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. 7383 _____ Бергалиев М.

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. 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

Ход работы.

0. Исходный код программы представлен в приложении.
1. Выберем модель сети. Возьмем один слой с 256 нейронами с функцией активации relu и 10 нейронов с softmax. В качестве оптимизатора будем использовать adam. Будем обучать 5 эпох с размером батча 128. Ошибка и точность во время обучения показаны на рис. 1, 2.

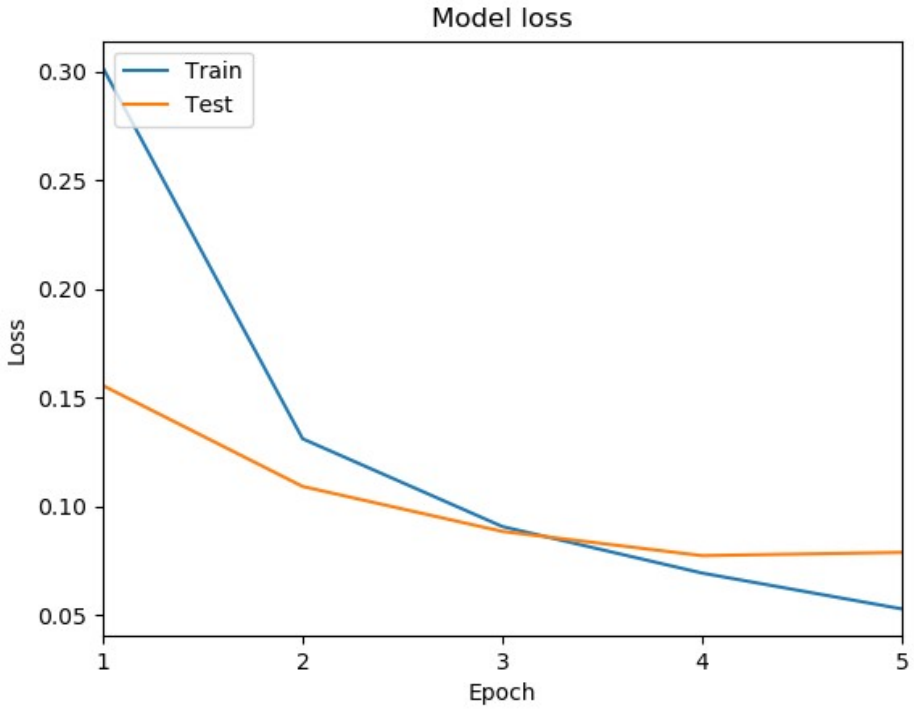

Рисунок 1 — График ошибок во время обучения модели

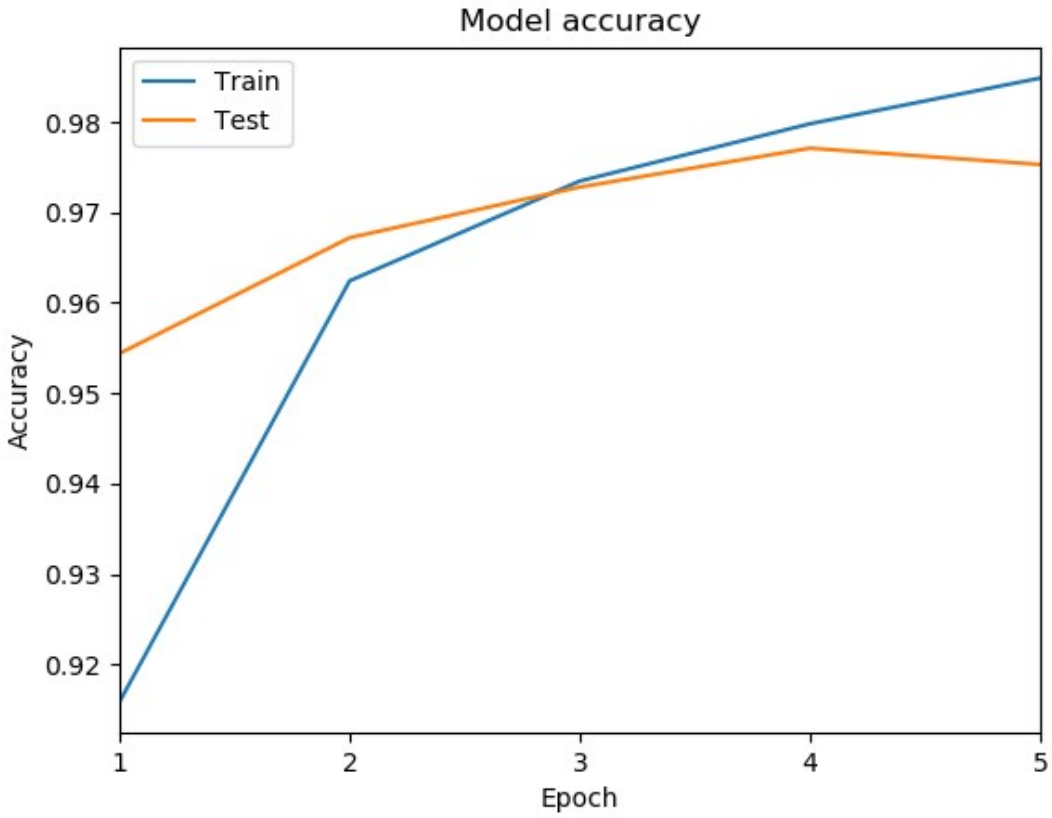

Рисунок 2 — График точности во время обучения модели

Как видно, точность обученной модели составляет не менее 97%.

2. Изучим влияние различных оптимизаторов на обучение модели. Будем рассматривать следующие: RMSprop, Adam, Adagrad, Adadelta, SGD. протестируем с различными значениями параметра скорости обучения. Результаты показаны на рис. 3-8.

