Assignment : 14

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

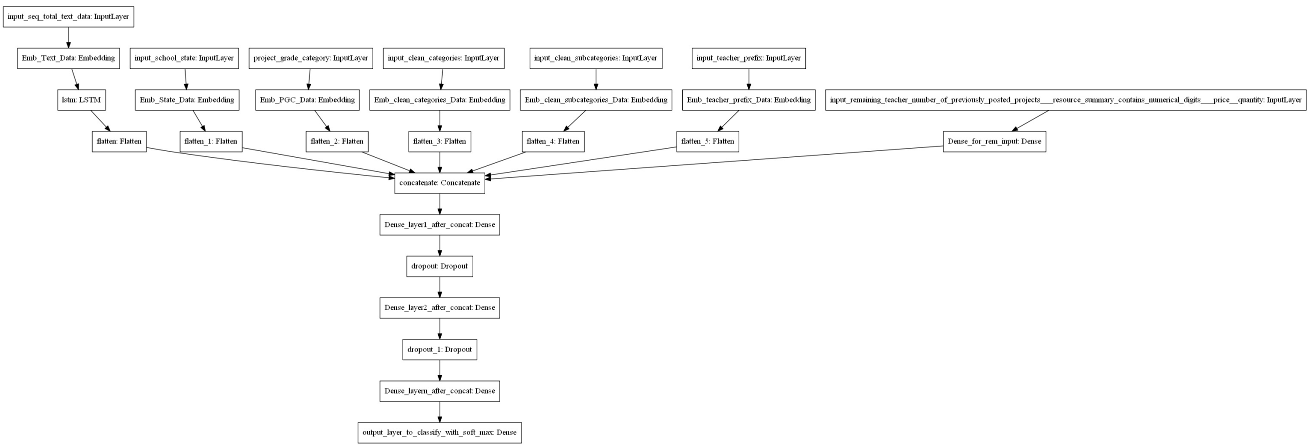
1. You can work with preprocessed_data.csv for the assignment. You can get the data from - Data folder
2. Load the data in your notebook.
3. After step 2 you have to train 3 types of models as discussed below.
4. For all the model use 'auc' as a metric. check this and this for using auc as a metric
5. 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.
7. For all the model's use TensorBoard and plot the Metric value and Loss with epoch. While submitting, take a screenshot of plots and include those images in a separate pad and write your observations about them.
8. Make sure that you are using GPU to train the given models.

In [ ]:
```
#you can use gdown modules to import dataset for the assignment
#for importing any file from drive to Colab you can write the syntax as !gdown --id file_id
#you can run the below cell to import the required preprocessed data.csv file and glove vector
```

In [ ]:
```
#!gdown --id 1GpATd_pM4mcnWWIs28-s1lgqdAg2Wdv-
#!gdown --id 1pGd5tLwA30M7wkbJKdXHaae9tYVDICJ_
```

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_contains_numerical_digits._price._qua** ---concatenate remaining columns and add a Dense layer after 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 [ ]:
```
# https://stats.stackexchange.com/questions/270546/how-does-keras-embedding-layer-work
input_layer = Input(shape=(n,))
embedding = Embedding(no_1, no_2, input_length=n)(input_layer)
flatten = Flatten()(embedding)
```

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/

2. Please go through this link 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 [1]:
```
import tensorflow
from tensorflow.keras.layers import Input,Dense,LSTM
import pandas as pd
```

In [ ]:

Model-1

In [1]:
```
# import all the libraries
#make sure that you import your libraries from tf.keras and not just keras
import tensorflow
from tensorflow.keras.layers import Input,Dense,LSTM
```

In [2]:
```
#read the csv file
import pandas as pd
p1 = '/content/drive/MyDrive/AAIC/Assignments/LSTM on Donors Choose/preprocessed_data_final.csv'
p2 = "C:/Users/darsh/Downloads/Srujan/Donars Choose Assignment/preprocessed_data_final.csv"
df = pd.read_csv(p2)
```

In [3]:
```
df.head()
```

Out[3]:

	teacher_number_of_previously_posted_projects	resource_summary_contains_numerical_digits	price	quantity	school_state	project_grade_c
0	0	0	154.60	23	in	grade:
1	7	0	299.00	1	fl	gra
2	1	0	516.85	22	az	gra
3	4	0	232.90	4	ky	grade:
4	1	0	67.98	4	tx	grade:

In [4]:
```
y = df['project_is_approved'].values
df.drop(['project_is_approved'],axis=1,inplace=True)
```

In [5]:
```
categorical_input = ['school_state','project_grade_category','clean_categories', 'clean_subcategories','teacher_
```

In [ ]:

In [6]:
```
# perform stratified train test split on the dataset
# perform stratified train test split on the dataset


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df, y,
                                                    stratify=y,
                                                    test_size=0.25,random_state=0)
```

