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. 8382 _____ Рочева А.К.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

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

Задание.

- 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

Выполнение работы.

Для выполнения работы был использован набор данных MNIST, входящий в состав Keras.

Была выбрана такая архитектура сети:

```
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

Эта архитектура полностью удовлетворяет условию точности классификации не менее 95 % (что подтверждается результатами обучения при различных оптимизаторах).

Для дальнейшей работы были выбраны 6 оптимизаторов с двумя разными наборами параметров в каждом:

1. optimizers.Adam(learning_rate=0.001,beta_1=0.9, beta_2=0.999, amsgrad=False),
2. optimizers.Adam(learning_rate=0.002, beta_1=0.9, beta_2=0.9, amsgrad=True),
3. optimizers.Adamax(learning_rate=0.001, beta_1=0.9, beta_2=0.999),
4. optimizers.Adamax(learning_rate=0.002, beta_1=0.9, beta_2=0.9),

5. optimizers.Nadam(learning_rate=0.001, beta_1=0.9, beta_2=0.999),
6. optimizers.Nadam(learning_rate=0.002, beta_1=0.9, beta_2=0.9),
7. optimizers.SGD(learning_rate=0.01, momentum=0.0, nesterov=False),
8. optimizers.SGD(learning_rate=0.03, momentum=0.1, nesterov=True),
9. optimizers.Adadelta(learning_rate=1.0, rho=0.95),
10. optimizers.Adadelta(learning_rate=1.1, rho=0.99),
11. optimizers.RMSprop(learning_rate=0.001, rho=0.9),
12. optimizers.RMSprop(learning_rate=0.005, rho=0.92)

Построение и обучение модели:

```
def compile_model(model, optimizer):
    model.compile(optimizer=optimizer,
loss='categorical_crossentropy', metrics=['accuracy'])
    history = model.fit(train_images, train_labels, epochs=5,
batch_size=128, verbose=0)
    test_loss, test_acc = model.evaluate(test_images,
test_labels)
    train_acc = history.history['accuracy'][-1]
    train_loss = history.history['loss'][-1]
    print('test_acc:', test_acc)
    return [train_acc, train_loss, test_acc, test_loss]
```

Результаты обучения представлены на рисунках 1 и 2.

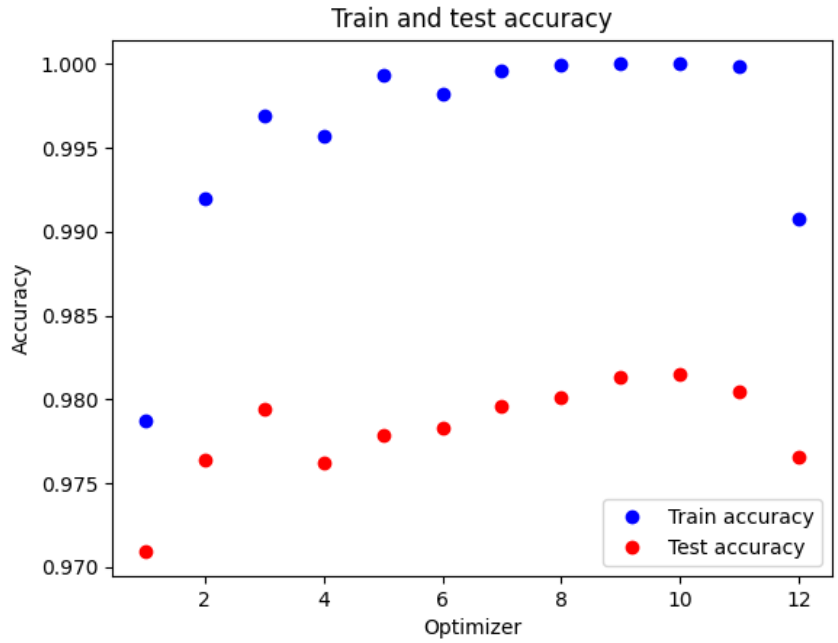

Рис. 1 — значения точности сетей при различных оптимизаторах

Проанализируем полученные данные. Сети, использующие оптимизаторы Nadam (с параметрами learning_rate=0.001, beta_1=0.9, beta_2=0.999), SGD, Adadelta и RMSprop (с параметрами learning_rate=0.001, rho=0.9) показали самую высокую точность (номера 5, 7-11). При этом точность на тестовых данных самая высокая в сетях, использующих оптимизатор Adadelta (9-10).

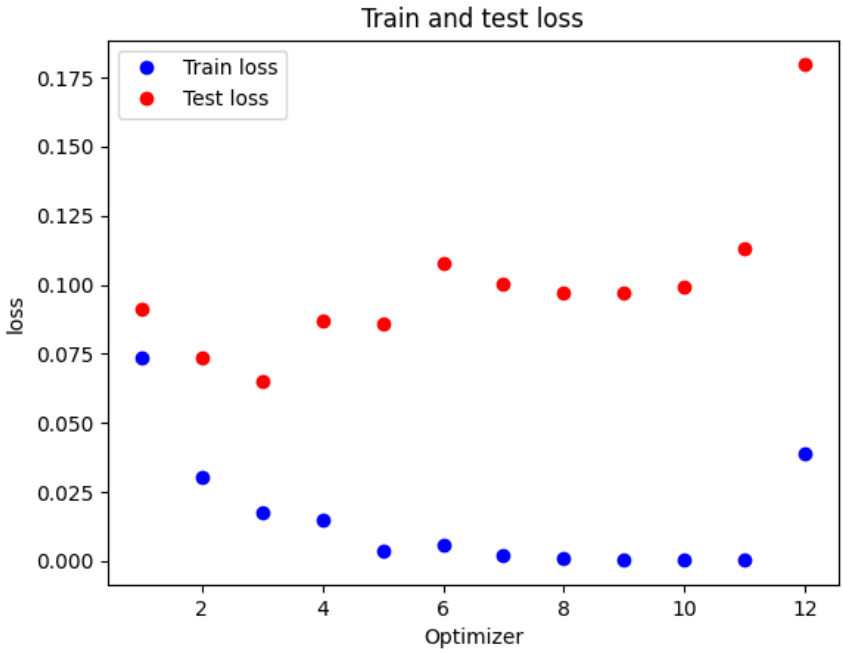

Рис. 2 — значение ошибок при работе сети

Из полученного графика видно, что при обучении наименьшие потери показывают сети с оптимизаторами Nadam, SGD, Adadelta и RMSprop (с параметрами learning_rate=0.001, rho=0.9). При этом на тестовых данных наименьшие потери наблюдаются при использовании оптимизатора Adamax(с параметрами learning_rate=0.001, beta_1=0.9, beta_2=0.999).

Из полученных данных можно сделать вывод, что для работы сети для распознавания символов можно использовать оптимизатор Adadelta (с параметрами learning_rate=1.0, rho=0.95), т. к. при его использовании точность на тестовых данных максимальна и потери достаточно низкие по сравнению с теми оптимизаторами, точность сети с которыми так же является максимальной.

Функция для загрузки пользовательского изображения:

```
def load_img(filename):
    image = Image.open(filename).convert('L')
    image = 255 - np.array(image)
    image = image/255
    return np.expand_dims(image, axis=0)
```

Результаты тестирования сети:

Изображение	Предсказание
	1
	2
	3
	4
	5
	6

	3
	8
	9

Из таблицы видно, что модель ошиблась только при распознавании цифры 7 (при нескольких запусках программы сеть выдает ответ 7/2/3). Это может быть связано с тем, как в тестовых данных писались цифры 7, 2 и 3.

Выводы.

В ходе выполнения данной лабораторной работы было изучено представление и обработка графических данных, был выявлен лучший оптимизатор для построения модели искусственной нейронной сети, распознающей рукописные цифры, была построена и протестирована на пользовательских изображениях модель.