```
!pip install rarfile
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting rarfile
  Downloading rarfile-4.0-py3-none-any.whl (28 kB)
Installing collected packages: rarfile
Successfully installed rarfile-4.0
```

importing libraries

```
import pandas as pd
import rarfile
import numpy as np
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.models import Sequential
from sklearn.preprocessing import LabelEncoder
from keras.utils import np_utils
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input,Dense,Conv1D,Flatten,Embedding,MaxPool1D,concatenate,Dropout
from tensorflow.keras.callbacks import ModelCheckpoint,TensorBoard,EarlyStopping
from tensorflow.keras.optimizers import Adam
```

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'ClassLabel_DocumentNumberInThatLabel'.
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.
```

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 those texts by '.'
after that remove the words whose length is less than or equal to 2 and also remove'com' word and then combine those words by space
In one doc, if we have 2 or more mails, get all.
Eg:[test@dm1.d.com, test2@dm2.dm3.com]-->[dm1.d.com, dm3.dm4.com]-->[dm1,d,com,dm2,dm3,com]-->[dm1,dm2,dm3]-->"dm1 dm2 dm3"
append all those into one list/array. ( This will give length of 18828 sentences i.e one list for each of the document).
Some sample output was shown below.

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

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

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

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

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

importing regular expression module

```
import os
import regex as re
from nltk import ne_chunk, pos_tag, word_tokenize
from nltk.tree import Tree
```

importing data from drive

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

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

```
rar_file = "/content/drive/MyDrive/documents.rar"
```

extracting the data

```
rar = rarfile.RarFile(rar_file)
rar.extractall()
```

```
# access the extracted data
data_dir = "/content/documents"
```

```
import os
import re
from bs4 import BeautifulSoup
```

```
def preprocess(file):
    """Do all the Preprocessing as shown above and
    return a tuple contain preprocess_email,preprocess_subject,preprocess_text for that Text_data"""

    class_ = file.split('_')[0]

    with open("/content/documents/" + file, 'rb') as f:
        text = f.read()
        original_data = text

        # Removing  tags using beautifull soup module
        soup = BeautifulSoup(text, 'lxml')
        text = soup.get_text()

        # Email
        emails = re.findall(r"[\w-]+@[\w\.-]+", text)
        process_emails = [e.split("@")[1].split(".")[0] for e in emails if len(e.split("@")[1].split(".")[0]) > 2 and e.split("@")[1].spl
        final_mail = " ".join(process_emails)

        # Subject
        subject = re.findall(r'Subject:.*', text)
        subject = subject[0].split(":")[-1] if subject else " "
        subject = re.sub(r'[^A-Za-z0-9]+', ' ', subject).lower()

        # Removing  subjects and emails
        text = re.sub(r'Subject:.*', '', text)
        text = re.sub(r"[\w-]+@[\w\.-]+", '', text)

        # removing word from text which length less than 2 and greater than 15
        text = text.lower()
        text = " ".join(word for word in text.split() if 2 < len(word) < 15)

        # Remove sentances starting with "Write to:" or "From:"
        text = re.sub(r'Write to:.', '', text)
        text = re.sub(r'From:.', '', text)

        # Remove new lines, tabs, '-' and "\"
        text = re.sub(r"\s+", " ", text)
        text = re.sub(r"/", ".", text)

        # Remove words ending with ":"
        text = re.sub(r"[a-zA-Z]+:", " ", text)

        # removing number
        text = re.sub(r"[0-9]","",text)

        # Delete  _word_ type words
        text = re.sub(r"(_?)([A-Za-z0-9])(_?)",r'\2',text)

        # remove  oneletter word  and two letter word
        text = re.sub(r"([A-Za-z]{1,2})(_)(A-Za-z)","\g<3>",text)

        # Replace all word except A-Za-z_
        text = re.sub(r'[^A-Za-z_]'," ",text)

        # Decontractions
        # refer : https://stackoverflow.com/questions/19790188/expanding-english-language-contractions-in-python
        contractions = {"can't": "can not", "'s": "is", "i've": "i have", "i'm": "i am", "you're": "you are", "i'll": "i will", "'d": "wo
        for contraction, replacement in contractions.items():
            text = text.replace(contraction, replacement)



        #Chunking
        # refer : https://pythonprogramming.net/chunking-nltk-tutorial/
        #  https://www.analyticsvidhya.com/blog/2021/10/what-is-chunking-in-natural-language-processing/
        chunks = ne_chunk(pos_tag(word_tokenize(text)),binary=True)
        for chunk in chunks:
            if isinstance(chunk, nltk.Tree):
                label = chunk.label()
                words = [word for word, pos in chunk.leaves()]
                string = " ".join(words)
                chunked_string = "_".join(words)
                if label == "PERSON":
                    text = re.sub(r"\b{}\b".format(string), " ", text)
                else:
                    text = re.sub(r"\b{}\b".format(string), chunked_string, text)
                    text = re.sub(r"\s+"," ",text)


        preprocess_email = final_mail
        preprocess_subject = subject
        preprocess_subject = subject
        preprocess_text = text

        return (class_,preprocess_email, preprocess_subject, preprocess_text)


folder = "/content/documents/"
rows = []
for file in os.listdir("/content/documents"):
    class_, preprocess_email, preprocess_subject, preprocess_text = preprocess(file)
    rows.append([class_, preprocess_email, preprocess_subject, preprocess_text])
```

```
import time
start  = time.time()
data_preproces= preprocess('alt.atheism_49960.txt')
end = time.time()
print(end-start)
```

