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»

Студентка гр. 7381 _____ Давкаева В.С.

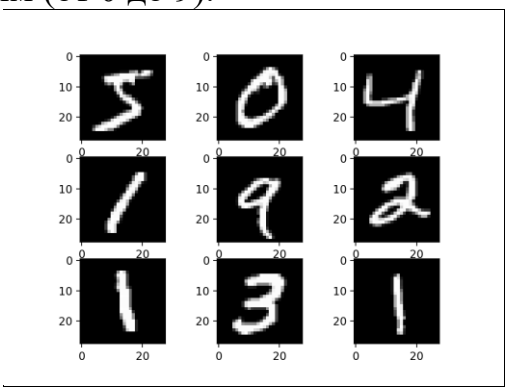
Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

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.

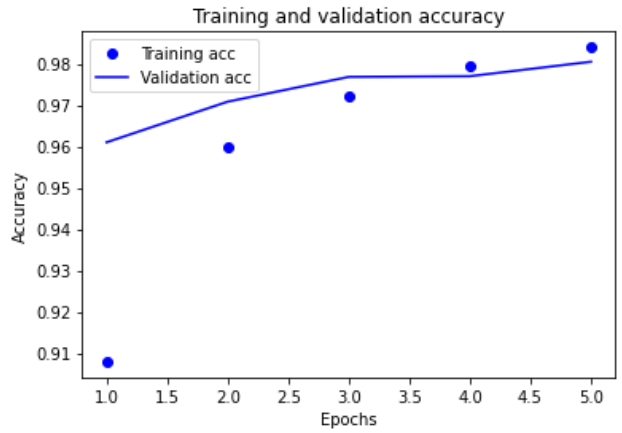
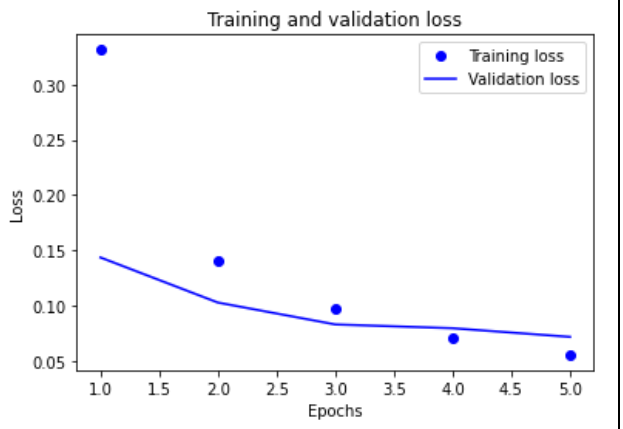
- Ознакомиться с представлением графических данных
- Ознакомиться с простейшим способом передачи графических данных нейронной сети
- Создать модель
- Настроить параметры обучения
- Написать функцию, позволяющая загружать изображение пользователи и классифицировать его

Требования.