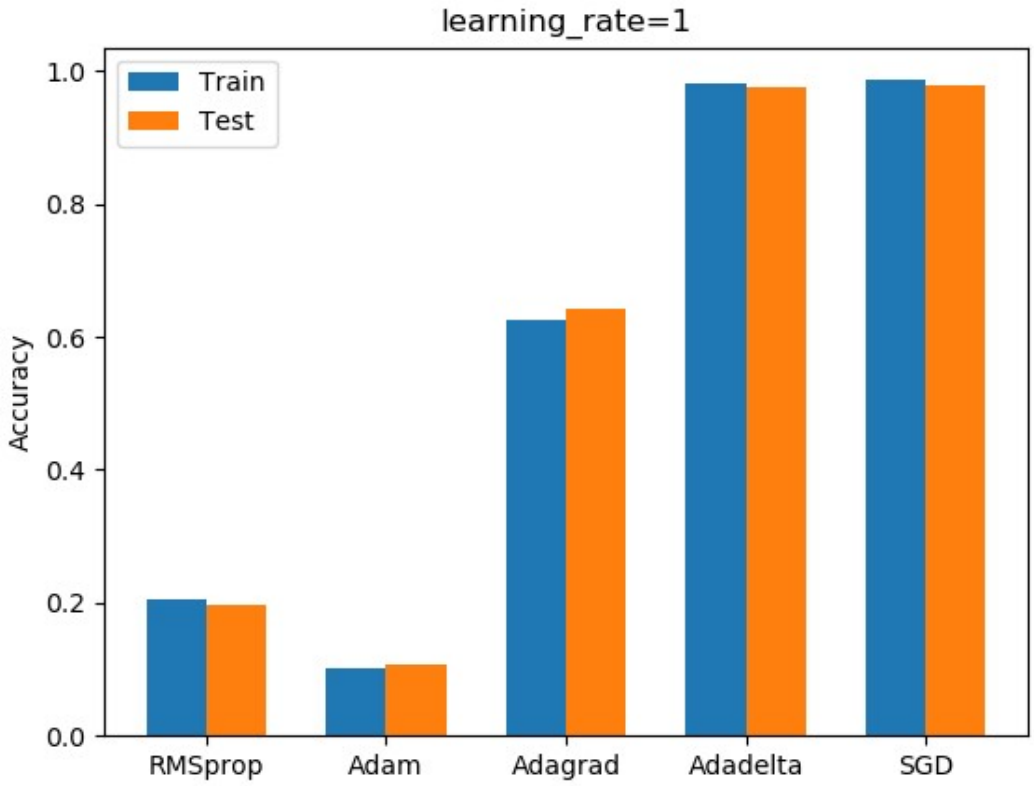

Рисунок 3 — Точность обучения при скорости обучения 1

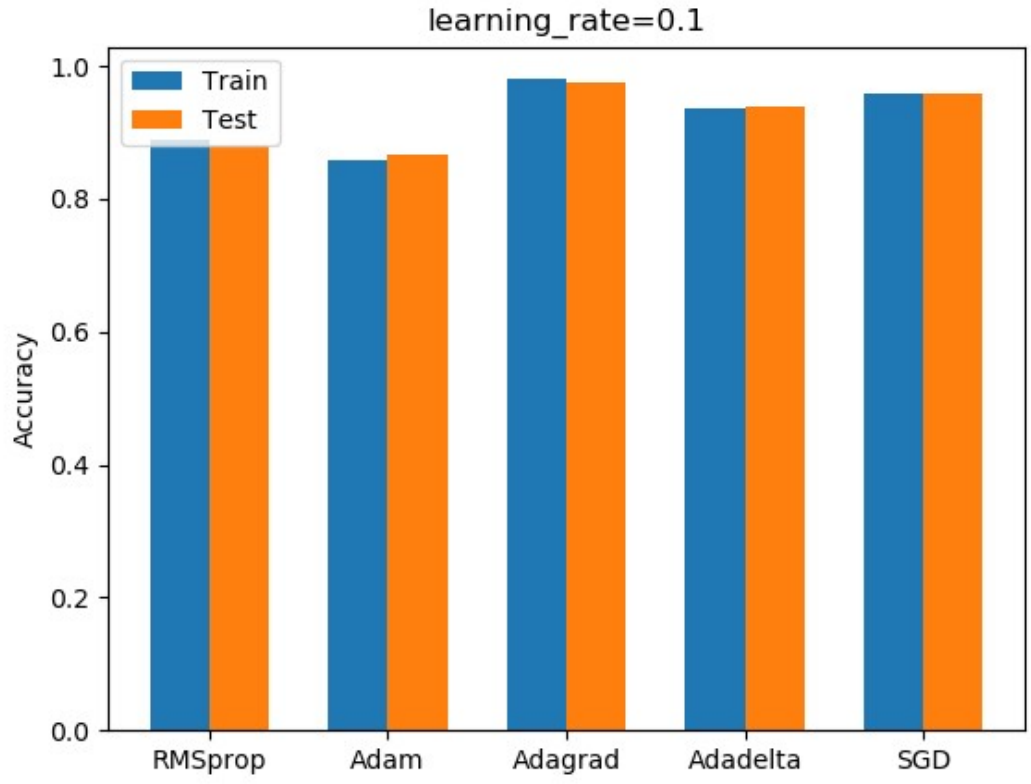

Рисунок 4 — Точность обучения при скорости обучения 0.1

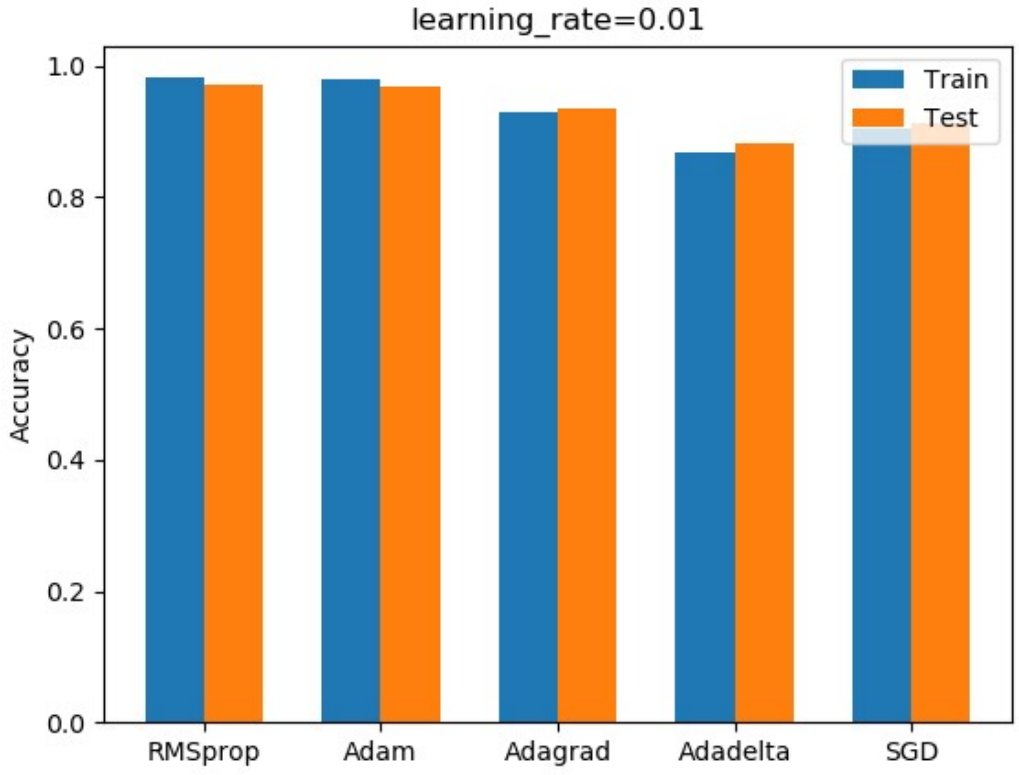

Рисунок 5 — Точность обучения при скорости обучения 0.01

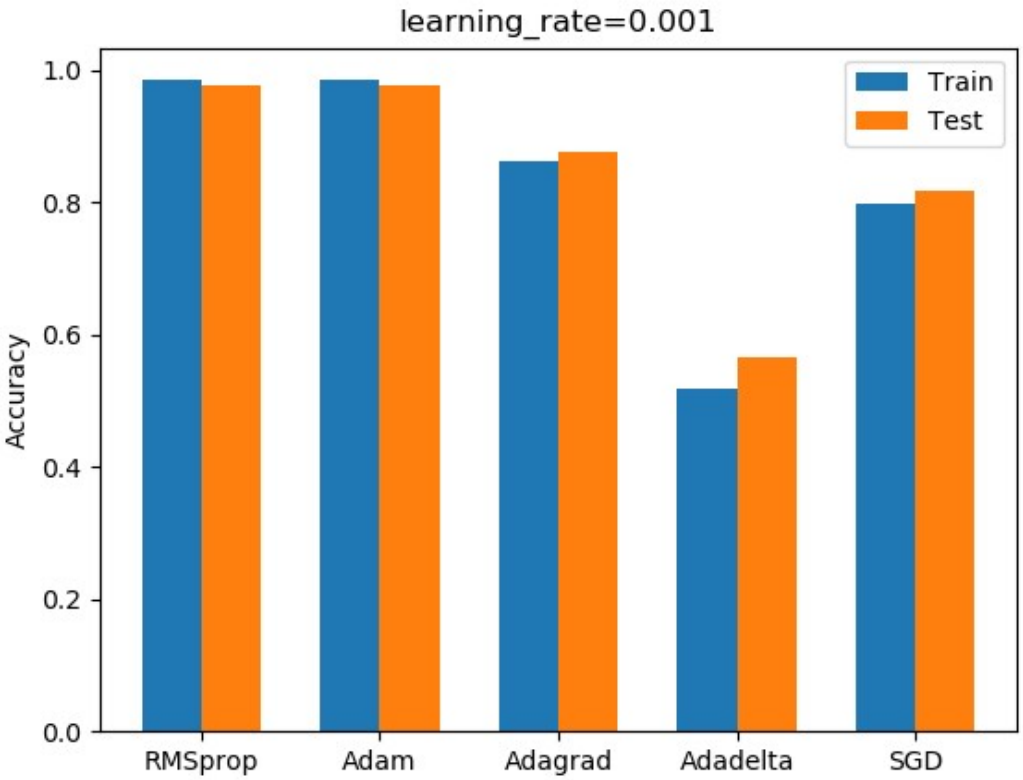

Рисунок 6 — Точность обучения при скорости обучения 0.001

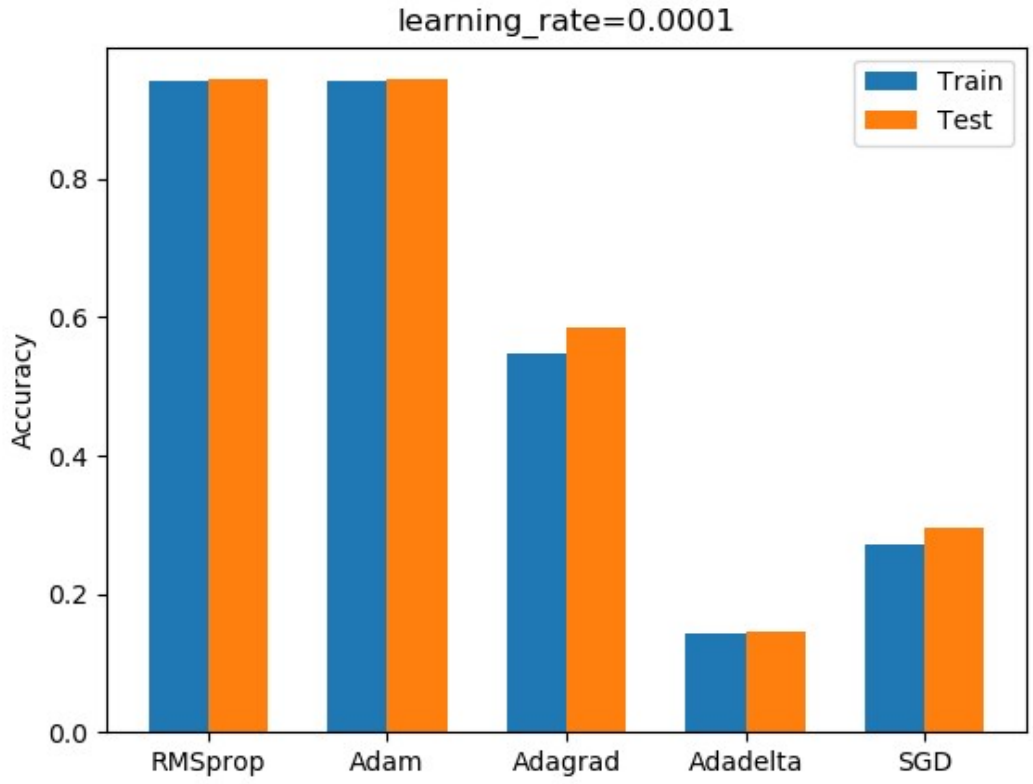

Рисунок 7 — Точность обучения при скорости обучения 0.0001

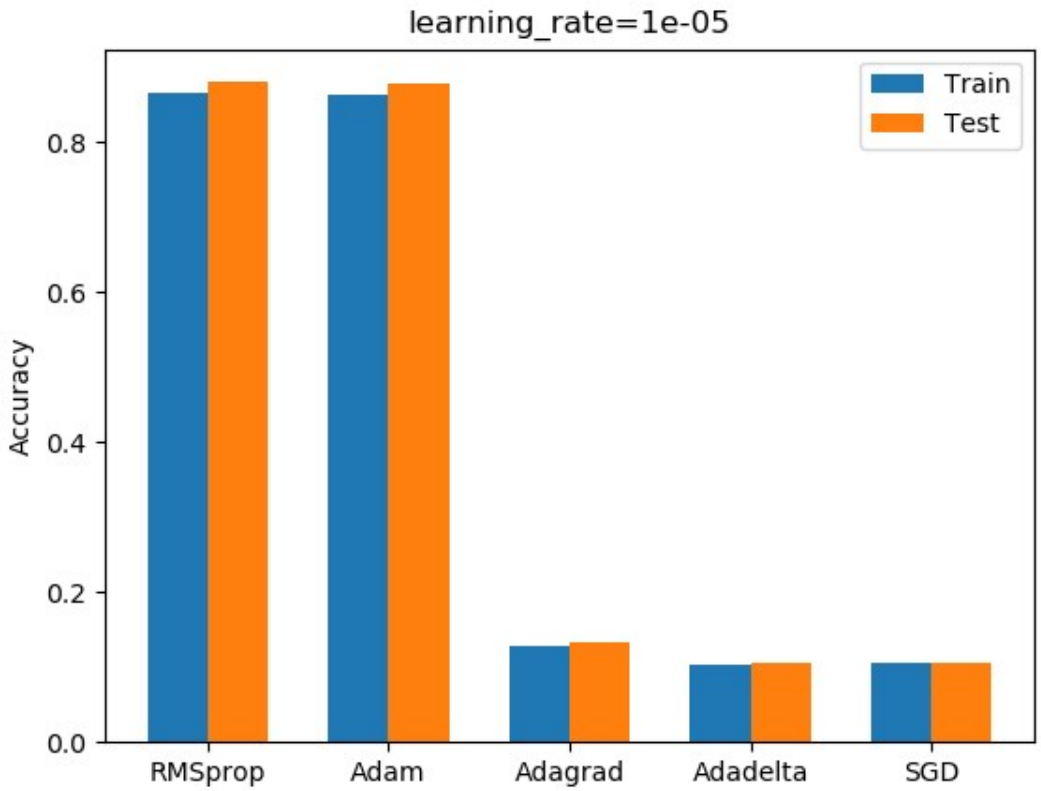

Рисунок 8 — Точность обучения при скорости обучения 0.00001

Как видно, Adadelta и SGD хорошо показывают себя при относительно больших значениях скорости обучения. Adagrad показывает себя лучше при средних значениях, а Adam и RMSprop при малых.