In [7]:
```
y_train.shape
```

Out[7]: (81936,)

1.1 Text Vectorization

In [8]:
```
#since the data is already preprocessed, we can directly move to vectorization part
#first we will vectorize the text data
#for vectorization of text data in deep learning we use tokenizer, you can go through below references
# https://www.kdnuggets.com/2020/03/tensorflow-keras-tokenization-text-data-prep.html
#https://stackoverflow.com/questions/51956000/what-does-keras-tokenizer-method-exactly-do
# after text vectorization you should get train_padded_docs and test_padded_docs
```

In [9]:
```
text_input = ['essay','project_title','project_resource_summary',]
X_train['total_text_input'] = X_train['essay'] + ' ' + X_train['project_title'] + ' ' + X_train['project_resour
X_test['total_text_input'] = X_test['essay'] + ' ' + X_test['project_title'] + ' ' + X_test['project_resource_su
```

In [ ]:

In [10]:
```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

num_words = 1000
oov_token = '<UNK>'
pad_type = 'post'
trunc_type = 'post'

# Tokenize our training data
tokenizer = Tokenizer(num_words=num_words, oov_token=oov_token)
tokenizer.fit_on_texts(X_train['total_text_input'])

# Get our training data word index
word_index = tokenizer.word_index

# Encode training data sentences into sequences
train_sequences = tokenizer.texts_to_sequences(X_train['total_text_input'])
test_sequences = tokenizer.texts_to_sequences(X_test['total_text_input'])

# Get max training sequence length
maxlen = max([len(x) for x in train_sequences])

# Pad the training sequences
train_padded = pad_sequences(train_sequences, padding=pad_type, truncating=trunc_type, maxlen=maxlen)
test_padded = pad_sequences(test_sequences, padding=pad_type, truncating=trunc_type, maxlen=maxlen)
```

In [11]:
```
# Output the results of our work
#print("Word index:\n", word_index)
#print("\nTraining sequences:\n", train_sequences)
#print("\nPadded training sequences:\n", train_padded)
print("\nPadded training shape, Test Shape:", train_padded.shape,test_padded.shape)
print("Training sequences data type:", type(train_sequences),type(test_sequences))
print("Padded Training sequences data type:", type(train_padded),type(test_padded))
```

```
Padded training shape, Test Shape: (81936, 355) (27312, 355)
Training sequences data type: <class 'list'> <class 'list'>
Padded Training sequences data type: <class 'numpy.ndarray'> <class 'numpy.ndarray'>
```

In [ ]:

In [ ]:

In [12]:
```
#after getting the padded_docs you have to use predefined glove vectors to get 300 dim representation for each w
# we will be storing this data in form of an embedding matrix and will use it while defining our model
# Please go through following blog's 'Example of Using Pre-Trained GloVe Embedding' section to understand how to
# https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/
```

In [13]:
```
from numpy import asarray
from numpy import zeros
from keras.preprocessing.text import Tokenizer
from keras_preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Embedding
from tqdm import tqdm
```

In [14]:
```
from tqdm import tqdm

# load the whole embedding into memory
embeddings_index = dict()
p1 = '/content/drive/MyDrive/AAIC/Assignments/LSTM on Donors Choose/glove.6B.300d.txt'
```

```
p2 = "C:/Users/darsh/Downloads/Srujan/Donars Choose Assignment/glove.6B.300d.txt"
f = open(p2,encoding="utf8")
for line in tqdm(f):
    values = line.split()
    word = values[0]
    coefs = asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
print('\nLoaded %s word vectors.' % len(embeddings_index))
```

```
400000it [00:20, 19299.49it/s]
Loaded 400000 word vectors.
```

In [15]:
```
vocab_size = len(tokenizer.word_index) + 1
```

In [16]:
```
# create a weight matrix for words in training docs
embedding_matrix = zeros((vocab_size, 300))
for word, i in tokenizer.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

In [57]:
```
e = Embedding(vocab_size, 300, weights=[embedding_matrix], input_length=maxlen, trainable=False)
```

In [58]:
```
embedding_matrix.shape
```

Out[58]:
```
(56772, 300)
```

In [ ]:

In [ ]:

1.2 Categorical feature Vectorization

In [59]:
```
# for model 1 and model 2, we have to assign a unique number to each feature in a particular categorical column
# you can either use tokenizer,label encoder or ordinal encoder to perform the task
# label encoder gives an error for 'unseen values' (values present in test but not in train)
# handle unseen values with label encoder - https://stackoverflow.com/a/56876351
# ordinal encoder also gives error with unseen values but you can use modify handle_unknown parameter
# documentation of ordianl encoder https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Ordi
# after categorical feature vectorization you will have column_train_data and column_test_data.
```

In [17]:
```
from sklearn.preprocessing import OrdinalEncoder
import numpy as np
from scipy.sparse import coo_matrix, hstack
from scipy.sparse import csr_matrix


