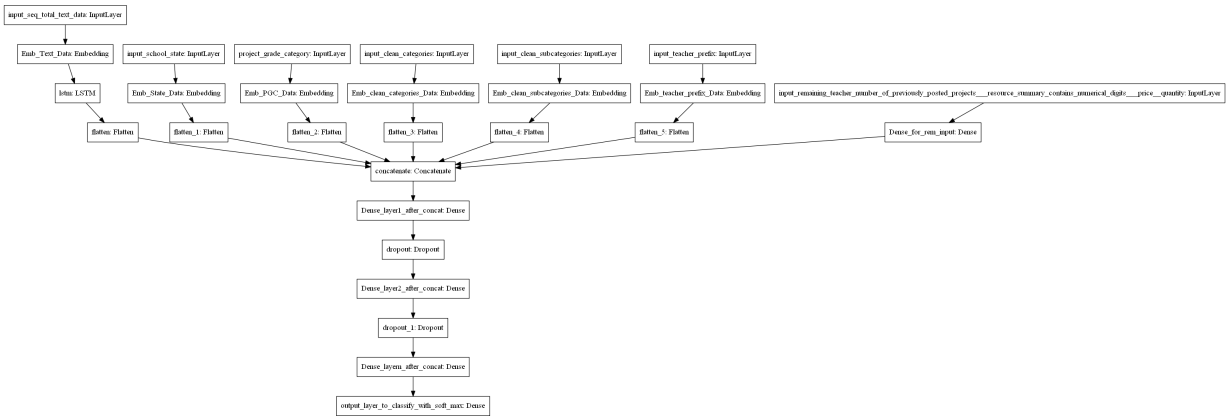
Assignment : 16

1. Download the preprocessed DonorsChoose data from here Dataset (https://drive.google.com/file/d/1GU3LIJJ3zS1xLXXe-sdItSJHtI5txjVO/view?usp=sharing)
2. Split the data into train, cv, and test
3. After step 2 you have to train 3 types of models as discussed below.
4. For all the model use 'auc' (https://scikit-learn.org/stable/modules/model_evaluation.html#roc-metrics) as a metric. check this (https://datascience.stackexchange.com/a/20192) for using auc as a metric. you need to print the AUC value for each epoch. Note: you should NOT use the tf.metric.auc
5. You are free to choose any number of layers/hiddden units but you have to use same type of architectures shown below.
6. You can use any one of the optimizers and choice of Learning rate and momentum, resources: cs231n class notes (http://cs231n.github.io/neural-networks-3/), cs231n class video (https://www.youtube.com/watch?v=hd_KFJ5ktUc).
7. You should Save the best model weights.
8. For all the model's use TensorBoard (https://www.youtube.com/watch?v=2U6Jl7oqRkM) and plot the Metric value and Loss with epoch. While submitting, take a screenshot of plots and include those images in .ipynb notebook and PDF.
9. Use Categorical Cross Entropy as Loss to minimize.
10. try to get AUC more than 0.75 for atleast one model

Model-1

Build and Train deep neural network as shown below



ref: https://i.imgur.com/w395Yk9.png

- **Input_seq_total_text_data** --- You have to give Total text data columns. After this use the

Embedding layer to get word vectors. Use given predefined glove word vectors, don't train any word vectors. After this use LSTM and get the LSTM output and Flatten that output.

- **Input_school_state** --- Give 'school_state' column as input to embedding layer and Train the Keras Embedding layer.
- **Project_grade_category** --- Give 'project_grade_category' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_categories** --- Give 'input_clean_categories' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_subcategories** --- Give 'input_clean_subcategories' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_subcategories** --- Give 'input_teacher_prefix' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_remaining_teacher_number_of_previously_posted_projects.*resource*summary_co** ---concatenate remaining columns and add a Dense layer after that.

- For LSTM, you can choose your sequence padding methods on your own or you can train your LSTM without padding, there is no restriction on that.

Below is an example of embedding layer for a categorical columns. In below code all are dummy values, we gave only for referance.

In [3]:
```
''''# https://stats.stackexchange.com/questions/270546/how-does-keras-embedding-l
input_layer = Input(shape=(n,))
embedding = Embedding(no_1, no_2, input_length=n)(input_layer)
flatten = Flatten()(embedding)'''
```

Out[3]:
```
"'# https://stats.stackexchange.com/questions/270546/how-does-keras-embedding-l
ayer-work\ninput_layer (https://stats.stackexchange.com/questions/270546/how-do
es-keras-embedding-layer-work\ninput_layer) = Input(shape=(n,))\nembedding = Em
bedding(no_1, no_2, input_length=n)(input_layer)\nflatten = Flatten()(embeddin
g)"
```

1. Go through this blog, if you have any doubt on using predefined Embedding values in Embedding layer - https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/ (https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/)

2. Please go through this link https://keras.io/getting-started/functional-api-guide/ (https://keras.io/getting-started/functional-api-guide/) and check the 'Multi-input and multi-output models' then you will get to know how to give multiple inputs.

In [4]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call d
rive.mount("/content/drive", force_remount=True).
```

In [0]:
```
path_train="/content/drive/My Drive/Colab Notebooks/preprocessed_data.csv"
```

In [6]:
```
#importing all the required lib
import pandas as pd
import numpy as np
import os
import math
from collections import defaultdict
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.layers import SpatialDropout1D, LSTM, BatchNormalization,concatenate,
from keras.models import Sequential
from keras import Model,Input
from keras.layers.convolutional import Conv2D,Conv1D
import keras.backend as k
from sklearn.metrics import roc_auc_score
import tensorflow as tf
import keras
from sklearn.utils import compute_class_weight
from keras.initializers import he_normal,glorot_normal
from keras.regularizers import l1,l2
from keras.callbacks import Callback, EarlyStopping, ModelCheckpoint,LearningRat
from time import time
from keras.callbacks import TensorBoard

from IPython.display import SVG, display
import pickle
import warnings
warnings.filterwarnings("ignore")
```

```
Using TensorFlow backend.
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade (https://www.tensorflow.org/guide/migrate) now or ensure your notebook will continue to use TensorFlow 1.x via the `%tensorflow_version 1.x` magic: more info (https://colab.research.google.com/notebooks/tensorflow_version.ipynb).

In [0]:
```
project_data=pd.read_csv(path_train)
```

In [8]:
```
project_data.shape
```

Out[8]:
```
(109248, 10)
```

In [9]:
```
project_data.head(1)
```

Out[9]:

	school_state	teacher_prefix	project_grade_category	Unnamed: 3	clean_categories	clean_subcate
0	ca	mrs	grades_prek_2	NaN	math_science	appliedsc health_lifes

In [0]:
```
class_label = project_data['project_is_approved']
class_wght = compute_class_weight("balanced", classes= np.unique(class_label),y=
```

In [0]:
```
y_label = project_data['project_is_approved'].values
project_data.drop(['project_is_approved'],axis=1,inplace=True)
x_train, x_test, y_train, y_test = train_test_split(project_data, y_label, strat
```

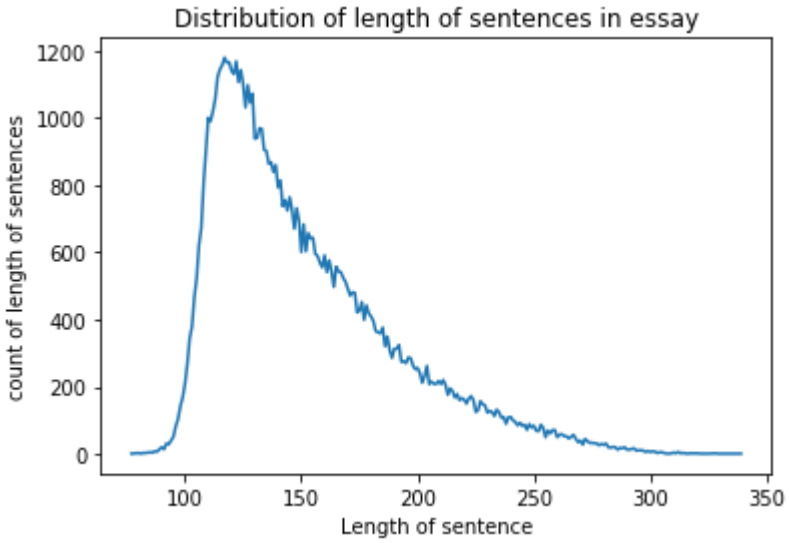
In [0]:
```
from keras.utils import to_categorical
y_train = to_categorical(y_train, num_classes=2)
y_test = to_categorical(y_test, num_classes=2)
```

In [13]:
```
token = Tokenizer()
token.fit_on_texts(x_train['essay'])
vocabulary_length = len(token.word_index) + 1
print('Total unique words in the x_train',vocabulary_length)
encoded_seq_train = token.texts_to_sequences(x_train['essay'])
encoded_seq_test = token.texts_to_sequences(x_test['essay'])
```

```
Total unique words in the x_train 49152
```