3. Была написана функция `loadImage`, которая загружает пользовательское изображение. Ее код представлен в приложении.

Выводы: Было изучено влияние различных оптимизаторов на обучение. Лучшие результаты дают Adam и RMSprop со скоростью обучения 0.001, Adagrad со скоростью 0.1, Adadelta и SGD со скоростью 1.

ПРИЛОЖЕНИЕ

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

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)

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

     model.compile(optimizer=optimizer,loss='categorical_crossentro
py', metrics=['accuracy'])
    return model

def test_model():
    model = build_model('adam')
       history  =  model.fit(train_images,  train_labels,  epochs=5,
batch_size=128, validation_data=(test_images, test_labels))
    x = range(1, 6)
    plt.plot(x, history.history['loss'])
    plt.plot(x, history.history['val_loss'])
    plt.title('Model loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.xlim(x[0], x[-1])
    pylab.xticks(x)
    plt.legend(['Train', 'Test'], loc='upper left')
```

```
    plt.show()
    plt.plot(x, history.history['accuracy'])
    plt.plot(x, history.history['val_accuracy'])
    plt.title('Model accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.xlim(x[0], x[-1])
    pylab.xticks(x)
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()
    return model

def test_optimizers():
    nrep = 10
    optimizers = [RMSprop, Adam, Adagrad, Adadelta, SGD]
    lrates = [1, 0.1, 0.01, 0.001, 0.0001, 0.00001]
