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

ОТЧЕТ

по лабораторной работе №7
"Классификация обзоров фильмов"
по дисциплине «Искусственные нейронные сети»

Студентка гр. 8382 _____ Ивлева О.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель.

Классификация последовательностей — это проблема прогнозирующего моделирования, когда у вас есть некоторая последовательность входных данных в пространстве или времени, и задача состоит в том, чтобы предсказать категорию для последовательности.

Проблема усложняется тем, что последовательности могут различаться по длине, состоять из очень большого словарного запаса входных символов и могут потребовать от модели изучения долгосрочного контекста или зависимостей между символами во входной последовательности.

В данной лабораторной работе также будет использоваться датасет IMDb, однако обучение будет проводиться с помощью рекуррентной нейронной сети

Задание.

- Ознакомиться с рекуррентными нейронными сетями
- Изучить способы классификации текста
- Ознакомиться с ансамблированием сетей
- Построить ансамбль сетей, который позволит получать точность не менее 97%
- Найти набор оптимальных ИНС для классификации текста
- Провести ансамблирование моделей
- Написать функцию/функции, которые позволят загружать текст и получать результат ансамбля сетей
- Провести тестирование сетей на своих текстах (привести в отчете)

Ход работы.

1. Были созданы следующие модели нейронных сетей: рекуррентная нейронная сеть (RNN), сверточная нейронная сеть (CNN) и сверточная рекуррентная нейронная сеть (CRNN). Код программы представлен в приложении А.

2. Обучим три модели при одинаковых параметрах обучения и проведем их ансамблирование. Оценим точность нейронных сетей и их ансамблей. Графики точностей для моделей представлены на рис. 1.