In [0]:
```
size = []
for sent in encoded_seq_train:
    size.append(len(sent))

s = list(set(size))
count = []
for i in s:
    count.append(size.count(i))
```

In [15]:
```
plt.plot(s,count)
plt.xlabel('Length of sentence')
plt.ylabel('count of length of sentences')
plt.title('Distribution of length of sentences in essay')
plt.show()
```



For padding the sequence we need to find est length which covers most of the length and we have found that around 95% of information is covered within length of 300. so we can take maximum length=300 for padding. 2.We will do post padding

```
# This is formatted as code
```

padding the encoded sequence

In [0]:
```
max_size = 300
pad_train = pad_sequences(encoded_seq_train, maxlen=max_size, padding='post')
pad_test = pad_sequences(encoded_seq_test, maxlen=max_size, padding='post')
```

In [17]:
```
print(len(pad_train[10]))
print(len(pad_test[1000]))
```

```
300
300
```

In [18]:
```
len(pad_train)
```

Out[18]: 76473

In [19]:
```
len(pad_test)
```

Out[19]: 32775

In [20]:
```
"""essay is vectorized into 300 dimension"""
```

Out[20]: 'essay is vectorized into 300 dimension'

In [0]:
```
tokenizing words using glove model
```

In [0]:
```
import pickle

from tqdm import tqdm
import os
```

In [0]:
```
import io
embeddings_index = {}
with io.open('/content/drive/My Drive/Colab Notebooks/glove_vectors','rb') as f:
  model = pickle.load(f)
  glove_words =  set(model.keys())

# for train
embedded_vector_train = np.zeros((vocabulary_length,300))
for word, i in token.word_index.items():
    if word in glove_words:
        embedded_vector = model[word]
        embedded_vector_train[i] = embedded_vector
```

In [23]:
```
len(embedded_vector_train[100])
```

Out[23]: 300

tokenizing features

In [0]:
```
def tokenization(feature):
    all_words = list(feature)
    distinct_words = list(set(feature))
    length = len(distinct_words)
    count = []
    for cat in distinct_words:
        count.append([all_words.count(cat),cat])
    count.sort()
    rank = {}
    for i in range(1,len(count)+1):
        rank.update({count[i-1][1] : i})
    return (rank,distinct_words,length)
```

Tokenizing clean categories, clean sub categories,state,techer prefix,project grade categories,essay

In [25]:
```
cty_rank,distinct_words,cty_length = tokenization(x_train['clean_categories'])
print(cty_rank)
print(distinct_words)
print(cty_length)
```

```
{'music_arts warmth care_hunger': 1, 'literacy_language warmth care_hunger': 2,
'music_arts appliedlearning': 3, 'appliedlearning warmth care_hunger': 4, 'hist
ory_civics health_sports': 5, 'math_science warmth care_hunger': 6, 'music_arts
history_civics': 7, 'health_sports warmth care_hunger': 8, 'music_arts health_s
ports': 9, 'specialneeds warmth care_hunger': 10, 'health_sports history_civic
s': 11, 'specialneeds health_sports': 12, 'history_civics appliedlearning': 13,
'literacy_language health_sports': 14, 'music_arts specialneeds': 15, 'health_s
ports music_arts': 16, 'health_sports appliedlearning': 17, 'appliedlearning hi
story_civics': 18, 'history_civics specialneeds': 19, 'health_sports math_scien
ce': 20, 'history_civics math_science': 21, 'history_civics music_arts': 22, 's
pecialneeds music_arts': 23, 'math_science health_sports': 24, 'appliedlearning
health_sports': 25, 'literacy_language appliedlearning': 26, 'math_science hist
ory_civics': 27, 'appliedlearning music_arts': 28, 'health_sports literacy_lang
uage': 29, 'literacy_language history_civics': 30, 'appliedlearning math_scienc
e': 31, 'math_science appliedlearning': 32, 'warmth care_hunger': 33, 'health_s
ports specialneeds': 34, 'history_civics literacy_language': 35, 'appliedlearni
ng specialneeds': 36, 'math_science music_arts': 37, 'literacy_language music_a
rts': 38, 'math_science specialneeds': 39, 'history_civics': 40, 'appliedlearni
ng literacy_language': 41, 'math_science literacy_language': 42, 'appliedlearni
ng': 43, 'literacy_language specialneeds': 44, 'specialneeds': 45, 'music_art
s': 46, 'health_sports': 47, 'literacy_language math_science': 48, 'math_scienc
e': 49, 'literacy_language': 50}
['appliedlearning specialneeds', 'math_science specialneeds', 'literacy_languag
e warmth care_hunger', 'math_science appliedlearning', 'literacy_language music
_arts', 'specialneeds warmth care_hunger', 'history_civics music_arts', 'music_
arts health_sports', 'history_civics', 'math_science warmth care_hunger', 'hist
ory_civics literacy_language', 'literacy_language appliedlearning', 'music_arts
history_civics', 'appliedlearning', 'health_sports appliedlearning', 'health_sp
orts math_science', 'specialneeds health_sports', 'health_sports', 'music_arts
specialneeds', 'warmth care_hunger', 'health_sports history_civics', 'literacy_
language', 'literacy_language math_science', 'appliedlearning literacy_languag
e', 'health_sports warmth care_hunger', 'history_civics appliedlearning', 'heal
th_sports literacy_language', 'math_science health_sports', 'appliedlearning he
alth_sports', 'appliedlearning history_civics', 'appliedlearning music_arts',
'math_science music_arts', 'appliedlearning math_science', 'history_civics math
_science', 'music_arts warmth care_hunger', 'specialneeds', 'math_science liter
acy_language', 'specialneeds music_arts', 'music_arts', 'music_arts appliedlear
ning', 'literacy_language health_sports', 'math_science', 'history_civics speci
alneeds', 'health_sports music_arts', 'history_civics health_sports', 'literacy
_language specialneeds', 'math_science history_civics', 'literacy_language hist
ory_civics', 'appliedlearning warmth care_hunger', 'health_sports specialneed
s']
50
```

In [26]:

```
enc_cty_train = []
enc_cty_test = []
clean_cat=x_train['clean_categories']
clean_cat1=x_test['clean_categories']
for cat in clean_cat:
    enc_cty_train.append(cty_rank[cat])

for cat in clean_cat1:
    if cat in distinct_words:
        enc_cty_test.append(cty_rank[cat])
    else:
        enc_cty_test.append(0)

enc_cty_train = np.asarray(enc_cty_train)
enc_cty_test = np.asarray(enc_cty_test)

print(enc_cty_train[0])
print(enc_cty_test[100])
```

```
49
46
```

In [27]:
```
sub_cty_rank,distinct_words,sub_cty_length = tokenization(x_train['clean_subcateg
print(sub_cty_rank)
print(distinct_words)
print(sub_cty_length)
enc_sub_cty_train = []
enc_sub_cty_test = []
clean_sub_cat=x_train['clean_subcategories']
clean_sub_cat1=x_test['clean_subcategories']
for cat in clean_sub_cat:
    enc_sub_cty_train.append(sub_cty_rank[cat])

for cat in clean_sub_cat1:
    if cat in distinct_words:
        enc_sub_cty_test.append(sub_cty_rank[cat])
    else:
        enc_sub_cty_test.append(0)

enc_sub_cty_train = np.asarray(enc_sub_cty_train)
enc_sub_cty_test = np.asarray(enc_sub_cty_test)

print(enc_sub_cty_train[0])
print(enc_sub_cty_test[100])
```

```
{'civics_government extracurricular': 1, 'civics_government foreignlanguage
s': 2, 'civics_government nutritioneducation': 3, 'civics_government parentin
volvement': 4, 'civics_government teamsports': 5, 'college_careerprep teamspo
rts': 6, 'college_careerprep warmth care_hunger': 7, 'communityservice financ
ialliteracy': 8, 'communityservice music': 9, 'economics literature_writing':
10, 'economics nutritioneducation': 11, 'economics other': 12, 'environmental
science teamsports': 13, 'esl economics': 14, 'esl nutritioneducation': 15,
'esl teamsports': 16, 'extracurricular financialliteracy': 17, 'financiallite
racy foreignlanguages': 18, 'financialliteracy health_wellness': 19, 'financi
alliteracy parentinvolvement': 20, 'financialliteracy performingarts': 21, 'f
inancialliteracy socialsciences': 22, 'foreignlanguages gym_fitness': 23, 'gy
m_fitness parentinvolvement': 24, 'gym_fitness socialsciences': 25, 'gym_fitn
ess warmth care_hunger': 26, 'parentinvolvement teamsports': 27, 'parentinvol
vement warmth care_hunger': 28, 'socialsciences teamsports': 29, 'appliedscie
nces nutritioneducation': 30, 'appliedsciences warmth care_hunger': 31, 'char
actereducation nutritioneducation': 32, 'college_careerprep economics': 33,
'college_careerprep gym_fitness': 34, 'communityservice nutritioneducation':
35, 'earlydevelopment economics': 36, 'earlydevelopment foreignlanguages': 3
7, 'earlydevelopment teamsports': 38, 'earlydevelopment warmth care_hunger':
```

