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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Искусственные нейронные сети»

Тема: «Распознавание рукописных символов»

Студентка гр. 8383 _____ Сырцова Е.А.

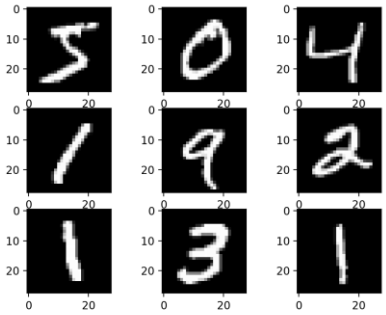
Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель

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

Требования

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

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 — в данном случае нас интересует точность (доля правильно классифицированных изображений).

Сравним работу сети под влиянием различных оптимизаторов и их параметров. На рис. 1-8 представлены графики ошибки и точности для оптимизаторов, а в табл.1-2 представлены значения ошибки и точности:

Таблица 1 – Значения ошибки

	Adagrad	Adam	RMSprop	SGD
0.001	0.5485913157463074	0.07070258259773254	0.06690067797899246	0.06485424190759659
0.01	0.06082149222493172	0.12872655689716634	0.22010715305805206	0.18006804585456848

Таблица 2 – Значения точности

	Adagrad	Adam	RMSprop	SGD
0.001	0.8701000213623047	0.9775999784469604	0.9822999835014343	0.9815999865531921
0.01	0.9822000265120146	0.9715999960899353	0.9749000072479248	0.9801999926567078