```
0.35096311569213867
```

```
data_preproces
```

```
('alt.atheism',
 'mantis',
 ' atheist resources',
 '  mathew archive   resources last   december      atheist resources addresses atheist organizations usa freedom from religion
foundation darwin fish bumper stickers and assorted other atheist paraphernalia are available from the freedom from religion
foundation the us  write   ffrf  p o  box   madison           evolution designs evolution designs sell the  darwin fish   it s
fish symbol  like the ones christians stick their cars  but with feet and the word  darwin  written inside  the deluxe moulded
plastic fish    postpaid the us  write   evolution designs   laurel canyon    north hollywood    people the san francisco bay
area can get darwin fish from lynn gold try mailing for net people who lynn directly  the price    per fish  american atheist
press aap publish various atheist books critiques the bible  lists biblical and on  one such book    the bible handbook  w p
ball and g w  foote  american atheist press   pp  isbn      nd edition    bible absurdities  atrocities  contains ball     the
bible contradicts itself   aap  based the king james version the bible  write   american atheist press  p o  box   austin
cameron road  austin                   prometheus books sell books including haught s  holy horrors   see below   write    east
amherst street  buffalo  new york           alternate address  which may newer older    prometheus books  glenn drive  buffalo
for humanism organization promoting black secular humanism and uncovering the history black freethought  they publish quarterly
newsletter  aah examiner  write   norm allen  jr   african americans for humanism  p o  box   buffalo    united kingdom
rationalist press association national secular society islington high street  holloway road london ew london n nl       british
humanist association south place ethical society lamb s conduit passage conway hall london wcr rh red lion square    london wcr
rl fax       the national secular society publish  the freethinker   monthly magazine founded   germany ibka e v  bund der und
atheisten postfach   d  berlin   germany  ibka publish   miz   materialien und informationen zur zeit  politisches journal der
und atheisten  hrsg  ibka e v   miz vertrieb  postfach   d  berlin   germany  for atheist books  write   ibdk  b ucherdienst der
postfach   d  hannover germany      books fiction thomas disch  the santa claus compromise  short story  the ultimate proof that
santa exists  all characters and events are fictitious  any similarity living dead gods uh  well    walter miller  canticle for
leibowitz  one gem this post atomic doomsday novel the monks who spent their lives copying blueprints from  saint leibowitz
filling the sheets paper with ink and leaving white lines and letters  edgar pangborn  davy  post atomic doomsday novel set
clerical states  the church  for example  forbids that anyone  produce  describe use any substance containing    atoms   philip
dick philip dick dick wrote many philosophical and short stories and novels  his stories are bizarre times  but very
approachable  wrote mainly sf  but wrote about people  truth and religion rather than technology  although often believed that
had met some sort god  remained sceptical  amongst his novels  the following are some    galactic pot healer  fallible alien
deity summons group earth craftsmen and women remote planet raise giant cathedral from beneath the oceans  when the deity begins
demand faith from the earthers  pot healer joe fernwright unable comply  polished  ironic and amusing novel  maze death
noteworthy for its description religion    valis  the schizophrenic hero searches for the hidden mysteries gnostic christianity
after reality fired into his brain pink laser beam unknown but possibly divine origin  accompanied his dogmatic and dismissively
atheist friend and assorted other odd characters   the divine invasion  god invades earth making young woman pregnant she
returns from another star system  unfortunately she terminally ill  and must assisted dead man whose brain wired  hour easy
listening music  margaret atwood  the handmaid s tale  story based the premise that the congress mysteriously assassinated  and
quickly take charge the nation set  right  again  the book the diary woman s life she tries live under the new christian
theocracy  women s right own property revoked  and their bank accounts are closed  sinful luxuries are outlawed  and the radio
only used for readings from the bible  crimes are punished   doctors who performed legal abortions the  old world  are hunted
down and hanged  atwood s writing style difficult get used first  but the tale grows more and more chilling goes on  various
authors  the bible  this somewhat dull and rambling work has often been criticized  however  probably worth reading  only that
you ll know what all the fuss about  exists many different versions  make sure you get the one true version  books non fiction
peter rosa  vicars christ   bantam press   although rosa seems christian even catholic this very enlighting history papal
immoralities  adulteries  fallacies etc   german    gottes erste diener  die dunkle seite des papsttums   droemer knaur
michael martin    philosophical temple university press  philadelphia  usa  detailed and scholarly justification atheism
contains outstanding appendix defining terminology and usage this  necessarily  tendentious area  argues both for  negative
atheism   i e  the  non belief the existence god s    and also for  positive atheism    the belief the non existence god s
includes great refutations the most challenging arguments for god  particular attention paid refuting contempory theists such
platinga and swinburne   pages  isbn      hardcover  paperback also available   the case against christianity   temple
university press comprehensive critique christianity  which considers the best contemporary defences christianity and
ultimately  demonstrates that they are unsupportable and or incoherent   pages  isbn     james turner  without god  without
creed   the johns hopkins university press  baltimore  md  usa subtitled  the origins unbelief america   examines the way which
unbelief  whether agnostic atheistic  became mainstream alternative world view  focusses the period    and while considering
france and britain the emphasis american  and particularly new england developments   neither religious history secularization
atheism  without god  without creed is  rather  the intellectual history the fate single idea  the belief that god exists
pages  isbn  hardcover     x  paper      george seldes  editor   the great thoughts    ballantine books  new york  usa
dictionary quotations  different kind  concentrating statements and writings which  explicitly implicitly  present the person s
philosophy and world view  includes obscure  and often suppressed  opinions from many people  for some popular observations
```

```
row = []
done = 0
for f in os.listdir('documents'):
    if done%1000==0:
        print(done)
    done+=1

    data_preproces = preprocess(f)
    row.append(data_preproces)
```

```
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
18000
```

```
data = pd.DataFrame(rows, columns=["class_", "email", "subject", "preprocess_text"])
```

```
data.shape
```

```
(18828, 4)
```

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

```
class_                                              rec.sport.hockey
email                                           cunixb columbia cunixc
subject                                           atlanta hockey hell
preprocess_text       gary dare  mamatha devineni ratnam   can t ...
Name: 400, dtype: object
```

now will load the pre_processed_csv files

```
data =pd.read_csv('/content/my_dataframe.csv')
```

```
data.head(5)
```

	class_	email	subject	preprocess_text
0	comp.sys.ibm.pc.hardware	tegra iastate	monitors nanao	article brian schaufenbuel what tube does the ...
1	rec.sport.hockey	cmsa andrew	wings ogrodnick	article mikemolloy haven seen any mention ogro...
2	sci.med	alcor	lasers for dermatologists	having had limited tinea pedis for more than y...
[illegible]	[illegible]	[illegible]	[illegible]	what the phone number for alias toll

column which we want to combine

```
columns = ['email','subject','preprocess_text']
```

code for combining the column as total_data

```
data['total_data'] = data[columns].astype(str).agg(' '.join, axis=1)
```

```
data.head(8)
```

	class_	email	subject	preprocess_text	total_data
0	comp.sys.ibm.pc.hardware	tegra iastate	monitors nanao	article brian schaufenbuel what tube does the ...	tegra iastate monitors nanao article brian s...
1	rec.sport.hockey	cmsa andrew	wings ogrodnick	article mikemolloy haven seen any mention ogro...	cmsa andrew wings ogrodnick article mikemollo...

now this is new_data after combining the column

diablo alias phone

```
new_data = data[['class_','total_data']]
new_data.head(8)
```

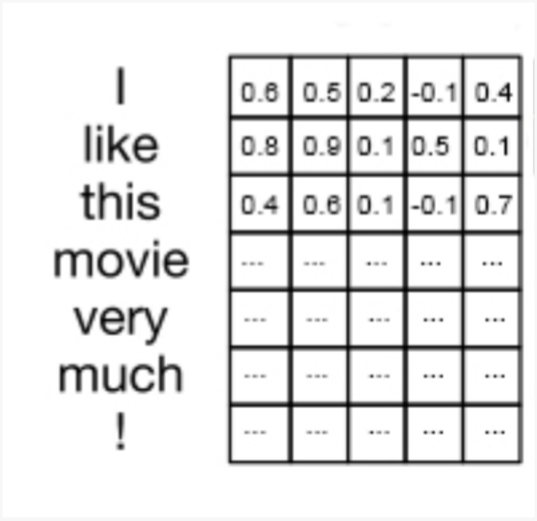
	class_	total_data
0	comp.sys.ibm.pc.hardware	tegra iastate monitors nanao article brian s...
1	rec.sport.hockey	cmsa andrew wings ogrodnick article mikemollo...
2	sci.med	alcor lasers for dermatologists having had li...
3	comp.graphics	diablo alias phone number wanted what the pho...
4	rec.autos	stdvax mimsy questions about insurance compan...
5	comp.sys.mac.hardware	nan looking for free share wares looking for ...
6	rec.sport.baseball	cybernet dcseq houston mailing list can anyon...
7	sci.space	ksr access access ksr keeping spacecraft on a...

Training The models to Classify:

```
1. Combine "preprocessed_text", "preprocessed_subject", "preprocessed_emails" into one column. use that column to model.

