## CSEE 5590 0001- Python and Deep Learning

## Fall 2018

## Python Lab Assignment 4

## Submitted On 12/5/2018

## Name: Alexandria Piatt and Alexander Larios

## Class IDs (respectively): 23 15

## TABLE OF CONTENTS

## 1. Author

## 2. Objective

## 3. Features

## 4. Configuration

5. Input/Output Screenshots

6. Implementation & Full Source Code

7. References

**AUTHORS**

This lab report corresponds to the third lab assignment in CS 5590 0001 Python and Deep-Learning. ALEXANDRIA PIATT (ID: 23) and ALEXANDER LARIOS (ID: 15) are both in their last year at UMKC, both are completing a B.S. in C.S. in December 2018. The leader of the course is SARIA GOUDARZVAND.

**OBJECTIVE**

This lab was given to practice creating LSTM, a CNN, and a RNN. The LSTM and the RNN were both applied to the Reuters dataset for text classification and the CNN was used to classify the CIFAR 10 dataset. Previous python libraries, such as Keras, and TensorBoard were used again. The point is to develop the skills of creating Neural Networks and honing our knowledge of the keras library and the use of TensorBoard.

**FEATURES**

Task 1:

***Perform text classification analysis on a data set using RNN***

Use RNN in to perform text classification on the Reuters data set. The data set included around 11,000 news wires from Reuters. The news wires are topic labels and includes 46 of them. A RNN model was implemented and trained to be able to classify each news wire based off of one of those 46 categories. TensorBoard was used to chart and log the accuracy and loss of the runs.

Task 2:

***Perform text classification analysis on a data set using LSTM***

Use LSTM in to perform text classification on the Reuters data set. The data set included around 11,000 news wires from Reuters. The news wires are topic labels and includes 46 of them. A LSTM model was implemented and trained to be able to classify each news wire based off of one of those 46 categories. TensorBoard was used to chart and log the accuracy and loss of runs.

Task 2.b:

Compare the results of the RNN analysis and the LSTM analysis. The results were very similar. Both had very high accuracy rates. The LSTM was approximately 98% accurate and the RNN wa about 97% accurate at categorizing the news wires. This makes sense considering the “buzzwords” that are typically apparent in news articles. It makes classifying them by category relatively easy for a machine learning algorithm.

Task 3:

***Perform image classification analysis on a data set using CNN***

Classify a set of images into their proper categories. The programmer used 3 convolutional layers with activation ‘relu’. There was also 3 Dense layers 2 with ‘relu’ the other with ‘softmax’. The data was a 100-category version of Cifar that was imported from keras. The data was spilt int training and validation sets. After about 60 epochs the model would over train and the accuracy would go down. Over training limited the accuracy of the validation set to about 47% even though training accuracy was over 90%

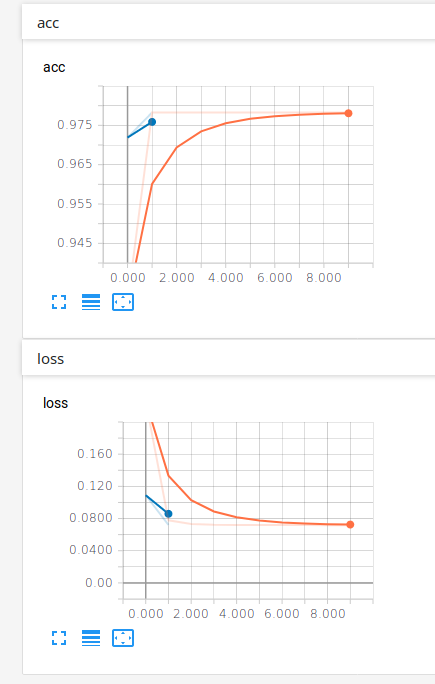
**CONFIGURATION**

All of the code in this lab was written and built using PYCHARM IDE, in an Anaconda3 environment and using Python 3.6.