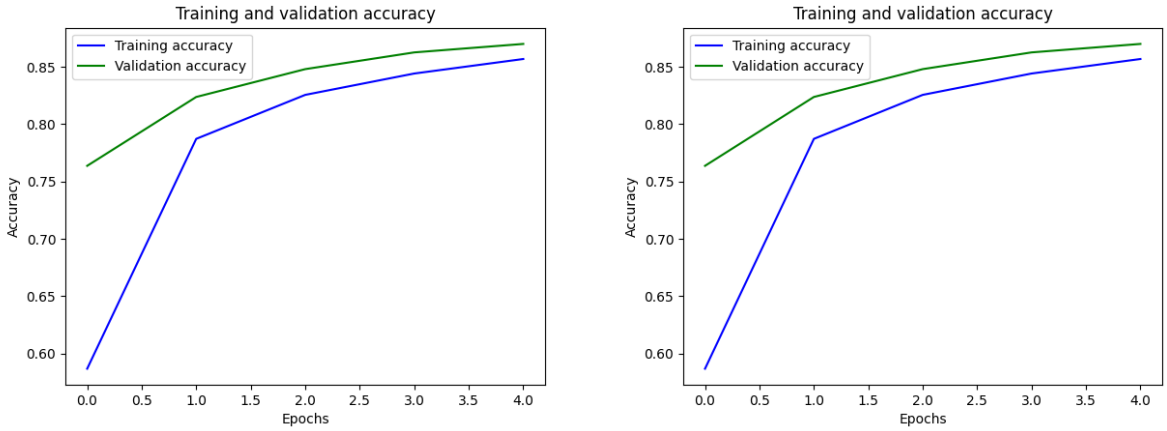


Рисунок 1 – Adagrad, learning_rate = 0.001

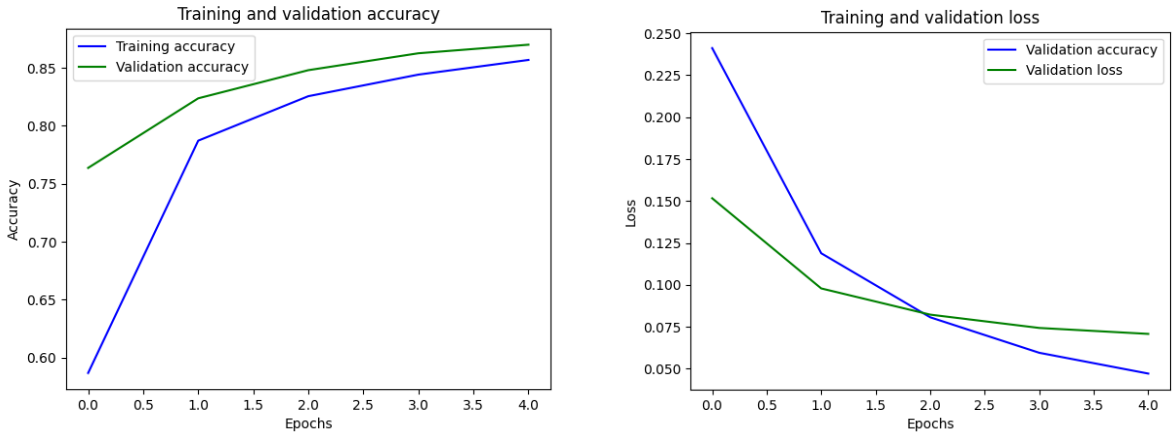


Рисунок 2 – Adam, learning_rate = 0.001

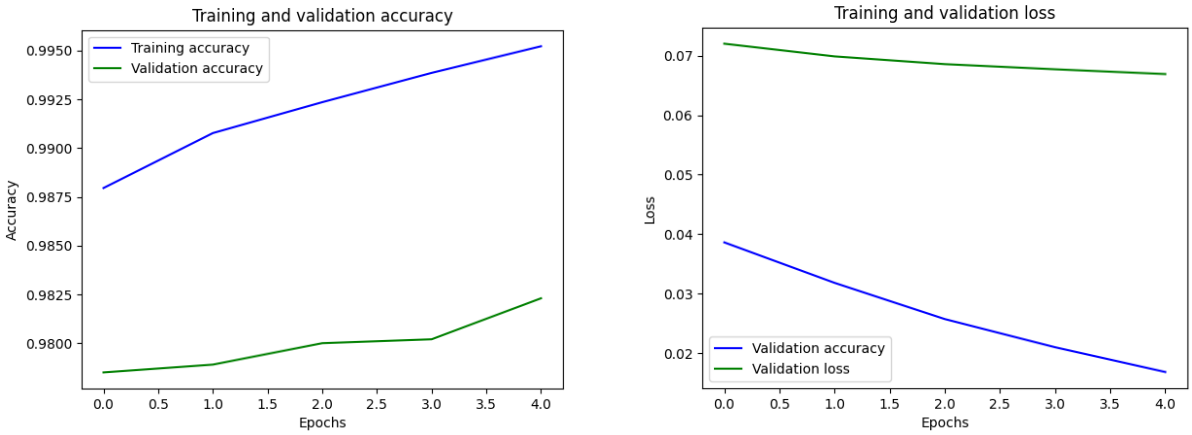


