CSEE5590/490: Python and Deep Learning Programming (2018 Fall)

Lab 4

Team ID:- 13

Partner 1: Kamal Tej Veerapaneni Id - 31

Partner 2: Vinay Maturi Id - 17

YouTube Link: https://youtu.be/2Y-uJLyzy-Y

Github Link: https://github.com/Kamaltejveerapaneni/Python-and-Deep-Learning/wiki/Deep-Learning-Lab-2

Introduction:

In this Lab Assignment we have worked on the following tasks.

1. Load a New data set (different from class)
2. Implement the text classification with CNN, RNN/LSTM model
3. Compare the results of CNN and RNN/LSTM models, for the text classification
4. Implement the image classification with CNN model

Objectives:

- Get familiar with Tensor flow Library
- Here we are going to create a model, fit it and then plot training and validation values for loss and accuracy
- We also have to change and compare CNN to RNN and observe how loss and accuracy varies in each case
- Image classification with Cifar-100

Question 1:

1. Implement the text classification with CNN model, with a new dataset which is not used in the class

Workflow:

The result is shown using Tensor board .Here we can see the code along with Loss and Accuracy values

Important to note: Used a different data set known as collections.csv and shown the graph plots

Observations:

Here we make some observations for text classification using CNN and find out Loss and Accuracy values . Since this question focuses on text classification we can see loss and accuracy values.We are going to plot Loss and Accuracy values for the dataset used

We keep the epochs and batch size along with other parameters the same for the first three observations

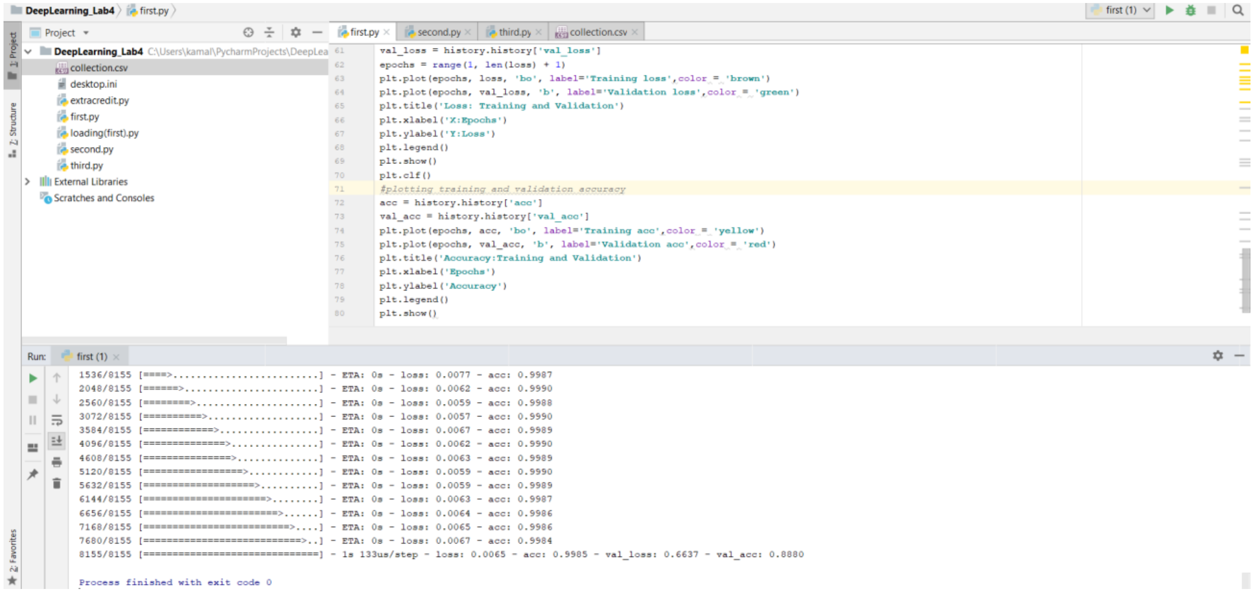
Code Snippet + Loss Values:

Batch Size:512

Epoch:20

Optimizer: Adam

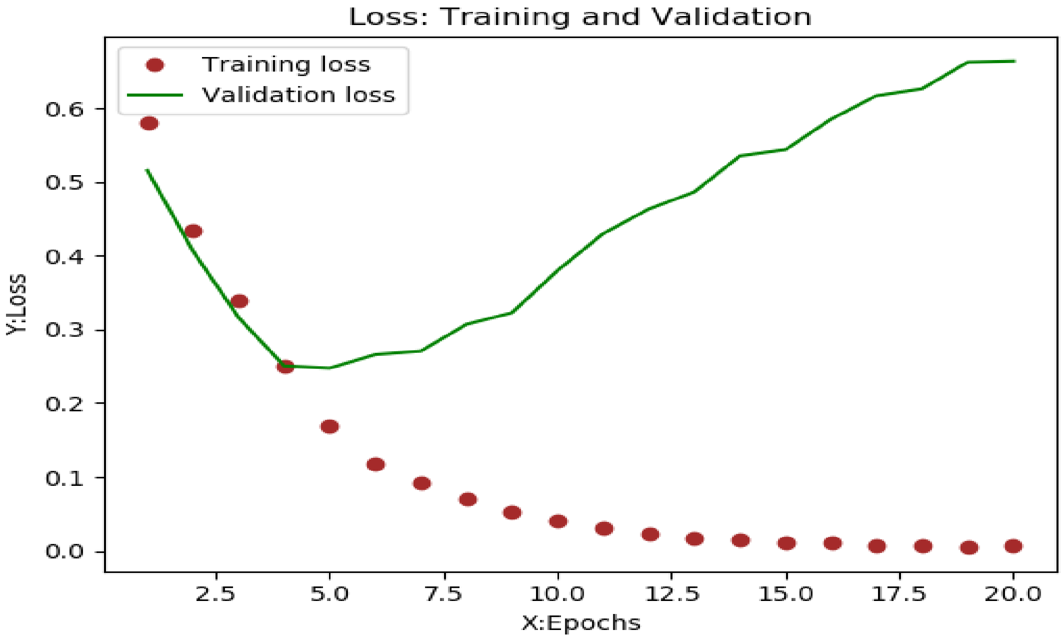
Activation Function: relu Loss:0.0065 ,Acc:0.9985



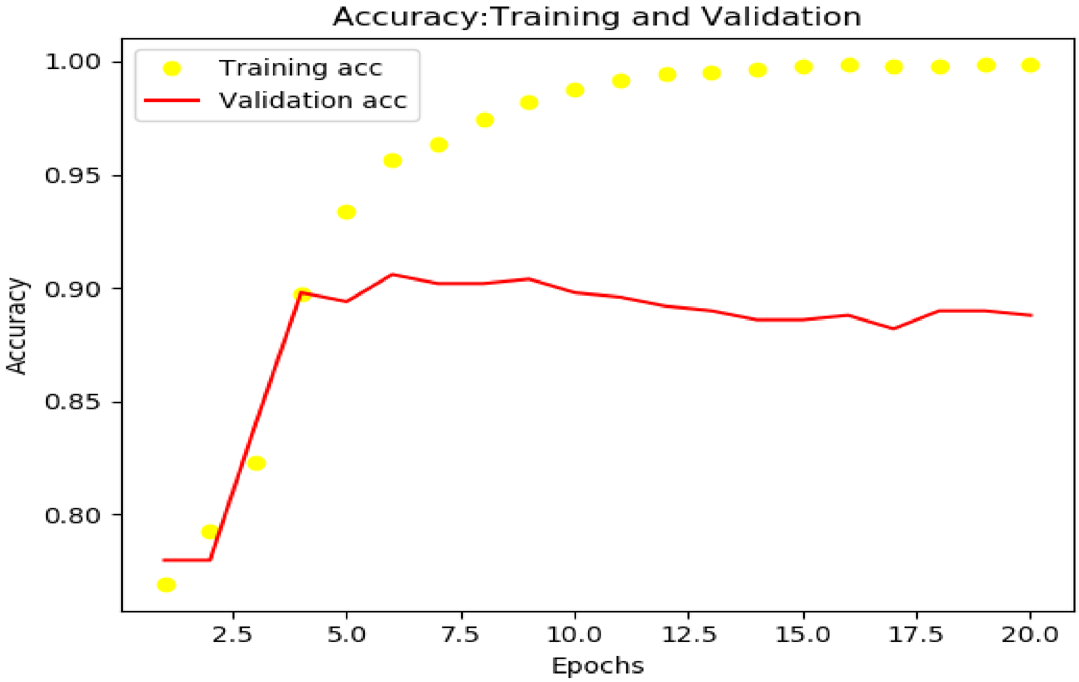
```
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'bo', label='Training loss',color = 'brown')
plt.plot(epochs, val_loss, 'b', label='Validation loss',color = 'green')
plt.title('Loss: Training and Validation')
plt.xlabel('X:Epochs')
plt.ylabel('Y:Loss')
plt.legend()
plt.show()
plt.clf()
#plotting training and validation accuracy
acc = history.history['acc']
val_acc = history.history['val_acc']
plt.plot(epochs, acc, 'bo', label='Training acc',color = 'yellow')
plt.plot(epochs, val_acc, 'b', label='Validation acc',color = 'red')
plt.title('Accuracy:Training and Validation')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
1536/8155 [====>.........................] - ETA: 0s - loss: 0.0077 - acc: 0.9987
2048/8155 [======>.......................] - ETA: 0s - loss: 0.0062 - acc: 0.9990
2560/8155 [========>.....................] - ETA: 0s - loss: 0.0059 - acc: 0.9988
3072/8155 [==========>...................] - ETA: 0s - loss: 0.0057 - acc: 0.9990
3584/8155 [============>.................] - ETA: 0s - loss: 0.0067 - acc: 0.9989
4096/8155 [==============>...............] - ETA: 0s - loss: 0.0062 - acc: 0.9990
4608/8155 [===============>..............] - ETA: 0s - loss: 0.0063 - acc: 0.9989
5120/8155 [=================>............] - ETA: 0s - loss: 0.0059 - acc: 0.9990
5632/8155 [===================>..........] - ETA: 0s - loss: 0.0059 - acc: 0.9989
6144/8155 [=====================>........] - ETA: 0s - loss: 0.0063 - acc: 0.9987
6656/8155 [=======================>......] - ETA: 0s - loss: 0.0064 - acc: 0.9986
7168/8155 [=========================>....] - ETA: 0s - loss: 0.0065 - acc: 0.9986
7680/8155 [===========================>..] - ETA: 0s - loss: 0.0067 - acc: 0.9984
8155/8155 [==============================] - 1s 133us/step - loss: 0.0065 - acc: 0.9985 - val_loss: 0.6637 - val_acc: 0.8880

Process finished with exit code 0
```