enc = OrdinalEncoder(handle_unknown='use_encoded_value',unknown_value=np.nan)
```

In [18]:
```
school_state_enc = (enc.fit_transform(np.array(X_train['school_state']).reshape(-1,1)))
teacher_prefix_enc = (enc.fit_transform(np.array(X_train['teacher_prefix']).reshape(-1,1)))
project_grade_category_enc = (enc.fit_transform(np.array(X_train['project_grade_category']).reshape(-1,1)))
clean_categories_enc = (enc.fit_transform(np.array(X_train['clean_categories']).reshape(-1,1)))
clean_subcategories_enc = (enc.fit_transform(np.array(X_train['clean_subcategories']).reshape(-1,1)))
```

In [19]:
```
school_state_enc_test = (enc.transform(np.array(X_test['school_state']).reshape(-1,1)))
teacher_prefix_enc_test = (enc.transform(np.array(X_test['teacher_prefix']).reshape(-1,1)))
project_grade_category_enc_test = (enc.transform(np.array(X_test['project_grade_category']).reshape(-1,1)))
clean_categories_enc_test = (enc.transform(np.array(X_test['clean_categories']).reshape(-1,1)))
clean_subcategories_enc_test = (enc.transform(np.array(X_test['clean_subcategories']).reshape(-1,1)))
```

In [ ]:

1.3 Numerical feature Vectorization

In [63]:
```
# you have to standardise the numerical columns
# stack both the numerical features
#after numerical feature vectorization you will have numerical_data_train and numerical_data_test
```

In [20]:
```
numerical_input = ['teacher_number_of_previously_posted_projects',
                   'resource_summary_contains_numerical_digits',
                   'price','quantity'
                  ]
```

In [21]:
```
from sklearn import preprocessing
```

```python
import numpy as np



scaler  = preprocessing.StandardScaler().fit(X_train[numerical_input])
std_data_train = pd.DataFrame(scaler.transform(X_train[numerical_input]),columns=numerical_input)
#std_data_train = ((std_data_train.astype(str).agg(','.join, axis=1)))
```

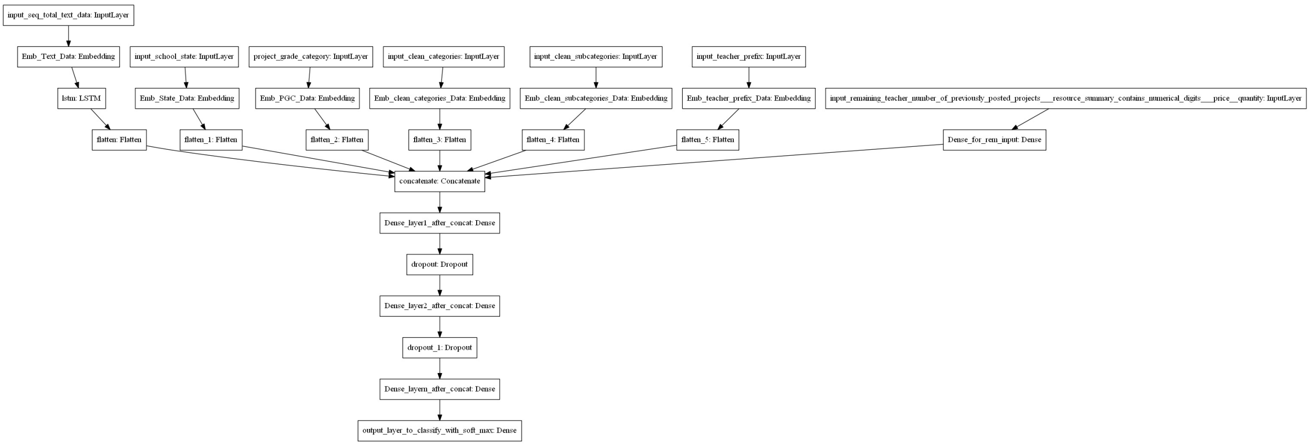
In [22]:
```python
std_data_test = pd.DataFrame(scaler.transform(X_test[numerical_input]),columns=numerical_input)
#std_data_test = ((std_data_test.astype(str).agg(', '.join, axis=1)))
```

In [67]:
```python
np.array(std_data_train).shape
```

Out[67]:
```
(81936, 4)
```

In [ ]:

1.4 Defining the model



In [68]:
```python
# as of now we have vectorized all our features now we will define our model.
# as it is clear from above image that the given model has multiple input layers and hence we have to use funct
# Please go through - https://keras.io/guides/functional_api/
# it is a good programming practise to define your complete model i.e all inputs , intermediate and output laye
# while defining your model make sure that you use variable names while defining any length,dimension or size.
#for ex.- you should write the code as 'input_text = Input(shape=(pad_length,))' and not as 'input_text = Input
# the embedding layer for text data should be non trainable
# the embedding layer for categorical data should be trainable
# https://stats.stackexchange.com/questions/270546/how-does-keras-embedding-layer-work
# https://towardsdatascience.com/deep-embeddings-for-categorical-variables-cat2vec-b05c8ab63ac0
#print model.summary() after you have defined the model
#plot the model using utils.plot_model module and make sure that it is similar to the above image
```

In [23]:
```python
import tensorflow
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Embedding
from tensorflow.keras import regularizers
from tensorflow.keras.regularizers import l2
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense, Input , Dropout
from tensorflow.keras.layers import concatenate
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.callbacks import TensorBoard
import tensorflow as tf
from sklearn.metrics import roc_auc_score
from tensorflow.keras.metrics import AUC
```

In [24]:
```python
elements_in_school_state = (len(set(pd.DataFrame(school_state_enc)[0])))
elements_in_teacher_prefix = (len(set(pd.DataFrame(teacher_prefix_enc)[0])))
elements_in_project_grade_category = (len(set(pd.DataFrame(project_grade_category_enc)[0])))
elements_in_clean_categories = (len(set(pd.DataFrame(clean_categories_enc)[0])))
elements_in_clean_subcategories = (len(set(pd.DataFrame(clean_subcategories_enc)[0])))
```

In [ ]:

In [25]:
```python
train_padded.shape
```

Out[25]:
```
(81936, 355)
```

In [26]:
```python
input_seq_total_text_data = Input(shape=(maxlen,),name='input_seq_total_text_data_')
```