    for lrate in lrates:
        res_train = [[] for _ in optimizers]
        res_test = [[] for _ in optimizers]
        for _ in range(0, nrep):
            for j in range(0, len(optimizers)):
                                model  =  build_model(optimizers[j]
(learning_rate=lrate))
                  history = model.fit(train_images, train_labels,
epochs=5, batch_size=128)
                 res_train[j].append(history.history['accuracy'][-
1])
                 test_loss, test_acc = model.evaluate(test_images,
test_labels, verbose=0)
                res_test[j].append(test_acc)
        res_train = list(map(lambda x: np.mean(x), res_train))
        res_test = list(map(lambda x: np.mean(x), res_test))
              plt.bar(np.arange(len(optimizers))  *  3,  res_train,
width=1)
           plt.bar(np.arange(len(optimizers)) * 3 + 1, res_test,
width=1)
         plt.xticks([3*i + 0.5 for i in range(0,len(optimizers))],
labels=[o.__name__ for o in optimizers])
        plt.title('learning_rate='+str(lrate))
        plt.ylabel('Accuracy')
        plt.legend(['Train', 'Test'])
        plt.savefig(str(lrate)+'.png')
        plt.clf()


def loadImage(filename):
```

```
    image = Image.open(filename)
    image = image.resize((28, 28))
    image = np.dot(np.asarray(image), np.array([1/3, 1/3, 1/3]))
    image /= 255
    image = 1 - image
    image = image.reshape((1, 28, 28))
    return image

if __name__ == '__main__':
    model = test_model()
    image = loadImage('7.png')
    res = model.predict(image)
    print(np.argmax(res))
```