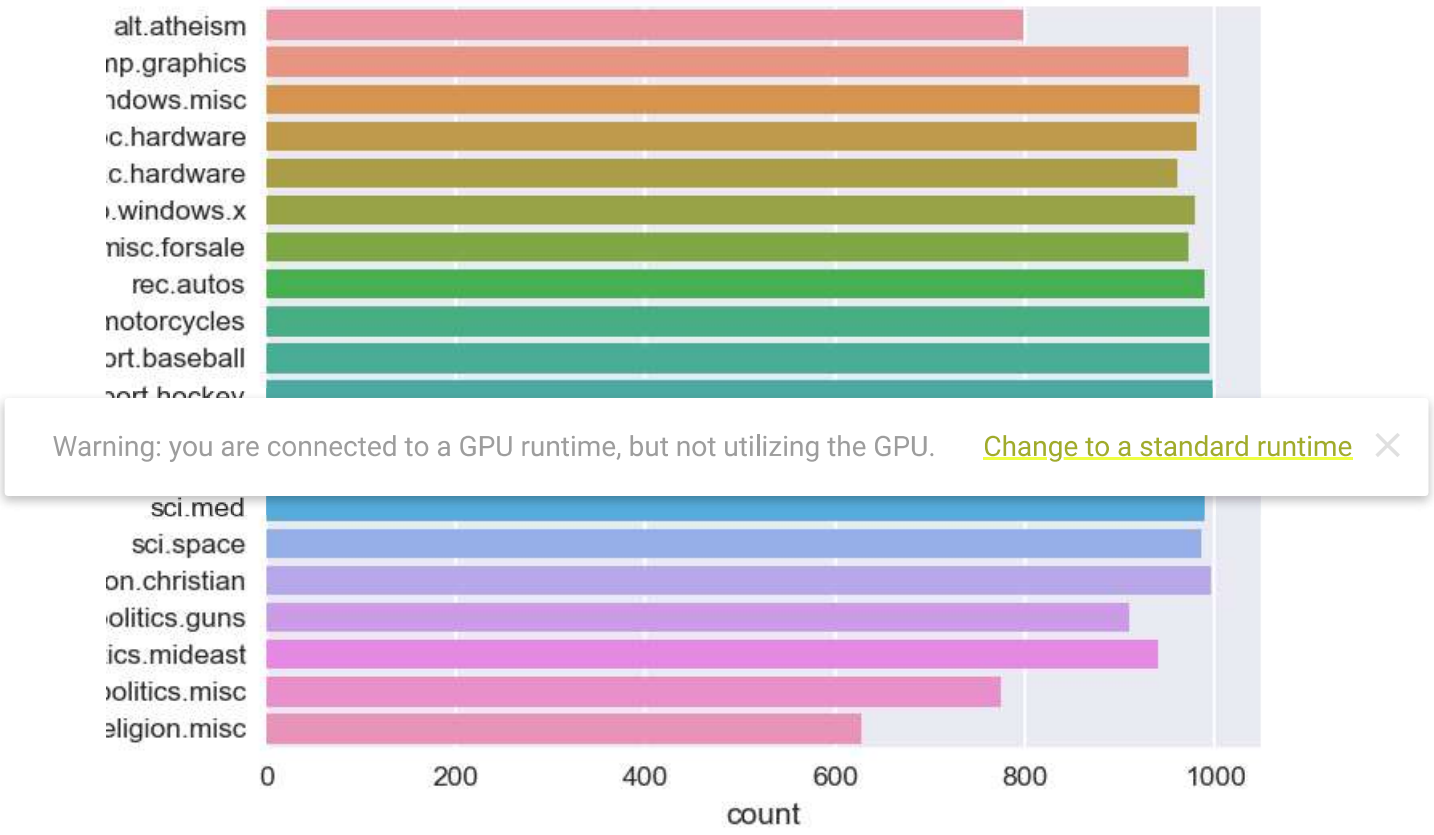
Text Classification:

Data

1. we have total of 20 types of documents(Text files) and total 18828 documents(text files)
2. You can download data from this link, in that you will get documents.rar folder.
If you unzip that, you will get total of 18828 documnets. document name is defined as'Class
so from document name, you can extract the label for that document.
4. Now our problem is to classify all the documents into any one of the class.
5. Below we provided count plot of all the labels in our data.

count plot of all the class labels.



Assignment:

sample document

```
Subject: A word of advice
From: jcopelan@nyx.cs.du.edu (The One and Only)

In article < 65882@mimsy.umd.edu > mangoe@cs.umd.edu (Charley Wingate) writes:
>
>I've said 100 times that there is no "alternative" that should think you
>might have caught on by now.  And there is no "alternative", but the point
>is, "rationality" isn't an alternative either.  The problems of metaphysical
>and religious knowledge are unsolvable-- or I should say, humans cannot
>solve them.

How does that saying go: Those who say it can't be done shouldn't interrupt
those who are doing it.

Jim
--
Have you washed your brain today?
```

Preprocessing:

useful links: http://www.pyregex.com/

1. Find all emails in the document and then get the text after the "@". and then split thos

Eg:[test@dm1.d.com, test2@dm2.dm3.com]-->[dm1.d.com, dm3.dm4.com]-->[dm1,d,com,dm2,dm3,com]
append all those into one list/array. (This will give length of 18828 sentences i.e one li
Some sample output was shown below.

> In the above sample document there are emails [jcopelan@nyx.cs.du.edu, 65882@mimsy.umd.ed

preprocessing:
[jcopelan@nyx.cs.du.edu, 65882@mimsy.umd.edu, mangoe@cs.umd.edu] ==> [nyx cs du edu mimsy u
[nyx edu mimsy umd edu umd edu]

2. Replace all the emails by space in the original text.

```
# we have collected all emails and preprocessed them, this is sample output
preprocessed_email
```

```
array(['juliet caltech edu',
       'coding bchs edu newsgate sps mot austlcm sps mot austlcm sps mot com  dna bchs e
       'batman bmd trw', ..., 'rbdc wsnc org dscomsa desy zeus  desy',
       'rbdc wsnc org morrow stanford edu pangea Stanford EDU',
       'rbdc wsnc org apollo apollo'], dtype=object)
```

```
len(preprocessed_email)
```

```
18828
```

3. Get subject of the text i.e. get the total lines where "Subject:" occur and remove the word which are before the ":" remove the newlines, tabs, punctuations, any special char
Eg: if we have sentance like "Subject: Re: Gospel Dating @ \r\r\n" --> You have to get "Gos
Save all this data into another list/array.

