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. 8383 _____ Колмыков В.Д.

Преподаватель _____

Санкт-Петербург

2021

Цель работы.

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

Ход работы.

1. Поиск архитектуры с точностью не менее 95%.

Был реализован код из методических указаний:

```
model = Sequential()
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, batch_size=128, verbose=False)
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('test_acc:', test_acc)
```

Точность сети с данной архитектурой равна 97.7 %.

2. Исследование влияния оптимизаторов.

Изначально программа написана с использованием оптимизатора Adam и шагом сходимости 0.001. Параметр был изменен на 0.1:

```
adam = tf.keras.optimizers.Adam(learning_rate=0.1)
model.compile(optimizer=adam, loss='categorical_crossentropy', metrics=['accuracy'])
```

С данным параметром точность работы упала до 88.7%.

Уменьшение параметра до 0.0001 также не дало увеличения точности - она составила 96.8%.

Для исследования был взят другой оптимизатор SGD (стохастический оптимизатор градиентного спуска):

```
opt = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.0, nesterov=True)
```

При шаге сходимости 0.01, ускорении 0, и использовании моментов Нестерова точность составляет 91%.

При использовании обычных моментов точность составляет 90%.

Увеличение шага сходимости до 0.1 и использование моментов Нестерова дают точность 96%.

Если установить значение ускорения в 0.9, точность возрастет до 98%.

Далее для исследования был взят оптимизатор RMSProp:

```
opt = tf.keras.optimizers.RMSprop(learning_rate=0.001, rho=0.9)
```

При шаге сходиости 0.001 втором параметре 0.9 точность обучения составляет 97%.

Увеличение шага до 0.1 ведет к уменьшению точности до 88%.

Уменьшение шага до 0.0001 дает точность 94%.

При шаге сходимости 0.001 изменение второго параметра не дает значительной разницы в точности – всегда 97% (в документации рекомендуется менять только шаг).

3. Добавление пользовательского ввода изображения.

Для пользовательского ввода изображений была написана следующая функция:

```
def interface():
    while(True):
        print('Input path to the file (or "stop"):')
        path = input()
        if path == 'stop':
            break
        try:
            img = get_image(path)
            predict(img)
        except PermissionError:
            print("Error: Permission denied")
            continue
```

```
            except FileNotFoundError:
                print("Error: File doesn't exists")
                continue
            except SizeException:
                print("Error: Size is not 28x28")
                continue
```

Пользователь вводит путь до файла пока не введет “stop”. Отлавливаются следующие случаи:

- Файла не существует или доступ запрещен
- Размер изображение не 28x28.

Функция получения изображения в нужном виде (в виде тензора):

```
def get_image(path):
    im = Image.open(path)
    arr = np.asarray(im)

    if len(arr) != 28 | len(arr[0]) != 28:
        print(arr)
        raise SizeException

    if arr[0][0].ndim != 0:
         arr = fix_arr(arr)

    arr = arr / 255.0
    t = np.array([arr])
    return t
```

Если на вход подана цветное изображение происходит его преобразование в черно-белое:

```
def fix_arr(arr):
    res = np.zeros((len(arr), len(arr[0])), dtype=int)
    for i in range(len(arr)):
        for j in range(len(arr[0])):
            res[i][j]    =    (arr[i][j][0]    +    arr[i][j][1]    +
arr[i][j][2])//3
    return res
```

Таким образом, на вход можно подать даже цветное изображение. Например, следующие изображения обработаются и даже корректно:

Функция получения результата на экране:

```
def predict(img):
```

```
pred = model.predict(img)
print(np.argmax(pred))
```

Выводы.

В ходе выполнения работы была реализована классификация черно-белых изображений рукописных цифр. Было исследовано влияние оптимизаторов с разными параметрами на точность работы нейросети, а также был реализован интерфейс ввода пользовательского изображения.