In [28]:
```
state_rank,distinct_words,state_length = tokenization(x_train['school_state'])
print(state_rank)
print(distinct_words)
print(state_length)
enc_state_train = []
enc_state_test = []
clean_state=x_train['school_state']
clean_state1=x_test['school_state']
for cat in clean_state:
    enc_state_train.append(state_rank[cat])

for cat in clean_state1:
    if cat in distinct_words:
        enc_state_test.append(state_rank[cat])
    else:
        enc_state_test.append(0)

enc_state_train = np.asarray(enc_state_train)
enc_state_test = np.asarray(enc_state_test)

print(enc_state_train[0])
print(enc_state_test[100])
```

```
{'vt': 1, 'wy': 2, 'nd': 3, 'mt': 4, 'ne': 5, 'ri': 6, 'sd': 7, 'de': 8, 'nh':
9, 'ak': 10, 'me': 11, 'hi': 12, 'wv': 13, 'dc': 14, 'nm': 15, 'ks': 16, 'ia':
17, 'id': 18, 'ar': 19, 'co': 20, 'mn': 21, 'or': 22, 'ky': 23, 'ms': 24, 'nv':
25, 'md': 26, 'tn': 27, 'ct': 28, 'ut': 29, 'al': 30, 'wi': 31, 'va': 32, 'az':
33, 'nj': 34, 'wa': 35, 'ok': 36, 'la': 37, 'ma': 38, 'oh': 39, 'mo': 40, 'in':
41, 'pa': 42, 'mi': 43, 'ga': 44, 'sc': 45, 'il': 46, 'nc': 47, 'fl': 48, 'ny':
49, 'tx': 50, 'ca': 51}
['al', 'ma', 'in', 'tn', 'me', 'dc', 'id', 'wv', 'pa', 'hi', 'wy', 'co', 'ca',
'ak', 'nm', 'va', 'nv', 'ky', 'or', 'mt', 'mo', 'sc', 'ks', 'ny', 'ga', 'vt',
'tx', 'ut', 'mi', 'ar', 'fl', 'ri', 'de', 'ok', 'sd', 'la', 'az', 'nh', 'oh',
'wa', 'nj', 'ne', 'mn', 'ia', 'nc', 'wi', 'il', 'ms', 'nd', 'md', 'ct']
51
19
47
```

In [29]:
```
teacher_rank,distinct_words,teacher_length = tokenization(x_train['teacher_prefi
print(teacher_rank)
print(distinct_words)
print(teacher_length)
enc_teacher_train = []
enc_teacher_test = []
clean_teacher=x_train['teacher_prefix']
clean_teacher1=x_test['teacher_prefix']
for cat in clean_teacher:
    enc_teacher_train.append(teacher_rank[cat])

for cat in clean_teacher1:
    if cat in distinct_words:
        enc_teacher_test.append(teacher_rank[cat])
    else:
        enc_teacher_test.append(0)

enc_teacher_train = np.asarray(enc_teacher_train)
enc_teacher_test = np.asarray(enc_teacher_test)

print(enc_teacher_train[0])
print(enc_teacher_test[100])
```

```
{'dr': 1, 'teacher': 2, 'mr': 3, 'ms': 4, 'mrs': 5}
['mr', 'dr', 'teacher', 'ms', 'mrs']
5
5
5
```

In [30]:
```
pgc_rank,distinct_words,pgc_length = tokenization(x_train['project_grade_category
print(pgc_rank)
print(distinct_words)
print(pgc_length)
enc_pgc_train = []
enc_pgc_test = []
clean_pgc=x_train['project_grade_category']
clean_pgc1=x_test['project_grade_category']
for cat in clean_pgc:
    enc_pgc_train.append(pgc_rank[cat])

for cat in clean_pgc1:
    if cat in distinct_words:
        enc_pgc_test.append(pgc_rank[cat])
    else:
        enc_pgc_test.append(0)

enc_pgc_train = np.asarray(enc_pgc_train)
enc_pgc_test = np.asarray(enc_pgc_test)

print(enc_pgc_train[0])
print(enc_pgc_test[100])
```

```
{'grades_9_12': 1, 'grades_6_8': 2, 'grades_3_5': 3, 'grades_prek_2': 4}
['grades_6_8', 'grades_3_5', 'grades_9_12', 'grades_prek_2']
4
4
3
```

Standardizing train and test data

In [31]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(x_train['price'].values.reshape(-1,1))
x_train_price= scaler.transform(x_train['price'].values.reshape(-1,1))
x_test_price= scaler.transform(x_test['price'].values.reshape(-1,1))
print(x_train_price.shape, y_train.shape)
print(x_test_price.shape, y_test.shape)
```

```
(76473, 1) (76473, 2)
(32775, 1) (32775, 2)
```

In [32]:
```
scaler.fit(x_train['teacher_number_of_previously_posted_projects'].values.reshape
x_train_ppp= scaler.transform(x_train['teacher_number_of_previously_posted_proje
x_test_ppp= scaler.transform(x_test['teacher_number_of_previously_posted_projects
print(x_train_ppp.shape, y_train.shape)
print(x_test_ppp.shape, y_test.shape)
```

```
(76473, 1) (76473, 2)
(32775, 1) (32775, 2)
```

In [0]:
```
numerical_train = np.hstack((x_train_price,x_train_ppp))
numerical_test = np.hstack((x_test_price,x_test_ppp))
```

In [58]:
```
'''Defining AUC'''
```

Out[58]:
```
'Defining AUC'
```

In [0]:
```
def auc1(y_true, y_pred):
    if len(np.unique(y_true[:,1])) == 1:
        return 0.5
    else:
        return roc_auc_score(y_true, y_pred)

def auc(y_true, y_pred):
    return tf.py_func(auc1, (y_true, y_pred), tf.double)
```

TensorBoard installation

In [35]:
```
!pip install tensorboardcolab
```

```
Requirement already satisfied: tensorboardcolab in /usr/local/lib/python3.6/dis
t-packages (0.0.22)
```

In [36]:
```
from tensorboardcolab import TensorBoardColab, TensorBoardColabCallback

tbc=TensorBoardColab()
```

```
Wait for 8 seconds...
TensorBoard link:
https://0bacd753.ngrok.io (https://0bacd753.ngrok.io)
```

LSTM MODEL 1-Architecture

In [0]:
```
keras.backend.clear_session()
essay = Input(shape=(300,), name='essay_input')
x1 = Embedding(vocabulary_length, 300, weights=[embedded_vector_train], input_le
lstm_out = LSTM(100,recurrent_dropout=0.5,return_sequences=True)(x1)
flatten_1 = Flatten()(lstm_out)

state = Input(shape=(1,), name='school_state')
x2 = Embedding(state_length,5, input_length=1)(state)
flatten_2 = Flatten()(x2)


pgc = Input(shape=(1,), name='project_grade_category')
x3 = Embedding(pgc_length,5, input_length=1)(pgc)
flatten_3 = Flatten()(x3)


clean_cty = Input(shape=(1,), name='clean_categories')
x4 = Embedding(cty_length,5, input_length=1)(clean_cty)
flatten_4 = Flatten()(x4)


clean_sub_cty = Input(shape=(1,), name='clean_sub_categories')
x5 = Embedding(sub_cty_length,5, input_length=1)(clean_sub_cty)
flatten_5 = Flatten()(x5)


teacher= Input(shape=(1,), name='teacher_prefix')
x6 = Embedding(teacher_length,5, input_length=1)(teacher)
flatten_6 = Flatten()(x6)


numerical_input = Input(shape=(2,), name='remaining_input')
dense_1 = Dense(16, activation='relu',kernel_initializer="he_normal",kernel_regu



x = concatenate([flatten_1,flatten_2,flatten_3,flatten_4,flatten_5,flatten_6,den

x = Dense(128, activation='relu',kernel_initializer="he_normal",kernel_regularize
x = Dropout(.5)(x)
x = Dense(64, activation='relu',kernel_initializer="he_normal",kernel_regularize
x = Dropout(.5)(x)
x = BatchNormalization()(x)

x = Dense(32, activation='relu',kernel_initializer="he_normal",kernel_regularize
output = Dense(2, activation='softmax')(x)

model1 = Model(inputs=[essay,state,pgc,clean_cty,clean_sub_cty,teacher,numerical_
model1.compile(loss='categorical_crossentropy', optimizer=keras.optimizers.Adam(
print(model1.summary())
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:107: The name tf.reset_default_graph is deprecated. Ple
ase use tf.compat.v1.reset_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
```