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

4. Do Tokenizer i.e convert text into numbers. please be careful while doing it.
if you are using tf.keras "Tokenizer" API, it removes the "_", but we need that.

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 the previous assignments.

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.
```

Model-1: Using 1D convolutions with word embeddings

```
Encoding of the Text  --> For a given text data create a Matrix with Embedding layer as shown Below.
In the example we have considered d = 5, but in this assignment we will get d = dimension of Word vectors we are using.
 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-keras/

How EMBEDDING LAYER WORKS

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/

1. all are Conv1D layers with any number of filter and filter sizes, there is no restriction on this.
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 params. (Only recommendation if you have less computing power)
5. You can use any number of layers after the Flatten Layer.

```
%load_ext tensorboard
```

```
    The tensorboard extension is already loaded. To reload it, use:
      %reload_ext tensorboard
```

```
x = new_data['total_data']
y = new_data['class_']

# refer : https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/
# encoding lables
encoder = LabelEncoder()
encoder.fit(y)
encoder_y = encoder.transform(y)
## converting it to a matrix
y = np_utils.to_categorical(encoder_y)
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,stratify=y)
```

```
length_sentence =[len(s) for s in x_train]
length_sentence.sort()
length_sentence = np.array(length_sentence)
```

```
percentile_90 = int(np.percentile(length_sentence,90))
percentile_90
```

```
    2241
```

```
percentile_98= int(np.percentile(length_sentence,98))
percentile_98
```

```
    6116
```

```
maxl = percentile_98
```

Since 98% of the size of the sentences are less than 6116 we will use maxlen

```
tokenizer = Tokenizer(filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n')
tokenizer.fit_on_texts(x_train)
x_train  = tokenizer.texts_to_sequences(x_train)
x_test = tokenizer.texts_to_sequences(x_test)
```

```
# refer : https://stackoverflow.com/questions/42943291/what-does-keras-io-preprocessing-sequence-pad-sequences-do#:~:text=pad_sequences%2

x_train = pad_sequences(x_train,maxlen=maxl,padding="post")
x_test= pad_sequences(x_test,maxlen=maxl,padding='post')
```

```
print(x_train.shape)
print(x_test.shape)
```

```
    (14121, 5787)
    (4707, 5787)
```

downloading the pre-trained glover vector

```
!wget http://nlp.stanford.edu/data/glove.6B.zip
```

```
    --2023-02-04 04:20:56--  http://nlp.stanford.edu/data/glove.6B.zip
    Resolving nlp.stanford.edu (nlp.stanford.edu)... 171.64.67.140
    Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:80... connected.
    HTTP request sent, awaiting response... 302 Found
    Location: https://nlp.stanford.edu/data/glove.6B.zip [following]
    --2023-02-04 04:20:56--  https://nlp.stanford.edu/data/glove.6B.zip
    Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:443... connected.
    HTTP request sent, awaiting response... 301 Moved Permanently
    Location: https://downloads.cs.stanford.edu/nlp/data/glove.6B.zip [following]
    --2023-02-04 04:20:56--  https://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
    Resolving downloads.cs.stanford.edu (downloads.cs.stanford.edu)... 171.64.64.22
    Connecting to downloads.cs.stanford.edu (downloads.cs.stanford.edu)|171.64.64.22|:443... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 862182613 (822M) [application/zip]
    Saving to: ‘glove.6B.zip’

    glove.6B.zip        100%[===================>] 822.24M  3.85MB/s    in 3m 14s

    2023-02-04 04:24:10 (4.24 MB/s) - ‘glove.6B.zip’ saved [862182613/862182613]
```

unzipping

```
!unzip glove.6B.zip
```

```
    Archive:  glove.6B.zip
      inflating: glove.6B.50d.txt
      inflating: glove.6B.100d.txt
      inflating: glove.6B.200d.txt
      inflating: glove.6B.300d.txt
```

```
##https://medium.com/analytics-vidhya/basics-of-using-pre-trained-glove-vectors-in-python-d38905f356db

import numpy as np
```

```
embedding_dict = {}
pretrain =  open("glove.6B.50d.txt")
for i in pretrain:
    value = i.split(" ")
    word = value[0]
    vector = np.asarray(value[1:])
    embedding_dict[word] = vector
pretrain.close()
```

now tokenizing the indexes words

```
len(tokenizer.index_word)
```

```
103137
```

```
#Converting embedding word to embedding matrix

import numpy as np

size = len(tokenizer.word_index) + 1
emb_matrix = np.zeros((size, 50))

for word, i in tokenizer.word_index.items():
    emb = embedding_dict.get(word)
    if emb is not None:
        emb_matrix[i] = emb
```

```
print(emb_matrix.shape)
```

```
(103138, 50)
```

```
size
```

```
103138
```