Рисунок 3 – RMSprop, learning_rate = 0.001

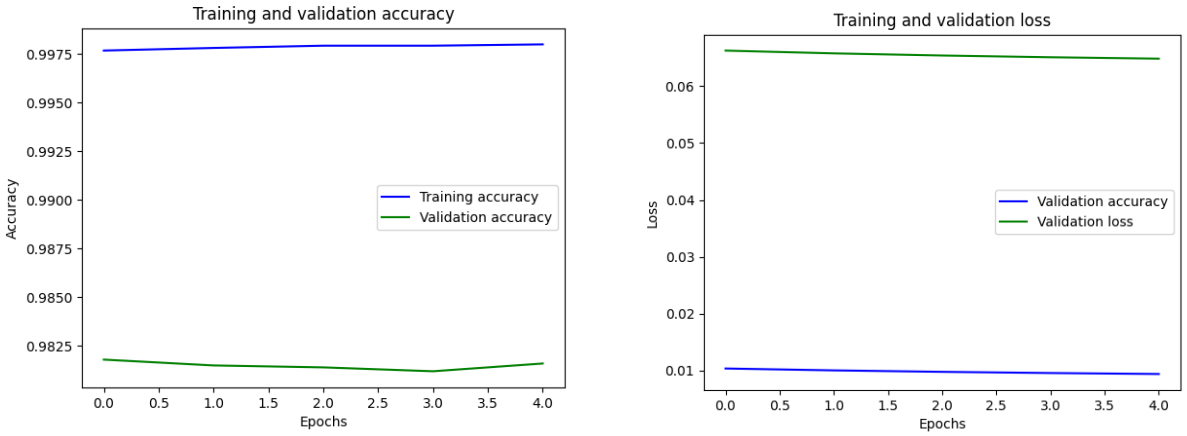


Рисунок 4 – SGD, learning_rate = 0.001

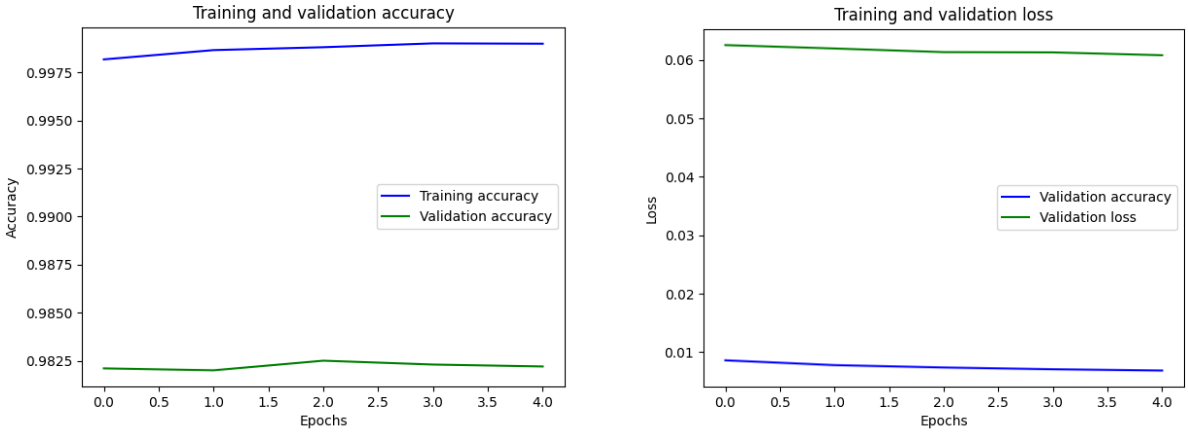


Рисунок 5 – Adagrad, learning_rate = 0.01

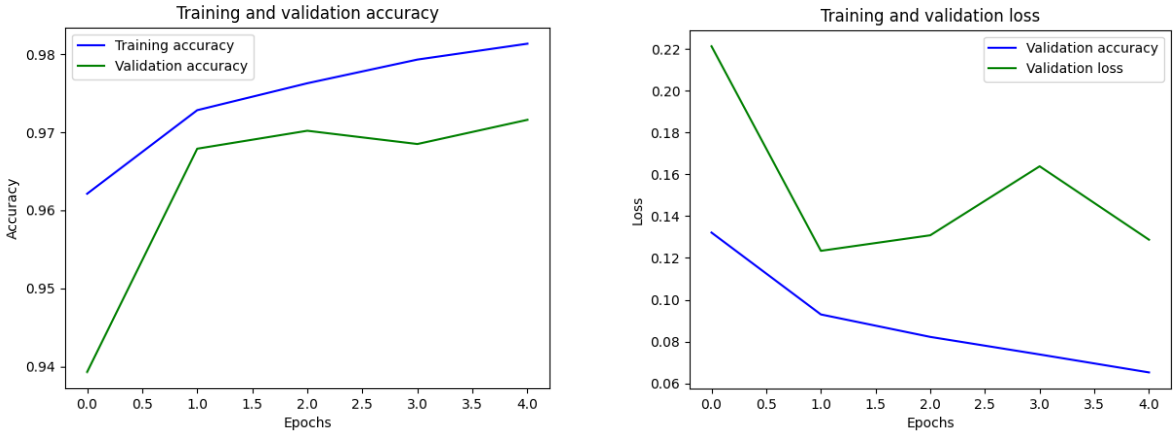