```
tensorflow_backend.py:111: The name tf.placeholder_with_default is deprecate
d. Please use tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:66: The name tf.get_default_graph is deprecated. Please
use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:541: The name tf.placeholder is deprecated. Please use
tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:4432: The name tf.random_uniform is deprecated. Please
use tf.random.uniform instead.
```

In [0]:
```
train_1 = [pad_train,enc_cty_train,enc_sub_cty_train,enc_state_train,enc_pgc_tra
test_1 = [pad_test,enc_cty_test,enc_sub_cty_test,enc_state_test,enc_pgc_test,enc_
```

In [0]:
```
print(pad_train.shape)
print(enc_cty_train.shape)
print(enc_sub_cty_train.shape)
print(enc_state_train.shape)
print(enc_pgc_train.shape)
print(enc_teacher_train.shape)
print(numerical_train.shape)
```

```
(76473, 300)
(76473,)
(76473,)
(76473,)
(76473,)
(76473,)
(76473, 2)
```

In [0]:
```
print(pad_test.shape)
print(enc_cty_test.shape)
print(enc_sub_cty_test.shape)
print(enc_state_test.shape)
print(enc_pgc_test.shape)
print(enc_teacher_test.shape)
print(numerical_test.shape)
```

```
(32775, 300)
(32775,)
(32775,)
(32775,)
(32775,)
(32775,)
(32775, 2)
```

In [0]:
```
#model fitting
#https://machinelearningmastery.com/check-point-deep-learning-models-keras/
filepath="weights_copy.best.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_auc', verbose=1, save_best_o
callbacks_list = [checkpoint,TensorBoardColabCallback(tbc)]
model1.fit(train_1, y_train,epochs=5,verbose=1,batch_size=256,callbacks =callbac
```

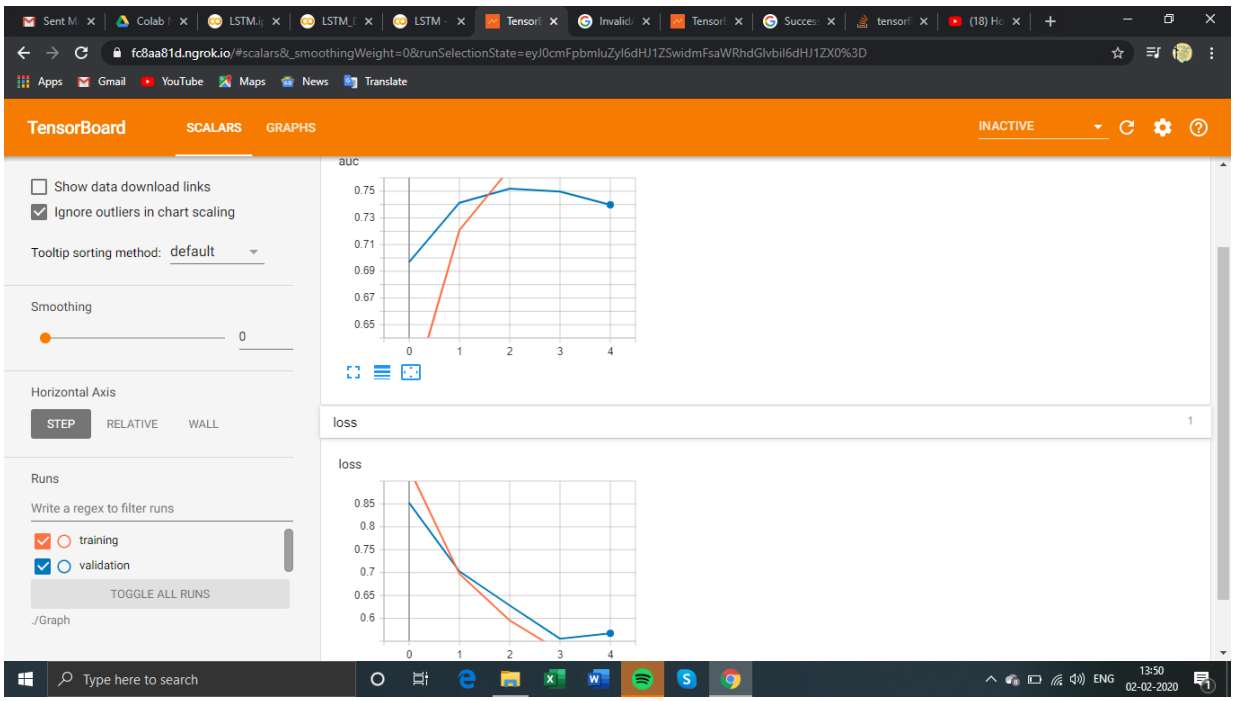
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/
python/ops/math_grad.py:1424: where (from tensorflow.python.ops.array_ops) is d
eprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:1033: The name tf.assign_add is deprecated. Please use tf.c
ompat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:1020: The name tf.assign is deprecated. Please use tf.compa
t.v1.assign instead.

Train on 76473 samples, validate on 32775 samples
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorboardcola
b/core.py:49: The name tf.summary.FileWriter is deprecated. Please use tf.compa
t.v1.summary.FileWriter instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbacks.
py:1122: The name tf.summary.merge_all is deprecated. Please use tf.compat.v1.s
ummary.merge_all instead.

Epoch 1/5
76473/76473 [==============================] - 186s 2ms/step - loss: 0.9272 - a
uc: 0.5910 - val_loss: 0.8525 - val_auc: 0.6968

Epoch 00001: val_auc improved from -inf to 0.69684, saving model to weights_cop
y.best.hdf5
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorboardcola
b/callbacks.py:51: The name tf.Summary is deprecated. Please use tf.compat.v1.S
ummary instead.

Epoch 2/5
76473/76473 [==============================] - 185s 2ms/step - loss: 0.6972 - a
uc: 0.7208 - val_loss: 0.7029 - val_auc: 0.7413

Epoch 00002: val_auc improved from 0.69684 to 0.74131, saving model to weights_
copy.best.hdf5
Epoch 3/5
76473/76473 [==============================] - 183s 2ms/step - loss: 0.5952 - a
uc: 0.7669 - val_loss: 0.6281 - val_auc: 0.7520

Epoch 00003: val_auc improved from 0.74131 to 0.75196, saving model to weights_
copy.best.hdf5
Epoch 4/5
76473/76473 [==============================] - 179s 2ms/step - loss: 0.5285 - a
uc: 0.7992 - val_loss: 0.5556 - val_auc: 0.7497
```

```
Epoch 00004: val_auc did not improve from 0.75196
Epoch 5/5
76473/76473 [==============================] - 177s 2ms/step - loss: 0.4749 - a
uc: 0.8310 - val_loss: 0.5672 - val_auc: 0.7398

Epoch 00005: val_auc did not improve from 0.75196
```

Out[43]: <keras.callbacks.History at 0x7f3e6f11cdd8>

In [0]:
```
from IPython.display import Image
Image(retina=True, filename='/content/drive/My Drive/m1.png')
```

Out[70]:



auc graph:

Blue curve validation AUC: 0.75196

Red curve Train AUC :0.7669

Model-2

Use the same model as above but for 'input_seq_total_text_data' give only some words in the sentance not all the words. Filter the words as below.

1. Train the TF-IDF on the Train data feature 'essay'

2. Get the idf value for each word we have in the train data.

3. Remove the low idf value and high idf value words from our data. Do some analysis on the Idf values and based on those values choose the low and high threshold value. Because very frequent words and very very rar

e words don't give much information. (you can plot a box plots and take only the idf scores within IQR range and corresponding words)

4. Train the LSTM after removing the Low and High idf value words. (In model-1 Train on total data but in Model-2 train on data after removing some words based on IDF values)

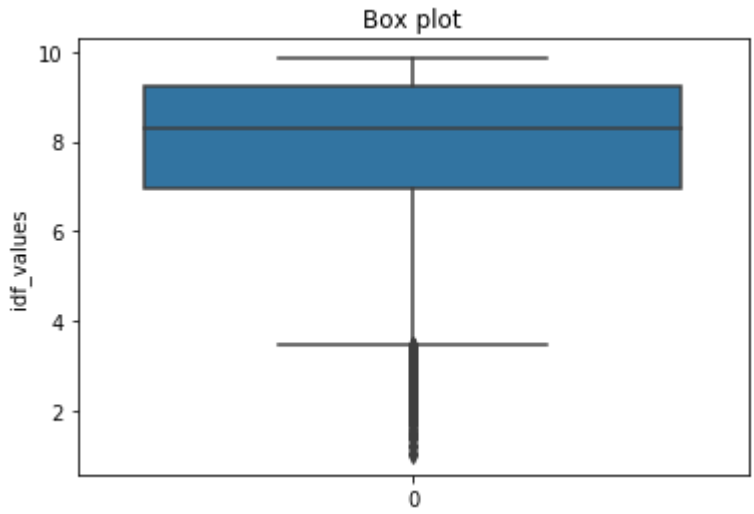
Performing tfid vectorization on essay data