4. After you store it in the list, Replace those sentances in original text by space.

5. Delete all the sentances where sentence starts with **"Write to:"** or **"From:"**.
> In the above sample document check the 2nd line, we should remove that

6. Delete all the tags like "< anyword >"
> In the above sample document check the 4nd line, we should remove that "< 65882@mimsy.umd

Eg: "AAIC-The course that gets you HIRED(AAIC - Der Kurs, der Sie anstellt)" --> "AAIC-The

> In the above sample document check the 4nd line, we should remove that "(Charley Wingate)

8. Remove all the newlines('\n'), tabs('\t'), "-", "\".

9. Remove all the words which ends with ":".
Eg: "Anyword:"
> In the above sample document check the 4nd line, we should remove that "writes:"

10. Decontractions, replace words like below to full words.
please check the donors choose preprocessing for this
Eg: can't -> can not, 's -> is, i've -> i have, i'm -> i am, you're -> you are, i'll --> i

There is no order to do point 6 to 10. but you have to get final output correctly

11. Do chunking on the text you have after above preprocessing.
Text chunking, also referred to as shallow parsing, is a task that
follows Part-Of-Speech Tagging and that adds more structure to the sentence.
So it combines the some phrases, named entities into single word.
So after that combine all those phrases/named entities by separating "_".
And remove the phrases/named entities if that is a "Person".
You can use **nltk.ne_chunk** to get these.
Below we have given one example. please go through it.

useful links:
https://www.nltk.org/book/ch07.html
https://stackoverflow.com/a/31837224/4084039
http://www.nltk.org/howto/tree.html
https://stackoverflow.com/a/44294377/4084039

```
#i am living in the New York
print("i am living in the New York -->", list(chunks))
print(" ")
print("-"*50)
print(" ")
#My name is Srikanth Varma
print("My name is Srikanth Varma -->", list(chunks1))
```

Warning: you are connected to a GPU runtime, but not utilizing the GPU. Change to a standard runtime

```
--------------------------------------------------

My name is Srikanth Varma --> [('My', 'PRP$'), ('name', 'NN'), ('is', 'VBZ'), Tree('PERS
```

We did chunking for above two lines and then We got one list where each word is mapped to a
POS(parts of speech) and also if you see "New York" and "Srikanth Varma",
they got combined and represented as a tree and "New York" was referred as "GPE" and "Srika
so now you have to Combine the "New York" with "_" i.e "New_York"
and remove the "Srikanth Varma" from the above sentence because it is a person.

13. Replace all the digits with space i.e delete all the digits.
> In the above sample document, the 6th line have digit 100, so we have to remove that.

14. After doing above points, we observed there might be few word's like
"_word_" (i.e starting and ending with the _), "_word" (i.e starting with the _), "word_" (i.e ending with the _) remove the _ from these type of words.

15. We also observed some words like **"OneLetter_word"- eg: d_berlin, "TwoLetters_word" - eg: dr_berlin** , in these words we remove the "OneLetter_" (d_berlin ==> "TwoLetters_" (de_berlin ==> berlin). i.e remove the words which are length less than or equal to 2 after spliiting those words by "_".

16. Convert all the words into lower case and lowe case
and remove the words which are greater than or equal to 15 or less than or equal to 2.

17. replace all the words except "A-Za-z_" with space.

18. Now You got Preprocessed Text, email, subject. create a dataframe with those.
Below are the columns of the df.

```
data = pd.read_csv('')
```

```
data.columns
```

```
Index(['text', 'class', 'preprocessed_text', 'preprocessed_subject',
```



```
data.iloc[400]
```

```
text                    From: arc1@ukc.ac.uk (Tony Curtis)\r\r\r\nSubj...
class                                                        alt.atheism
preprocessed_text       said re is article if followed the quoting rig...
preprocessed_subject                                christian morality is
preprocessed_emails                                   ukc mac macalstr edu
Name: 567, dtype: object
```

To get above mentioned data frame --> Try to Write Total Preprocessing steps in One Function Named Preprocess as below.

```
def preprocess(Input_Text):
    """Do all the Preprocessing as shown above and
    return a tuple contain preprocess_email,preprocess_subject,preprocess_text for that Text_
    return (list_of_preproessed_emails,subject,text)
```

Code checking:

After Writing preprocess function. call that functoin with the input text of 'alt.atheism_49960' doc and print the output of the preprocess function
This will help us to evaluate faster, based on the output we can suggest you if there are any changes.

After writing Preprocess function, call the function for each of the document(18828 docs) and then create a dataframe as mentioned above.

Training The models to Classify:

1. Combine "preprocessed_text", "preprocessed_subject", "preprocessed_emails" into one colu

2. Now Split the data into Train and test. use 25% for test also do a stratify split.

3. Analyze your text data and pad the sequnce if required.
Sequnce length is not restricted, you can use anything of your choice.
you need to give the reasoning

5. code the model's (Model-1, Model-2) as discussed below
and try to optimize that models.

6. For every model use predefined Glove vectors.
Don't train any word vectors while Training the model.

7. Use "categorical_crossentropy" as Loss.

8. Use **Accuracy and Micro Avgeraged F1 score** as your as Key metrics to evaluate your model.

9. Use Tensorboard to plot the loss and Metrics based on the epoches.

10. Please save your best model weights in to **'best_model_L.h5' (L = 1 or 2).**

11. You are free to choose any Activation function, learning rate, optimizer.
But have to use the same architecture which we are giving below.

12. You can add some layer to our architecture but you **deletion** of layer is not acceptable.

13. Try to use **Early Stopping** technique or any of the callback techniques that you did in t

14. For Every model save your model to image (Plot the model) with shapes
and inlcude those images in the notebook markdown cell,
upload those imgages to Classroom. You can use "plot_model"
please refer this if you don't know how to plot the model with shapes.

```
import pandas as pd
import os
import pickle
import tensorflow as tf
from tensorflow.keras.callbacks import EarlyStopping
from numpy import array
from numpy import asarray
from numpy import zeros
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential,Model
from tensorflow.keras.layers import Dense,MaxPooling1D
```



```
import numpy as np
from tensorflow.keras.layers import Conv1D
from tensorflow.keras.callbacks import TensorBoard
import tensorflow as tf
import itertools
import re
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split


import nltk
nltk.download('punkt')
nltk.download('maxent_ne_chunker')
nltk.download('words')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Unzipping corpora/words.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
True
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!unrar x "/content/drive/MyDrive/documents.rar" "/content/drive/MyDrive/"
```

```
Streaming output truncated to the last 5000 lines.
Extracting  /content/drive/MyDrive/documents/sci.space_60893.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60894.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60895.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60896.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60897.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60898.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60899.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60900.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60901.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60902.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60903.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60904.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60905.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60906.txt       OK
```



```
Extracting  /content/drive/MyDrive/documents/sci.space_60910.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60911.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60912.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60913.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60914.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60915.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60916.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60917.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60918.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60919.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60920.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60921.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60922.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60923.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60924.txt       OK
Extracting  /content/drive/MyDrive/documents/sci.space_60925.txt       OK
```

```
Extracting  /content/drive/MyDrive/documents/sci.space_60926.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60927.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60928.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60929.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60930.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60931.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60932.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60933.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60934.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60935.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60936.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60937.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60938.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60939.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60940.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60941.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60942.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60943.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60944.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60945.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60946.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60947.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60948.txt      OK
Extracting  /content/drive/MyDrive/documents/sci.space_60949.txt      OK
```

```
#Fetching the data
comp_data = []
doc = []
documents=os.listdir('/content/drive/MyDrive/documents')
for d in documents:
    with open('/content/drive/MyDrive/documents/' + str(d),'r',encoding='utf8',errors='replac
        file_read = f.read()
        doc.append(file_read)

comp_data.append(doc)
```