```
emb_text_data = Embedding(input_dim=vocab_size,output_dim=300,
                          weights=[embedding_matrix], input_length=maxlen,trainable=False,
                          name='emb_text_data')(input_seq_total_text_data)
lstm = LSTM(units=128,activation='tanh',return_sequences=True)(emb_text_data)
flatten_text = Flatten()(lstm)

input_school_state = Input(shape=1,name='input_school_state')
input_school_state_emb = Embedding(input_dim=elements_in_school_state,
                                   output_dim=int(min(elements_in_school_state / 2, 50)),
                                   input_length=1,
                                  name='input_school_state_emb')(input_school_state)
flatten_school_state = Flatten()(input_school_state_emb)


input_grade_category = Input(shape=1,name='input_grade_category')
input_grade_category_emb = Embedding(input_dim=elements_in_project_grade_category,
                                   output_dim=int(min(elements_in_project_grade_category / 2, 50)),
                                   input_length=1,
                                  name='input_grade_category_emb')(input_grade_category)
flatten_grade_category = Flatten()(input_grade_category_emb)


input_clean_categories = Input(shape=1,name='input_clean_categories')
input_clean_categories_emb = Embedding(input_dim=elements_in_clean_categories,
                                      output_dim=int(min(elements_in_clean_categories / 2, 50)),
                                      input_length=1,
                                      name='input_clean_categories_emb')(input_clean_categories)
flatten_clean_categories = Flatten()(input_clean_categories_emb)



input_clean_sub_categories = Input(shape=1,name='input_clean_sub_categories')
input_clean_sub_categories_emb = Embedding(input_dim=elements_in_clean_subcategories,
                                      output_dim=int(min(elements_in_clean_subcategories / 2, 50)),
                                      input_length=1,
                                      name='input_clean_sub_categories_emb')(input_clean_sub_categories)
flatten_clean_sub_categories = Flatten()(input_clean_sub_categories_emb)


input_teacher_prefix = Input(shape=1,name='input_teacher_prefix')
input_teacher_prefix_emb = Embedding(input_dim=elements_in_teacher_prefix,
                                      output_dim=int(min(elements_in_teacher_prefix / 2, 50)),
                                      input_length=1,
                                      name='input_teacher_prefix_emb')(input_teacher_prefix)
flatten_teacher_prefix = Flatten()(input_teacher_prefix_emb)



input_remaining = Input(shape=4,name='input_remaining')
input_remaining_dense = Dense(units=256,activation='relu',
                               kernel_initializer='he_normal',kernel_regularizer=l2(0.1),
                              name='input_remaining_dense')(input_remaining)
flatten_remaining = Flatten()(input_remaining_dense)

concat_layer = concatenate([flatten_text,flatten_school_state,flatten_grade_category,
                            flatten_clean_categories,flatten_clean_sub_categories,
                            flatten_teacher_prefix,flatten_remaining],)

dense_layer1_after_concat = Dense(units=256,activation='relu',
                                  kernel_initializer='he_normal',kernel_regularizer=l2(0.001),
                                  name='dense_layer1_after_concat')(concat_layer)
drop1 = Dropout(0.5)(dense_layer1_after_concat)