Рисунок 6 – Adam, learning_rate = 0.01

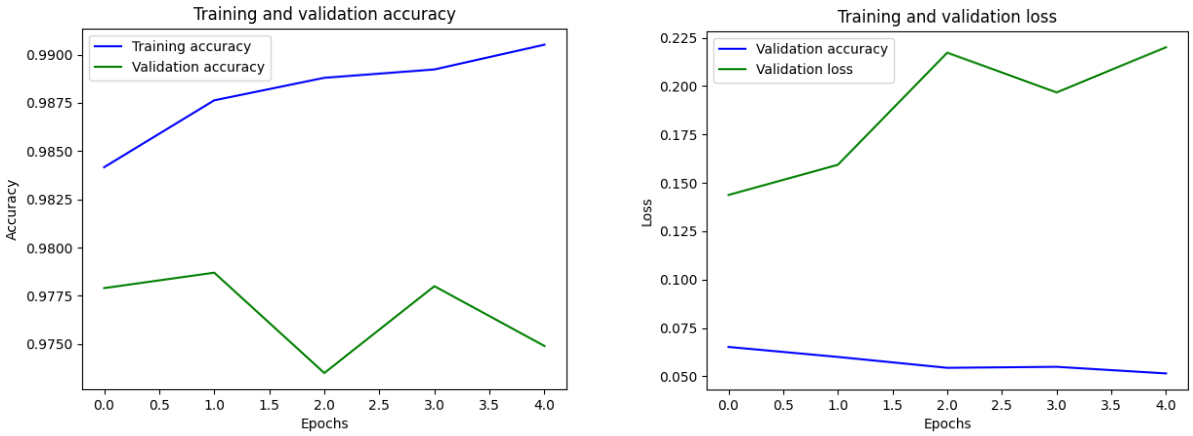


Рисунок 7 – RMSprop, learning_rate = 0.01

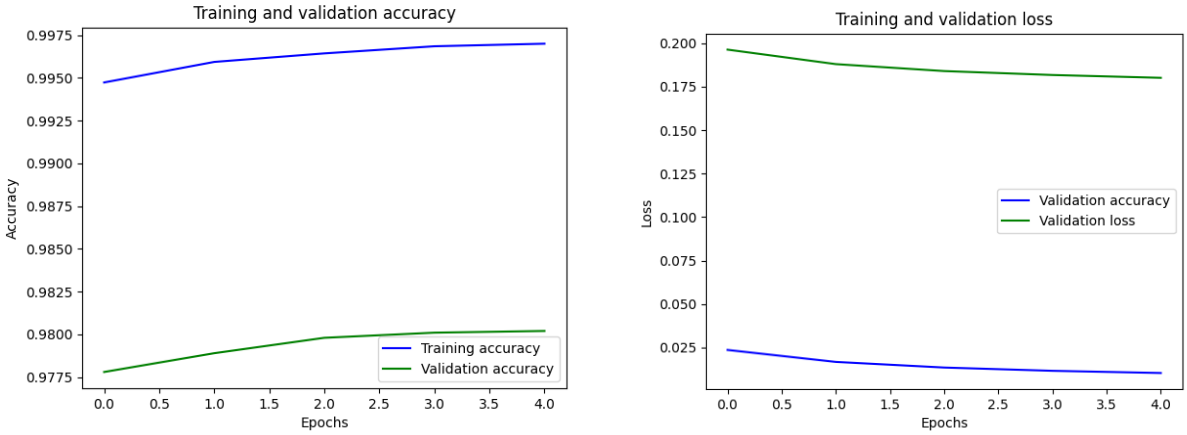
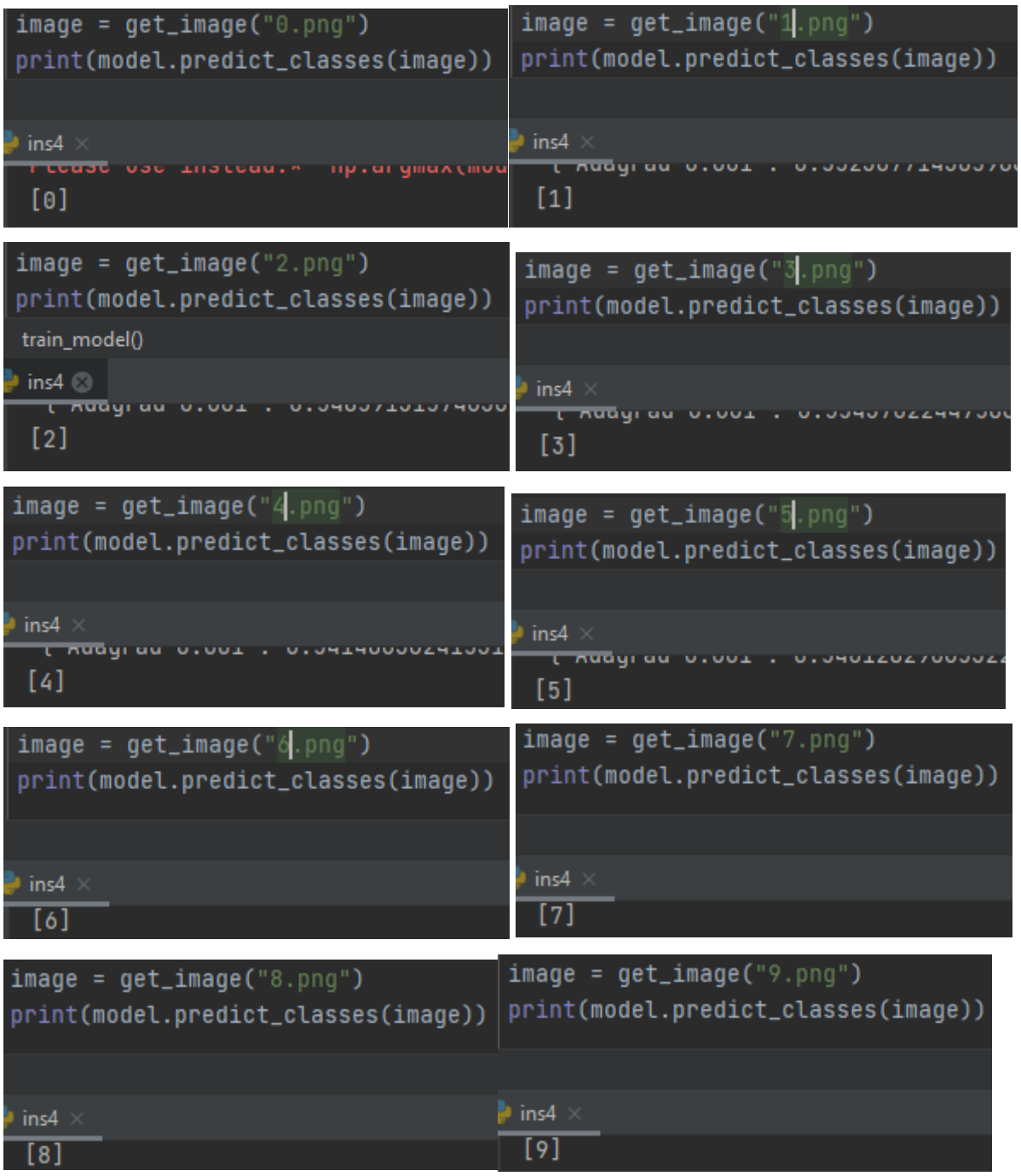


Рисунок 8 – SGD, learning_rate = 0.01

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

Из графиков и значений следует, что наилучшая работа сети наблюдается при заданном оптимизаторе Adagrad с параметром learning_rate = 0.01.

Загрузим пользовательское изображение и проверим результат работы программы при выбранном оптимизаторе Adagrad с параметром learning_rate = 0.01.



Вывод

В процессе 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.

ПРИЛОЖЕНИЕ А

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

res = dict()
los = dict()


def get_image(filename):
    image = Image.open(filename).convert('L')
    image = image.resize((28, 28))
    image = asarray(image) / 255.0
    output = np.expand_dims(image, axis=0)
    return output


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', input_shape=(28 * 28,)))
model.add(Dense(10, activation='softmax'))


def train_model(optimizer):
    optimizer_config = optimizer.get_config()
    model.compile(optimizer=optimizer, loss='categorical_crossentropy',
metrics=['accuracy'])
    history = model.fit(train_images, train_labels, epochs=5, batch_size=128,
                        validation_data=(test_images, test_labels))

    loss, acc = model.evaluate(test_images, test_labels)
    print('test_acc:', acc)
```

```
    plt.title('Training and validation accuracy')
    plt.plot(history.history['accuracy'], 'b', label='Training accuracy')
    plt.plot(history.history['val_accuracy'], 'g', label='Validation
accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()
    plt.savefig("%s_%s_%s_acc.png" % (optimizer_config["name"],
optimizer_config["learning_rate"], acc), format='png')
    plt.clf()

    plt.title('Training and validation loss')
    plt.plot(history.history['loss'], 'b', label='Validation accuracy')
    plt.plot(history.history['val_loss'], 'g', label='Validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
    plt.savefig("%s_%s_%s_loss.png" % (optimizer_config["name"],
optimizer_config["learning_rate"], acc), format='png')
    plt.clf()

    res["%s %s" % (optimizer_config["name"],
optimizer_config["learning_rate"])] = acc
    los["%s %s" % (optimizer_config["name"],
optimizer_config["learning_rate"])] = loss


for learning_rate in [0.001, 0.01]:
    train_model(optimizers.Adagrad(learning_rate=learning_rate))
    train_model(optimizers.Adam(learning_rate=learning_rate))
    train_model(optimizers.RMSprop(learning_rate=learning_rate))
    train_model(optimizers.SGD(learning_rate=learning_rate))


model.compile(optimizer=optimizers.Adagrad(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))

print("accuracy:\n", res)
print("loss:\n", los)

image = get_image("2.png")
print(model.predict_classes(image))
```