Warning: you are connected to a GPU runtime, but not utilizing the GPU. Change to a standard runtime

```
mail=[]
txt = comp_data[0]
for d in txt:
    em = re.findall(r'[\w\.-]+@[\w\.-]+', d)
    email = em
    for i in email:
        d=d.replace(i," ")
    mail.append(d)
    emails.append(email)
lst1=[]
for i in emails:
    lst2=[]
    for email in i:
        for j in range(len(email)):
            if(email[j]=='@'):
```

```
                g = email[j+1:].split('.')
                text= g
                lst2.append(text)
        l1.append(l2)
    lst3=[]
    for i in lst1:
        lst2=[]
        for j in i:
            for word in j:
                if((len(word)>2) & (word!='com')):
                    lst2.append(word)
                    flag = True
        lst3.append(l2)
    preprocessed_email=[]
    for val in lst3:
        k=""
        k=" ".join(val)
        preprocessed_email.append(k)


    #Removing subject
    subject=[]
    wo_sub_data=[]
    for i in mail:
        c = re.findall('Subject:.*',i)
        sub= c
        for j in sub:
            str_j = str(j)
            i=i.replace(str_j," ")
        wo_sub_data.append(i)
        subject.append(sub)


    subject_pp=[]
    #Removing pucntuation https://stackoverflow.com/questions/5843518/remove-all-special-characte
```



```
        for word in s:
            word=word.replace("Subject: ",'')
            word=re.sub(r'[^a-zA-Z0-9 ]',r'',word)
            word=word.replace("Re: ",'')
        subject_pp.append(word)


    data_wo_write_from=[]
    for txt in wo_sub_data:
        f=re.findall("Write to:.*",txt)
        g=re.findall("From:.*",txt)
        for i in f:
            txt=txt.replace(str(i),'')
        for j in g:
            txt=txt.replace(str(j),'')
```

```
    data_wo_write_from.append(txt)


data_wo_colon=[]
for i in data_wo_write_from:
    i_lst = i.split()
    for j in i_lst:
        if(j[-1]==':'):
            str_j = str(j)
            i=i.replace(str_j,'')
    data_wo_colon.append(i)



#https://stackoverflow.com/questions/54396405/how-can-i-preprocess-nlp-text-lowercase-remove-
data_wo_contr=[]
r = 8
for phrase in data_wo_colon:
    phrase = re.sub(r"\'m", " am", phrase)
    phrase = re.sub(r"won\'t", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    data_wo_contr.append(phrase)



data_wo_tag=[]
for i in data_wo_contr:
    s=re.findall('<.*?>',i)
```



```
    s=re.findall('\(.*\)',i)
    for j in s:
        i=i.replace(str(j),'')
    s=re.findall('\n',i)
    for j in s:
        i=i.replace(str(j),'')
    s=re.findall('\t',i)
    for j in s:
        i=i.replace(str(j),'')
    i = re.sub(r'[>|?|\|.|-]',r'',i)
    i = re.sub(r'[=|!|<]',r'',i)
    data_wo_tag.append(i)
```

```
#file:///C:/Users/LENOVO/Desktop/ch/Text_Classification_Assignment_colab.pdf.pdf
from nltk import word_tokenize, pos_tag, ne_chunk
from nltk import Tree
def contgp_chunks(text, label):

    prev = None
    continuous_chunk = []
    word=[]
    continuos_word=[]
    current_chunk = []
    i=0
    chunked = ne_chunk(pos_tag(word_tokenize(text)))
    for subtree in chunked:
        if type(subtree) == Tree and subtree.label() == label:
            i+=1
            current_chunk.append("_".join([token for token, pos in subtree.leaves()]))
            word.append(" ".join([token for token, pos in subtree.leaves()]))
        if current_chunk:
            named_entity = " ".join(current_chunk)
            named_word=" ".join(word)
            named_word=" ".join(word)
            flag = False
            if named_entity not in continuous_chunk:
                continuous_chunk.append(named_entity)
                current_chunk = []
            if named_word not in continuos_word:
                continuos_word.append(named_word)
                word=[]
        else:
            continue
    return continuous_chunk,continuos_word

def contin_chunks(text, label):
    flag = True
```



```
    prev = None
    chunked = ne_chunk(pos_tag(word_tokenize(text)))
    for subtree in chunked:

        if type(subtree) == Tree and subtree.label() == label:
            current_chunk.append(" ".join([token for token, pos in subtree.leaves()]))
        if current_chunk:
            named_entity = " ".join(current_chunk)
            if named_entity not in chunks_cont:
                chunks_cont.append(named_entity)
                current_chunk = []
        else:
            continue
    return chunks_cont
```

```
from nltk import word_tokenize, pos_tag, ne_chunk
from nltk import Tree
def contin_chunks(text, label):
    chunked = ne_chunk(pos_tag(word_tokenize(text)))
    prev = None
    continuous_chunk = []
    current_chunk = []
    for subtree in chunked:
        if type(subtree) == Tree and subtree.label() == label:
            current_chunk.append(" ".join([token for token, pos in subtree.leaves()]))
        if current_chunk:
            named_entity = " ".join(current_chunk)
            if named_entity not in continuous_chunk:
                continuous_chunk.append(named_entity)
                current_chunk = []
        else:
            continue
    return continuous_chunk

def contingp_chunks(text, label):
    chunked = ne_chunk(pos_tag(word_tokenize(text)))
    prev = None
    continuous_chunk = []
    word=[]
    continuos_word=[]
    current_chunk = []
    i=0
    for subtree in chunked:
        if type(subtree) == Tree and subtree.label() == label:
            i+=1
            current_chunk.append("_".join([token for token, pos in subtree.leaves()]))
            word.append(" ".join([token for token, pos in subtree.leaves()]))
        if current_chunk:
```



```
            named_word=   .join(word)
            if named_entity not in continuous_chunk:
                continuous_chunk.append(named_entity)
                current_chunk = []
            if named_word not in continuos_word:
                continuos_word.append(named_word)
                word=[]
        else:
            continue
    return continuous_chunk,continuos_word



from tqdm import tqdm
chunked_data=[]
```

```
for i in tqdm(data_wo_tag):
    p=contin_chunks(i, "PERSON")
    for j in p:
        i=i.replace(str(j),'')
    g,w=contingp_chunks(i, "GPE")
    for k in range(len(g)):
        i=i.replace(w[k],g[k])
    chunked_data.append(i)
```

```
100%|██████████| 9028/9028 [15:48<00:00,  9.52it/s]
```

```
data_wo_unscore=[]
for i in chunked_data:
    txt_lst = i.split()
    for word in txt_lst:
        if word.startswith('_'):
            i=i.replace(word[0],"")
        if word.endswith("_"):
            i=i.replace(word[-1],"")
    data_wo_unscore.append(i)
```

```
data_wo_dig=[]
for i in data_wo_unscore:
    digits=re.findall('[0-9]+',i)
    for d in digits:
        rep_val =i.replace(d,"")
        i=rep_val
    data_wo_dig.append(i)
```



```
    txt_lst = i.split()
    j= ' '.join([w for w in txt_lst if len(w)>2])
    j_lst  = j.split()
    i_new=' '.join([w for w in j_lst if (len(w)<=15)])
    data_lc.append(i_new.lower())
```

```
preprocessed_text=[]
for txt in data_lc:
    val_to_put = re.sub(r'[^A-Za-z_ ]','',i)
    txt=val_to_put
    preprocessed_text.append(txt)
```

```
#Fetching the labels
txt_labels = []
for d in documents:
    each_label = d.split('_')
    val = each_label[0]
    txt_labels.append(val)
```

```
print("Preprocessed Email")
print(preprocessed_email[documents.index("alt.atheism_49960.txt")])
print("-"*100)
print("Preprocessed Subject")
print(subject_pp[documents.index("alt.atheism_49960.txt")])
print("-"*100)
print("Preprocessed Text")
print(preprocessed_text[documents.index("alt.atheism_49960.txt")])
```

```
Preprocessed Email
mantis netcom mantis
----------------------------------------------------------------------------------------------------
Preprocessed Subject
AltAtheism FAQ Atheist Resources
----------------------------------------------------------------------------------------------------
Preprocessed Text
 In article     Looking at historical evidence such perfect utopian islamic statesdid nc
```

```
doc_txt = comp_data[0]
preprocessed_data=pd.DataFrame({"text":doc_txt,"class":txt_labels,"preprocessed_text":preproc
                                "preprocessed_subject":subject_pp,"preprocessed_emails":prepr
```