dense_layer2_after_concat = Dense(units=256,activation='relu',
                                  kernel_initializer='he_normal',kernel_regularizer=l2(0.001),
                                  name='dense_layer2_after_concat')(drop1)
drop2 = Dropout(0.5)(dense_layer2_after_concat)

bn1 = BatchNormalization()(drop2)

dense_layer3_after_concat = Dense(units=256,activation='relu',
                                  kernel_initializer='he_normal',kernel_regularizer=l2(0.001),
                                  name='dense_layer3_after_concat')(bn1)
drop3 = Dropout(0.5)(dense_layer3_after_concat)
#bn2 =  BatchNormalization()(drop3)

output = Dense(units=2,activation='softmax')(drop3)
```

In [27]:

```
m1 = Model(inputs=[input_seq_total_text_data,
                   input_school_state,
                   input_grade_category,
                   input_clean_categories,
                   input_clean_sub_categories,
```

```
                  input_teacher_prefix,
                  input_remaining],
          outputs=[output])
```

In [28]:
```
m1.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to                     
==================================================================================================
input_seq_total_text_data_ (Inp [(None, 355)]        0                                            
__________________________________________________________________________________________________
emb_text_data (Embedding)       (None, 355, 300)     17031600    input_seq_total_text_data_[0][0] 
__________________________________________________________________________________________________
input_school_state (InputLayer) [(None, 1)]          0                                            
__________________________________________________________________________________________________
input_grade_category (InputLaye [(None, 1)]          0                                            
__________________________________________________________________________________________________
input_clean_categories (InputLa [(None, 1)]          0                                            
__________________________________________________________________________________________________
input_clean_sub_categories (Inp [(None, 1)]          0                                            
__________________________________________________________________________________________________
input_teacher_prefix (InputLaye [(None, 1)]          0                                            
__________________________________________________________________________________________________
input_remaining (InputLayer)    [(None, 4)]          0                                            
__________________________________________________________________________________________________
lstm (LSTM)                     (None, 355, 128)     219648      emb_text_data[0][0]              
__________________________________________________________________________________________________
input_school_state_emb (Embeddi (None, 1, 25)        1275        input_school_state[0][0]         
__________________________________________________________________________________________________
input_grade_category_emb (Embed (None, 1, 2)         8           input_grade_category[0][0]       
__________________________________________________________________________________________________
input_clean_categories_emb (Emb (None, 1, 25)        1275        input_clean_categories[0][0]     
__________________________________________________________________________________________________
input_clean_sub_categories_emb  (None, 1, 50)        19650       input_clean_sub_categories[0][0] 
__________________________________________________________________________________________________
input_teacher_prefix_emb (Embed (None, 1, 2)         10          input_teacher_prefix[0][0]       
__________________________________________________________________________________________________
input_remaining_dense (Dense)   (None, 256)          1280        input_remaining[0][0]            
__________________________________________________________________________________________________
flatten (Flatten)               (None, 45440)        0           lstm[0][0]                       
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 25)           0           input_school_state_emb[0][0]     
__________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 2)            0           input_grade_category_emb[0][0]   
__________________________________________________________________________________________________
flatten_3 (Flatten)             (None, 25)           0           input_clean_categories_emb[0][0] 
__________________________________________________________________________________________________
flatten_4 (Flatten)             (None, 50)           0           input_clean_sub_categories_emb[0]
__________________________________________________________________________________________________
flatten_5 (Flatten)             (None, 2)            0           input_teacher_prefix_emb[0][0]   
__________________________________________________________________________________________________
flatten_6 (Flatten)             (None, 256)          0           input_remaining_dense[0][0]      
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 45800)        0           flatten[0][0]                    
                                                                 flatten_1[0][0]                  
                                                                 flatten_2[0][0]                  
                                                                 flatten_3[0][0]                  
                                                                 flatten_4[0][0]                  
                                                                 flatten_5[0][0]                  
                                                                 flatten_6[0][0]                  
__________________________________________________________________________________________________
dense_layer1_after_concat (Dens (None, 256)          11725056    concatenate[0][0]                
__________________________________________________________________________________________________
dropout (Dropout)               (None, 256)          0           dense_layer1_after_concat[0][0]  
__________________________________________________________________________________________________
dense_layer2_after_concat (Dens (None, 256)          65792       dropout[0][0]                    
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 256)          0           dense_layer2_after_concat[0][0]  
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 256)          1024        dropout_1[0][0]                  
__________________________________________________________________________________________________
dense_layer3_after_concat (Dens (None, 256)          65792       batch_normalization[0][0]        
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 256)          0           dense_layer3_after_concat[0][0]  
__________________________________________________________________________________________________
dense (Dense)                   (None, 2)            514         dropout_2[0][0]                  
==================================================================================================
Total params: 29,132,924
Trainable params: 12,100,812
Non-trainable params: 17,032,112
__________________________________________________________________________________________________
```

In [29]:
```
from keras.utils import np_utils
```

