МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Искусственные нейронные сети»

Тема: Прогноз успеха фильма по обзорам

Студент гр. 7383 _____ Медведев И.С.

Преподаватель _____ Жукова Н. А.

Санкт-Петербург

2020

Цель работы.

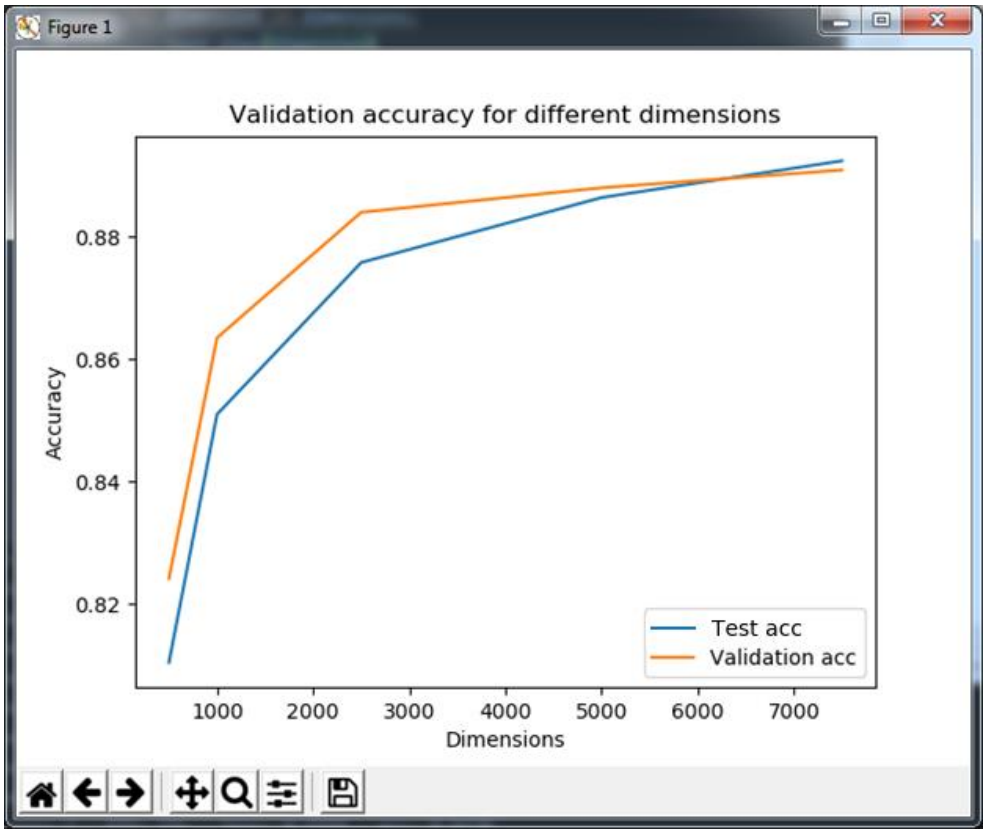
Реализовать предсказание успеха обзора фильма по обзорам.

Выполнение работы.

В ходе выполнения лабораторной работы была написана программа на языке Python, код программы представлен в приложении А.

Было проведено исследование зависимости результата от длины вектора представления текста. Для исследования были взяты длины: 500, 100, 2500, 5000, 7500 (не была взята большая длина из-за того что мой персональный компьютер отказывается выделять память для больших значений и выдает ошибку `MemoryError: Unable to allocate 3.73 GiB`).

Модель сети представлена в приложении А. Результаты тестирования представлены на рис. 1



По рис. 1 можно сделать вывод о том, что наибольшая точность достигается при длине вектора 7500 и равна ~0.89, однако начиная с длины вектора 3000 точность менялась незначительно.

Также была написана функция, которая загружала пользовательский текст из файла и позволяла оценить отзыв. Данная программа была протестирована на двух отзывах:

1) `the film is dump! Bad scenario`

2) `Amazing film, actors are very good, fantastic soundtrack`

Для первого отзыва программа выдала результат 0.2714578, что означает, что она отнесла его больше к отрицательному отзыву, чем к положительному. На второй отзыв программа выдала результат 0.6398537. Это означает что программа оценила данный отзыв как положительный, нежели отрицательный.

Выводы.

В ходе выполнения данной работы была создана сеть для прогноза успеха фильмов по отзывам. Было исследовано влияние размера вектора представления текста на результат. Было выявлено, что чем больше размер, тем больше точность выдает сеть, хотя при длине вектора от 3000 и более точность меняется незначительно. Также была написана функция оценивающая пользовательский отзыв. Результаты тестирования показали, что данная функция работает корректно.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А: КОД ПРОГРАММЫ

```
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Convolution2D, MaxPooling2D,
Dense, Dropout, Flatten
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
import numpy as np

batch_size = 256
num_epochs = 20
kernel_size =7
pool_size = 2
conv_depth_1 = 32
conv_depth_2 = 64
drop_prob_1 = 0.25
drop_prob_2 = 0.5
hidden_size = 512

(X_train, y_train), (X_test, y_test) = cifar10.load_data()
num_train, depth, height, width = X_train.shape
num_test = X_test.shape[0]
num_classes = np.unique(y_train).shape[0]
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= np.max(X_train)
X_test /= np.max(X_train)


Y_train = to_categorical(y_train, num_classes)
Y_test = to_categorical(y_test, num_classes)

def build_model(dropout = True):
    inp = Input(shape=(depth, height, width))
    conv_1  =  Convolution2D(conv_depth_1,  (kernel_size,  kernel_size),
padding='same', activation='relu')(inp)
    conv_2  =  Convolution2D(conv_depth_1,  (kernel_size,  kernel_size),
padding='same', activation='relu')(conv_1)
    pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
    if dropout:
        drop_1 = Dropout(drop_prob_1)(pool_1)
    else:
        drop_1 = pool_1
    conv_3  =  Convolution2D(conv_depth_2,  (kernel_size,  kernel_size),
padding='same', activation='relu')(drop_1)
    conv_4   =   Convolution2D(conv_depth_2,   kernel_size,   kernel_size,
padding='same', activation='relu')(conv_3)
    pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_4)
```

```
    if dropout:
        drop_2 = Dropout(drop_prob_1)(pool_2)
    else:
        drop_2 = pool_2
    flat = Flatten()(drop_2)
    hidden = Dense(hidden_size, activation='relu')(flat)
    drop_3 = Dropout(drop_prob_2)(hidden)
    out = Dense(num_classes, activation='softmax')(drop_3)
    model = Model(inp, out)
    model.compile(loss='categorical_crossentropy',      optimizer='adam',
metrics=['accuracy'])
    history   =   model.fit(X_train,   Y_train,     batch_size=batch_size,
epochs=num_epochs, verbose=1, validation_split=0.1)
    print(model.evaluate(X_test, Y_test, verbose=1))

    x = range(1, num_epochs + 1)
    plt.plot(x, history.history['loss'], label='Training loss')
    plt.plot(x, history.history['val_loss'], label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
    plt.clf()

    plt.plot(x, history.history['acc'], label='Training acc')
    plt.plot(x, history.history['val_acc'], label='Validation acc')
    plt.title('Training and validation accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend()
    plt.show()


build_model()
```