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

ОТЧЕТ

по лабораторной работе №8

по дисциплине «Искусственные нейронные сети»

Тема: «Генерация текста на основе «Алисы в стране чудес»»

Студент гр. 8383 _____ Муковский Д.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы

Рекуррентные нейронные сети также могут быть использованы в качестве генеративных моделей. Это означает, что в дополнение к тому, что они используются для прогнозных моделей (создания прогнозов), они могут изучать последовательности проблемы, а затем генерировать совершенно новые вероятные последовательности для проблемной области. Подобные генеративные модели полезны не только для изучения того, насколько хорошо модель выявила проблему, но и для того, чтобы узнать больше о самой проблемной области.

Задачи

1. Ознакомиться с генерацией текста
2. Ознакомиться с системой Callback в Keras

Требования

1. Реализовать модель ИНС, которая будет генерировать текст
2. Написать собственный CallBack, который будет показывать то, как генерируется текст во время обучения (то есть раз в какое-то количество эпох генерировать и выводить текст у необученной модели)
3. Отследить процесс обучения при помощи TensorFlowCallBack (TensorBoard), в отчете привести результаты и их анализ.

Ход работы

Была реализована модель ИНС для генерации текста, которая представлена ниже.

```
model = Sequential()

model.add(LSTM(256, input_shape=(X.shape[1], X.shape[2])))

model.add(Dropout(0.2))

model.add(Dense(y.shape[1], activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam')
```

Пример генерации текста:

```
Seed:" ' said alice, who always took a great interest in
questions of eating and drinking.
'they lived on treacle,' said the dormouse, after thinking a minute or
two.
'they couldn't have done that, you kno "

Text:
said the monk, and see boo of thes sae io the court, and the woide sas soin
it soeer to tee that she had hoen the harter woted the had hoen the whide
and the tored oo the horse so the horsh oo th
```

Далее был описан собственный CallBack для генерации текста необученной модели каждые 4 эпохи. Callback представлен ниже:

```
class CustomCallback(Callback):
    def generate_text(self, size=200):
        start = np.random.randint(0, n_patterns-1)
        pattern = dataX[start]
        text = []
        for i in range(size):
            x = np.reshape(pattern, (1, len(pattern), 1))
            x = x / float(n_vocab)
            prediction = model.predict(x, verbose=0)
            index = np.argmax(prediction)
            result = int_to_char[index]
            text.append(result)
            pattern.append(index)
            pattern = pattern[1:len(pattern)]
        print("".join(text))

    def on_epoch_end(self, epoch, logs=None):
        if epoch % 4 == 0:
            print(f'Epoch № {epoch+1}')
            self.generate_text()
```

Пример генерации текста во время обучения:

```
Epoch № 5
```

```
tas ho the toete the tas ho the toete the tas ho the toete the tas ho the
toete the tas ho the toete the tas ho the toete the tas ho the toete the
tas ho the toete the tas ho the toete the tas ho the
```

```
Epoch № 9
nd the wan iort the white was ao anl oo the thne  she was aol the was oo
the thne  bnd the was oo the whnt oas here the was oo the tas of the caree
an the was oo the haree hn the was oo the tas of the
```

```
Epoch № 13
ol a lote oatee hatter of the haad,
'that soe tait to toar a setel tou dave to tei thuh the dense   she hatter
waid  tothing to the more of the sabbit sesee thet she was oot io the tast
of the hadt,
```

```
Epoch № 17
on, and the tooed seter the whst whrh the horsh to teet the was oo the
woode  'att it was the mabt th the tai oo the toide 'she fare te thene
thet sould th the toede of the coort, and the white was a
```

Как не трудно заметить: чем более обучена нейронная сеть, тем лучше текс она генерирует.

Прогресс обучения с помощью TensorBoard

Для наблюдения за процессом обучения модели был использован TensorBoard. График потерь и гистограмма распределения весов представлен на рис. 1 и 2 соответственно.

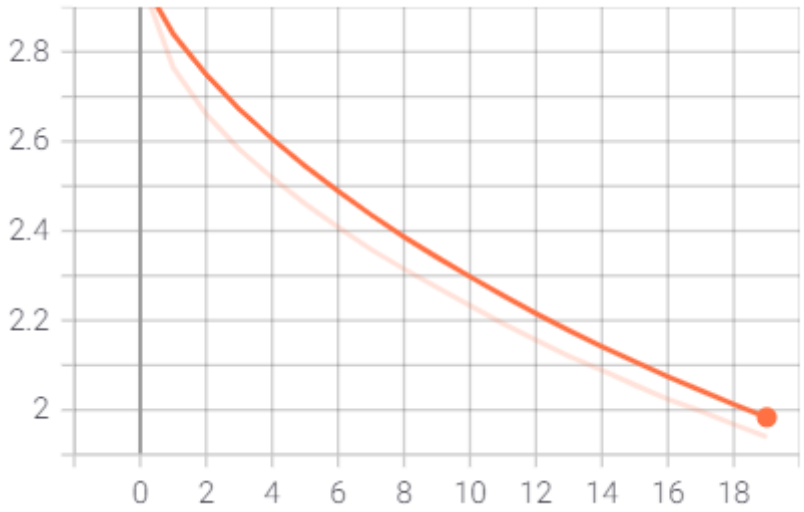

Рисунок 1 – График потерь

Как видим сеть не переобучалась, а с каждой эпохой уменьшала значение ошибки.