```
test_data = [test_padded,school_state_enc_test,project_grade_category_enc_test,
            clean_categories_enc_test,clean_subcategories_enc_test,teacher_prefix_enc_test,np.array(std_data_te

train_data = [train_padded,school_state_enc,project_grade_category_enc,
              clean_categories_enc,clean_subcategories_enc,teacher_prefix_enc,np.array(std_data_train)]

y_train_enc =  tensorflow.keras.utils.to_categorical(y_train, 2)
y_test_enc =  tensorflow.keras.utils.to_categorical(y_test, 2)
```

In [ ]:

In [30]:

```
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, average='macro', sample_weight=None).astype('double')

def auroc(y_true, y_pred):
    return tensorflow.numpy_function(auc1, (y_true, y_pred), tensorflow.double)

callbacks = [
    tf.keras.callbacks.ModelCheckpoint('./LSTM_Model_1.h5', save_weights_only=False,save_best_only=True, \
                                       mode='max', monitor='val_auroc',verbose=1),
    tf.keras.callbacks.ReduceLROnPlateau(monitor='val_auroc', patience=2,mode='max',verbose=1),
]
```

In [ ]:

In [31]:

```
m1.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[auroc])
```

In [32]:

```
m1.fit(train_data,y_train_enc,
       validation_data=(test_data,y_test_enc),
       batch_size=128,
       epochs=50,
       callbacks=callbacks,
       verbose=1)
```

```
Epoch 1/50
641/641 [==============================] - 35s 45ms/step - loss: 28.4536 - auroc: 0.6084 - val_loss: 12.5123 -
val_auroc: 0.7056

Epoch 00001: val_auroc improved from -inf to 0.70555, saving model to .\LSTM_Model_1.h5
Epoch 2/50
641/641 [==============================] - 28s 44ms/step - loss: 6.3930 - auroc: 0.6689 - val_loss: 2.7273 - va
l_auroc: 0.7120

Epoch 00002: val_auroc improved from 0.70555 to 0.71196, saving model to .\LSTM_Model_1.h5
Epoch 3/50
641/641 [==============================] - 28s 44ms/step - loss: 1.4311 - auroc: 0.7118 - val_loss: 0.7703 - va
l_auroc: 0.7289

Epoch 00003: val_auroc improved from 0.71196 to 0.72890, saving model to .\LSTM_Model_1.h5
Epoch 4/50
641/641 [==============================] - 28s 43ms/step - loss: 0.5491 - auroc: 0.7316 - val_loss: 0.4499 - va
l_auroc: 0.7381

Epoch 00004: val_auroc improved from 0.72890 to 0.73811, saving model to .\LSTM_Model_1.h5
Epoch 5/50
641/641 [==============================] - 29s 45ms/step - loss: 0.4298 - auroc: 0.7423 - val_loss: 0.4308 - va
l_auroc: 0.7468

Epoch 00005: val_auroc improved from 0.73811 to 0.74678, saving model to .\LSTM_Model_1.h5
Epoch 6/50
641/641 [==============================] - 29s 45ms/step - loss: 0.4161 - auroc: 0.7533 - val_loss: 0.4157 - va
l_auroc: 0.7494

Epoch 00006: val_auroc improved from 0.74678 to 0.74943, saving model to .\LSTM_Model_1.h5
Epoch 7/50
641/641 [==============================] - 29s 45ms/step - loss: 0.4074 - auroc: 0.7633 - val_loss: 0.4160 - va
l_auroc: 0.7485

Epoch 00007: val_auroc did not improve from 0.74943
Epoch 8/50
641/641 [==============================] - 29s 45ms/step - loss: 0.4055 - auroc: 0.7716 - val_loss: 0.4240 - va
l_auroc: 0.7446

Epoch 00008: val_auroc did not improve from 0.74943

Epoch 00008: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
Epoch 9/50
641/641 [==============================] - 29s 45ms/step - loss: 0.3679 - auroc: 0.8031 - val_loss: 0.3936 - va
```

```
l_auroc: 0.7461

Epoch 00009: val_auroc did not improve from 0.74943
Epoch 10/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3511 - auroc: 0.8113 - val_loss: 0.3916 - va
l_auroc: 0.7447

Epoch 00010: val_auroc did not improve from 0.74943

Epoch 00010: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
Epoch 11/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3407 - auroc: 0.8228 - val_loss: 0.3933 - va
l_auroc: 0.7441

Epoch 00011: val_auroc did not improve from 0.74943
Epoch 12/50
641/641 [==============================] - 29s 45ms/step - loss: 0.3368 - auroc: 0.8246 - val_loss: 0.3934 - va
l_auroc: 0.7435