In [0]:
```
vectorizer = TfidfVectorizer(min_df=10)
x_tfidf_train = vectorizer.fit_transform(x_train['essay'].values)
print(x_tfidf_train.shape)
```

```
(76473, 14523)
```

visualizing idf values using box plot to remove redundant and rare words

In [0]:
```
import seaborn as sns
idf_values = vectorizer.idf_
sns.boxplot(data=idf_values)
plt.title('Box plot')
plt.ylabel('idf_values')
plt.show()
```



Filtering words

In [0]:
```
features = np.asarray(vectorizer.get_feature_names())

f_i = []
for i in range(len(idf_values)):
    if idf_values[i] >= 5 and idf_values[i] <=9:
        f_i.append(i)

filtered_words = []
for i in f_i:
    filtered_words.append(features[i])
```

In [0]:
```
print('all words = ', len(features))
print('Filtered words = ',len(filtered_words))
```

```
all words =  14523
Filtered words =  8900
```

In [0]:
```
# keeping words present in filtered words in train_data
from tqdm import tqdm
x_tfidf_train_new = []
for sent in tqdm(x_train['essay']):
    line = []
    for word in sent.split():
        if word in filtered_words:
            line.append(word)
    x_tfidf_train_new.append(' '.join(line))

#keeping words present in filtered data in test_data
x_tfidf_test_new = []
for sent in tqdm(x_test['essay']):
    line = []
    for word in sent.split():
        if word in filtered_words:
            line.append(word)
    x_tfidf_test_new.append(' '.join(line))

print(len(x_tfidf_train_new))
print(len(x_tfidf_test_new))
```

```
100%|██████████| 76473/76473 [19:46<00:00, 64.44it/s]
100%|██████████| 32775/32775 [08:21<00:00, 65.30it/s]

76473
32775
```

tokenizing the idf sentences

In [0]:
```
token_tfidf = Tokenizer()
token_tfidf.fit_on_texts(x_tfidf_train_new)
vocabulary_length = len(token_tfidf.word_index) + 1
print('Total distinct words present in the x_tfidf_train_new',vocabulary_length)
enc_tfidf_train_new = token_tfidf.texts_to_sequences(x_tfidf_train_new)
enc_tfidf_test_new = token_tfidf.texts_to_sequences(x_tfidf_test_new)
```

```
Total distinct words present in the x_tfidf_train_new 8901
```

Padding

In [0]:
```
max_size = 300
pad_tfidf_train = pad_sequences(enc_tfidf_train_new, maxlen=max_size, padding='p
pad_tfidf_test = pad_sequences(enc_tfidf_test_new, maxlen=max_size, padding='pos
print(len(pad_tfidf_train[10]))
print(len(pad_tfidf_test[1000]))
```

```
300
300
```

In [0]:
```
# for train
embedded_vector_train_2 = np.zeros((vocabulary_length,300))
for word, i in token_tfidf.word_index.items():
    if word in glove_words:
        embedded_vector = model[word]
        embedded_vector_train_2[i] = embedded_vector
```

In [0]:
```
vocabulary_length
```

Out[47]:
```
8901
```

Model architecture 2:

In [0]:
```
keras.backend.clear_session()
essay = Input(shape=(300,), name='essay_input')
x1 = Embedding(vocabulary_length, 300, weights=[embedded_vector_train_2], input_
lstm_out = LSTM(100,recurrent_dropout=0.5,return_sequences=True)(x1)
flatten_1 = Flatten()(lstm_out)

state = Input(shape=(1,), name='school_state')
x2 = Embedding(state_length,5, input_length=1)(state)
flatten_2 = Flatten()(x2)


pgc = Input(shape=(1,), name='project_grade_category')
x3 = Embedding(pgc_length,5, input_length=1)(pgc)
flatten_3 = Flatten()(x3)


clean_cty = Input(shape=(1,), name='clean_categories')
x4 = Embedding(cty_length,5, input_length=1)(clean_cty)
flatten_4 = Flatten()(x4)


clean_sub_cty = Input(shape=(1,), name='clean_sub_categories')
x5 = Embedding(sub_cty_length,5, input_length=1)(clean_sub_cty)
flatten_5 = Flatten()(x5)


teacher= Input(shape=(1,), name='teacher_prefix')
x6 = Embedding(teacher_length,5, input_length=1)(teacher)
flatten_6 = Flatten()(x6)


numerical_input = Input(shape=(2,), name='remaining_input')
dense_1 = Dense(16, activation='relu',kernel_initializer="he_normal",kernel_regu



x = concatenate([flatten_1,flatten_2,flatten_3,flatten_4,flatten_5,flatten_6,den

x = Dense(128, activation='relu',kernel_initializer="he_normal",kernel_regulariz
x = Dropout(.5)(x)
x = Dense(64, activation='relu',kernel_initializer="he_normal",kernel_regularize
x = Dropout(.5)(x)
x = BatchNormalization()(x)

x = Dense(32, activation='relu',kernel_initializer="he_normal",kernel_regularize
output = Dense(2, activation='softmax')(x)

model2 = Model(inputs=[essay,state,pgc,clean_cty,clean_sub_cty,teacher,numerical_
tensorboard = TensorBoard(log_dir="logs".format(time()))
model2.compile(loss='categorical_crossentropy', optimizer=keras.optimizers.Adam(
print(model2.summary())
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:107: The name tf.reset_default_graph is deprecated. Ple
ase use tf.compat.v1.reset_default_graph instead.
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:111: The name tf.placeholder_with_default is deprecate
d. Please use tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:66: The name tf.get_default_graph is deprecated. Please
use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:541: The name tf.placeholder is deprecated. Please use
tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:4432: The name tf.random_uniform is deprecated. Please
use tf.random.uniform instead.
```

In [0]:
```
train_2 = [pad_tfidf_train,enc_cty_train,enc_sub_cty_train,enc_state_train,enc_pg
test_2 = [pad_tfidf_test,enc_cty_test,enc_sub_cty_test,enc_state_test,enc_pgc_te
```

In [0]:
```
print(pad_tfidf_train.shape)
print(enc_cty_train.shape)
print(enc_sub_cty_train.shape)
print(enc_state_train.shape)
print(enc_pgc_train.shape)
print(enc_teacher_train.shape)
print(numerical_train.shape)
```

```
(76473, 300)
(76473,)
(76473,)
(76473,)
(76473,)
(76473,)
(76473, 2)
```

In [0]:
```
print(pad_tfidf_test.shape)
print(enc_cty_test.shape)
print(enc_sub_cty_test.shape)
print(enc_state_test.shape)
print(enc_pgc_test.shape)
print(enc_teacher_test.shape)
print(numerical_test.shape)
```

```
(32775, 300)
(32775,)
(32775,)
(32775,)
(32775,)
(32775,)
(32775, 2)
```

In [0]:
```
filepath="weights_copy2.best.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_auc', verbose=1, save_best_o
callbacks_list = [checkpoint,tbCallBack]
model2.fit(train_2, y_train,epochs=5,verbose=1,batch_size=256,callbacks =callbacl
```