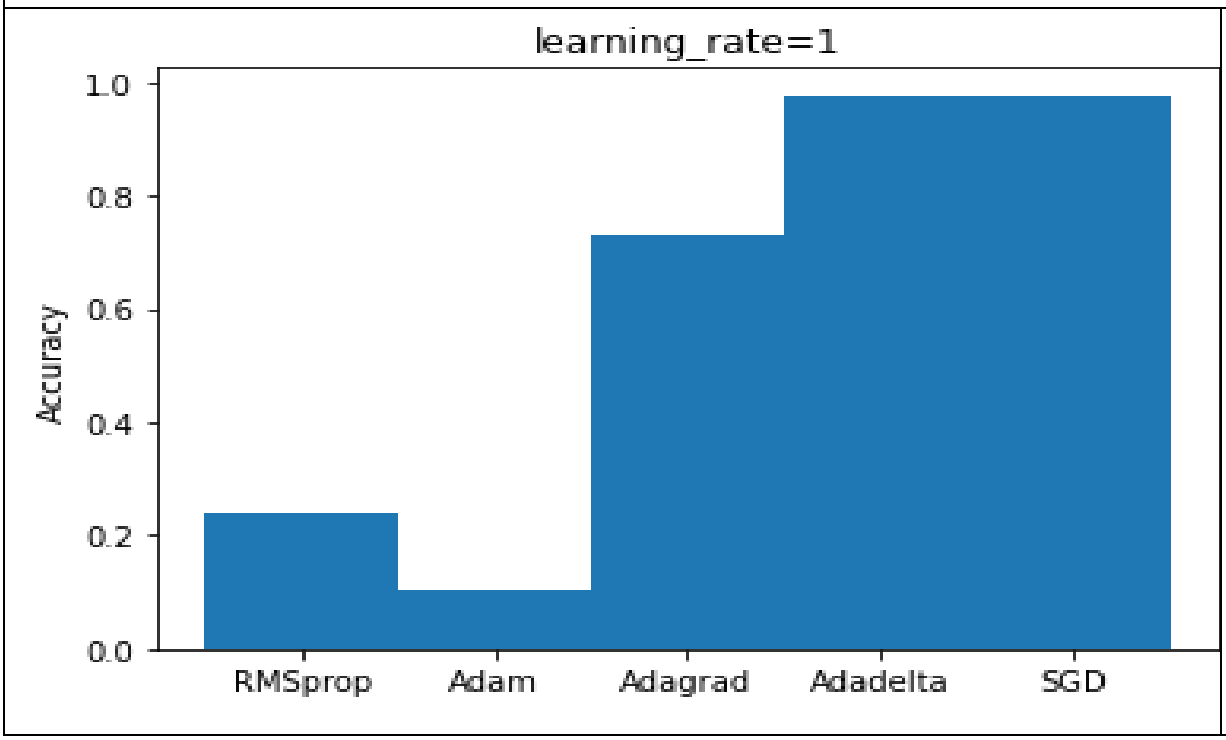
1. Найти архитектуру сети, при которой точность классификации будет не менее 95%
2. Исследовать влияние различных оптимизаторов, а также их параметров, на процесс обучения
3. Написать функцию, которая позволит загружать пользовательское изображение не из датасета

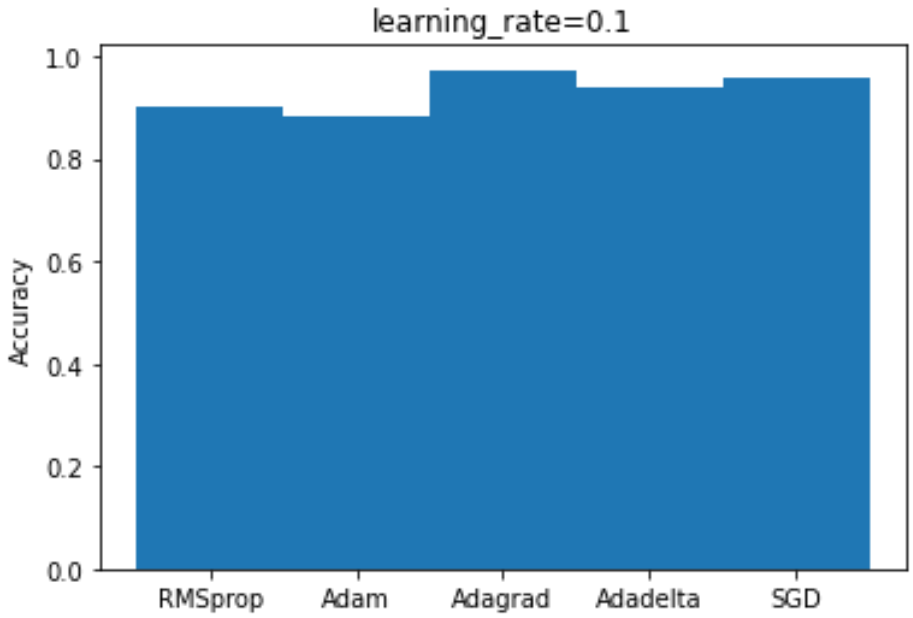
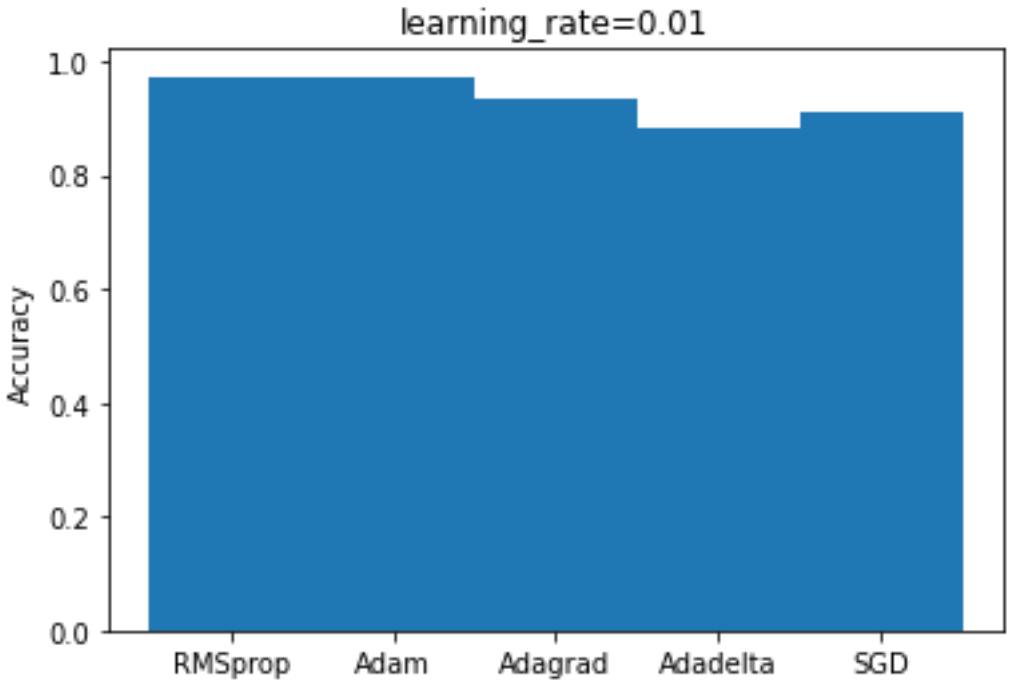
Ход работы.