Graphs



Accuracy:



Question 2:

2. Implement the text classification with RNN/LSTM model, with a new dataset which is not used in the class

Workflow:

The result is shown using Tensor board .Here we can see the code along with Loss and Accuracy values

Important to note: Used a different data set known as collections.csv and shown the graph plots

Observations:

Here we make some observations for text classification using LSTM and find out Loss and Accuracy values. Since this question focuses on text classification we can see loss and accuracy values. We are going to plot Loss and Accuracy values for the dataset used

We keep the epochs and batch size along with other parameters the same for the first three observations

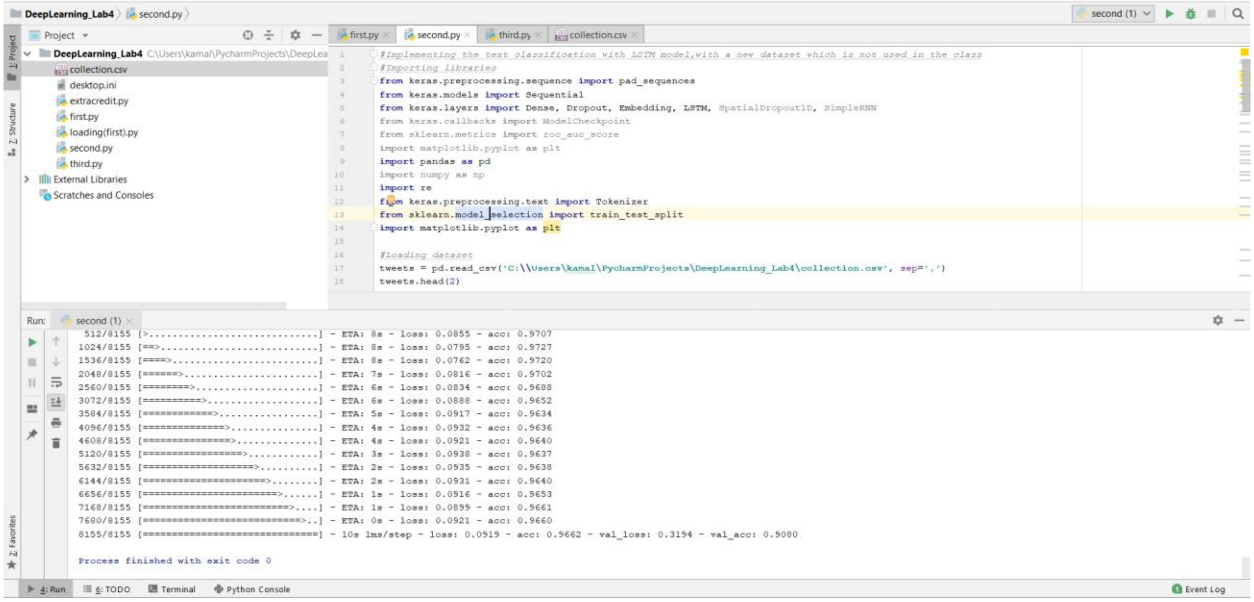
Code Snippet + Loss Values:

