**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. 7382 |  | Головина Е.С. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2020

**Цель работы.**

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

**Задачи.**

* Ознакомиться с представлением графических данных
* Ознакомиться с простейшим способом передачи графических данных нейронной сети
* Создать модель
* Настроить параметры обучения
* Написать функцию, позволяющую загружать изображение пользователя и классифицировать его

**Требования.**

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

**Ход работы.**

1. **Ознакомиться с представлением графических данных**

Для операций с графическими данными (изображениями) в python существует модуль Pillow. Загрузка изображения происходит следующим образом.

img = Image.open(filename)

Если необходимо работать напрямую с двумерным массивом, каждый элемент которого кодирует цвет соответствующего пиксела, тогда можно вытащить этот массив. Например, следующим образом:

width, height = img.size

array\_img = np.array(img.getdata(), dtype='uint8')

if array\_img.ndim > 1:

array\_img = array\_img[:,0]

array\_img = np.reshape(array\_img,(width,height))

В данном примере происходит считывание размеров изображения, затем создается массив numpy, из которого выделяются только необходимые данные и массив преобразовывается в двумерный.

1. **Ознакомиться с простейшим способом передачи графических данных нейронной сети**

Изображения представлены в виде массивов чисел в интервале [0, 255], поэтому перед обучением их необходимо преобразовать так, чтобы все значения оказались в интервале [0, 1].

train\_images = train\_images / 255.0

test\_images = test\_images / 255.0

Также при создании модели ИНС при использовании изображений в качестве данных – первым слоем необходимо поставить слой Flatten(), который преобразует изображения из двумерного массива в одномерный.

1. **Создание модели и настройка параметров обучения**

Создание модели происходит следующим образом:

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'])

Таким образом, построена нейронная сеть с тремя слоями. Сеть начинается со слоя Flatten() – т.к. в качестве данных для обучения используем изображения.

Заканчивается слоем с активационной функцией softmax, т.к. в данном случае решаем задачу многоклассовой классификации. По это же причине сеть компилируется с функцией потерь categorical\_crossentropy.

1. **Написать функцию, позволяющую загружать изображение пользователя и классифицировать его**

Для загрузки изображения используем модуль Pillow. Функция выглядит следующим образом:

def getImage(filename):

img = Image.open(filename)

img = img.resize((28,28))

width, height = img.size

array\_img = np.array(img.getdata(), dtype='uint8')

img.close()

if array\_img.ndim > 1:

array\_img = array\_img[:,0]

array\_img = np.reshape(array\_img,(width,height))

return array\_img

С помощью программы Paint было создано изображение, с рукописной цифрой 4: E:\Coding\INS\lab4\ex4.png. Проверим работу полученной нейронной сети.

image = np.array([getImage('ex4.png')])

print(model.predict(image))

Результат: [[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]]

Проверим работу ИНС на другом изображении с цифрой 2: E:\Coding\INS\lab4\ex2.png.

Результат: [[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]]

Как видно – результаты проверок оказались правильными.

1. **Эксперименты**

Исходная нейронная сеть дает точность 97.5%. Значит она соответствует требованию быть больше 95%.

Можно попробовать еще увеличить точность сделав количество выходов 2го слоя равным количеству пикселов (784).

В этом случае получили точность 98.1%.

**Оптимизаторы.**

От выбора алгоритма оптимизации зависит то, насколько быстро сеть обучится. Если сеть будет обучаться слишком быстро, то может не сойтись к глобальному минимуму, каждый раз пролетая через него, а если слишком медленно, то может застрять в локальном минимуме, либо сходиться очень долго. Соответственно важно правильно выбрать алгоритм и его шаг, а также другие сопутствующие параметры, свои для каждого алгоритма.

В начальной модели использовался алгоритм adam – это достаточно распространенный и эффективный алгоритм. Его параметры по умолчанию такие: learning\_rate=0.001, beta\_1=0.9, beta\_2=0.999, amsgrad=False.

Можно попробовать увеличить шаг до 0.002.

Точность уменьшилась до 97.6%. Значит нет смысла в увеличении шага.

Попробуем каждый оптимизатор:

sgd 91.5

rmsprop 97.8

adagrad 97.8

adadelta 97.9

adam 98.0

adamax 97.8

nadam 97.9

Видно, что ‘adam’ дает лучший результат с параметрами по умолчанию.

**Вывод.**

В ходе выполнения работы была реализована классификация черно-белых изображений рукописных цифр (28x28) по 10 категориям (от 0 до 9). Для этого создали искусственную нейронную сеть с помощью библиотеки keras в python, протестировали получившуюся модель, затем, проведя ряд экспериментов, выбрали наилучшую конфигурацию модели.