МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЁТ

по лабораторной работе №4

по дисциплине «Искусственные нейронные сети»

Тема: Распознавание рукописных символов

Студент гр.8382 _____ Фильцин И.В.

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

Ход работы

Рассмотрим следующую архитектуру сети. В модели 3 слоя: 1ый слой для преобразования матрицы в вектор, второй слой состоит из 256 нейронов с функцией активации relu, выходной слой состоит из 10 нейронов с функцией активации softmax. Проведем обучение сети с оптимизатором Adam со всеми параметрами по умолчанию. Точность на тестовых данных составляет 0.97, на тренировочных 0.97.

Проверим влияние параметров оптимизатора на результаты обучения. Алгоритм оптимизатора заключается в том, что он вычисляет экспоненциальное скользящее среднее градиента и квадратичных градиент, а параметры β_1, β_2 управляют скоростью затухания этих скользящих средних.

Попробуем изменить параметр β_1, уменьшив его с 0.9 0.6. Получаем точность на тестовых данных составляет 0.97, на тренировочных 0.97. Таким образом, точность модели осталась на прежнем уровне.

Попробуем изменить параметр β_2, уменьшив его с 0.999 до 0.6 Получаем

точность на тестовых данных составляет 0.97, на тренировочных 0.97. Таким образом, точность модели также осталась на прежнем уровне.

Попробуем изменить скорость обучения, увеличив ее с 0.001 до 0.01. Получаем точность на тестовых данных составляет 0.96, на тренировочных 0.96. Таким образом, видно, что точность модели стала ниже.

Рассмотрим точность модели с использованием оптимизатора Adagrad. Точность модели упала до 0.87 на тестовых и тренировочных данных.

Рассмотрим точность модели с использованием оптимизатора RmsProp. Точность на тестовых данных составляет 0.97, на тренировочных 0.97. Таким образом результат обучения практически ничем не отличается от обучения с использованием Adam.

Вывод

В ходе лабораторной работы была реализована сеть, осуществляющая классификацию черно-белых изображений рукописных цифр. Были исследование различные оптимизатор и их вклад в процесс обучения модели. Лучшие результат показал алгоритм Adam и RmsProp.

Приложение А.

Исходный код

```
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Sequential, load_model
import tensorflow.keras.datasets.mnist as mnist
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import Adam, Adagrad, RMSprop
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np


def build_model():
    model = Sequential()

    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    optimizer = Adam()
    model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

    return model


def train_model():
    (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 = build_model()

    model.fit(train_images, train_labels, epochs=5, batch_size=128)
    test_loss, test_acc = model.evaluate(test_images, test_labels)

    print(test_acc)

    model.save("model")


def load_image(path):
```

```
    img = load_img(path,  color_mode="grayscale",  target_size=(28,    28))

    vec = img_to_array(img)
    vec —= 255
    vec = vec /  —255.0

    return  vec


def run_model():
    model = load_model("model")
    img = load_image("./3.png")

    r = model.predict(np.asarray([img]))
    print(r)
    print(np.argmax(r,    1)[0])


train_model()
#run_model()
```