Batch Size:512

Epoch:20

Optimizer: Adam

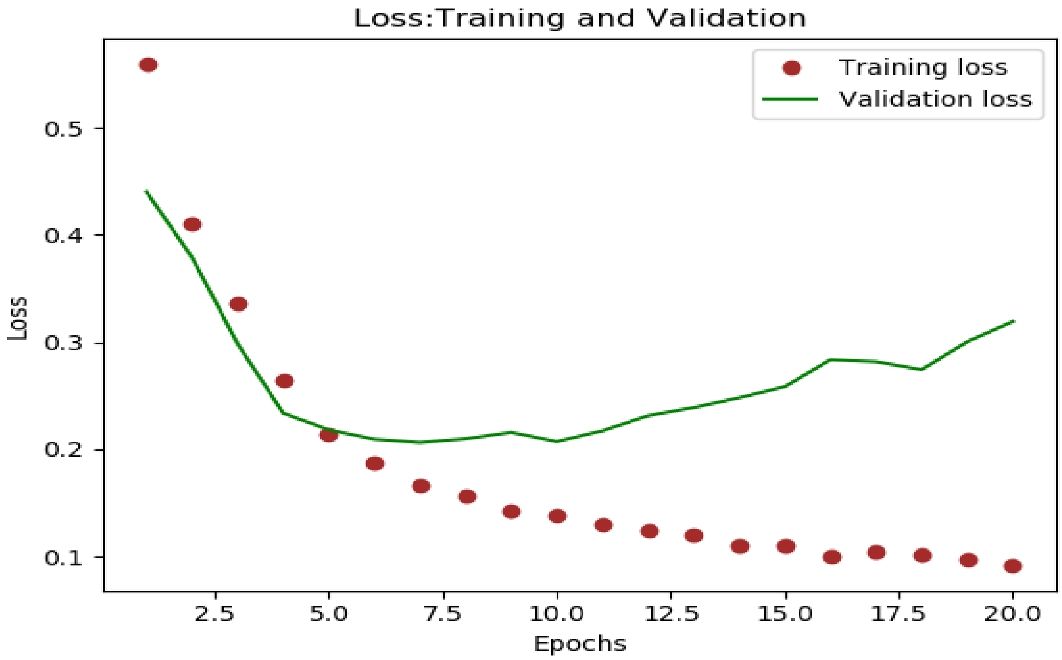
Activation Function: relu Loss:0.0919,Acc:0.9662