Warning: you are connected to a GPU runtime, but not utilizing the GPU. Change to a standard runtime

	text	class	preprocessed_text	preprocessed_subject	prep
	From:				

```
preprocessed_data.to_csv('preprocessed_data (1)', encoding = 'utf-8-sig')
```

Modeling

From:

Task 1

```
data = pd.read_csv('/content/drive/MyDrive/preprocessed_data (1).csv')
```

```
data.head()
```

	text	class	preprocessed_text	preprocessed_subject	preproce
0	From: mathew <mathew@mantis.co.uk>\n Subject: ...	alt.atheism	atheism resources resources december atheist r...	alt atheism faq atheist sources	m
1	From: mathew <mathew@mantis.co.uk>\n Subject: ...	alt.atheism	atheism introduction introduction april egin p...	alt atheism faq introduction to atheism	
	From:		article well has quite		

Warning: you are connected to a GPU runtime, but not utilizing the GPU. Change to a standard runtime

```
X_train,X_test,y_train,y_test= train_test_split(X,y,random_state=42,test_size=0.20,stratify=y
```

```
print(X_train.shape)
print(X_test.shape)
```

```
(15062, 4)
(3766, 4)
```

Model-1: Using 1D convolutions with word embeddings

Encoding of the Text --> For a given text data create a Matrix with Embedding layer as sho
In the example we have considered d = 5, but in this assignment we will get d = dimension o
i.e if we have maximum of 350 words in a sentence and embedding of 300 dim word vector,
we result in 350*300 dimensional matrix for each sentance as output after embedding layer

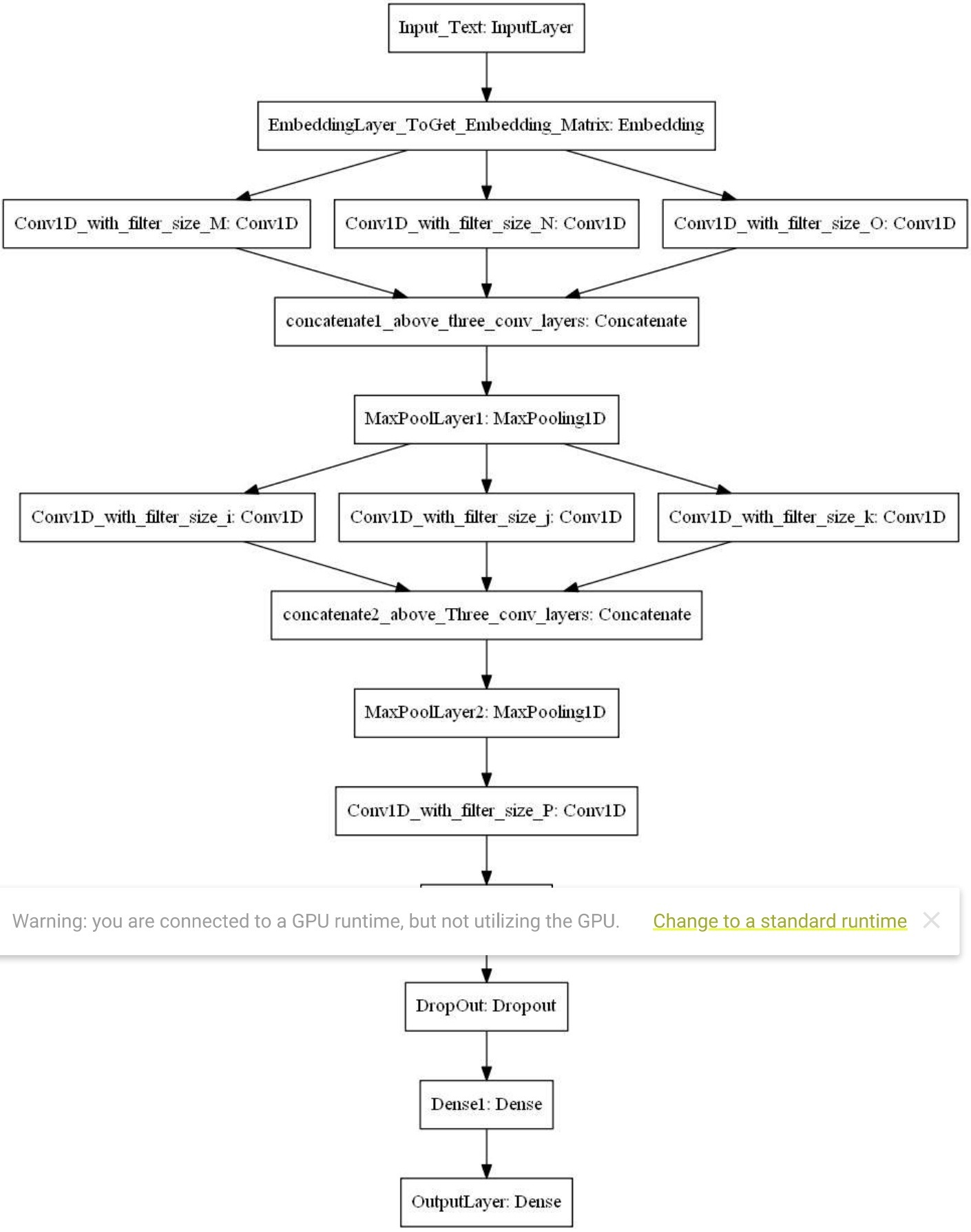
Ref: https://i.imgur.com/kiVQuk1.png

Reference:
https://stackoverflow.com/a/43399308/4084039
https://missinglink.ai/guides/keras/keras-conv1d-working-1d-convolutional-neural-networks-k

How EMBEDDING LAYER WORKS

Warning: you are connected to a GPU runtime, but not utilizing the GPU. Change to a standard runtime

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/



Warning: you are connected to a GPU runtime, but not utilizing the GPU. Change to a standard runtime

ref: 'https://i.imgur.com/fv1GvFJ.png'

1. all are Conv1D layers with any number of filter and filter sizes, there is no restrictio

2. use concatenate layer is to concatenate all the filters/channels.

3. You can use any pool size and stride for maxpooling layer.

4. Don't use more than 16 filters in one Conv layer becuase it will increase the no of para
(Only recommendation if you have less computing power)

5. You can use any number of layers after the Flatten Layer.

```
from sklearn.preprocessing import LabelEncoder
le_tr = LabelEncoder()
le_te = LabelEncoder()

y_tr = le_tr.fit_transform(y_train)
y_te = le_te.fit_transform(y_test)
lab= y_tr
y_train_label = tf.keras.utils.to_categorical(y_tr, 20)
y_test_label = tf.keras.utils.to_categorical(y_te, 20)



#tokenizing
tok = Tokenizer(filters='!"#$%&()*+,-/:;<=>?@[\\]^`{|}~\t\n', oov_token="<OOV>")
```



```
seq_tr= tok.texts_to_sequences(txt_tr)
txt_te = X_test['preprocessed_text'].astype(str)
seq_te= tok.texts_to_sequences(txt_te)


#padding
pad_tr = pad_sequences(seq_tr, maxlen=1050, padding='post')
pad_te = pad_sequences(seq_te, maxlen=1050, padding='post')



#taken from reference notebook
embeddings_index = {}
f = open('/content/drive/MyDrive/glove.6B.100d.txt')
for line in f:
```

```
        l= line.split()
        val = l
        coefs = np.asarray(val[1:], dtype='float32')
        word = val[0]
        embeddings_index[word] = coefs
    f.close()
    print('Found %s word vectors.' % len(embeddings_index))
```

```
Found 400000 word vectors.
```

```
#taken from reference notebook
embedding_matrix = np.zeros((len(tok.word_index) + 1, 100))
wrd_ind_items = tok.word_index.items()
for word, i in wrd_ind_items:
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
# words not found in embedding index will be all-zeros.
        embedding_matrix[i] = embedding_vector
```

```
vocab_size=len(tok.word_index) + 1
vocab_size
```

```
70153
```

Callbacks

```
from tensorflow.keras.callbacks import Callback
from sklearn.metrics import f1_score
class F1score(tf.keras.callbacks.Callback):
```



```
        self.x_test=x_test
        self.history={'auc': [],'F1_score':[]}

    def on_epoch_end(self, epoch, logs={}):
        y_pred=self.model.predict(self.x_test)
        mat=[]
        f1_sc = 0
        for val in y_pred:
            y_val=np.argmax(val)
            arr=np.zeros(20,dtype=int)
            arr[y_val]=1
            mat.append(arr)
            y_pred_a = np.array(mat)
```

```
        f1_sc=f1_score(self.y_test,y_pred_a,average='micro')
```

```
#  Save model at every epoch if your validation accuracy is improved from previous epoch.
filepath = 'model_save/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5'
from tensorflow.keras.callbacks import ModelCheckpoint
checkpoint = ModelCheckpoint(filepath=filepath, save_best_only = True, save_weights_only=Fals



# Use tensorboard for every model and analyse your gradients. (you need to upload the screens
from tensorflow.keras.callbacks import TensorBoard
import datetime
log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = TensorBoard(log_dir = log_dir, histogram_freq=1)


#rate scheduler
from tensorflow.keras.callbacks import ReduceLROnPlateau
lr_reduce = ReduceLROnPlateau(monitor = 'val_accuracy', factor=0.1, min_lr=0.001)

def scheduler(epoch, lr):
  if( (epoch+1)%3 == 0):
    lr = 0.5*lr
    return lr
  else:
    return lr
from tensorflow.keras.callbacks import LearningRateScheduler
sch_lr_reduce = LearningRateScheduler(scheduler, verbose=1)


tf.keras.backend.clear_session()
input = Input(shape=(1050,),dtype='int32')
emb = Embedding(vocab_size, 100, weights=[embedding_matrix], input_length=1050, trainable=Tru

c1=Conv1D(3,11,kernel_initializer='glorot_uniform',activation='relu')(emb)
```

Warning: you are connected to a GPU runtime, but not utilizing the GPU. Change to a standard runtime

```
mp_1=MaxPooling1D(pool_size=2, padding='same')(concat_1)

cc1=Conv1D(3,7,kernel_initializer='glorot_uniform',activation='relu')(mp_1)
cc2=Conv1D(3,5,kernel_initializer='glorot_uniform',activation='relu')(mp_1)
cc3=Conv1D(3,3,kernel_initializer='glorot_uniform',activation='relu')(mp_1)
concat_2 =concatenate([cc1, cc2, cc3],axis=1)

mp_2=MaxPooling1D(pool_size=2, padding='same')(concat_2)

conv_layer3 = Conv1D(3,12, activation='relu')(mp_2)
flatten=Flatten()(conv_layer3)
dp =Dropout(0.5)(flatten)
dense_layer=Dense(100, activation='relu')(dp)
output_layer=Dense(20, activation='softmax')(dense_layer)
```

```
opt = tf.keras.optimizers.Adam()
f1score=F1score(x_test=pad_te, y_test=y_test_label)
callbacks_lst = [f1_score, checkpoint,tensorboard_callback]

model = Model(inputs=input,outputs=output_layer)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
print(model.summary())
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 1050)]       0           []

 embedding (Embedding)          (None, 1050, 100)    7015300     ['input_1[0][0]']

 conv1d (Conv1D)                (None, 1040, 3)      3303        ['embedding[0][0]']

 conv1d_1 (Conv1D)              (None, 1042, 3)      2703        ['embedding[0][0]']

 conv1d_2 (Conv1D)              (None, 1044, 3)      2103        ['embedding[0][0]']

 concatenate (Concatenate)      (None, 3126, 3)      0           ['conv1d[0][0]',
                                                                  'conv1d_1[0][0]',
                                                                  'conv1d_2[0][0]']

 max_pooling1d (MaxPooling1D)   (None, 1563, 3)      0           ['concatenate[0][0]']

 conv1d_3 (Conv1D)              (None, 1557, 3)      66          ['max_pooling1d[0][0]']

 conv1d_4 (Conv1D)              (None, 1559, 3)      48          ['max_pooling1d[0][0]']

 conv1d_5 (Conv1D)              (None, 1561, 3)      30          ['max_pooling1d[0][0]']
                                                                  'conv1d_4[0][0]',
                                                                  'conv1d_5[0][0]']

 max_pooling1d_1 (MaxPooling1D)  (None, 2339, 3)     0           ['concatenate_1[0][0]']

 conv1d_6 (Conv1D)              (None, 2328, 3)      111         ['max_pooling1d_1[0][0]

 flatten (Flatten)              (None, 6984)         0           ['conv1d_6[0][0]']

 dropout (Dropout)              (None, 6984)         0           ['flatten[0][0]']

 dense (Dense)                  (None, 100)          698500      ['dropout[0][0]']

 dense_1 (Dense)                (None, 20)           2020        ['dense[0][0]']

==================================================================================================
```

```
Total params: 7,724,184
Trainable params: 7,724,184
Non-trainable params: 0
```

```
None
```

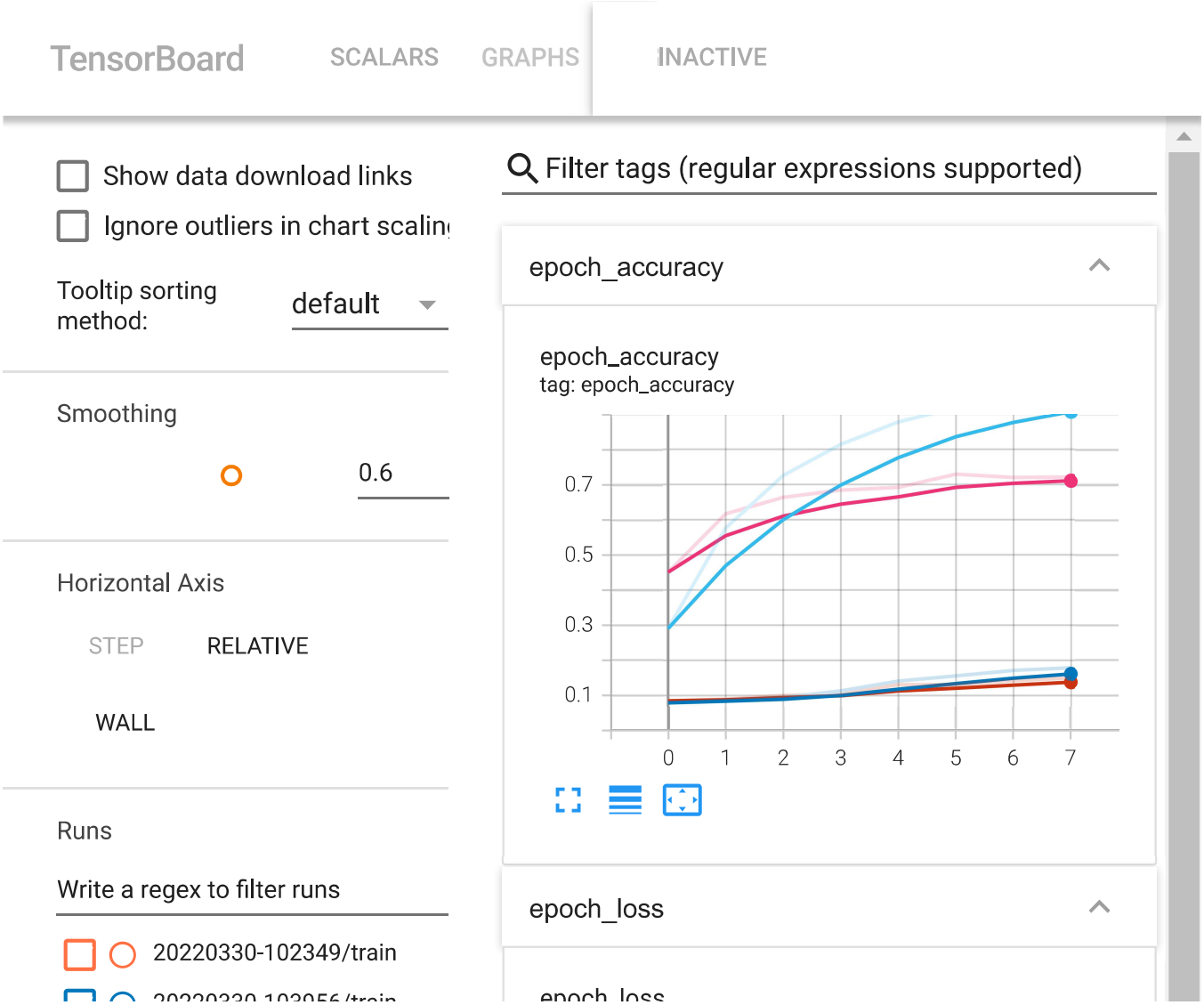
```
callbacks_lst = [f1score, tensorboard_callback]
history=model.fit(pad_tr, y_train_label,validation_data=(pad_te, y_test_label),verbose=1,epoc
```

```
Epoch 1/8
471/471 [==============================] - ETA: 0s - loss: 2.1219 - accuracy: 0.2904The
471/471 [==============================] - 353s 747ms/step - loss: 2.1219 - accuracy: 0
Epoch 2/8
471/471 [==============================] - ETA: 0s - loss: 1.2207 - accuracy: 0.5763The
471/471 [==============================] - 369s 783ms/step - loss: 1.2207 - accuracy: 0
Epoch 3/8
471/471 [==============================] - ETA: 0s - loss: 0.8036 - accuracy: 0.7256The
471/471 [==============================] - 368s 782ms/step - loss: 0.8036 - accuracy: 0
Epoch 4/8
471/471 [==============================] - ETA: 0s - loss: 0.5462 - accuracy: 0.8140The
471/471 [==============================] - 362s 768ms/step - loss: 0.5462 - accuracy: 0
Epoch 5/8
471/471 [==============================] - ETA: 0s - loss: 0.3694 - accuracy: 0.8778The
471/471 [==============================] - 361s 766ms/step - loss: 0.3694 - accuracy: 0
Epoch 6/8
471/471 [==============================] - ETA: 0s - loss: 0.2540 - accuracy: 0.9182The
471/471 [==============================] - 373s 793ms/step - loss: 0.2540 - accuracy: 0
Epoch 7/8
471/471 [==============================] - ETA: 0s - loss: 0.2018 - accuracy: 0.9344The
471/471 [==============================] - 381s 810ms/step - loss: 0.2018 - accuracy: 0
Epoch 8/8
471/471 [==============================] - ETA: 0s - loss: 0.1552 - accuracy: 0.9511The
471/471 [==============================] - 385s 818ms/step - loss: 0.1552 - accuracy: 0
```

Warning: you are connected to a GPU runtime, but not utilizing the GPU. Change to a standard runtime

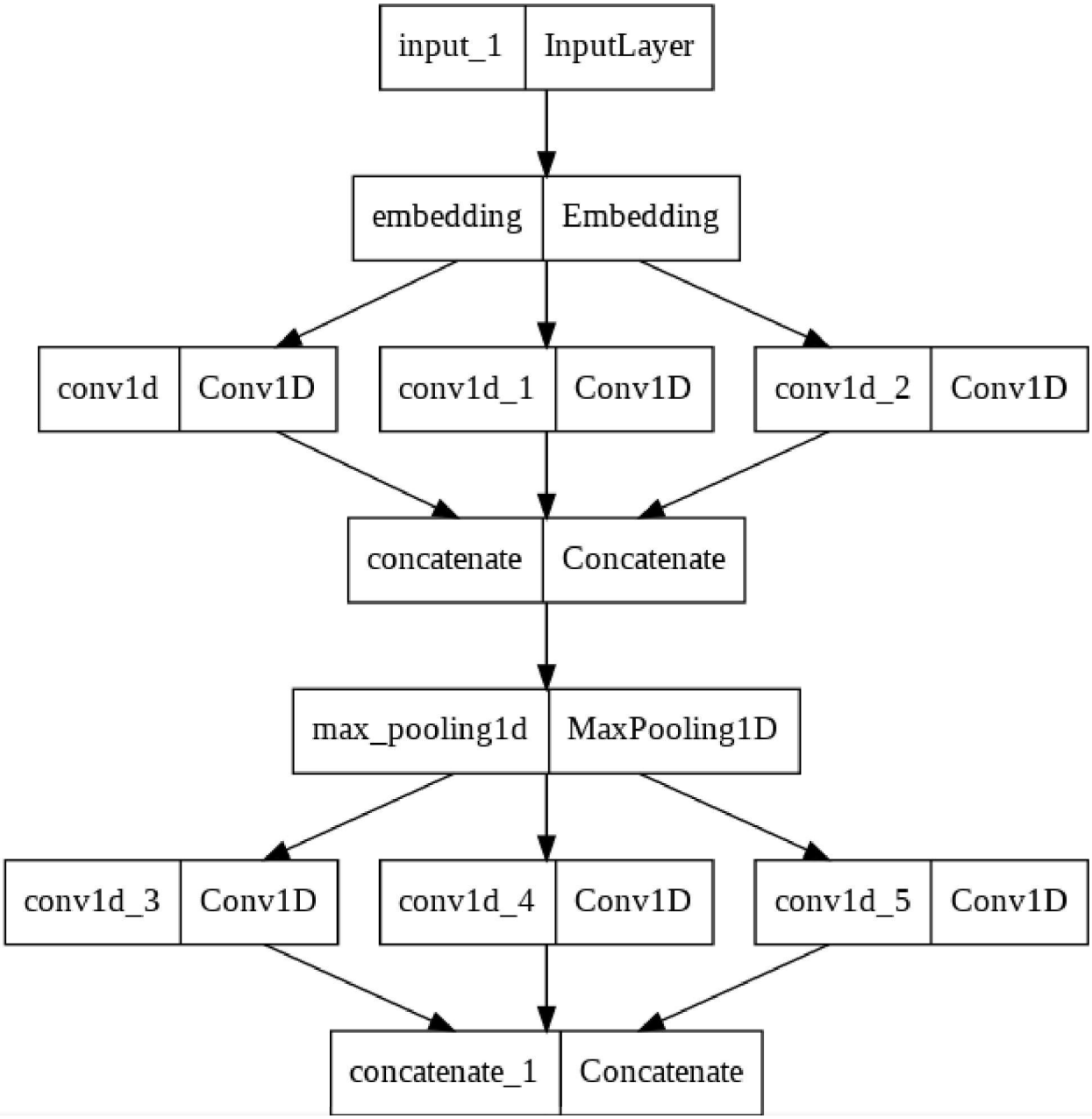
```
%tensorboard --logdir logs/fit
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
Reusing TensorBoard on port 6006 (pid 817), started 1:29:17 ago. (Use '!kill 817' to ki
```

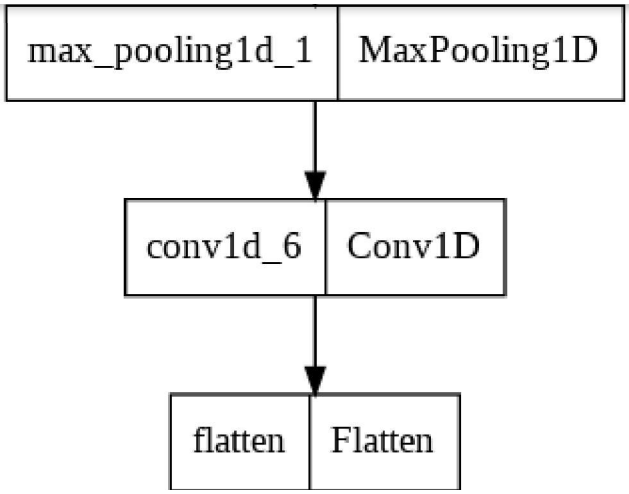



```
tf.keras.utils.plot_model(model,to_file='model.png',show_shapes=False,show_layer_names=True,r
```

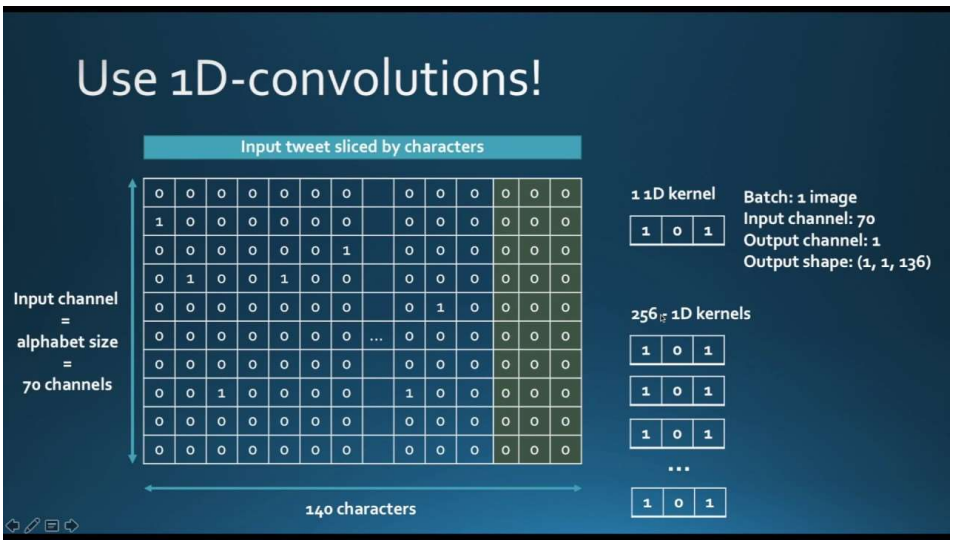
Warning: you are connected to a GPU runtime, but not utilizing the GPU. Change to a standard runtime



Warning: you are connected to a GPU runtime, but not utilizing the GPU. Change to a standard runtime



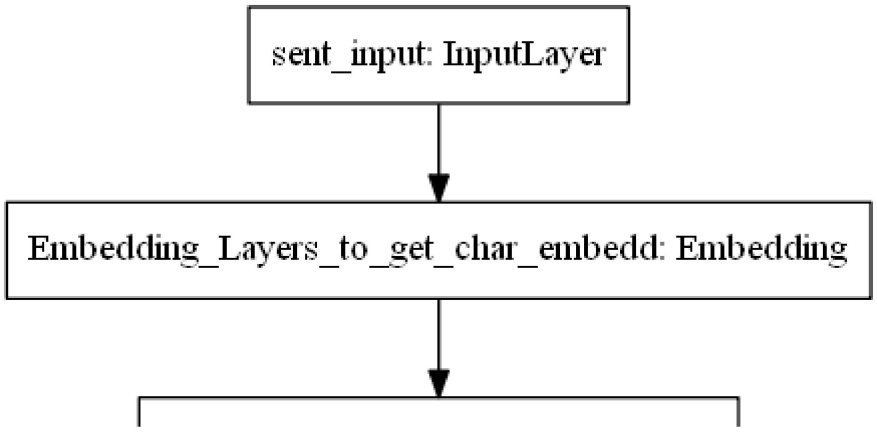
Model-2 : Using 1D convolutions with character embedding

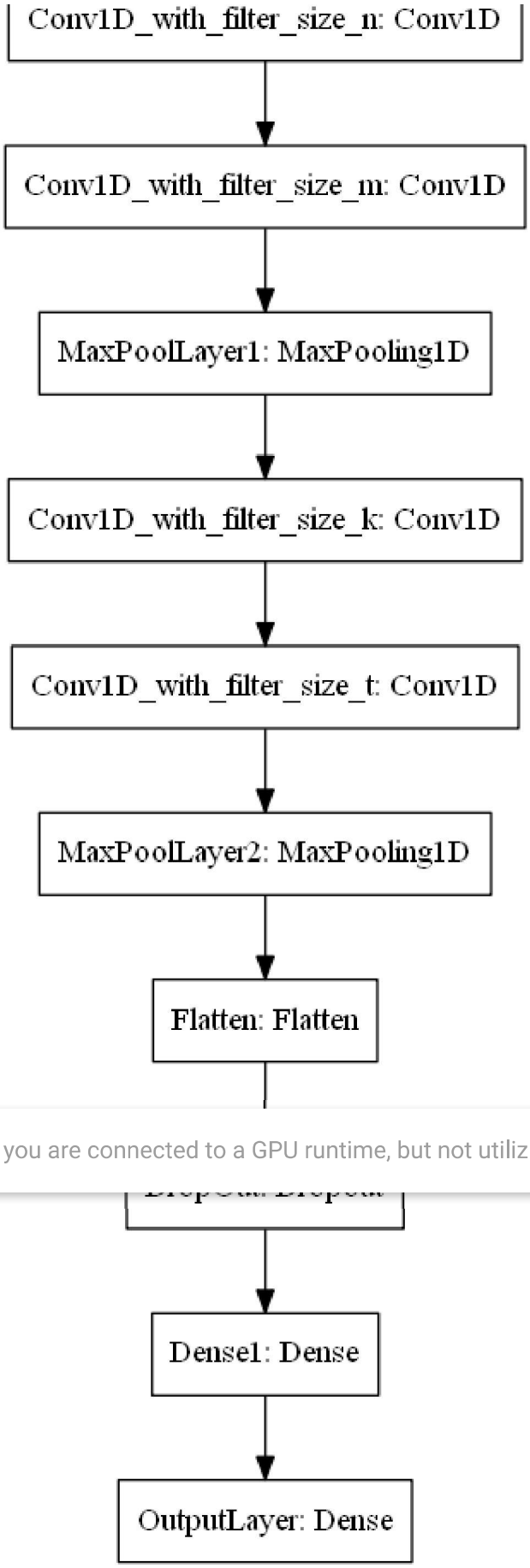


Here are the some papers based on Char-CNN

1. Xiang Zhang, Junbo Zhao, Yann LeCun. Character-level Convolutional Networks for Text
2. Yoon Kim, Yacine Jernite, David Sontag, Alexander M. Rush. Character-Aware Neural Lar
3. Shaojie Bai, J. Zico Kolter, Vladlen Koltun. An Empirical Evaluation of Generic Convc
4. Use the pratrained char embeddings https://github.com/minimaxir/char-embeddings/blob/

Warning: you are connected to a GPU runtime, but not utilizing the GPU. Change to a standard runtime





Task 2

```
!rm -rf ./logs/


tr_data = X_train['preprocessed_text'].astype(str)
te_data = X_test['preprocessed_text'].astype(str)
tok_2 = Tokenizer(char_level=True,num_words=None, oov_token='UNK')

tok_2.fit_on_texts(tr_data)
seq_char_tr= tok_2.texts_to_sequences(tr_data)
seq_char_te= tok_2.texts_to_sequences(te_data)

cd = {}
alphabets = "abcdefghijklmnopqrstuvwxyz"
for j, chr in enumerate(alphabets):
    cd[chr] = j + 1
tok_2.word_index = cd.copy()
tok_2.word_index[tok_2.oov_token] = max(cd.values()) + 1


max_length =1014
pad_tr_2 = pad_sequences(seq_char_tr, maxlen=max_length, padding='post')
pad_te_2 = pad_sequences(seq_char_te, maxlen=max_length, padding='post')



vocab_size

     28


emb_wt = []
mat = np.zeros(vocab_size)
vocab_size=len(tok_2.word_index)+1
```



```
for char, i in tok_2.word_index.items():
    onehot = np.zeros(vocab_size)
    onehot[i - 1] = 1
    emb_wt.append(onehot)
emb_wt = np.array(emb_wt)
embedding_weights = emb_wt

     vocab size: 28


inputs=Input(shape=(1014,),dtype='int32')
emb = Embedding(vocab_size,28, weights=[embedding_weights], input_length=1014, trainable=True
c1=Conv1D(3,9,kernel_initializer='glorot_uniform',activation='relu')(emb)
c2=Conv1D(3,7,kernel_initializer='glorot_uniform',activation='relu')(c1)
```

```
maxpool_1=MaxPooling1D(pool_size=2, strides=2, padding='same',)(c2)

cc1=Conv1D(3,7,activation='relu',kernel_initializer='glorot_uniform')(maxpool_1)
cc2=Conv1D(3,5,activation='relu',kernel_initializer='glorot_uniform')(cc1)

maxpool_2=MaxPooling1D(pool_size=2, strides=2, padding='same')(cc2)

flatten=Flatten()(maxpool_2)
dropout =Dropout(0.3)(flatten)
dense_layer=Dense(32, activation='relu')(dropout)
output_layer=Dense(20, activation='softmax')(dense_layer)



opt = tf.keras.optimizers.Adam()
f1score_2=F1score(x_test=pad_te_2, y_test=y_test_label)
callbacks_lst = [f1score_2,tensorboard_callback]

model_2 = Model(inputs=inputs,outputs=output_layer)
model_2.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
print(model_2.summary())
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 1014)]            0

 embedding (Embedding)       (None, 1014, 28)          784

 conv1d (Conv1D)             (None, 1006, 3)           759

 conv1d_1 (Conv1D)           (None, 1000, 3)           66

 max_pooling1d (MaxPooling1D  (None, 500, 3)           0
```



```
 conv1d_3 (Conv1D)           (None, 490, 3)            48

 max_pooling1d_1 (MaxPooling  (None, 245, 3)           0
 1D)

 flatten (Flatten)           (None, 735)               0

 dropout (Dropout)           (None, 735)               0

 dense (Dense)               (None, 32)                23552

 dense_1 (Dense)             (None, 20)                660

=================================================================
Total params: 25,935
```

```
Trainable params: 25,935
Non-trainable params: 0

_________________________________________________________________
None
```

```
history2=model_2.fit(pad_tr_2, y_train_label, epochs=8,validation_data=(pad_te_2, y_test_labe
```

```
Epoch 1/8
471/471 [==============================] - ETA: 0s - loss: 2.9471 - accuracy: 0.0793The
471/471 [==============================] - 59s 124ms/step - loss: 2.9471 - accuracy: 0.0
Epoch 2/8
471/471 [==============================] - ETA: 0s - loss: 2.9297 - accuracy: 0.0868The
471/471 [==============================] - 48s 101ms/step - loss: 2.9297 - accuracy: 0.0
Epoch 3/8
471/471 [==============================] - ETA: 0s - loss: 2.9001 - accuracy: 0.0946The
471/471 [==============================] - 47s 101ms/step - loss: 2.9001 - accuracy: 0.0
Epoch 4/8
471/471 [==============================] - ETA: 0s - loss: 2.7932 - accuracy: 0.1133The
471/471 [==============================] - 48s 102ms/step - loss: 2.7932 - accuracy: 0.1
Epoch 5/8
471/471 [==============================] - ETA: 0s - loss: 2.6693 - accuracy: 0.1411The
471/471 [==============================] - 46s 97ms/step - loss: 2.6693 - accuracy: 0.14
Epoch 6/8
471/471 [==============================] - ETA: 0s - loss: 2.5999 - accuracy: 0.1559The
471/471 [==============================] - 67s 142ms/step - loss: 2.5999 - accuracy: 0.1
Epoch 7/8
471/471 [==============================] - ETA: 0s - loss: 2.5435 - accuracy: 0.1714The
471/471 [==============================] - 66s 141ms/step - loss: 2.5435 - accuracy: 0.1
Epoch 8/8
471/471 [==============================] - ETA: 0s - loss: 2.5123 - accuracy: 0.1790The
471/471 [==============================] - 74s 158ms/step - loss: 2.5123 - accuracy: 0.1
```

```
%load_ext tensorboard
```

Warning: you are connected to a GPU runtime, but not utilizing the GPU. Change to a standard runtime

TensorBoard SCALARS GRAPHS INACTIVE

Show data download links

Ignore outliers in chart scalin

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis

STEP RELATIVE

WALL

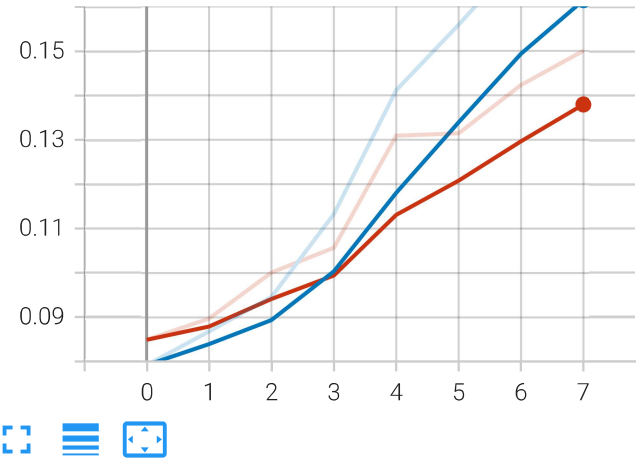
Runs

Write a regex to filter runs

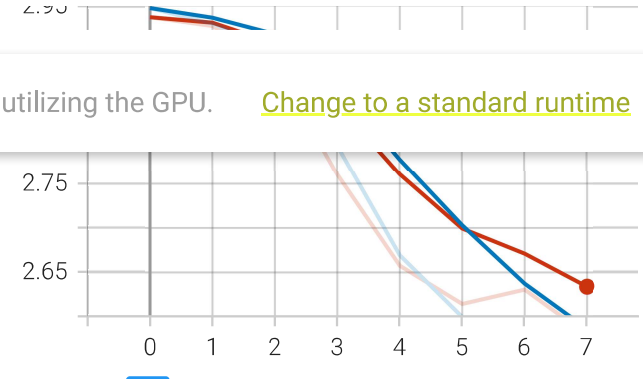
Filter tags (regular expressions supported)

epoch_accuracy

epoch_accuracy
tag: epoch_accuracy



epoch_loss



Warning: you are connected to a GPU runtime, but not utilizing the GPU. Change to a standard runtime