**INPUT/OUTPUT SCREENSHOTS**

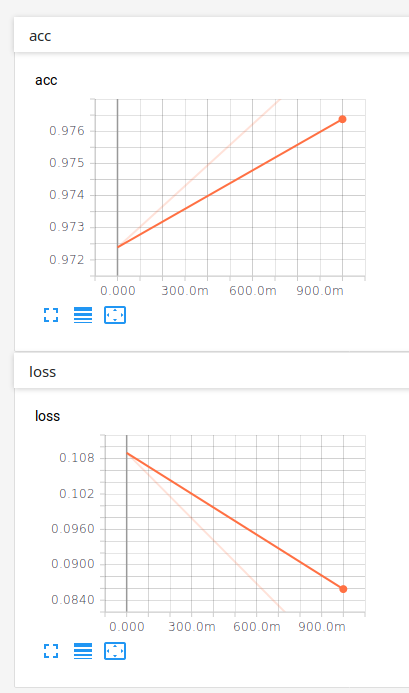
Task 1:

The programmer used the Reuters data set from Keras. After reading in the data from Keras, the programmer cleans up and pads the data. To ensure the inputs were all the same length, the programmer padded any sequences less than 600 and truncated the ones greater than 600 down to 600. The RNN model used ‘sigmoid’ Activation, an ‘adam’ optimizer and the loss metric was ‘binary\_crossentropy’. For the fit function, the programmer used 10 epochs and a batch size of 50. TensorBoard logs were used to chart the loss and accuracy across the runs.



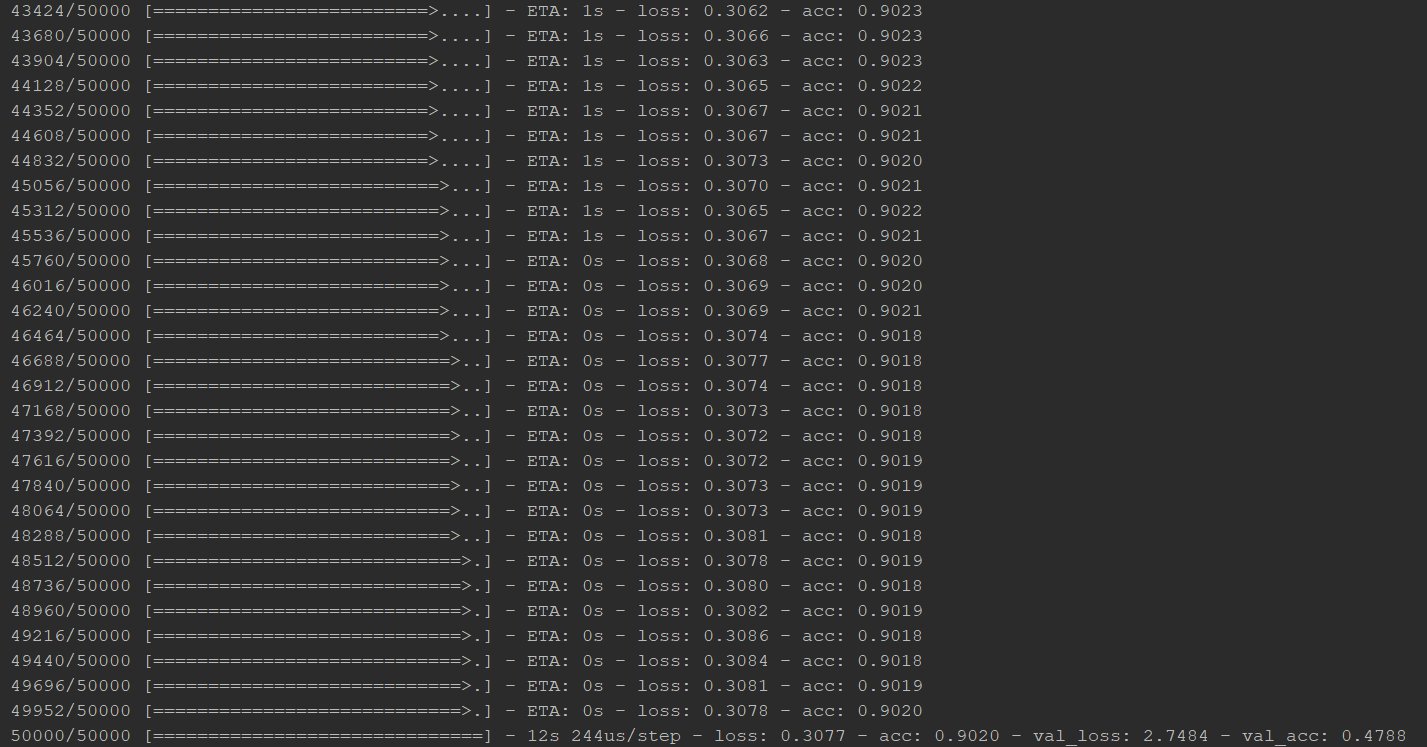
Task 2:

The programmer used the Reuters data set from Keras. After reading in the data from Keras, the programmer cleans up and pads the data. To ensure the inputs were all the same length, the programmer padded any sequences less than 600 and truncated the ones greater than 600 down to 600.For the Conv1D layer the activation is ‘relu’. The LSTM model used ‘sigmoid’ Activation, an ‘adam’ optimizer and the loss metric was ‘binary\_crossentropy’. For the fit function, the programmer used 15 epochs and a batch size of 25. TensorBoard logs were used to chart the loss and accuracy across the runs.



**Task 3:**

The data was a 100-category version of Cifar that was imported from keras. The data was spilt int training and validation sets. After about 60 epochs the model would over train and the accuracy would go down. Over training limited the accuracy of the validation set to about 47% even though training accuracy was over 90%



**IMPLEMENTATION & FULL SOURCE CODE**

Task 1:

**from** keras.datasets **import** reuters

**from** time **import** time

**from** keras.models **import** Sequential

**from** keras.layers **import** Activation, Dense, Dropout, Embedding, LSTM, Conv1D, MaxPooling1D

**from** keras.preprocessing.sequence **import** pad\_sequences

**from** keras.callbacks **import** TensorBoard

**from** keras.utils **import** to\_categorical

(x\_train, y\_train), (x\_test, y\_test) = reuters.load\_data(path=**"reuters.npz"**, num\_words=10000)

x\_train = pad\_sequences(x\_train, 600, padding=**'pre'**, truncating=**'pre'**)

x\_test = pad\_sequences(x\_test, 600, padding=**'pre'**, truncating=**'pre'**)

y\_train = to\_categorical(y\_train)

y\_test = to\_categorical(y\_test)

tb = TensorBoard(log\_dir=**"logs/lstm/{}"**.format(time()))

lstm\_model = Sequential()

lstm\_model.add(Embedding(10000, 8, input\_length=600))

lstm\_model.add(Dropout(.25))

lstm\_model.add(Conv1D(64, 5, padding=**'valid'**, activation=**'relu'**))

lstm\_model.add(MaxPooling1D(pool\_size=4))

lstm\_model.add(LSTM(70))

lstm\_model.add(Dense(46))

lstm\_model.add(Activation(**'sigmoid'**))

lstm\_model.compile(loss=**'binary\_crossentropy'**, optimizer=**'adam'**, metrics=[**'accuracy'**])

lstm\_model.fit(x\_train, y\_train, epochs = 15, batch\_size = 25, callbacks=[tb])

score, accuracy = lstm\_model.evaluate(x\_test, y\_test, verbose = 0)

Task 2:

**from** keras.datasets **import** reuters

**from** time **import** time

**from** keras.models **import** Sequential

**from** keras.layers **import** Activation, Dense, SimpleRNN, Embedding

**from** keras.preprocessing.sequence **import** pad\_sequences

**from** keras.callbacks **import** TensorBoard

**from** keras.utils **import** to\_categorical

(x\_train, y\_train), (x\_test, y\_test) = reuters.load\_data(path=**"reuters.npz"**, num\_words=10000)

x\_train = pad\_sequences(x\_train, 600, padding=**'pre'**, truncating=**'pre'**)

x\_test = pad\_sequences(x\_test, 600, padding=**'pre'**, truncating=**'pre'**)

y\_train = to\_categorical(y\_train)

y\_test = to\_categorical(y\_test)

tb = TensorBoard(log\_dir=**"logs/{}"**.format(time()))

rnn = Sequential()

rnn.add(Embedding(10000, 8, input\_length=600))

rnn.add(SimpleRNN(32))

rnn.add(Dense(46))

rnn.add(Activation(**'sigmoid'**))

rnn.summary()

rnn.compile(optimizer=**'adam'**, loss=**'binary\_crossentropy'**, metrics=[**'accuracy'**])

rnn.fit(x\_train, y\_train, epochs=10, batch\_size=50, callbacks=[tb])

score, accuracy = rnn.evaluate(x\_test, y\_test, verbose=0)

print(**'Test score'**, score)

print(**'Test accuracy'**, accuracy)

Task 3:

import numpy  
from keras.datasets import cifar100  
from keras.models import Sequential  
from keras.layers import Dense  
from keras.layers import Dropout  
from keras.layers import Flatten  
from keras.constraints import maxnorm  
from keras.optimizers import SGD  
from keras.layers.convolutional import Conv2D  
from keras.layers.convolutional import MaxPooling2D  
from keras.utils import np\_utils  
from keras import backend as K  
K.set\_image\_dim\_ordering('th')  
from keras.callbacks import TensorBoard  
from time import time  
  
tensor\_board = TensorBoard(log\_dir="linear\_logs/{}".format(time()))  
# fix random seed for reproducibility  
seed = 7  
numpy.random.seed(seed)  
# load data  
#Data set is different from in class data set as it has 100 catagories and is the only other image data set from keras.  
(X\_train, y\_train), (X\_test, y\_test) = cifar100.load\_data()  
  
# normalize inputs from 0-255 to 0.0-1.0  
X\_train = X\_train.astype('float32')  
X\_test = X\_test.astype('float32')  
X\_train = X\_train / 255.0  
X\_test = X\_test / 255.0  
# one hot encode outputs  
y\_train = np\_utils.to\_categorical(y\_train)  
y\_test = np\_utils.to\_categorical(y\_test)  
num\_classes = y\_test.shape[1]  
# Create the model  
model = Sequential()  
  
model.add(Conv2D(32, (3, 3), input\_shape=(3, 32, 32), padding='same', activation='relu', kernel\_constraint=maxnorm(3)))  
model.add(Dropout(0.2))  
model.add(Conv2D(32, (3, 3), input\_shape=(3, 32, 32), padding='same', activation='relu', kernel\_constraint=maxnorm(3)))  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Conv2D(64, (3, 3), input\_shape=(3, 32, 32), padding='same', activation='relu', kernel\_constraint=maxnorm(3)))  
model.add(Dropout(0.2))  
model.add(Conv2D(64, (3, 3), input\_shape=(3, 32, 32), padding='same', activation='relu', kernel\_constraint=maxnorm(3)))  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Conv2D(128, (3, 3), input\_shape=(3, 32, 32), padding='same', activation='relu', kernel\_constraint=maxnorm(3)))  
model.add(Dropout(0.2))  
model.add(Conv2D(128, (3, 3), input\_shape=(3, 32, 32), padding='same', activation='relu', kernel\_constraint=maxnorm(3)))  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Flatten())  
model.add(Dropout(0.2))  
model.add(Dense(1024, activation='relu', kernel\_constraint=maxnorm(3)))  
model.add(Dropout(0.2))  
model.add(Dense(512, activation='relu', kernel\_constraint=maxnorm(3)))  
model.add(Dropout(0.2))  
model.add(Dense(100, activation='softmax', kernel\_constraint=maxnorm(3)))  
  
  
# Compile model  
epochs = 60  
lrate = 0.01  
decay = lrate/epochs  
sgd = SGD(lr=lrate, momentum=0.9, decay=decay, nesterov=False)  
model.compile(loss='categorical\_crossentropy', optimizer=sgd, metrics=['accuracy'])  
print(model.summary())  
# Fit the model  
model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test), epochs=epochs, batch\_size=32, callbacks=[tensor\_board])  
# Final evaluation of the model  
scores = model.evaluate(X\_test, y\_test, verbose=0)  
prediction = model.predict(X\_test[:4], batch\_size=32, verbose=0, steps=None)  
print(prediction)  
print("Accuracy: %.2f%%" % (scores[1]\*100))  
  
model\_json = model.to\_json()  
with open("model.json", "w") as json\_file:  
 json\_file.write(model\_json)  
model.save\_weights("model.cnnimg")

**REFERENCES**

keras.io