Model1

```
pip install -U tensorflow-estimator
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tensorflow-estimator in /usr/local/lib/python3.8/dist-packages (2.11.0)
```

```
## Embedding layer
embedding_layer = Embedding(len(tokenizer.word_index)+1, 50, embeddings_initializer=tf.keras.initializers.Constant(emb_matrix),trainable=
```

```
first_layer = Input(shape=(maxl))

embed = embedding_layer(first_layer)

# conv1D layer with relu activation function and he.normal intilizer with embedding layer
conv1 = Conv1D(32,4,activation="relu",kernel_initializer =tf.keras.initializers.he_normal(),kernel_regularizer=tf.keras.regularizers.l2()

# con1d layer ith relu activation function with he.normal intilizer with l2 regularizer
conv2 = Conv1D(32,4,activation="relu",kernel_initializer =tf.keras.initializers.he_normal(),kernel_regularizer=tf.keras.regularizers.l2()

# con1d layer ith relu activation function with he.normal intilizer with l2 regularizer
conv3 = Conv1D(32,4,activation="relu",kernel_initializer = tf.keras.initializers.he_normal(),kernel_regularizer=tf.keras.regularizers.l2(

# now we are concating the all 3 layers named as second as layer
second_layer = concatenate([conv1,conv2,conv3])

# here we are adding max_pool with second layer
max_pool_1 = MaxPool1D(3)(second_layer)

# conv 1d layer with relu as activation fucntion and he.noral intilizers  and l2 regularizeer
conv4 = Conv1D(32,3,activation="relu",kernel_initializer = tf.keras.initializers.he_normal(),kernel_regularizer=tf.keras.regularizers.l2(

# conv 1d layer with relu activation function with he.normal intilizer with l2 regularizer with max_pool_1
conv5 = Conv1D(32,3,activation="relu",kernel_initializer = tf.keras.initializers.he_normal(),kernel_regularizer=tf.keras.regularizers.l2(
# conv 1D layer with relu activation function with he.normal initilizer with l2 regularizer with conacting the max_pool_1
conv6 =  Conv1D(32,3,activation="relu",kernel_initializer = tf.keras.initializers.he_normal(),kernel_regularizer=tf.keras.regularizers.l2

# now will concat the all 3 previous layers
third_layer = concatenate([conv4,conv5,conv6])

# now we using maxpool 1d layer with third layer
max_pool_2 = MaxPool1D(3)(third_layer)

# now we are adding 1 more conv1d layer named as our fourth layer with relu activation functio and l2 regularizer
fourth_layer = Conv1D(32,3,activation='relu', kernel_initializer = tf.keras.initializers.he_normal(seed=42),kernel_regularizer=tf.keras.r

# now finally will add the flatten layer
flatten = Flatten()(fourth_layer)

# now we are adding the dropout layer with the value of 0.3
dropout_layer = Dropout(0.3)(flatten)


# now we are adding dense fully connecting layer with relu activation function
dense_layer = Dense(64,activation="relu",kernel_initializer = tf.keras.initializers.he_normal())(dropout_layer)

# now finally we are adding our output layer with softmax function with glorot normal
output_layer = Dense(20,activation="softmax",kernel_initializer= tf.keras.initializers.glorot_normal())(dense_layer)

# now will compile the model
model =Model(inputs=first_layer,outputs=output_layer)
```

```
model.summary()
```

```
Model: "model_2"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to                     
==================================================================================================
 input_3 (InputLayer)           [(None, 5787)]       0           []                               
                                                                                                  
 embedding_2 (Embedding)        (None, 5787, 50)     5156900     ['input_3[0][0]']                
                                                                                                  
 conv1d_12 (Conv1D)             (None, 5784, 32)     6432        ['embedding_2[0][0]']            
                                                                                                  
 conv1d_13 (Conv1D)             (None, 5784, 32)     6432        ['embedding_2[0][0]']            
                                                                                                  
 conv1d_14 (Conv1D)             (None, 5784, 32)     6432        ['embedding_2[0][0]']            
                                                                                                  
 concatenate_2 (Concatenate)    (None, 5784, 96)     0           ['conv1d_12[0][0]',              
                                                                  'conv1d_13[0][0]',              
                                                                  'conv1d_14[0][0]']              
                                                                                                  
 max_pooling1d_5 (MaxPooling1D)  (None, 1928, 96)    0           ['concatenate_2[0][0]']          
                                                                                                  
 conv1d_15 (Conv1D)             (None, 1926, 32)     9248        ['max_pooling1d_5[0][0]']        
                                                                                                  
 conv1d_16 (Conv1D)             (None, 1926, 32)     9248        ['max_pooling1d_5[0][0]']        
                                                                                                  
 conv1d_17 (Conv1D)             (None, 1926, 32)     9248        ['max_pooling1d_5[0][0]']        
                                                                                                  
 concatenate_3 (Concatenate)    (None, 1926, 96)     0           ['conv1d_15[0][0]',              
                                                                  'conv1d_16[0][0]',              
                                                                  'conv1d_17[0][0]']              
                                                                                                  
 max_pooling1d_6 (MaxPooling1D)  (None, 642, 96)     0           ['concatenate_3[0][0]']          
                                                                                                  
 conv1d_18 (Conv1D)             (None, 640, 32)      9248        ['max_pooling1d_6[0][0]']        
                                                                                                  
 flatten_2 (Flatten)            (None, 20480)        0           ['conv1d_18[0][0]']              
                                                                                                  
 dropout_2 (Dropout)            (None, 20480)        0           ['flatten_2[0][0]']              
                                                                                                  
 dense_4 (Dense)                (None, 64)           1310784     ['dropout_2[0][0]']              
                                                                                                  
 dense_5 (Dense)                (None, 20)           1300        ['dense_4[0][0]']                
                                                                                                  
==================================================================================================
Total params: 6,525,272
Trainable params: 1,368,372
Non-trainable params: 5,156,900
__________________________________________________________________________________________________
```

```
!pip install tensorflow-addons==0.16.1
import tensorflow_addons as tfa
```

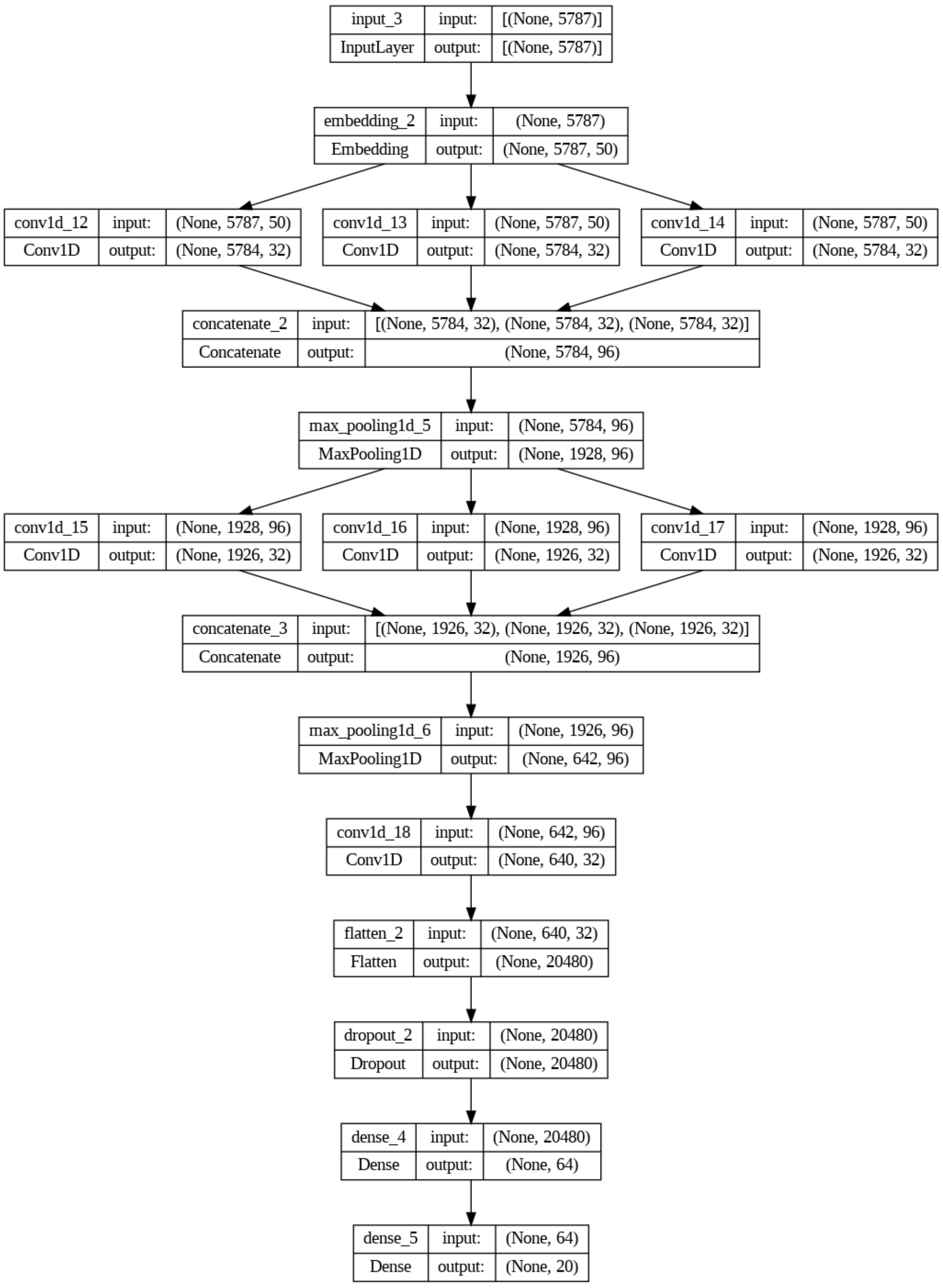
```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tensorflow-addons==0.16.1 in /usr/local/lib/python3.8/dist-packages (0.16.1)
Requirement already satisfied: typeguard>=2.7 in /usr/local/lib/python3.8/dist-packages (from tensorflow-addons==0.16.1) (2.7.1)
```

```
from tensorflow_addons.metrics import F1Score
```

```
## f1_score_callback
#custom_callback = custom()

## Callback for saving best model
checkpoint = ModelCheckpoint(filepath='best_model_1.h5',verbose=1,monitor='val_accuracy',
                           mode='max',save_best_only=True)

## Callback for earlystopping
early_stop = EarlyStopping(monitor="val_accuracy",mode='max',patience=2)

## Tensorboard
log_dir = "logs"
tensorboard = TensorBoard(log_dir=log_dir,histogram_freq=1,write_graph=True)

## all callbacks
callbacks =[checkpoint,early_stop,tensorboard]

## compile model
model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['accuracy',F1Score(average='micro',num_class


## Trainning


model.fit(x_train,y_train,epochs=15,verbose=2,validation_data=(x_test,y_test),batch_size =64,callbacks=callbacks)
```