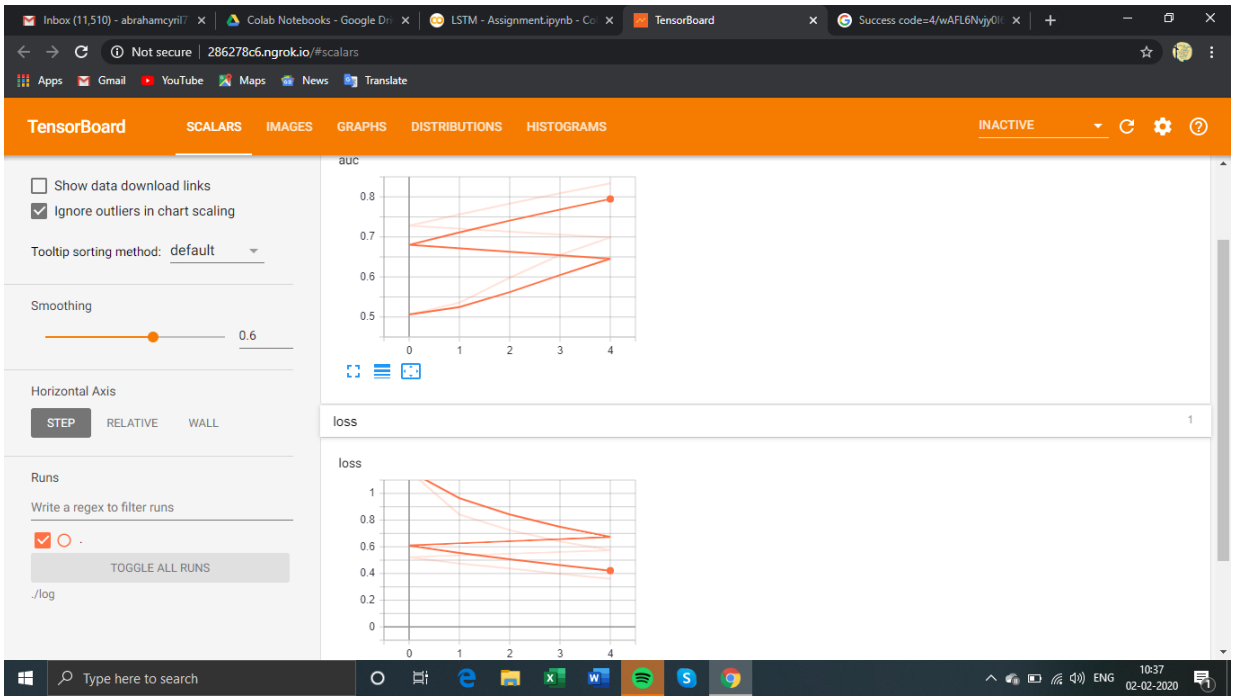
```
Train on 76473 samples, validate on 32775 samples
Epoch 1/5
76473/76473 [==============================] - 177s 2ms/step - loss: 0.5215 - a
uc: 0.7281 - val_loss: 0.5292 - val_auc: 0.6698

Epoch 00001: val_auc improved from -inf to 0.66984, saving model to weights_cop
y2.best.hdf5
Epoch 2/5
76473/76473 [==============================] - 173s 2ms/step - loss: 0.4755 - a
uc: 0.7562 - val_loss: 0.5072 - val_auc: 0.6784

Epoch 00002: val_auc improved from 0.66984 to 0.67841, saving model to weights_
copy2.best.hdf5
Epoch 3/5
76473/76473 [==============================] - 173s 2ms/step - loss: 0.4384 - a
uc: 0.7834 - val_loss: 0.5024 - val_auc: 0.6603

Epoch 00003: val_auc did not improve from 0.67841
Epoch 4/5
76473/76473 [==============================] - 171s 2ms/step - loss: 0.3975 - a
uc: 0.8089 - val_loss: 0.4980 - val_auc: 0.6630

Epoch 00004: val_auc did not improve from 0.67841
Epoch 5/5
76473/76473 [==============================] - 172s 2ms/step - loss: 0.3594 - a
uc: 0.8339 - val_loss: 0.5004 - val_auc: 0.6545

Epoch 00005: val_auc did not improve from 0.67841
```

Out[54]:
```
<keras.callbacks.History at 0x7f6051006cc0>
```

Train AUC and loss:

In [0]:
```
Image(retina=True, filename='/content/drive/My Drive/Colab Notebooks/train_auc_2
```

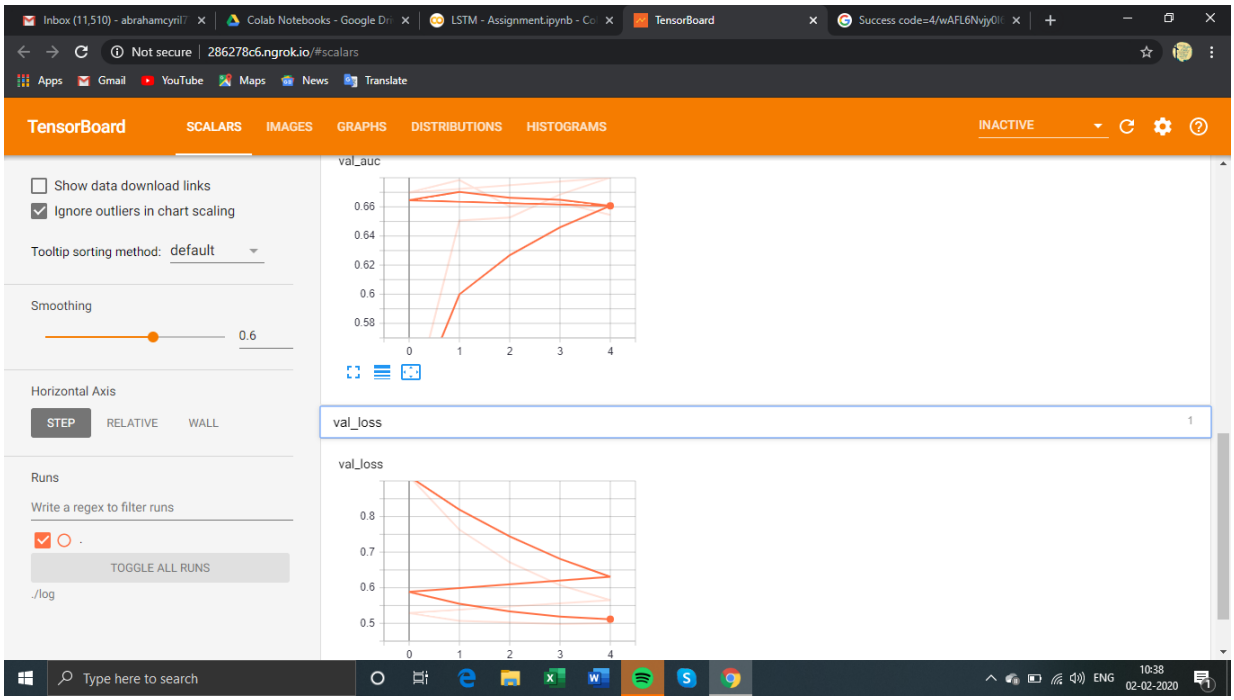
Out[47]:



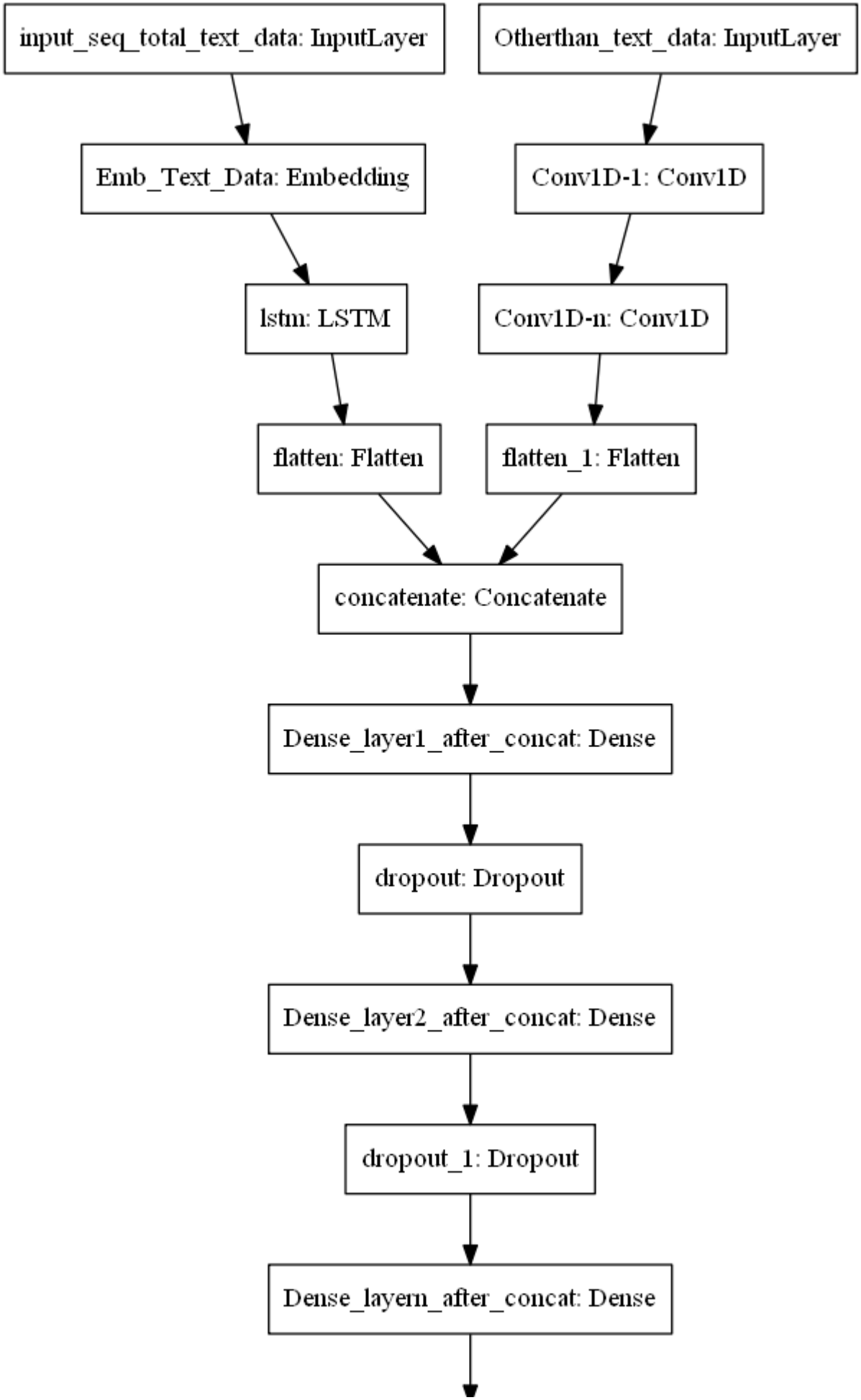
Validation AUC and loss

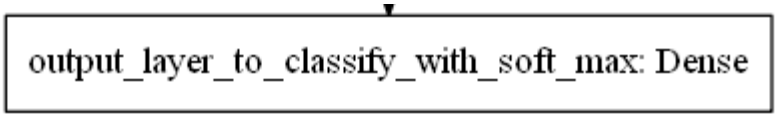
In [0]:
```
Image(retina=True, filename='/content/drive/My Drive/Colab Notebooks/val_auc_2.pr
```