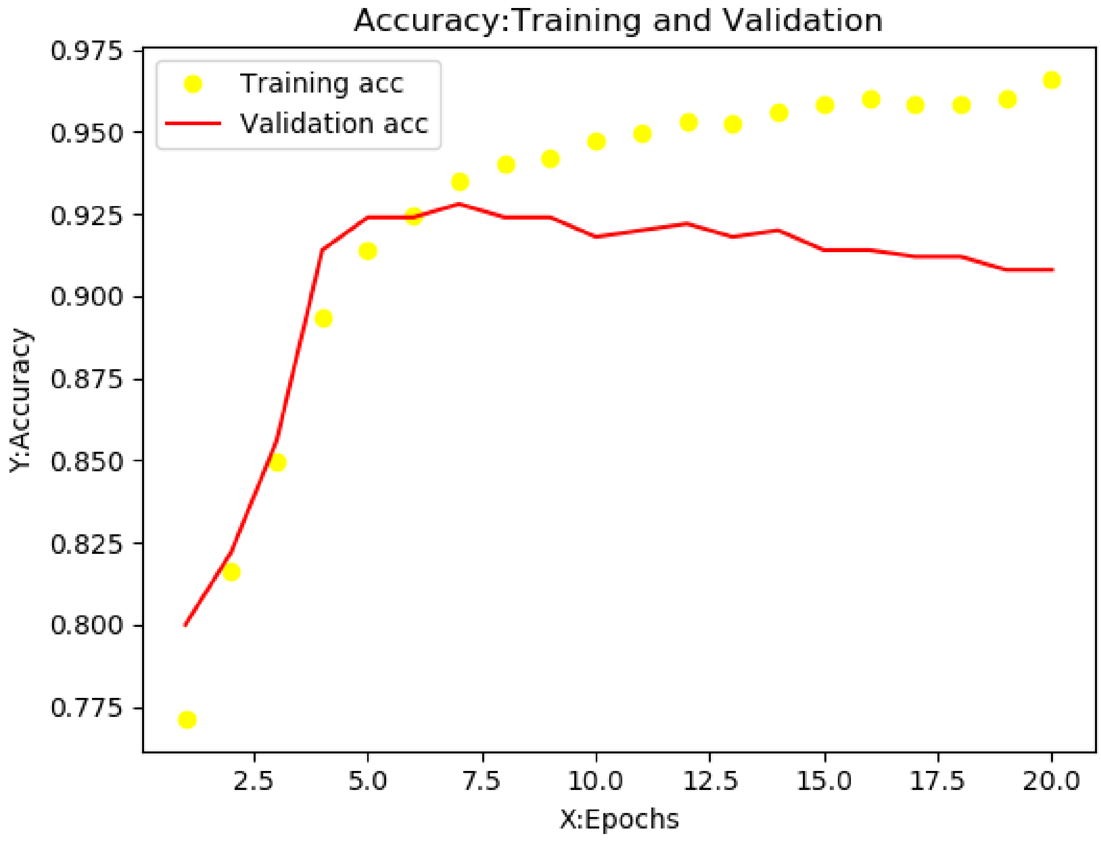


```
#Implementing the text classification with LSTM model,with a new dataset which is not used in the class
#Importing libraries
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Dropout, Embedding, LSTM, SpatialDropout1D, SimpleRNN
from keras.callbacks import ModelCheckpoint
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import re
from keras.preprocessing.text import Tokenizer
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

#Loading dataset
tweets = pd.read_csv('C:\\Users\kamal\PycharmProjects\DeepLearning_Lab4\collection.csv', sep=',')
tweets.head(2)
```

```
 512/8155 [>.............................] - ETA: 8s - loss: 0.0855 - acc: 0.9707
1024/8155 [==>...........................] - ETA: 8s - loss: 0.0795 - acc: 0.9727
1536/8155 [====>.........................] - ETA: 8s - loss: 0.0762 - acc: 0.9720
2048/8155 [======>.......................] - ETA: 7s - loss: 0.0816 - acc: 0.9702
2560/8155 [========>.....................] - ETA: 6s - loss: 0.0834 - acc: 0.9688
3072/8155 [==========>...................] - ETA: 6s - loss: 0.0888 - acc: 0.9652
3584/8155 [============>.................] - ETA: 5s - loss: 0.0917 - acc: 0.9634
4096/8155 [==============>...............] - ETA: 4s - loss: 0.0932 - acc: 0.9636
4608/8155 [===============>..............] - ETA: 4s - loss: 0.0921 - acc: 0.9640
5120/8155 [=================>............] - ETA: 3s - loss: 0.0938 - acc: 0.9637
5632/8155 [===================>..........] - ETA: 2s - loss: 0.0935 - acc: 0.9638
6144/8155 [=====================>........] - ETA: 2s - loss: 0.0931 - acc: 0.9640
6656/8155 [=======================>......] - ETA: 1s - loss: 0.0916 - acc: 0.9653
7168/8155 [=========================>....] - ETA: 1s - loss: 0.0899 - acc: 0.9661
7680/8155 [===========================>..] - ETA: 0s - loss: 0.0921 - acc: 0.9660
8155/8155 [==============================] - 10s 1ms/step - loss: 0.0919 - acc: 0.9662 - val_loss: 0.3194 - val_acc: 0.9080

Process finished with exit code 0
```

Graphs:


Loss:Training and Validation
Training loss
Validation loss
0.5
0.4
0.3
0.2
0.1
Loss
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs


Accuracy:Training and Validation
Training acc
Validation acc
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
0.775
Y:Accuracy
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
X:Epochs

Question 3:

3. Compare the results of CNN and RNN/LSTM models, for the text classification (same dataset for 2 models to compare) and describe, which model is best for the text classification based on your results

Workflow:

The result is shown using Tensor board .Here we can see the code along with Loss and Accuracy values

Important to note: Used a different data set known as collections.csv and shown the graph plots

Observations:

Comparing results of text classification for CNN and LSTM and find out Loss and Accuracy values . Since this question focuses on text classification we can see loss and accuracy values. We are going to plot Loss and Accuracy values for the dataset used

We keep the epochs and batch size along with other parameters the same for the first three observations

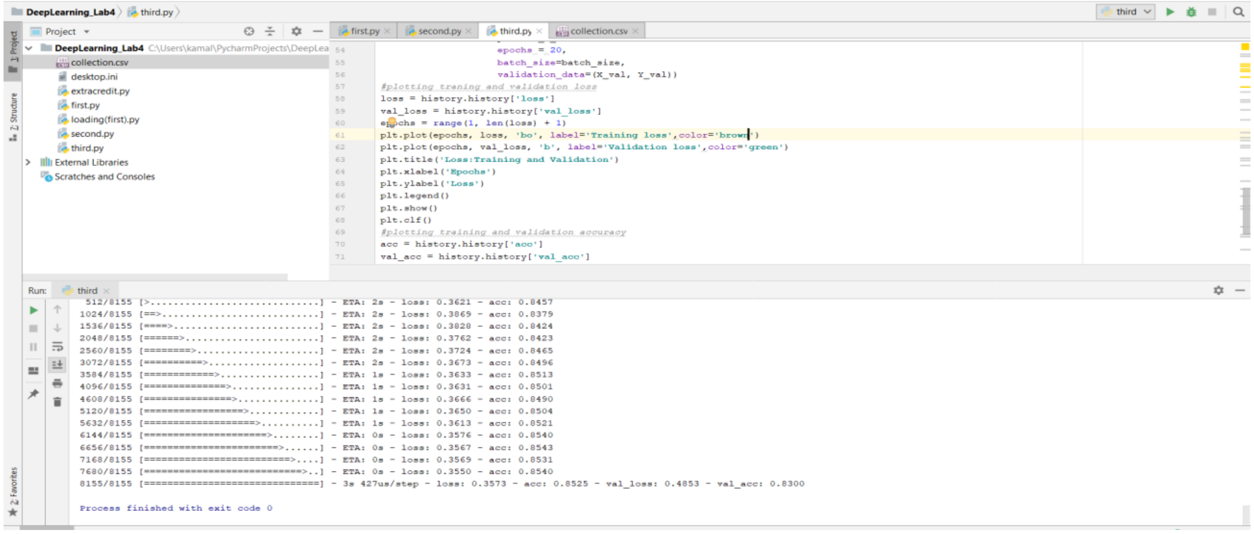
Code Snippet + Loss Values:

Batch Size:512

Epoch:20

Optimizer: Adam

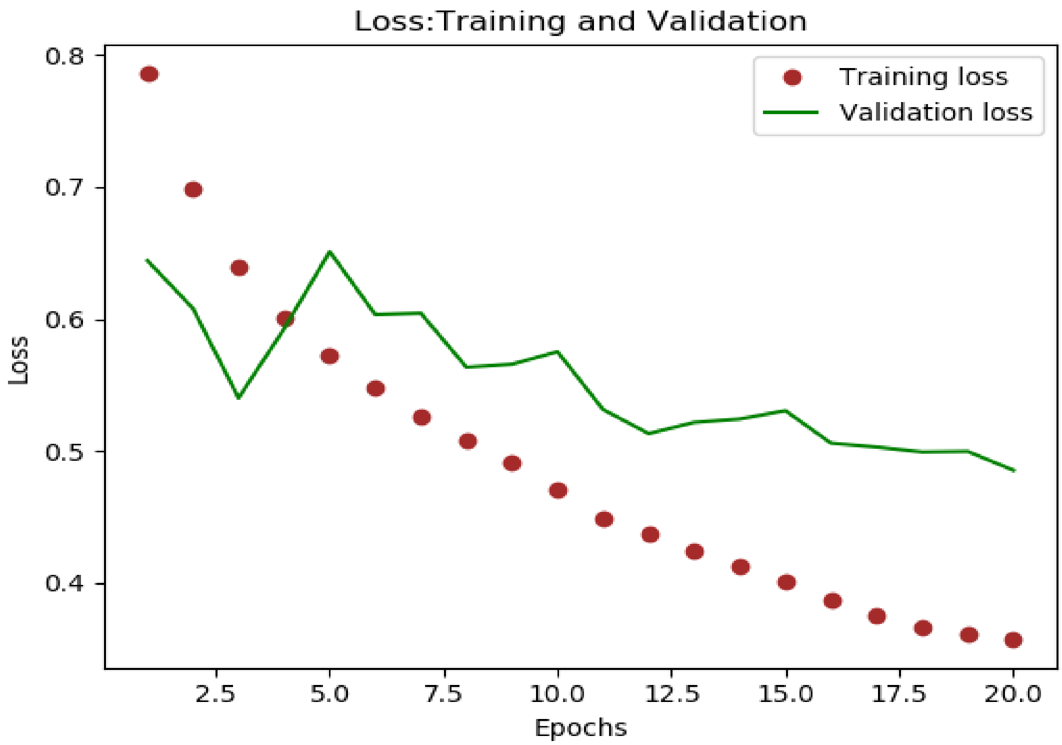
Activation Function: relu Loss:0.3573 , Acc:0.8525