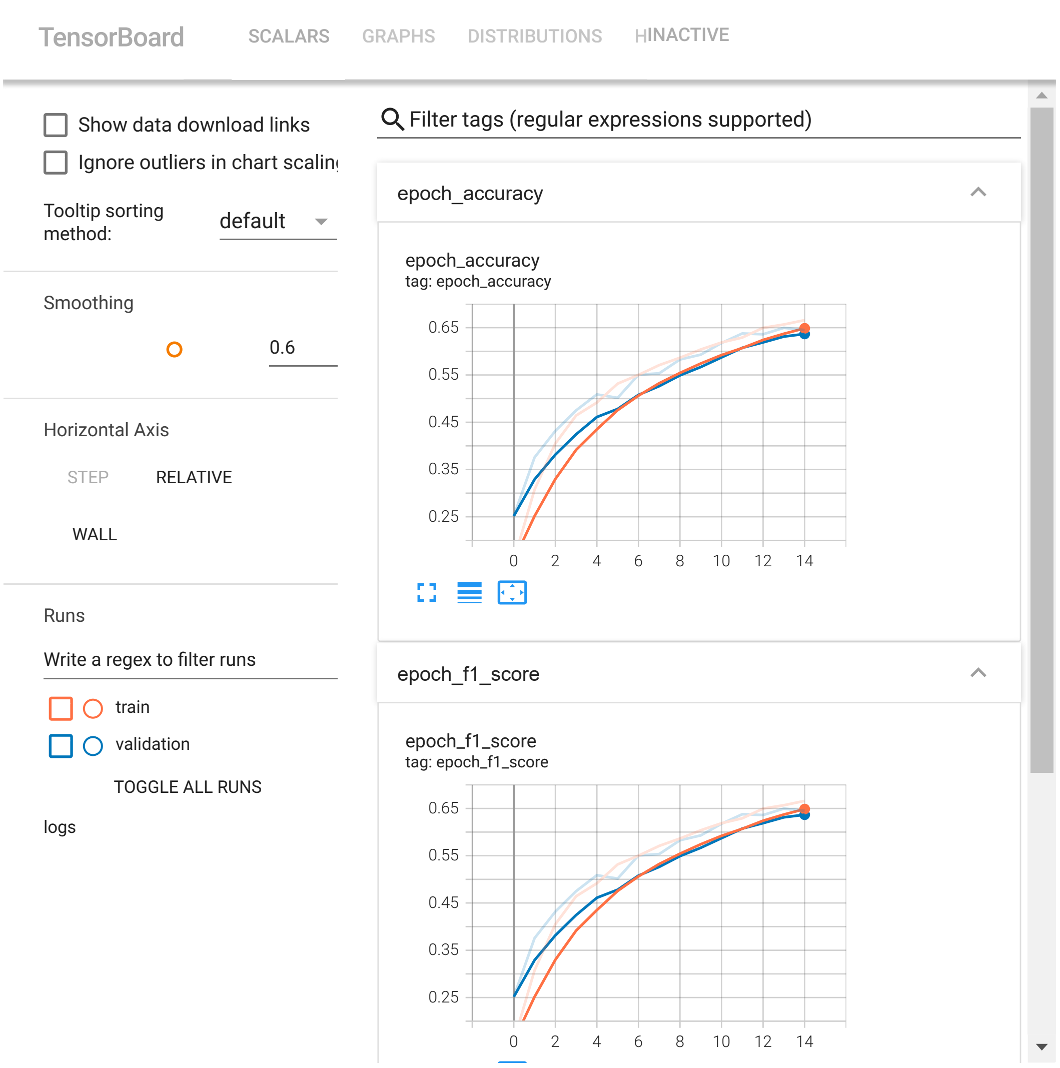
```
Epoch 1/15

Epoch 1: val_accuracy improved from -inf to 0.25154, saving model to best_model_1.h5
221/221 - 22s - loss: 4.0689 - accuracy: 0.1583 - f1_score: 0.1583 - val_loss: 2.6624 - val_accuracy: 0.2515 - val_f1_score: 0.2
Epoch 2/15

Epoch 2: val_accuracy improved from 0.25154 to 0.37561, saving model to best_model_1.h5
221/221 - 20s - loss: 2.3512 - accuracy: 0.3074 - f1_score: 0.3074 - val_loss: 2.1054 - val_accuracy: 0.3756 - val_f1_score: 0.3
Epoch 3/15

Epoch 3: val_accuracy improved from 0.37561 to 0.43170, saving model to best_model_1.h5
221/221 - 21s - loss: 1.9724 - accuracy: 0.4051 - f1_score: 0.4051 - val_loss: 1.8733 - val_accuracy: 0.4317 - val_f1_score: 0.4
Epoch 4/15

Epoch 4: val_accuracy improved from 0.43170 to 0.47504, saving model to best_model_1.h5
221/221 - 22s - loss: 1.7811 - accuracy: 0.4642 - f1_score: 0.4642 - val_loss: 1.7315 - val_accuracy: 0.4750 - val_f1_score: 0.4
Epoch 5/15

Epoch 5: val_accuracy improved from 0.47504 to 0.50882, saving model to best_model_1.h5
221/221 - 20s - loss: 1.6786 - accuracy: 0.4917 - f1_score: 0.4917 - val_loss: 1.6646 - val_accuracy: 0.5088 - val_f1_score: 0.5
Epoch 6/15

Epoch 6: val_accuracy did not improve from 0.50882
221/221 - 19s - loss: 1.5704 - accuracy: 0.5316 - f1_score: 0.5316 - val_loss: 1.6709 - val_accuracy: 0.5016 - val_f1_score: 0.5
Epoch 7/15

Epoch 7: val_accuracy improved from 0.50882 to 0.55046, saving model to best_model_1.h5
221/221 - 20s - loss: 1.5264 - accuracy: 0.5501 - f1_score: 0.5501 - val_loss: 1.5493 - val_accuracy: 0.5505 - val_f1_score: 0.5
Epoch 8/15

Epoch 8: val_accuracy improved from 0.55046 to 0.55343, saving model to best_model_1.h5
221/221 - 20s - loss: 1.4666 - accuracy: 0.5708 - f1_score: 0.5708 - val_loss: 1.5474 - val_accuracy: 0.5534 - val_f1_score: 0.5
Epoch 9/15

Epoch 9: val_accuracy improved from 0.55343 to 0.58275, saving model to best_model_1.h5
221/221 - 20s - loss: 1.4338 - accuracy: 0.5864 - f1_score: 0.5864 - val_loss: 1.4357 - val_accuracy: 0.5827 - val_f1_score: 0.5
Epoch 10/15

Epoch 10: val_accuracy improved from 0.58275 to 0.59295, saving model to best_model_1.h5
221/221 - 20s - loss: 1.3849 - accuracy: 0.6039 - f1_score: 0.6039 - val_loss: 1.4367 - val_accuracy: 0.5929 - val_f1_score: 0.5
Epoch 11/15

Epoch 11: val_accuracy improved from 0.59295 to 0.61759, saving model to best_model_1.h5
221/221 - 21s - loss: 1.3550 - accuracy: 0.6187 - f1_score: 0.6187 - val_loss: 1.3891 - val_accuracy: 0.6176 - val_f1_score: 0.6
Epoch 12/15

Epoch 12: val_accuracy improved from 0.61759 to 0.63777, saving model to best_model_1.h5
221/221 - 21s - loss: 1.3172 - accuracy: 0.6292 - f1_score: 0.6292 - val_loss: 1.3466 - val_accuracy: 0.6378 - val_f1_score: 0.6
Epoch 13/15

Epoch 13: val_accuracy did not improve from 0.63777
221/221 - 19s - loss: 1.2845 - accuracy: 0.6497 - f1_score: 0.6497 - val_loss: 1.3434 - val_accuracy: 0.6361 - val_f1_score: 0.6
Epoch 14/15

Epoch 14: val_accuracy improved from 0.63777 to 0.64967, saving model to best_model_1.h5
221/221 - 20s - loss: 1.2587 - accuracy: 0.6568 - f1_score: 0.6568 - val_loss: 1.3094 - val_accuracy: 0.6497 - val_f1_score: 0.6
Epoch 15/15
```

```
# refer : https://www.tensorflow.org/api_docs/python/tf/keras/utils/plot_model

tf.keras.utils.plot_model(model,to_file = 'model1.png',show_shapes=True,show_layer_names=True)
```