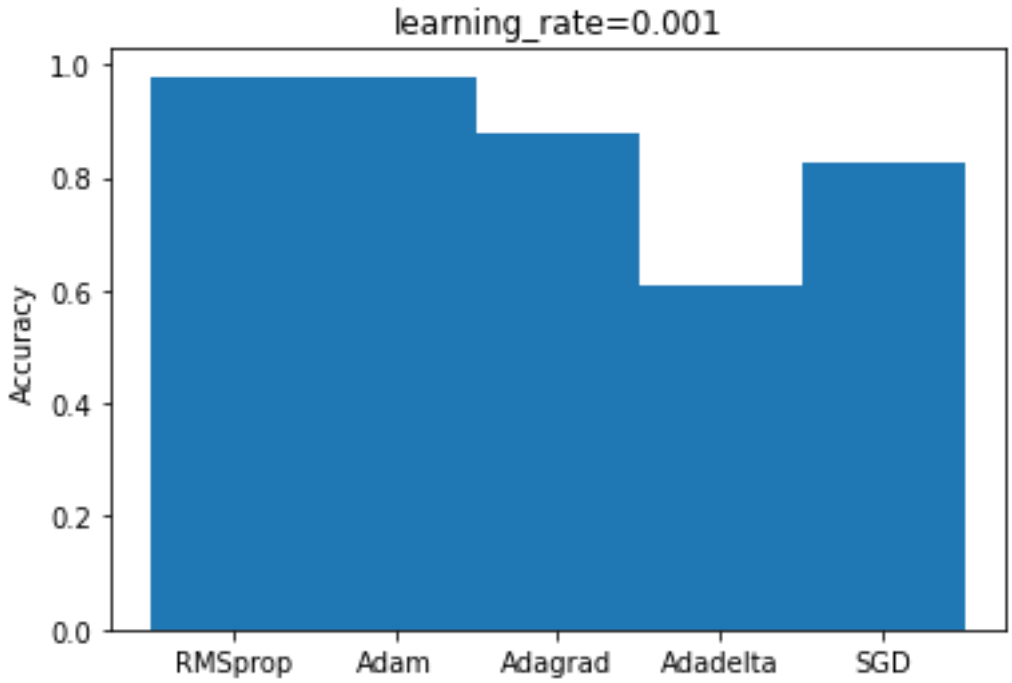
Была выбрана модель с 256 нейронами на входном слое и оптимизатором adam. Модель обучается на 5 эпохах. Ниже представлены графики точности и потерь.