Out[48]:



Model-3


input_seq_total_text_data: InputLayer
Otherthan_text_data: InputLayer
Emb_Text_Data: Embedding
Conv1D-1: Conv1D
lstm: LSTM
Conv1D-n: Conv1D
flatten: Flatten
flatten_1: Flatten
concatenate: Concatenate
Dense_layer1_after_concat: Dense
dropout: Dropout
Dense_layer2_after_concat: Dense
dropout_1: Dropout
Dense_layern_after_concat: Dense



ref: https://i.imgur.com/fkQ8nGo.png

- **input_seq_total_text_data**:

  . Use text column('essay'), and use the Embedding layer to get word vectors.

  . Use given predefined glove word vectors, don't train any word vectors.

  . Use LSTM that is given above, get the LSTM output and Flatten that output.

  . You are free to preprocess the input text as you needed.

- **Other_than_text_data**:

  . Convert all your Categorical values to onehot coded and then concatenate all these onehot vectors

  . Neumerical values and use CNN1D (https://keras.io/getting-started/sequential-model-guide/#sequence-classification-with-1d-convolutions) as shown in above figure.

  . You are free to choose all CNN parameters like kernel sizes, stride.

For essay the processing is same as for we did for first model.

Tokenizing subject category,subcategory,teacher prefix,state,project grade categories using *countvectorizer*

In [37]:
```
token_cat= CountVectorizer()

# integer encode the documents
proj_cat_train = token_cat.fit_transform(x_train['clean_categories'])
proj_cat_test = token_cat.transform(x_test['clean_categories'])


print(proj_cat_train.shape)
print(proj_cat_test.shape)
```

```
(76473, 9)
(32775, 9)
```

In [38]:
```
token_sub_cat = CountVectorizer()


subcat_train = token_sub_cat.fit_transform(x_train['clean_subcategories'])
subcat_test = token_sub_cat.transform(x_test['clean_subcategories'])

print(subcat_train.shape)
print(subcat_test.shape)
```

```
(76473, 30)
(32775, 30)
```

In [39]:
```
token_state = CountVectorizer()

# integer encode the documents
state_train = token_state.fit_transform(x_train['school_state'])
state_test = token_state.transform(x_test['school_state'])


print(state_train.shape)
print(state_test.shape)
```

```
(76473, 51)
(32775, 51)
```

In [40]:
```
token_pgc = CountVectorizer()

# integer encode the documents
pgc_train = token_pgc.fit_transform(x_train['project_grade_category'])
pgc_test = token_pgc.transform(x_test['project_grade_category'])


print(pgc_train.shape)
print(pgc_test.shape)
```

```
(76473, 4)
(32775, 4)
```

In [41]:
```
token_teacher = CountVectorizer()
# integer encode the documents
teacher_train = token_teacher.fit_transform(x_train['teacher_prefix'])
teacher_test = token_teacher.transform(x_test['teacher_prefix'])

print(teacher_train.shape)
print(teacher_test.shape)
```

```
(76473, 5)
(32775, 5)
```

In [42]:
```
print(pad_train.shape)
print(proj_cat_train.shape)
print(subcat_train.shape)
print(state_train.shape)
print(pgc_train.shape)
print(teacher_train.shape)
print(numerical_train.shape)
```

```
(76473, 300)
(76473, 9)
(76473, 30)
(76473, 51)
(76473, 4)
(76473, 5)
(76473, 2)
```

In [0]:
```
from scipy.sparse import hstack
```

In [0]:
```
train_3 = hstack((proj_cat_train,subcat_train,state_train,pgc_train,teacher_trai
test3 = hstack((proj_cat_test,subcat_test,state_test,pgc_test,teacher_test,numer
```

In [45]:
```
train_3.shape
```

Out[45]:
```
(76473, 101)
```

In [46]:
```
test3.shape
```

Out[46]:
```
(32775, 101)
```

In [0]:
```
train_m3=train_3.todense()
test_m3 =test3.todense()
```

In [0]:
```
train_3 = np.resize(train_m3,new_shape=(76473,101,1))
test_3 =np.resize(test_m3,new_shape=(32775,101,1))
```

In [49]:
```
train_3.shape
```

Out[49]:
```
(76473, 101, 1)
```

In [50]:
```
embedded_vector_train.shape[0]
```

Out[50]: 49152

In [51]:
```
vocabulary_length
```

Out[51]: 49152

defining model 3 architecture

In [52]:
```
essay = Input(shape=(300,), name='essay_input')#from model_1

x = Embedding(embedded_vector_train.shape[0], 300, weights=[embedded_vector_trai
lstm = LSTM(100,recurrent_dropout=0.5,return_sequences=True)(x)
flatten = Flatten()(lstm)


remaining_input = Input(shape=(101,1),name='rest_all')
x = Conv1D(filters=256, kernel_size = 3, padding='valid', kernel_initializer='gl
x = Conv1D(filters=256, kernel_size = 3, padding='valid', kernel_initializer='gl
flatten_1 = Flatten()(x)

#concatenate
concat = concatenate([flatten,flatten_1])

x = Dense(300, activation='relu',kernel_initializer="glorot_normal",kernel_regul
x = Dropout(.5)(x)
x = Dense(256, activation='relu',kernel_initializer="glorot_normal",kernel_regul
x = Dropout(.5)(x)
x = BatchNormalization()(x)
x = Dense(128, activation='relu',kernel_initializer="glorot_normal",kernel_regul
output = Dense(2, activation='softmax')(x)

model3 = Model(inputs=[essay,remaining_input], outputs=[output])
print(model3.summary())
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:541: The name tf.placeholder is deprecated. Please use tf.c
ompat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:66: The name tf.get_default_graph is deprecated. Please use
tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:4432: The name tf.random_uniform is deprecated. Please use
tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:190: The name tf.get_default_session is deprecated. Please
use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:197: The name tf.ConfigProto is deprecated. Please use tf.c
ompat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:203: The name tf.Session is deprecated. Please use tf.compa
t.v1.Session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:207: The name tf.global_variables is deprecated. Please use
tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:216: The name tf.is_variable_initialized is deprecated. Ple
ase use tf.compat.v1.is_variable_initialized instead.
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:223: The name tf.variables_initializer is deprecated. Pleas
e use tf.compat.v1.variables_initializer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:148: The name tf.placeholder_with_default is deprecated. Pl
ease use tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:3733: calling dropout (from tensorflow.python.ops.nn_ops) w
ith keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - kee
p_prob`.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:4479: The name tf.truncated_normal is deprecated. Please us
e tf.random.truncated_normal instead.

Model: "model_1"
__________________________________________________________________________________
_____________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
==================
essay_input (InputLayer)        (None, 300)          0
__________________________________________________________________________________
_____________________
rest_all (InputLayer)           (None, 101, 1)       0
__________________________________________________________________________________
_____________________
embedding_1 (Embedding)         (None, 300, 300)     14745600    essay_input[0]
[0]
__________________________________________________________________________________
_____________________
conv1d_1 (Conv1D)               (None, 99, 128)      512         rest_all[0][0]
__________________________________________________________________________________
_____________________
lstm_1 (LSTM)                   (None, 300, 100)     160400      embedding_1[0]
[0]
__________________________________________________________________________________
_____________________
conv1d_2 (Conv1D)               (None, 97, 128)      49280       conv1d_1[0][0]
__________________________________________________________________________________
_____________________
flatten_1 (Flatten)             (None, 30000)        0           lstm_1[0][0]
__________________________________________________________________________________
_____________________
flatten_2 (Flatten)             (None, 12416)        0           conv1d_2[0][0]
__________________________________________________________________________________
_____________________
concatenate_1 (Concatenate)     (None, 42416)        0           flatten_1[0]
[0]
                                                                 flatten_2[0]
[0]
__________________________________________________________________________________
_____________________
```

```
dense_1 (Dense)                 (None, 300)          12725100    concatenate_1
[0][0]
__________________________________________________________________________________
_____________________
dropout_1 (Dropout)             (None, 300)          0           dense_1[0][0]
__________________________________________________________________________________
_____________________
dense_2 (Dense)                 (None, 256)          77056       dropout_1[0]
[0]
__________________________________________________________________________________
_____________________
dropout_2 (Dropout)             (None, 256)          0           dense_2[0][0]
__________________________________________________________________________________
_____________________
batch_normalization_1 (BatchNor (None, 256)          1024        dropout_2[0]
[0]
__________________________________________________________________________________
_____________________
dense_3 (Dense)                 (None, 128)          32896       batch_normaliz
ation_1[0][0]
__________________________________________________________________________________
_____________________
dense_4 (Dense)                 (None, 2)            258         dense_3[0][0]
==================================================================================
==================
Total params: 27,792,126
Trainable params: 27,791,614
Non-trainable params: 512
__________________________________________________________________________________
_____________________
None
```