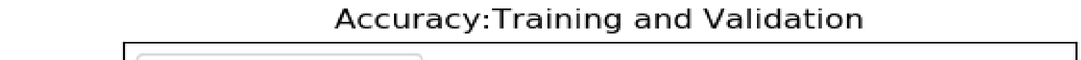


```
                    epochs = 20,
                    batch_size=batch_size,
                    validation_data=(X_val, Y_val))
#plotting traning and validation loss
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'bo', label='Training loss',color='brown')
plt.plot(epochs, val_loss, 'b', label='Validation loss',color='green')
plt.title('Loss:Training and Validation')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
plt.clf()
#plotting training and validation accuracy
acc = history.history['acc']
val_acc = history.history['val_acc']
```

```
 512/8155 [>.............................] - ETA: 2s - loss: 0.3621 - acc: 0.8457
1024/8155 [==>...........................] - ETA: 2s - loss: 0.3869 - acc: 0.8379
1536/8155 [====>.........................] - ETA: 2s - loss: 0.3828 - acc: 0.8424
2048/8155 [======>.......................] - ETA: 2s - loss: 0.3762 - acc: 0.8423
2560/8155 [========>.....................] - ETA: 2s - loss: 0.3724 - acc: 0.8465
3072/8155 [==========>...................] - ETA: 2s - loss: 0.3673 - acc: 0.8496
3584/8155 [============>.................] - ETA: 1s - loss: 0.3633 - acc: 0.8513
4096/8155 [==============>...............] - ETA: 1s - loss: 0.3631 - acc: 0.8501
4608/8155 [===============>..............] - ETA: 1s - loss: 0.3666 - acc: 0.8490
5120/8155 [=================>............] - ETA: 1s - loss: 0.3650 - acc: 0.8504
5632/8155 [===================>..........] - ETA: 1s - loss: 0.3613 - acc: 0.8521
6144/8155 [=====================>........] - ETA: 0s - loss: 0.3576 - acc: 0.8540
6656/8155 [=======================>......] - ETA: 0s - loss: 0.3567 - acc: 0.8543
7168/8155 [=========================>....] - ETA: 0s - loss: 0.3569 - acc: 0.8531
7680/8155 [===========================>..] - ETA: 0s - loss: 0.3550 - acc: 0.8540
8155/8155 [==============================] - 3s 427us/step - loss: 0.3573 - acc: 0.8525 - val_loss: 0.4853 - val_acc: 0.8300

Process finished with exit code 0
```

Graphs:


Loss:Training and Validation
Training loss
Validation loss
Loss
0.8
0.7
0.6
0.5
0.4
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs


Accuracy:Training and Validation
Training acc
Validation acc
Y:Accuracy
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
X:Epochs

Question 4:

4. Implement the image classification with CNN model, with a new dataset which is not used in the class (E.g. CIFAR 10 dataset)

Workflow:

The result is shown using Tensor board .Here we can see the code along with Loss and Accuracy values

