**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 |  | Киреев К.А. |
| 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**

* Ознакомиться с генерацией текста
* Ознакомиться с системой Callback в Keras

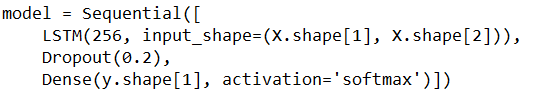
**Требования**

* Реализовать модель ИНС, которая будет генерировать текст
* 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)
* Отследить процесс обучения при помощи TensorFlowCallBack (TensorBoard), в отчете привести результаты и их анализ

**Ход работы**

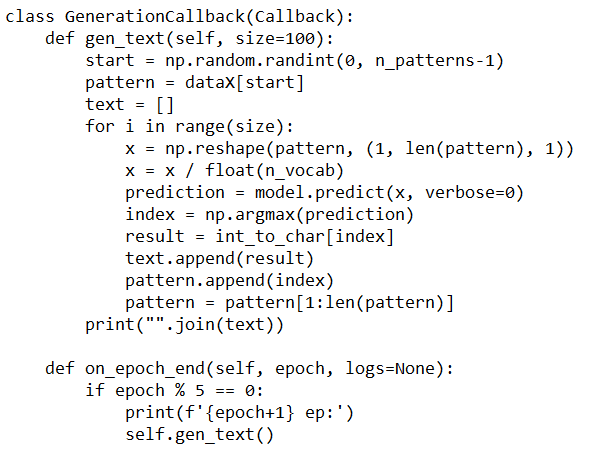
***Модель ИНС для генерации текста***

Была реализована модель ИНС для генерации текста на основе “Алисы в стране чудес”. Листинг модели представлен ниже.

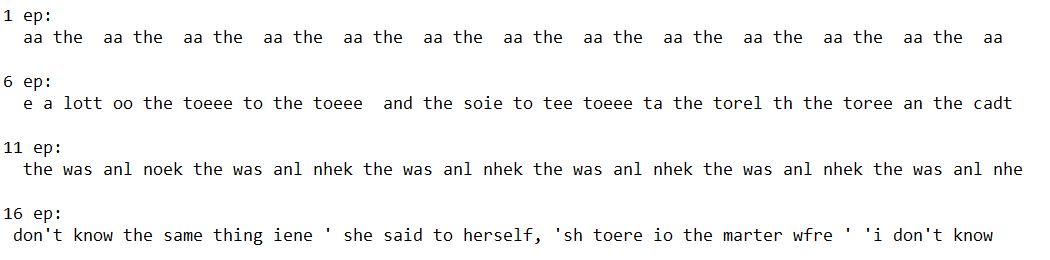


***Собственный CallBack***

Был написан собственный CallBack – GenerationCallback(), который раз в 5 эпох генерирует и выводит текст у необученной модели. Листинг представлен ниже.



Пример генерации текста на каждой пятой эпохе.



Видно, что с увеличением количества эпох, улучшается качество генерации текста.

***Прогресс обучения с помощью TensorBoard***

Для наблюдения за процессом обучения модели был использован TensorBoard.

Графики процесса обучения представлены на рис. 1-3.