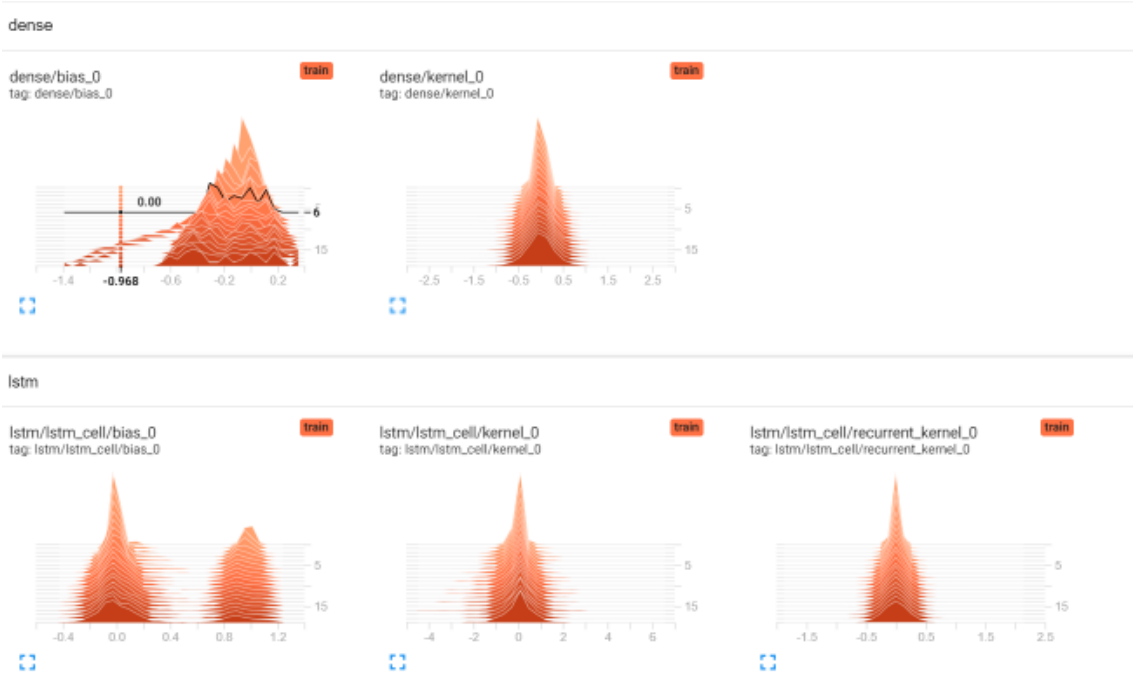

Рисунок 1 – Гистограмма распределения весов

Вывод

В ходе выполнения лабораторной работы была реализована ИНС, которая генерирует текст на основе книги «Алиса в стране чудес». Также был написан собственная колбэк-функция для мониторинга прогресса обучения сети.

ПРИЛОЖЕНИЕ А. Исходный код программы

```
import numpy as np
from keras.layers import Dense, Dropout, LSTM
from keras.models import Sequential
from keras.callbacks import ModelCheckpoint, Callback, TensorBoard
from keras.utils import np_utils

filename = "alice.txt"
raw_text = open(filename).read()
raw_text = raw_text.lower()

chars = sorted(list(set(raw_text)))
print(set(raw_text))
print(chars)
char_to_int = dict((c, i) for i, c in enumerate(chars))
int_to_char = dict((i, c) for i, c in enumerate(chars))

n_chars = len(raw_text)
n_vocab = len(chars)
print("Total Characters: ", n_chars)
print("Total Vocab: ", n_vocab)

seq_length = 200
dataX = []
dataY = []
for i in range(0, n_chars - seq_length, 1):
    seq_in = raw_text[i:i + seq_length]
    seq_out = raw_text[i + seq_length]
    dataX.append([char_to_int[char] for char in seq_in])
    dataY.append(char_to_int[seq_out])
n_patterns = len(dataX)
print("Total Patterns: ", n_patterns)

X = np.reshape(dataX, (n_patterns, seq_length, 1))
X = X / float(n_vocab)
y = np_utils.to_categorical(dataY)

model = Sequential()
model.add(LSTM(256, input_shape=(X.shape[1], X.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')

filepath = "weights-improvement-{epoch:02d}.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1,
save_best_only=True, mode='min')

class CustomCallback(Callback):
    def generate_text(self, size=200):
        start = np.random.randint(0, n_patterns - 1)
        pattern = dataX[start]
        text = []
        for i in range(size):
            x = np.reshape(pattern, (1, len(pattern), 1))
            x = x / float(n_vocab)
```

```
            prediction = model.predict(x, verbose=0)
            index = np.argmax(prediction)
            result = int_to_char[index]
            text.append(result)
            pattern.append(index)
            pattern = pattern[1:len(pattern)]
        print("".join(text))

    def on_epoch_end(self, epoch, logs=None):
        if epoch % 4 == 0:
            print(f'{epoch + 1} ep:')
            self.generate_text()

callbacks_list = [checkpoint, CustomCallback(), TensorBoard(log_dir='logs',
histogram_freq=1, embeddings_freq=1)]

model.fit(X, y, epochs=20, batch_size=128, callbacks=callbacks_list,
verbose=2)

filename = "weights-improvement-20.hdf5"
model.load_weights(filename)

start = np.random.randint(0, n_patterns - 1)
pattern = dataX[start]
print("Seed:")
print("\"", ''.join([int_to_char[value] for value in pattern]), "\"")
text = []

for i in range(200):
    x = np.reshape(pattern, (1, len(pattern), 1))
    x = x / float(n_vocab)
    prediction = model.predict(x, verbose=0)
    index = np.argmax(prediction)
    result = int_to_char[index]
    text.append(result)
    pattern.append(index)
    pattern = pattern[1:len(pattern)]
print("".join(text))
```