Epoch 00012: val_auroc did not improve from 0.74943

Epoch 00012: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
Epoch 13/50
641/641 [==============================] - 29s 45ms/step - loss: 0.3334 - auroc: 0.8264 - val_loss: 0.3934 - va
l_auroc: 0.7434

Epoch 00013: val_auroc did not improve from 0.74943
Epoch 14/50
641/641 [==============================] - 29s 45ms/step - loss: 0.3329 - auroc: 0.8282 - val_loss: 0.3937 - va
l_auroc: 0.7433

Epoch 00014: val_auroc did not improve from 0.74943

Epoch 00014: ReduceLROnPlateau reducing learning rate to 1.0000001111620805e-07.
Epoch 15/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3330 - auroc: 0.8277 - val_loss: 0.3939 - va
l_auroc: 0.7432

Epoch 00015: val_auroc did not improve from 0.74943
Epoch 16/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3334 - auroc: 0.8274 - val_loss: 0.3935 - va
l_auroc: 0.7432

Epoch 00016: val_auroc did not improve from 0.74943

Epoch 00016: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-08.
Epoch 17/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3337 - auroc: 0.8268 - val_loss: 0.3934 - va
l_auroc: 0.7432

Epoch 00017: val_auroc did not improve from 0.74943
Epoch 18/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3330 - auroc: 0.8277 - val_loss: 0.3937 - va
l_auroc: 0.7432

Epoch 00018: val_auroc did not improve from 0.74943

Epoch 00018: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-09.
Epoch 19/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3329 - auroc: 0.8279 - val_loss: 0.3938 - va
l_auroc: 0.7432

Epoch 00019: val_auroc did not improve from 0.74943
Epoch 20/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3335 - auroc: 0.8271 - val_loss: 0.3940 - va
l_auroc: 0.7432

Epoch 00020: val_auroc did not improve from 0.74943

Epoch 00020: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-10.
Epoch 21/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3333 - auroc: 0.8271 - val_loss: 0.3935 - va
l_auroc: 0.7432

Epoch 00021: val_auroc did not improve from 0.74943
Epoch 22/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3323 - auroc: 0.8282 - val_loss: 0.3936 - va
l_auroc: 0.7432

Epoch 00022: val_auroc did not improve from 0.74943

Epoch 00022: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-11.
Epoch 23/50
```

```
641/641 [==============================] - 30s 46ms/step - loss: 0.3334 - auroc: 0.8271 - val_loss: 0.3935 - va
l_auroc: 0.7432

Epoch 00023: val_auroc did not improve from 0.74943
Epoch 24/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3331 - auroc: 0.8272 - val_loss: 0.3936 - va
l_auroc: 0.7432

Epoch 00024: val_auroc did not improve from 0.74943

Epoch 00024: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-12.
Epoch 25/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3330 - auroc: 0.8278 - val_loss: 0.3939 - va
l_auroc: 0.7432

Epoch 00025: val_auroc did not improve from 0.74943
Epoch 26/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3334 - auroc: 0.8280 - val_loss: 0.3937 - va
l_auroc: 0.7432

Epoch 00026: val_auroc did not improve from 0.74943

Epoch 00026: ReduceLROnPlateau reducing learning rate to 1.0000001044244145e-13.
Epoch 27/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3328 - auroc: 0.8284 - val_loss: 0.3937 - va
l_auroc: 0.7432

Epoch 00027: val_auroc did not improve from 0.74943
Epoch 28/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3332 - auroc: 0.8283 - val_loss: 0.3937 - va
l_auroc: 0.7432

Epoch 00028: val_auroc did not improve from 0.74943

Epoch 00028: ReduceLROnPlateau reducing learning rate to 1.0000001179769417e-14.
Epoch 29/50
641/641 [==============================] - 30s 46ms/step - loss: 0.3330 - auroc: 0.8281 - val_loss: 0.3933 - va
l_auroc: 0.7432

Epoch 00029: val_auroc did not improve from 0.74943
Epoch 30/50
641/641 [==============================] - 30s 46ms/step - loss: 0.3331 - auroc: 0.8278 - val_loss: 0.3936 - va
l_auroc: 0.7432

Epoch 00030: val_auroc did not improve from 0.74943

Epoch 00030: ReduceLROnPlateau reducing learning rate to 1.0000001518582595e-15.
Epoch 31/50
641/641 [==============================] - 30s 46ms/step - loss: 0.3328 - auroc: 0.8280 - val_loss: 0.3940 - va
l_auroc: 0.7432

Epoch 00031: val_auroc did not improve from 0.74943
Epoch 32/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3324 - auroc: 0.8300 - val_loss: 0.3935 - va
l_auroc: 0.7432