График точности	График потерь
	

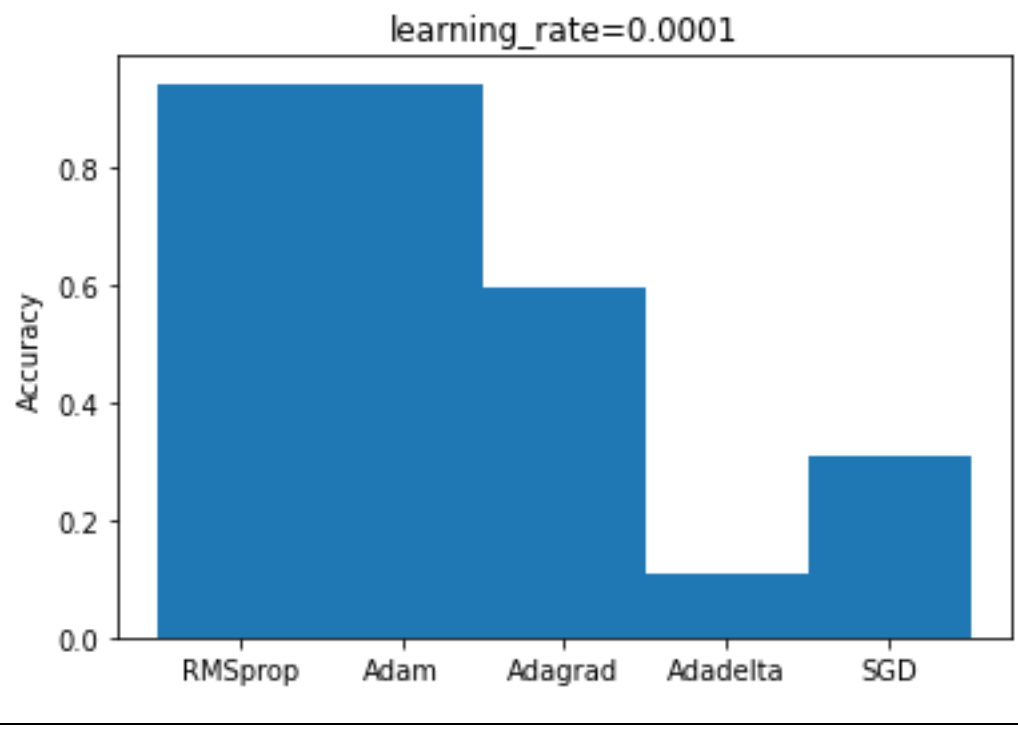
Изучение влияния RMSprop, Adam, Adagard, Adadelta,SGD оптимизаторов на обучение модели с различными значениями скорости обучения.

Скорость обучения = 1


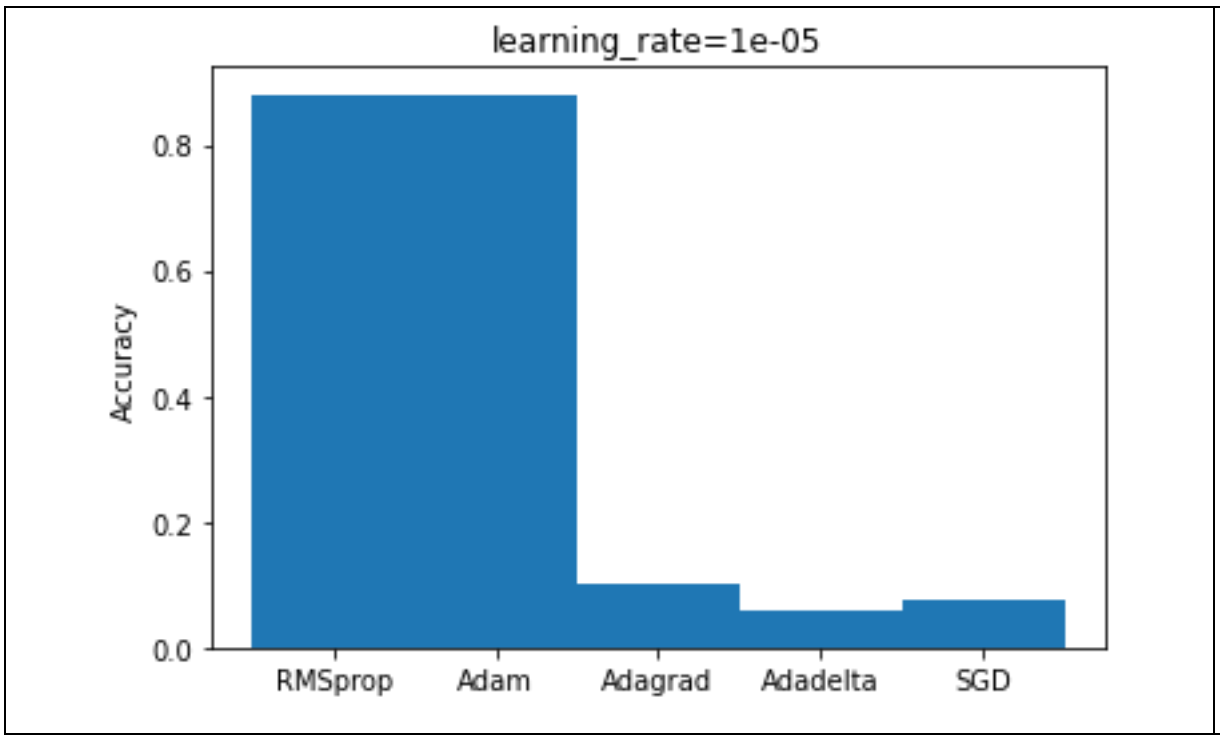
Скорость обучения = 0.1

Скорость обучения = 0.01

Скорость обучения = 0.001



Скорость обучения = 0.0001



Скорость обучения = 0.00001


learning_rate=1e-05
Accuracy
0.0
0.2
0.4
0.6
0.8
RMSprop
Adam
Adagrad
Adadelta
SGD

Выводы.

RMSprop и Adam показывают более точные результаты при малых значениях скорости обучения, Adadelta и SGD при больших значениях, Adagrad при средних.

ПРИЛОЖЕНИЕ

Исходный код

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

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_crossentropy',
metrics=['accuracy'])
    return model

def loadImg(path):
    img = Image.open(path)
    img = img.resize((28, 28))
    img = np.dot(np.asarray(img), np.array([1/3, 1/3, 1/3]))
```

```
    img /= 255
    img = 1 - img
    return img.reshape((1, 28, 28))

def prediction(path, model):
    return np.argmax(model.predict(loadImg(path)))

model = build_model('adam')

h = model.fit(train_images, train_labels, epochs=5, batch_size=128,
validation_split=0.1)
print(h.history.keys())
epochs = range(1, len(h.history['loss']) + 1)
plt.plot(epochs, h.history['loss'], 'bo', label='Training loss')
plt.plot(epochs, h.history['val_loss'], 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.clf()
plt.plot(epochs, h.history['accuracy'], 'bo', label='Training acc')
plt.plot(epochs, h.history['val_accuracy'], 'b', label='Validation
acc')
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_acc:', test_acc)
```

```
print(prediction("/2.jpg", model))

optimizers = [RMSprop, Adam, Adagrad, Adadelta, SGD]
lrates = [1, 0.1, 0.01, 0.001, 0.0001, 0.00001]
for lrate in lrates:
    acc = []
    for i in range(len(optimizers)):
        model = build_model(optimizers[i](learning_rate=lrate))
        history = model.fit(train_images, train_labels, epochs=5,
batch_size=128)
        test_loss, test_acc = model.evaluate(test_images, test_labels,
verbose=0)
        acc.append(test_acc)
    labeles_x = [o.__name__ for o in optimizers]
    plt.bar(labeles_x, acc, width=1)
    plt.title('learning_rate='+str(lrate))
    plt.ylabel('Accuracy')
    plt.show()
    plt.clf()
```