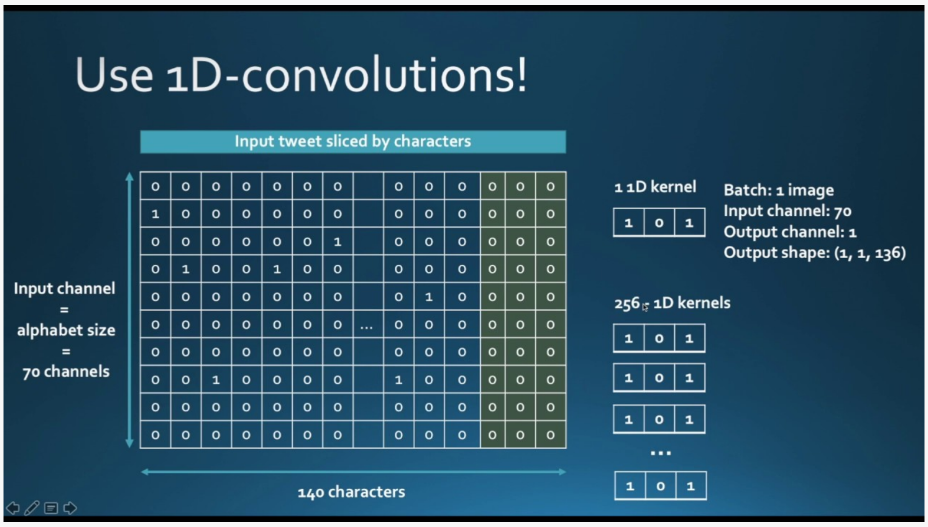


input_3 InputLayer input: [(None, 5787)] output: [(None, 5787)]
embedding_2 Embedding input: (None, 5787) output: (None, 5787, 50)
conv1d_12 Conv1D input: (None, 5787, 50) output: (None, 5784, 32)
conv1d_13 Conv1D input: (None, 5787, 50) output: (None, 5784, 32)
conv1d_14 Conv1D input: (None, 5787, 50) output: (None, 5784, 32)
concatenate_2 Concatenate input: [(None, 5784, 32), (None, 5784, 32), (None, 5784, 32)] output: (None, 5784, 96)
max_pooling1d_5 MaxPooling1D input: (None, 5784, 96) output: (None, 1928, 96)
conv1d_15 Conv1D input: (None, 1928, 96) output: (None, 1926, 32)
conv1d_16 Conv1D input: (None, 1928, 96) output: (None, 1926, 32)
conv1d_17 Conv1D input: (None, 1928, 96) output: (None, 1926, 32)
concatenate_3 Concatenate input: [(None, 1926, 32), (None, 1926, 32), (None, 1926, 32)] output: (None, 1926, 96)
max_pooling1d_6 MaxPooling1D input: (None, 1926, 96) output: (None, 642, 96)
conv1d_18 Conv1D input: (None, 642, 96) output: (None, 640, 32)
flatten_2 Flatten input: (None, 640, 32) output: (None, 20480)
dropout_2 Dropout input: (None, 20480) output: (None, 20480)
dense_4 Dense input: (None, 20480) output: (None, 64)
dense_5 Dense input: (None, 64) output: (None, 20)

```
%tensorboard --logdir logs
```