Epoch 00032: val_auroc did not improve from 0.74943

Epoch 00032: ReduceLROnPlateau reducing learning rate to 1.0000001095066122e-16.
Epoch 33/50
641/641 [==============================] - 30s 46ms/step - loss: 0.3331 - auroc: 0.8273 - val_loss: 0.3935 - va
l_auroc: 0.7432

Epoch 00033: val_auroc did not improve from 0.74943
Epoch 34/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3332 - auroc: 0.8275 - val_loss: 0.3937 - va
l_auroc: 0.7432

Epoch 00034: val_auroc did not improve from 0.74943

Epoch 00034: ReduceLROnPlateau reducing learning rate to 1.0000000830368326e-17.
Epoch 35/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3328 - auroc: 0.8275 - val_loss: 0.3940 - va
l_auroc: 0.7432

Epoch 00035: val_auroc did not improve from 0.74943
Epoch 36/50
641/641 [==============================] - 30s 46ms/step - loss: 0.3334 - auroc: 0.8272 - val_loss: 0.3937 - va
l_auroc: 0.7432

Epoch 00036: val_auroc did not improve from 0.74943

Epoch 00036: ReduceLROnPlateau reducing learning rate to 1.0000000664932204e-18.
```

```
Epoch 37/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3336 - auroc: 0.8271 - val_loss: 0.3936 - va
l_auroc: 0.7432

Epoch 00037: val_auroc did not improve from 0.74943
Epoch 38/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3324 - auroc: 0.8287 - val_loss: 0.3935 - va
l_auroc: 0.7432

Epoch 00038: val_auroc did not improve from 0.74943

Epoch 00038: ReduceLROnPlateau reducing learning rate to 1.000000045813705e-19.
Epoch 39/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3327 - auroc: 0.8282 - val_loss: 0.3937 - va
l_auroc: 0.7432

Epoch 00039: val_auroc did not improve from 0.74943
Epoch 40/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3330 - auroc: 0.8277 - val_loss: 0.3938 - va
l_auroc: 0.7432

Epoch 00040: val_auroc did not improve from 0.74943

Epoch 00040: ReduceLROnPlateau reducing learning rate to 1.000000032889008e-20.
Epoch 41/50
641/641 [==============================] - 30s 46ms/step - loss: 0.3325 - auroc: 0.8284 - val_loss: 0.3936 - va
l_auroc: 0.7432

Epoch 00041: val_auroc did not improve from 0.74943
Epoch 42/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3335 - auroc: 0.8276 - val_loss: 0.3940 - va
l_auroc: 0.7432

Epoch 00042: val_auroc did not improve from 0.74943

Epoch 00042: ReduceLROnPlateau reducing learning rate to 1.0000000490448793e-21.
Epoch 43/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3329 - auroc: 0.8288 - val_loss: 0.3940 - va
l_auroc: 0.7433

Epoch 00043: val_auroc did not improve from 0.74943
Epoch 44/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3329 - auroc: 0.8285 - val_loss: 0.3935 - va
l_auroc: 0.7432

Epoch 00044: val_auroc did not improve from 0.74943

Epoch 00044: ReduceLROnPlateau reducing learning rate to 1.0000000692397185e-22.
Epoch 45/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3320 - auroc: 0.8294 - val_loss: 0.3938 - va
l_auroc: 0.7432

Epoch 00045: val_auroc did not improve from 0.74943
Epoch 46/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3328 - auroc: 0.8271 - val_loss: 0.3932 - va
l_auroc: 0.7432

Epoch 00046: val_auroc did not improve from 0.74943

Epoch 00046: ReduceLROnPlateau reducing learning rate to 1.0000000944832675e-23.
Epoch 47/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3329 - auroc: 0.8283 - val_loss: 0.3935 - va
l_auroc: 0.7433

Epoch 00047: val_auroc did not improve from 0.74943
Epoch 48/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3333 - auroc: 0.8265 - val_loss: 0.3940 - va
l_auroc: 0.7432

Epoch 00048: val_auroc did not improve from 0.74943

Epoch 00048: ReduceLROnPlateau reducing learning rate to 1.0000000787060494e-24.
Epoch 49/50
641/641 [==============================] - 29s 46ms/step - loss: 0.3333 - auroc: 0.8268 - val_loss: 0.3937 - va
l_auroc: 0.7432

Epoch 00049: val_auroc did not improve from 0.74943
Epoch 50/50
641/641 [==============================] - 30s 46ms/step - loss: 0.3342 - auroc: 0.8255 - val_loss: 0.3937 - va
l_auroc: 0.7432

Epoch 00050: val_auroc did not improve from 0.74943
```

```
Epoch 00050: ReduceLROnPlateau reducing learning rate to 1.0000001181490946e-25.
```

Out[32]:
```
<keras.callbacks.History at 0x1c763fa7700>
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: