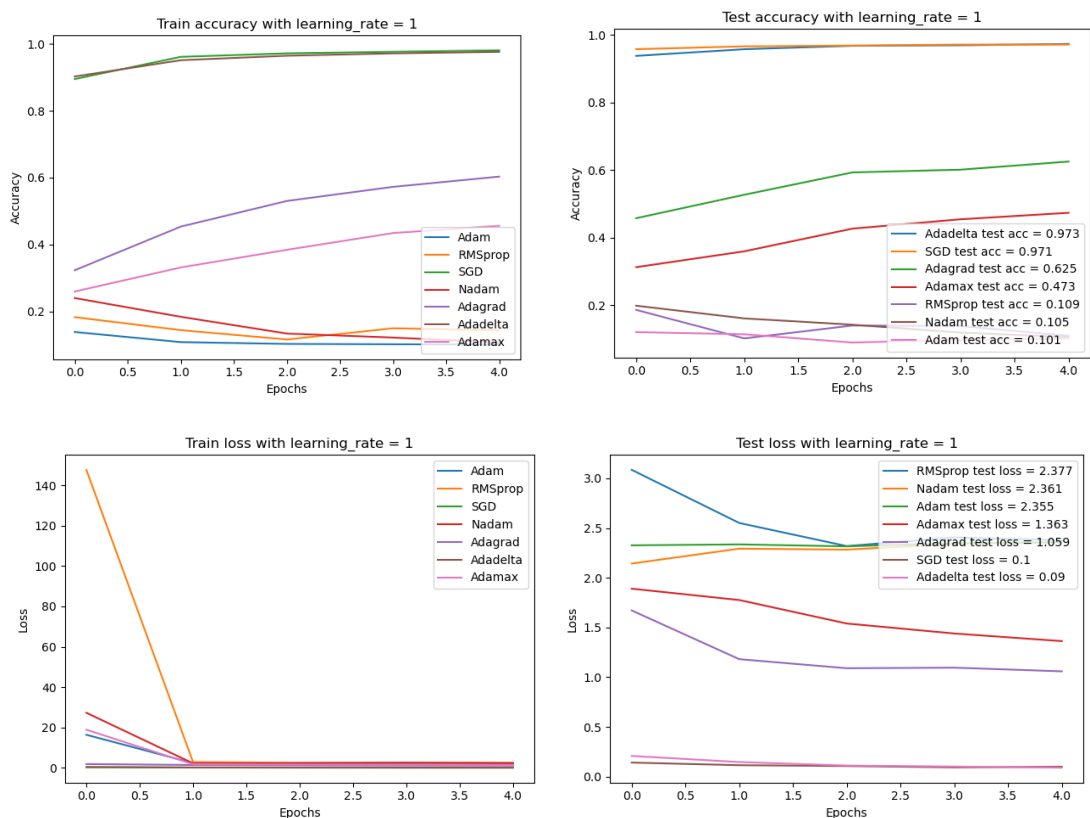


Рисунок 7 – Результаты при learning rate = 1.

Наилучший результат принадлежит Adadelta.

Из полученных результатов можно сделать вывод, что для данной задачи при меньшей скорости обучения (ближе к 0) лучшие результаты показывают RMSProp и Adam, а при скорости ближе к 1 лучшие результаты показывает Adadelta и SGD, а при learning rate = 0.01 лучшие результаты показывает Adamax и RMSProp.

3) Написать функцию, которая позволит загружать пользовательское изображение не из датасета.

Надо помнить, что картинка должна быть 28x28px и черно-белой. `train_images.shape=(60000, 28, 28)`, поэтому надо добавить к массиву, описывающему картинку с цифрой еще измерение слева, перед тем как отдавать этот массив в `model.predict()`. Код задания описан в функции `testImage(model)`.

Вывод.

В ходе выполнения данной работы были изучено влияние оптимизаторов и их показателей скорости обучения на результат обучения модели, а также заданная модель была протестирована на собственных данных, а не из датасета. При меньшей скорости обучения (ближе к 0) лучшие результаты показывают RMSProp и Adam, а при скорости ближе к 1 лучшие результаты показывает Adadelta и SGD, а при learning rate = 0.01 лучшие результаты показывает Adamax и RMSProp.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
from keras.layers import Dense

from tensorflow.keras.optimizers import * # import all
from keras.models import Sequential
from keras.datasets import boston_housing
import numpy as np
import matplotlib.pyplot as plt

import numpy as np
from tensorflow.keras.layers import Dense, Activation, Flatten
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets import boston_housing
import tensorflow as tf
from keras import backend as K
from PIL import Image #pip install Pillow
from numpy import newaxis

learning_rate = 0.01
mnist = tf.keras.datasets.mnist
optimizers_names_arr =
["Adam", "RMSprop", "SGD", "Nadam", "Adagrad", "Adadelata", "Adamax"]
optimizers_arr = [Adam(learning_rate, beta_1=0.9, beta_2=0.999,
amsgrad=False),
                  RMSprop(learning_rate, rho=0.9),
                  SGD(learning_rate, momentum=0.0, nesterov=False),
                  Nadam(learning_rate, beta_1=0.9, beta_2=0.999),
                  Adagrad(learning_rate),
                  Adadelata(learning_rate, rho=0.95),
                  Adamax(learning_rate, beta_1=0.9, beta_2=0.999)]

def build_model(optimizer):
    model = Sequential()
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    model.compile(optimizer, loss='categorical_crossentropy',
metrics=['accuracy'])
    return model

def sortSecond(val):
    return val[1]

def sortFifth(val):
```

```

    return val[4]

def plots(histories, test_acc):
    #print(test_acc)
    test_acc = np.around(test_acc, 3)
    for history in histories:
        plt.plot(history.history['acc'])
        plt.title('Train accuracy with learning_rate = ' + str(learning_rate))
        plt.ylabel('Accuracy')
        plt.xlabel('Epochs')
        plt.legend(optimizers_names_arr, loc='lower right')
        plt.show()

    optimizers_names_arr2 = []
    ona2 = []

    val_acc_history = []
    for history in histories:
        val_acc_history.append(history.history['val_acc'])
    val_acc_history.sort(key=sortFifth, reverse=True)
    for history in val_acc_history:
        plt.plot(history)
    i = 0
    while i < len(optimizers_names_arr):
        optimizers_names_arr2.append((optimizers_names_arr[i] + "
test acc = ", test_acc[i][1]))
        i = i + 1
    #print(optimizers_names_arr2)
    optimizers_names_arr2.sort(key=sortSecond, reverse=True)
    #print(optimizers_names_arr2)
    i = 0
    while i < len(optimizers_names_arr):
        ona2.append(optimizers_names_arr2[i][0] +
str(optimizers_names_arr2[i][1]))
        i += 1
        plt.title('Test accuracy with learning_rate = ' + str(learning_rate))
        plt.ylabel('Accuracy')
        plt.xlabel('Epochs')
        plt.legend(ona2, loc='lower right')
        plt.show()

    for history in histories:
        plt.plot(history.history['loss'])
    plt.title('Train loss with learning_rate = ' + str(learning_rate))
    plt.ylabel('Loss')
    plt.xlabel('Epochs')
    plt.legend(optimizers_names_arr, loc='upper right')
    plt.show()

```

```

optimizers_names_arr3 = []
ona3 = []

val_loss_history = []
for history in histories:
    val_loss_history.append(history.history['val_loss'])
val_loss_history.sort(key=sortFifth, reverse=True)
for history in val_loss_history:
    plt.plot(history)

i = 0
while i < len(optimizers_names_arr):
    optimizers_names_arr3.append((optimizers_names_arr[i]+ "
test loss = ", test_acc[i][0]))
    i = i + 1
#print(optimizers_names_arr3)
optimizers_names_arr3.sort(key=sortSecond, reverse=True)
#print(optimizers_names_arr3)
i = 0
while i < len(optimizers_names_arr):
    ona3.append(optimizers_names_arr3[i][0] +
str(optimizers_names_arr3[i][1]))
    i += 1
plt.title('Test loss with learning_rate = '+str(learning_rate))
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(ona3, loc='upper right')
plt.show()
return

def test_optimizers():
    histories = []
    test_acc = []
    for optimizer in optimizers_arr:
        model = build_model(optimizer)
        histories.append(model.fit(train_images, train_labels,
epochs=5, batch_size=128, validation_data=(test_images,
test_labels)))
        test_acc.append(model.evaluate(test_images, test_labels))
        K.clear_session() # it will destroy keras object
    plots(histories, test_acc)
    return

def singleModelPlots(history):
    title = []
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
    plt.title('Accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')

```

```

plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
return

def testImage(model):
    image_names = ['img1.bmp', 'img2.bmp', 'img3.bmp']
    for image in image_names:
        img = Image.open(image).convert("L") # translating a color
image to black and white (mode "L")
        image_array = np.asarray(img) / 255
        image = (image_array)[newaxis, :, :] # increase the
dimension of the existing array
        print(np.argmax(model.predict(image)))
    return

if __name__ == '__main__':
    (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 = build_model('adam')

    test_loss, test_acc = model.evaluate(test_images, test_labels)
    #history = model.fit(train_images, train_labels, epochs=5,
batch_size=128, validation_data=(test_images, test_labels))

    print('test_acc:', test_acc)
    #print(history.history)
    singleModelPlots(model.fit(train_images, train_labels, epochs=5,
batch_size=128, validation_data=(test_images, test_labels)))

    test_optimizers()
    testImage(model)

    #array = np.array([[1,2,3],[4,5,6]])
    #print(array)
    #print(((array)[newaxis,:,:]).shape)

```