In [53]:
```
model3.compile(loss='categorical_crossentropy', optimizer=keras.optimizers.adam(
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizer
s.py:793: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.tr
ain.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:3576: The name tf.log is deprecated. Please use tf.math.log
instead.

WARNING:tensorflow:From <ipython-input-34-70948b21cc0e>:8: py_func (from tensor
flow.python.ops.script_ops) is deprecated and will be removed in a future versi
on.
Instructions for updating:
tf.py_func is deprecated in TF V2. Instead, there are two
    options available in V2.
    - tf.py_function takes a python function which manipulates tf eager
    tensors instead of numpy arrays. It's easy to convert a tf eager tensor to
    an ndarray (just call tensor.numpy()) but having access to eager tensors
    means `tf.py_function`s can use accelerators such as GPUs as well as
    being differentiable using a gradient tape.
    - tf.numpy_function maintains the semantics of the deprecated tf.py_func
    (it is not differentiable, and manipulates numpy arrays). It drops the
    stateful argument making all functions stateful.
```

In [0]:
```
train_3_n = [pad_train,train_3]
test_3_n = [pad_test,test_3]
```

In [55]:
```
#model fitting
filepath="weights_3.best_copy.hdf5"
checkpoint = ModelCheckpoint(filepath,monitor='val_auc', verbose=1, save_best_on

earlystop = EarlyStopping(monitor = 'val_auroc',
                          mode="max",
                          min_delta = 0,
                          patience = 2,
                          verbose = 1,)
callbacks_list = [checkpoint,earlystop,TensorBoardColabCallback(tbc)]
model3.fit(train_3_n, y_train,epochs=5,verbose=1,batch_size=256,callbacks =callb
```

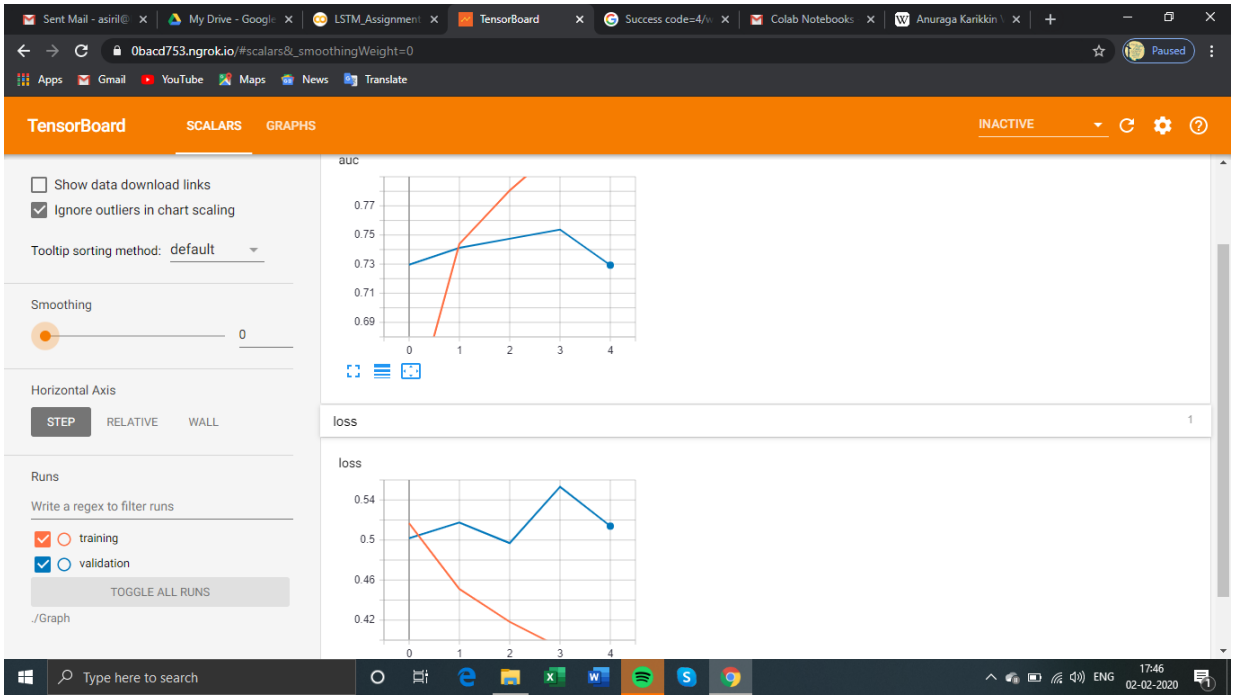
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/
python/ops/math_grad.py:1424: where (from tensorflow.python.ops.array_ops) is d
eprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:1033: The name tf.assign_add is deprecated. Please use tf.c
ompat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:1020: The name tf.assign is deprecated. Please use tf.compa
t.v1.assign instead.

Train on 76473 samples, validate on 32775 samples
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorboardcola
b/core.py:49: The name tf.summary.FileWriter is deprecated. Please use tf.compa
t.v1.summary.FileWriter instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbacks.
py:1122: The name tf.summary.merge_all is deprecated. Please use tf.compat.v1.s
ummary.merge_all instead.

Epoch 1/5
76473/76473 [==============================] - 191s 2ms/step - loss: 0.5167 - a
uc: 0.6181 - val_loss: 0.5017 - val_auc: 0.7296

Epoch 00001: val_auc improved from -inf to 0.72956, saving model to weights_3.b
est_copy.hdf5
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorboardcola
b/callbacks.py:51: The name tf.Summary is deprecated. Please use tf.compat.v1.S
ummary instead.

Epoch 2/5
76473/76473 [==============================] - 181s 2ms/step - loss: 0.4511 - a
uc: 0.7438 - val_loss: 0.5175 - val_auc: 0.7412

Epoch 00002: val_auc improved from 0.72956 to 0.74119, saving model to weights_
3.best_copy.hdf5
Epoch 3/5
76473/76473 [==============================] - 181s 2ms/step - loss: 0.4182 - a
uc: 0.7806 - val_loss: 0.4968 - val_auc: 0.7474

Epoch 00003: val_auc improved from 0.74119 to 0.74744, saving model to weights_
```

```
3.best_copy.hdf5
Epoch 4/5
76473/76473 [==============================] - 181s 2ms/step - loss: 0.3928 - a
uc: 0.8106 - val_loss: 0.5531 - val_auc: 0.7537

Epoch 00004: val_auc improved from 0.74744 to 0.75367, saving model to weights_
3.best_copy.hdf5
Epoch 5/5
76473/76473 [==============================] - 180s 2ms/step - loss: 0.3693 - a
uc: 0.8497 - val_loss: 0.5139 - val_auc: 0.7293

Epoch 00005: val_auc did not improve from 0.75367
```

Out[55]: <keras.callbacks.History at 0x7f1a3e235208>

In [56]:
```
from IPython.display import Image
Image(retina=True, filename='/content/drive/My Drive/image.png')
```

Out[56]:



In [57]:
```
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Model", "Train AUC", "Cross validation AUC"]

x.add_row(["Model 1", 0.7992,0.7520])
x.add_row(["Model 2",0.7562,0.6784] )
x.add_row(["Model 3", 0.8106, 0.7537])


print(x)
```

```
+---------+-----------+----------------------+
|  Model  | Train AUC | Cross validation AUC |
+---------+-----------+----------------------+
| Model 1 |   0.7992  |        0.752         |
| Model 2 |   0.7562  |        0.6784        |
| Model 3 |   0.8106  |        0.7537        |
+---------+-----------+----------------------+
```

Conclusion:

We got 2 models that have cross validation AUC above 0.75