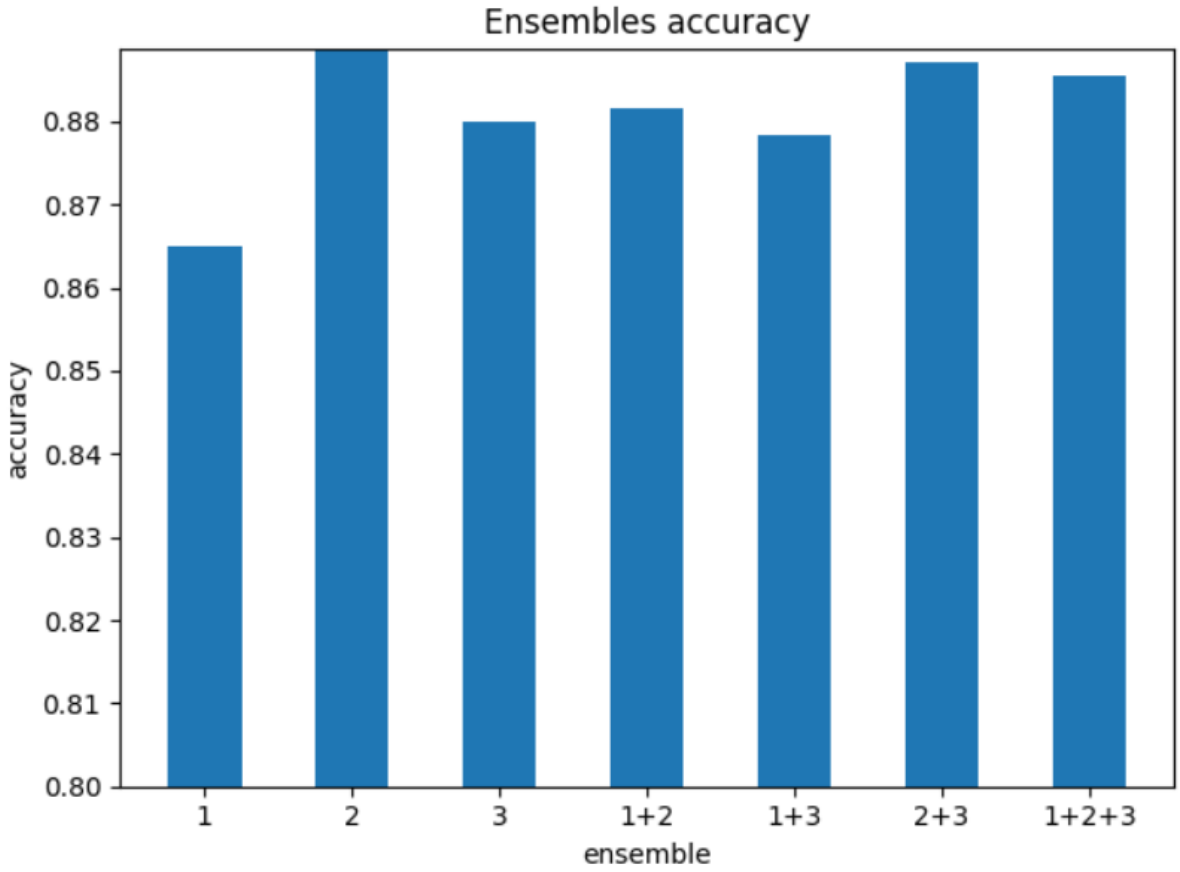

Рисунок 1 – График точностей моделей и их ансамблей

Значения точностей из графика: 86.5% RNN, 88.9% CNN, 88.0% CRNN, 88.2% RNN + CNN, 87.8% RNN + CRNN, 88.7% CNN + CRNN,88.6% RNN + CNN + CRNN. Все модели и их ансамбли дают примерно одинаковую точность, однако, наибольшую точность более 0.88 дает CNN модель и ансамбль CNN и CRNN моделей.

3. Была написана функция, которая позволяет загружать пользовательский текст из файла. Получим результат ансамбля сетей, загрузив следующий отзыв: The acting was amazing and the film was good overall but I think 'masterpiece' and 'film of the year' are a bit overused throughout the reviews. In no way did I dislike

this film, I thought it was really good, just overrated. Ответ ансамбля сетей: 0.7085517, что говорит о том, что данный отзыв относится к положительному.

Выводы.

В результате выполнения данной работы были созданы несколько моделей нейронных сетей и проведено их ансамблирование. Также были оценены точности каждой из моделей и ансамблей и получен результат оценки пользовательского отзыва ансамблем.

ПРИЛОЖЕНИЕ А

Исходный код программы. Файл lr4.py

```
import numpy as np
import matplotlib.pyplot as plt

from keras.datasets import cifar10
from keras.layers import Input, Convolution2D, MaxPooling2D, Dense, Dropout,
Flatten
from keras.models import Model
from keras.utils import np_utils

# Setting hyper-parameters:
batch_size = 150
num_epochs = 10
kernel_size = 4
pool_size = 2
conv_depth_1 = 32
conv_depth_2 = 64
hidden_size = 512
with_dropout = True
if with_dropout:
    drop_prob_1 = 0.25
    drop_prob_2 = 0.5

# Loading data:
(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]

# Normalizing data:
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255.0
X_test /= 255.0

# Converting labels:
Y_train = np_utils.to_categorical(y_train, num_classes)
Y_test = np_utils.to_categorical(y_test, num_classes)

# Building model:
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 with_dropout:
    drop_1 = Dropout(drop_prob_1)(pool_1)
conv_3 = Convolution2D(conv_depth_2, (kernel_size, kernel_size), padding='same',
activation='relu')(drop_1 if with_dropout else pool_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 with_dropout:
    drop_2 = Dropout(drop_prob_1)(pool_2)
flat = Flatten()(drop_2 if with_dropout else pool_2)
hidden = Dense(hidden_size, activation='relu')(flat)
if with_dropout:
    drop_3 = Dropout(drop_prob_2)(hidden)
out = Dense(num_classes, activation='softmax')(drop_3 if with_dropout else
hidden)

model = Model(inp, out)
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

# Fitting model:
h = model.fit(X_train, Y_train, batch_size=batch_size, epochs=num_epochs,
verbose=1, validation_data=(X_test, Y_test))

print(model.evaluate(X_test, Y_test))

# Plotting:
loss = h.history['loss']
val_loss = h.history['val_loss']
acc = h.history['accuracy']
val_acc = h.history['val_accuracy']
epochs = range(1, len(loss) + 1)

plt.plot(range(1, num_epochs + 1), loss, 'b', label='Training loss')
plt.plot(range(1, num_epochs + 1), val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
plt.clf()

plt.plot(range(1, num_epochs + 1), acc, 'b', label='Training acc')
plt.plot(range(1, num_epochs + 1), val_acc, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
```

```
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```