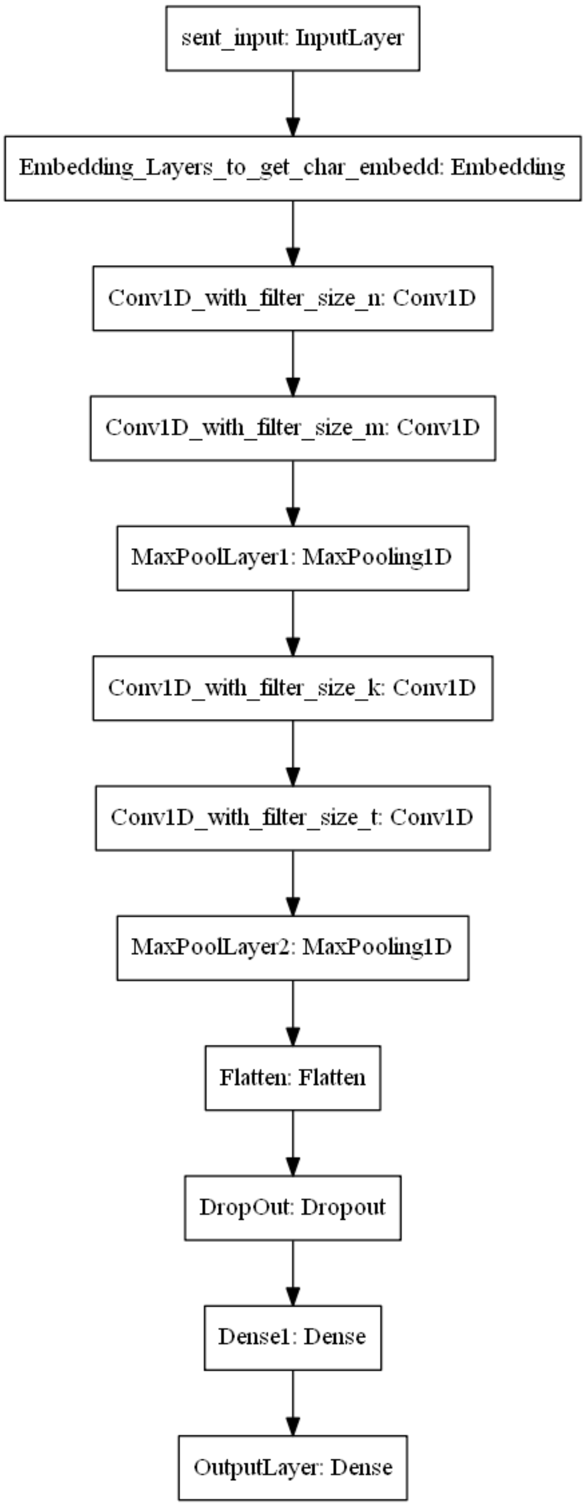


Model-2 : Using 1D convolutions with character embedding



Here are the some papers based on Char-CNN

1. Xiang Zhang, Junbo Zhao, Yann LeCun. Character-level Convolutional Networks for Text Classification.NIPS 2015
2. Yoon Kim, Yacine Jernite, David Sontag, Alexander M. Rush. Character-Aware Neural Language Models. AAAI 2016
3. Shaojie Bai, J. Zico Kolter, Vladlen Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequ
4. Use the pratrained char embeddings https://github.com/minimaxir/char-embeddings/blob/master/glove.840B.300d-char.txt


sent_input: InputLayer
Embedding_Layers_to_get_char_embedd: Embedding
Conv1D_with_filter_size_n: Conv1D
Conv1D_with_filter_size_m: Conv1D
MaxPoolLayer1: MaxPooling1D
Conv1D_with_filter_size_k: Conv1D
Conv1D_with_filter_size_t: Conv1D
MaxPoolLayer2: MaxPooling1D
Flatten: Flatten
DropOut: Dropout
Dense1: Dense
OutputLayer: Dense

```
#https://towardsdatascience.com/besides-word-embedding-why-you-need-to-know-character-embedding-6096a34a3b10
#https://towardsdatascience.com/character-level-cnn-with-keras-50391c3adf33
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,stratify=y)
```

```
tokenize_char = Tokenizer(filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',char_level= True,oov_token='UNK')
#training the train data
tokenize_char.fit_on_texts(x_train)
```

```
print(tokenize_char.word_index)
```

```
{'UNK': 1, ' ': 2, 'e': 3, 't': 4, 'a': 5, 'i': 6, 'o': 7, 'n': 8, 's': 9, 'r': 10, 'h': 11, 'l': 12, 'd': 13, 'c': 14, 'u': 15, 'm
```

```
size_char = len(tokenize_char.word_index)+1
print(size_char)
```

```
41
```

```
## Tokenize them
x_train = tokenize_char.texts_to_sequences(x_train)
x_test = tokenize_char.texts_to_sequences(x_test)
```

```
maxl = int(np.percentile(length_sentence,99))
```

```


print(maxl)
```

```
8959
```

```
x_train = pad_sequences(x_train,maxlen=maxl,padding="post")
x_test = pad_sequences(x_test,maxlen=maxl,padding="post")
print(f"x_train_shape{x_train.shape}")
print(f"x_train_shape{x_test.shape}")
```

```
x_train_shape(14121, 8959)
x_train_shape(4707, 8959)
```

```
# Make a embedding matrix

emb_matrix_char = np.zeros((41,41))

#print(tokenize_char.word_index)
for i,j in tokenize_char.word_index.items():
  emb_matrix_char[j][j]=1
```

```
print(emb_matrix_char)
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 [0. 0. 1. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 1. 0. 0.]
 [0. 0. 0. ... 0. 1. 0.]
 [0. 0. 0. ... 0. 0. 1.]]
```

```
# https://www.tensorflow.org/text/guide/word_embeddings#:~:text=The%20Embedding%20layer%20takes%20the,batch%2C%20sequence%2C%20embedding)

embedding_layer_char = Embedding(len(tokenize_char.word_index)+1,41, embeddings_initializer=tf.keras.initializers.Constant(emb_matrix_cha
```

```
first_layer = Input(shape=(maxl))
embed = embedding_layer_char(first_layer)
```

```
# conv1D layer with relu activation function and he.normal intilizer with embedding layer
con1 = Conv1D(64, 3, activation='relu', kernel_initializer=tf.keras.initializers.he_normal(seed=42), kernel_regularizer=tf.keras.regulari
# con1d layer ith relu activation function with he.normal intilizer with l1 regularizer
conv2 = Conv1D(64, 3, activation='relu', kernel_initializer=tf.keras.initializers.he_normal(seed=42), kernel_regularizer=tf.keras.regular

# max pool 1d layer
max_pool_1 = MaxPool1D(5)(conv2)

# conv1D layer he.normal intilizer with l1 regularizer
conv3= Conv1D(64, 3, activation='relu', kernel_initializer=tf.keras.initializers.he_normal(seed=42), kernel_regularizer=tf.keras.regulari

# conv1D layer with relu activation function he_normal intilizer
conv4 = Conv1D(64, 3, activation='relu', kernel_initializer=tf.keras.initializers.he_normal(seed=42), kernel_regularizer=tf.keras.regular

# maxpool 1d layer
max_pool_2 = MaxPool1D(5)(conv4)

# conv1D layer with relu activation fucntion
conv5 = Conv1D(64, 3, activation='relu', kernel_initializer=tf.keras.initializers.he_normal(seed=42), kernel_regularizer=tf.keras.regular
max_pool_3 = MaxPool1D(5)(conv5)

# flattened layer
flatten = Flatten()(max_pool_3)

# adding droput layer with value 0.5
dropout = Dropout(0.5)(flatten)

# dense layer with relu activation after adding dropout layer
dense1 = Dense(256, activation='relu', kernel_initializer=tf.keras.initializers.he_normal(seed=42))(dropout)
```

```
# now will write the output layer with softmax function because of multiclass classification problem
output = Dense(20, activation='softmax', kernel_initializer=tf.keras.initializers.glorot_normal(seed=42))(dense1)

# now model is ready to compile
model = Model(inputs=first_layer, outputs=output)
```

```
# model summery
model.summary()
```

```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_4 (InputLayer)        [(None, 8959)]            0         
                                                                 
 embedding_3 (Embedding)     (None, 8959, 41)          1681      
                                                                 
 conv1d_19 (Conv1D)          (None, 8957, 64)          7936      
                                                                 
 conv1d_20 (Conv1D)          (None, 8955, 64)          12352     
                                                                 
 max_pooling1d_7 (MaxPooling  (None, 1791, 64)         0         
 1D)                                                             
                                                                 
 conv1d_21 (Conv1D)          (None, 1789, 64)          12352     
                                                                 
 conv1d_22 (Conv1D)          (None, 1787, 64)          12352     
                                                                 
 max_pooling1d_8 (MaxPooling  (None, 357, 64)          0         
 1D)                                                             
                                                                 
 conv1d_23 (Conv1D)          (None, 355, 64)           12352     
                                                                 
 max_pooling1d_9 (MaxPooling  (None, 71, 64)           0         
 1D)                                                             
                                                                 
 flatten_3 (Flatten)         (None, 4544)              0         
                                                                 
 dropout_3 (Dropout)         (None, 4544)              0         
                                                                 
 dense_6 (Dense)             (None, 256)               1163520   
                                                                 
 dense_7 (Dense)             (None, 20)                5140      
                                                                 
=================================================================
Total params: 1,227,685
Trainable params: 1,226,004
Non-trainable params: 1,681
_________________________________________________________________
```