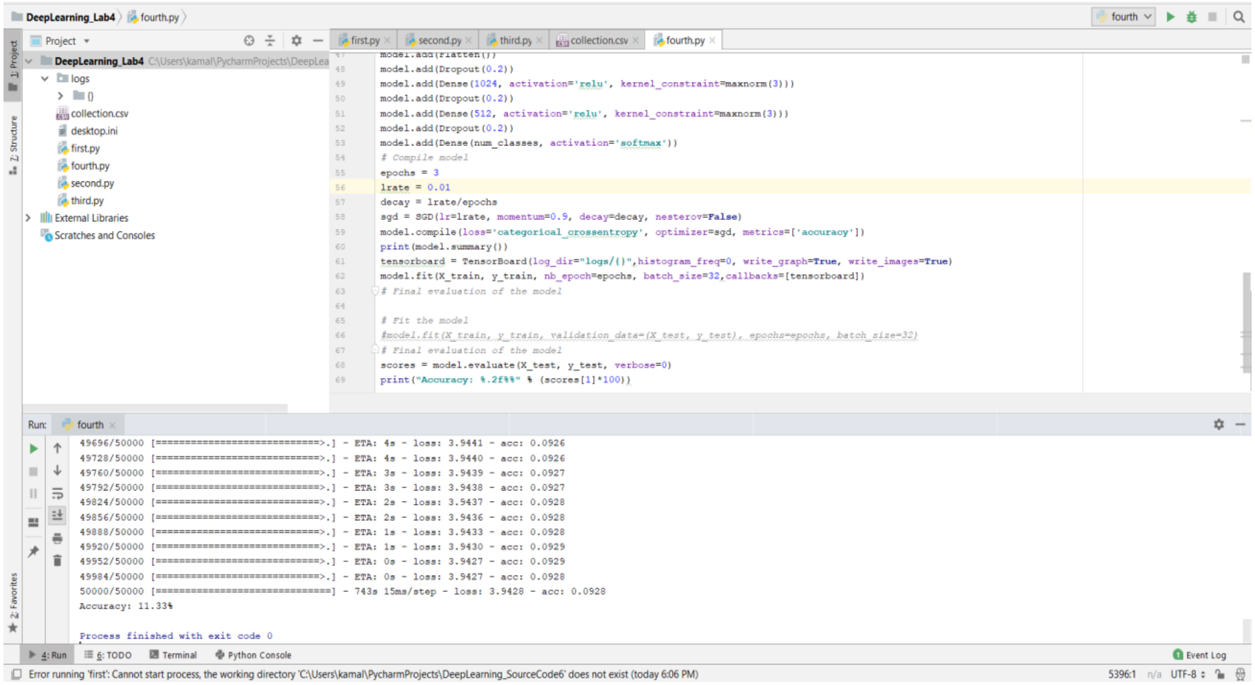
Important to note: Used a different data set known as CIFAR-100

Observations:

Here we make some observations for Image classification using CNN and find out Loss and Accuracy values. Since this question focuses on image classification we can see loss and accuracy values. We are going to plot Loss and Accuracy values for the dataset used

Code Snippet + Loss Values:

Epoch:3
Learning rate:0.01
Optimizer: sgd
Activation: relu
Loss:3.9
Accuracy:11.33



```
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(num_classes, activation='softmax'))
# Compile model
epochs = 3
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())
tensorboard = TensorBoard(log_dir="logs/{}",histogram_freq=0, write_graph=True, write_images=True)
model.fit(X_train, y_train, nb_epoch=epochs, batch_size=32,callbacks=[tensorboard])
# Final evaluation of the model

# Fit the model
#model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
49696/50000 [============================>.] - ETA: 4s - loss: 3.9441 - acc: 0.0926
49728/50000 [============================>.] - ETA: 4s - loss: 3.9440 - acc: 0.0926
49760/50000 [============================>.] - ETA: 3s - loss: 3.9439 - acc: 0.0927
49792/50000 [============================>.] - ETA: 3s - loss: 3.9438 - acc: 0.0927
49824/50000 [============================>.] - ETA: 2s - loss: 3.9437 - acc: 0.0928
49856/50000 [============================>.] - ETA: 2s - loss: 3.9436 - acc: 0.0928
49888/50000 [============================>.] - ETA: 1s - loss: 3.9433 - acc: 0.0928
49920/50000 [============================>.] - ETA: 1s - loss: 3.9430 - acc: 0.0929
49952/50000 [============================>.] - ETA: 0s - loss: 3.9427 - acc: 0.0929
49984/50000 [============================>.] - ETA: 0s - loss: 3.9427 - acc: 0.0928
50000/50000 [==============================] - 743s 15ms/step - loss: 3.9428 - acc: 0.0928
Accuracy: 11.33%

Process finished with exit code 0
```

Error running 'first': Cannot start process, the working directory 'C:\Users\kamal\PycharmProjects\DeepLearning_SourceCode6' does not exist (today 6:06 PM)

5396:1 n/a UTF-8