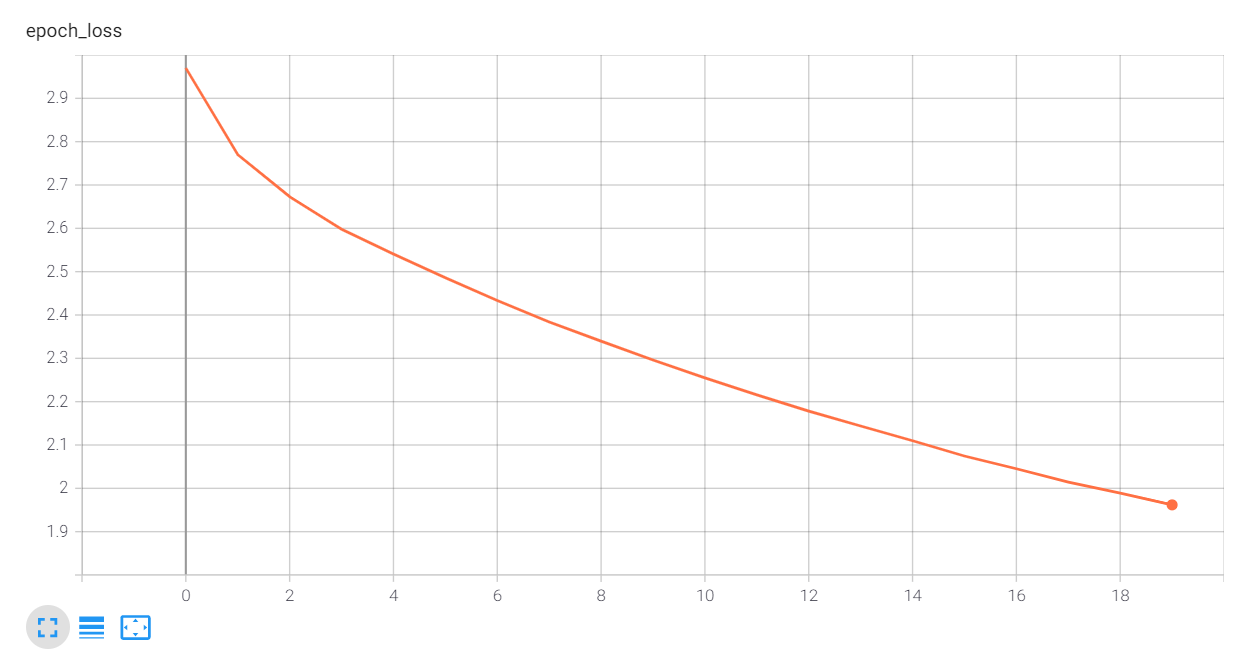


Рисунок 1 – График потерь



Рисунок 2 – Гистограммы активаций

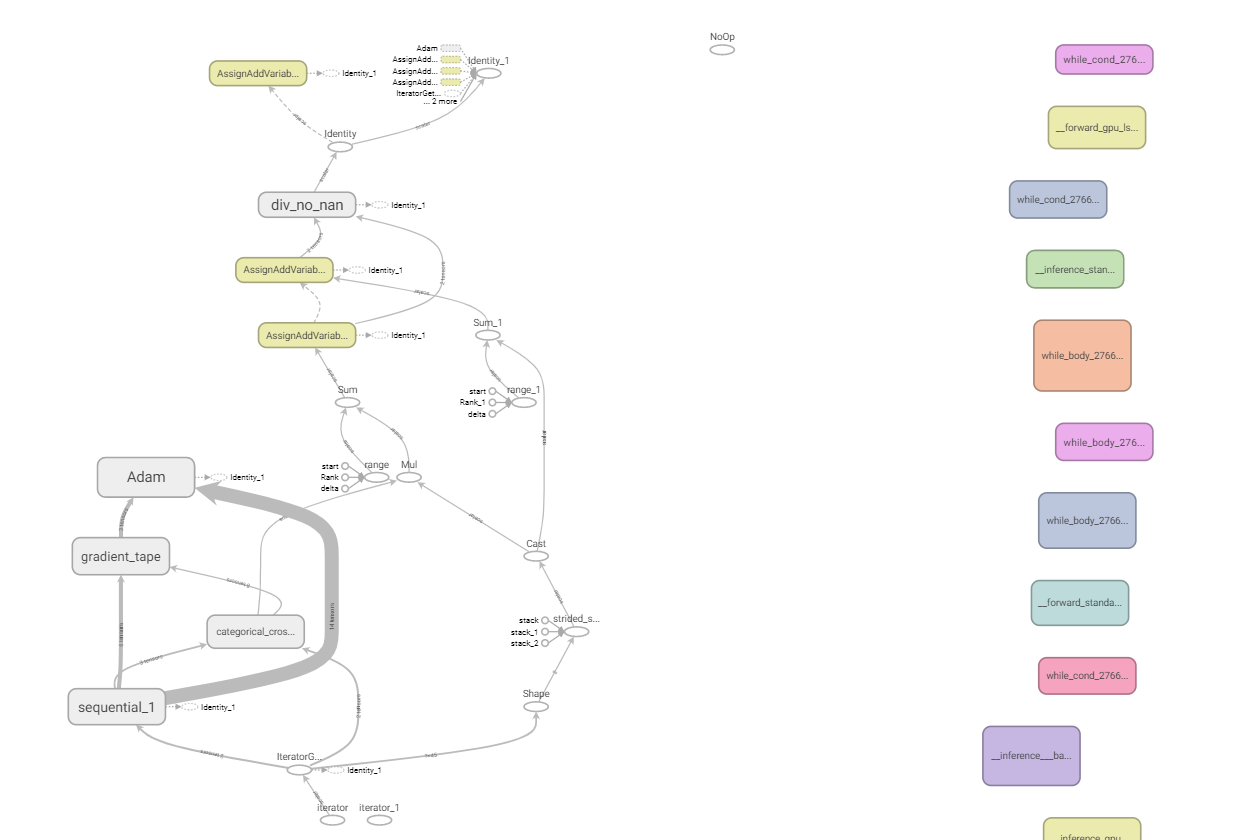
****

Рисунок 3 – Интерактивные диаграммы низкоуровневых операций

**Выводы**

В ходе лабораторной работы была реализована нейросеть для генерации текста на основе “Алисы в стране чудес”. Был также написан собственный Callback, позволяющий отслеживать прогресс обучения сети.

Приложение А. Исходный код программы

import os

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'

os.environ['TF\_FORCE\_GPU\_ALLOW\_GROWTH'] = 'true'

from matplotlib import gridspec

import matplotlib

import matplotlib.pyplot as plt

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

class GenerationCallback(Callback):

def gen\_text(self, size=100):

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 % 5 == 0:

print(f'{epoch+1} ep:')

self.gen\_text()

filename = "wonderland.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))

print(char\_to\_int)

print(int\_to\_char)

n\_chars = len(raw\_text)

n\_vocab = len(chars)

print("Total Characters: ", n\_chars)

print("Total Vocab: ", n\_vocab)

seq\_length = 100

dataX = [] # 144331 x 100

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)

print(dataX[0])

print(dataY[0])

X = np.reshape(dataX, (n\_patterns, seq\_length, 1)) # образцы, временные шаги, признаки

X = X / float(n\_vocab) # нормализация

y = np\_utils.to\_categorical(dataY)

# print(X.shape) # (144331, 100, 1)

# print(y.shape) # (144331, 45)

# print(X[0])

# print(y[0])

model = Sequential([

LSTM(256, input\_shape=(X.shape[1], X.shape[2])),

Dropout(0.2),

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')

callbacks\_list = [checkpoint, GenerationCallback(), 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(100):

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))