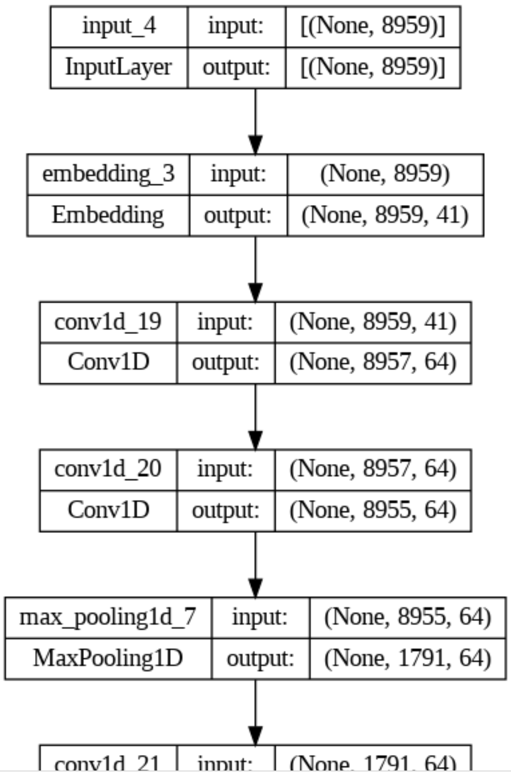
```
# refer : https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/Callback

from sklearn.metrics import f1_score
class custom_callback(tf.keras.callbacks.Callback):

  def on_train_begin(self,logs={}):
    self.f1_score_list = []

  def on_epoch_end(self,epoch,logs={}):

    x_val,y_val = x_test,y_test
    # here we have taken x_test, y test as validation data


    pred_y = self.model.predict(x_val)

    y_true = np.zeros(y_val.shape[0])
    y_predicted  = np.zeros(pred_y.shape[0])

    for i in range(len(y_true)):
      y_true[i] = int(np.argmax(y_val[i]))
      y_predicted[i] = int(np.argmax(y_predicted[i]))

    # printing the f1 score
    f1_value = f1_score(y_true,y_predicted,average="macro")
    print("f1_score:",f1_value)

# now will append the f1 score to variable f1_call
    self.f1_score_list.append(f1_value)
```

```
f1_call = custom_callback()
```

```
## Callback for saving best model
checkpoint = ModelCheckpoint(filepath='best_model_1.h5',verbose=1,monitor='val_accuracy',
                           mode='auto',save_best_only=True)

# using early stopping if the model got the certain condtion
# REF : https://stackoverflow.com/questions/50284898/keras-earlystopping-which-min-delta-and-patience-to-use
early_stop = EarlyStopping(monitor="val_accuracy",mode='max',min_delta=0.35,patience=2,verbose=1)

## Tensorboard
log_dir = "logs"
tensorboard = TensorBoard(log_dir=log_dir,histogram_freq=1,write_graph=True)

#  HERE WITH HELP OF REDUCE_LR WE ARE STOPPING OUR LEARNING RATE AFTER NOT IMPROVEMENT IN ACCURACY
#REF: https://stackoverflow.com/questions/51889378/how-to-use-keras-reducelronplateau
reduce_learning_rate = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_accuracy',patience=1,mode='auto',verbose=1,factor=0.9,min_lr=0.0

## all callbacks
callbacks =[reduce_learning_rate , f1_call,checkpoint,early_stop,tensorboard]

# compile model
model.compile(loss='categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(learning_rate=0.001), metrics=['accuracy'])


#train the model


model.fit(x_train,y_train,epochs=15,validation_data=(x_test,y_test),batch_size =64,callbacks=callbacks)
```

```
Epoch 1/15
  6/221 [..............................] - ETA: 16s - loss: 51.3087 - accuracy: 0.0260WARNING:tensorflow:Callback method `on_train_
148/148 [==============================] - 2s 14ms/step
f1_score: 0.004075810067250866

Epoch 1: val_accuracy improved from -inf to 0.05163, saving model to best_model_1.h5
221/221 [==============================] - 24s 99ms/step - loss: 18.0384 - accuracy: 0.0636 - val_loss: 3.3793 - val_accuracy: 0.05
Epoch 2/15
148/148 [==============================] - 2s 13ms/step
f1_score: 0.004075810067250866

Epoch 2: val_accuracy improved from 0.05163 to 0.05290, saving model to best_model_1.h5
221/221 [==============================] - 22s 99ms/step - loss: 3.0900 - accuracy: 0.0511 - val_loss: 3.0601 - val_accuracy: 0.052
Epoch 3/15
148/148 [==============================] - 2s 13ms/step
f1_score: 0.004075810067250866

Epoch 3: val_accuracy improved from 0.05290 to 0.05311, saving model to best_model_1.h5
221/221 [==============================] - 22s 98ms/step - loss: 3.0595 - accuracy: 0.0479 - val_loss: 3.0585 - val_accuracy: 0.053
Epoch 3: early stopping
<keras.callbacks.History at 0x7ff06de4ca90>
```

```
tf.keras.utils.plot_model(model,to_file = 'model2.png',show_shapes=True,show_layer_names=True)
```


input_4 | InputLayer | input: [(None, 8959)] | output: [(None, 8959)]
embedding_3 | Embedding | input: (None, 8959) | output: (None, 8959, 41)
conv1d_19 | Conv1D | input: (None, 8959, 41) | output: (None, 8957, 64)
conv1d_20 | Conv1D | input: (None, 8957, 64) | output: (None, 8955, 64)
max_pooling1d_7 | MaxPooling1D | input: (None, 8955, 64) | output: (None, 1791, 64)
conv1d_21 | input: (None, 1791, 64)

✓ 3s completed at 11:06 AM