Spoken Digit Recognition

May 7, 2021

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

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

[2]:
```
import numpy as np
import pandas as pd
import librosa
import os
import seaborn as sns
##if you need any imports you can do that here.
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19:
FutureWarning: pandas.util.testing is deprecated. Use the functions in the
public API at pandas.testing instead.
  import pandas.util.testing as tm
```

We shared recordings.zip, please unzip those.

[ ]:
```
path = '/content/drive/My Drive/recordings'
```

[ ]:
```
cd /content/drive/My Drive/30) spoken digit assignement
```

[ ]:
```
#read the all file names in the recordings folder given by us
#(if you get entire path, it is very useful in future)
#save those files names as list in "all_files"
all_files = []
label_list = []
for i in os.listdir('recordings'):
  name = i
  all_files.append(path+'/'+i)
  label = name.split('_')[0]
  label_list.append(label)
```

[ ]:
```
print('path:',all_files[1020],'--------','Label:',label_list[1020])
```

Create a dataframe(name=df_audio) with two columns(path, label).
You can get the label from the first letter of name.

Eg: 0_jackson_0 –> 0
0_jackson_43 –> 0

[ ]:
```python
#Create a dataframe(name=df_audio) with two columns(path, label).
#You can get the label from the first letter of name.
#Eg: 0_jackson_0 --> 0
#0_jackson_43 --> 0
data = { 'path': all_files,
        'label': label_list

}
df_audio = pd.DataFrame(data,columns=['path','label'])
```

[ ]:
```python

```

[ ]:
```python
#info
df_audio.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   path    2000 non-null   object
 1   label   2000 non-null   object
dtypes: object(2)
memory usage: 31.4+ KB
```

[ ]:
```python
from sklearn.utils import shuffle
df_audio = shuffle(df_audio, random_state=33)#don't change the random state
```

[ ]:
```python
df_audio.to_pickle('/content/drive/My Drive/30) spoken digit assignement/
↪df_audio.pkl')
```

[ ]:
```python
df_audio = pd.read_pickle('/content/drive/My Drive/30) spoken digit assignement/
↪df_audio.pkl')
```

[ ]:
```python
X = df_audio.path
Y = df_audio.label
```

[ ]:
```python
#split the data into train and validation and save in X_train, X_test, y_train,␣
↪y_test
#use stratify sampling
#use random state of 45
#use test size of 30%
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size=0.30,stratify=Y)
```

[ ]:
```python
sample_rate = 22050
def load_wav(x, get_duration=True):
    '''This return the array values of audio with sampling rate of 22050 and␣
↪Duration'''
    #loading the wav file with sampling rate of 22050
    samples, sample_rate = librosa.load(x, sr=22050)
    if get_duration:
        duration = librosa.get_duration(samples, sample_rate)
        return samples, duration
    else:
        return samples
```

[ ]:
```python
#use load_wav function that was written above to get every wave.
#save it in X_train_processed and X_test_processed
# X_train_processed/X_test_processed should be dataframes with two␣
↪columns(raw_data, duration) with same index of X_train/y_train
from tqdm.notebook import tqdm

X_train_data = []
X_test_data = []
X_train_processed = pd.DataFrame(columns = ["raw_data", "duration"])
X_test_processed = pd.DataFrame(columns = ["raw_data", "duration"])

for i in tqdm(X_train.index):
    load_X_train = load_wav(X_train[i])
    X_train_processed.loc[i] = load_X_train
    X_train_data.append(load_X_train)

for i in tqdm(X_test.index):
    load_X_test  = load_wav(X_test[i])
    X_test_processed.loc[i] = load_X_test
    X_test_data.append(load_X_test)
```

```
HBox(children=(FloatProgress(value=0.0, max=1400.0), HTML(value='')))
```

```
HBox(children=(FloatProgress(value=0.0, max=600.0), HTML(value='')))
```

[ ]:
```python
X_train_processed.head(-1)//////
```

[ ]:

	raw_data	duration
86	[-0.0007788757, -0.0044405614, -0.006944217, -…	0.399138
948	[-0.0005402197, -0.000797918, -0.0010040042, -…	0.418639

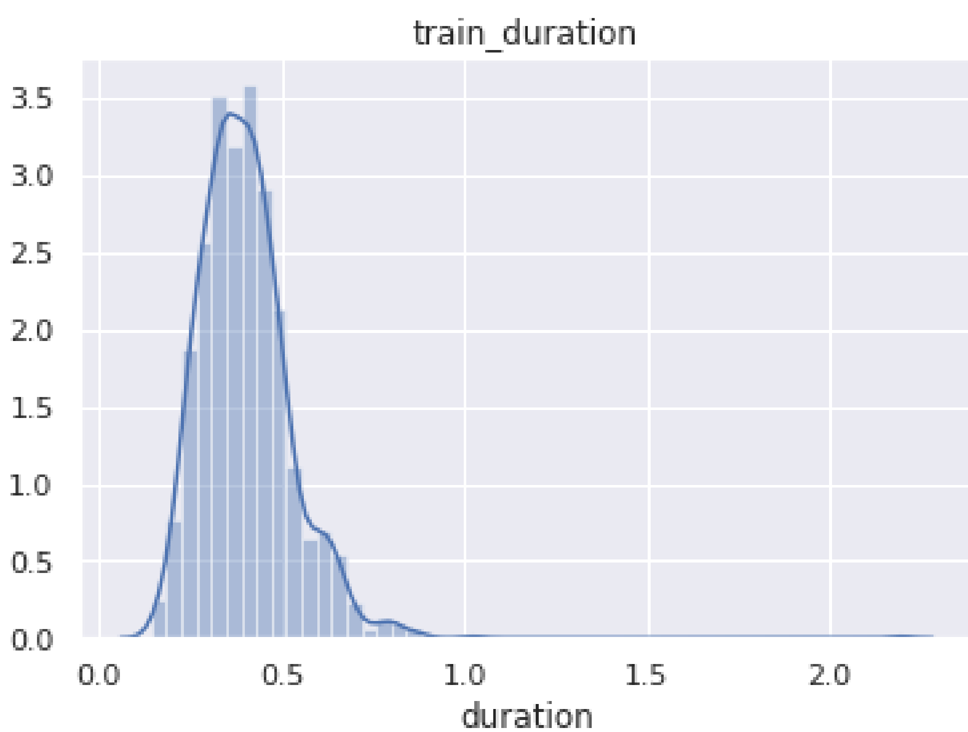
```
1105  [-0.0015830346, -0.0018493982, -0.0018068022, …  0.422766
1793  [0.00024614544, 0.00035170314, 0.00037052337, …  0.309252
920   [-0.00010898901, -8.4288484e-05, -8.509773e-05…  0.869161
…                                                 …        …
624   [-0.0052208863, 0.004611595, 0.009536311, 0.00…  0.439909
878   [-0.0064763697, -0.008053252, -0.0091387555, -…  0.413016
1821  [-0.016771669, -0.020051673, -0.018642116, -0…  0.474376
286   [-0.0075149927, -0.013656148, -0.017080104, -0…  0.292789
1154  [2.4389848e-05, -7.690136e-05, -0.00014365127,…  0.393787

[1399 rows x 2 columns]
```

[ ]:
```
X_train_processed.to_pickle('/content/drive/My Drive/30) spoken digit␣
↪assignement/X_train_processed.pkl')
```

[ ]:
```
X_test_processed.to_pickle('/content/drive/My Drive/30) spoken digit␣
↪assignement/X_test_processed.pkl')
```

[ ]:
```
#plot the histogram of the duration for train
sns.set()
sns.distplot(X_train_processed['duration']).set(title='train_duration')
```

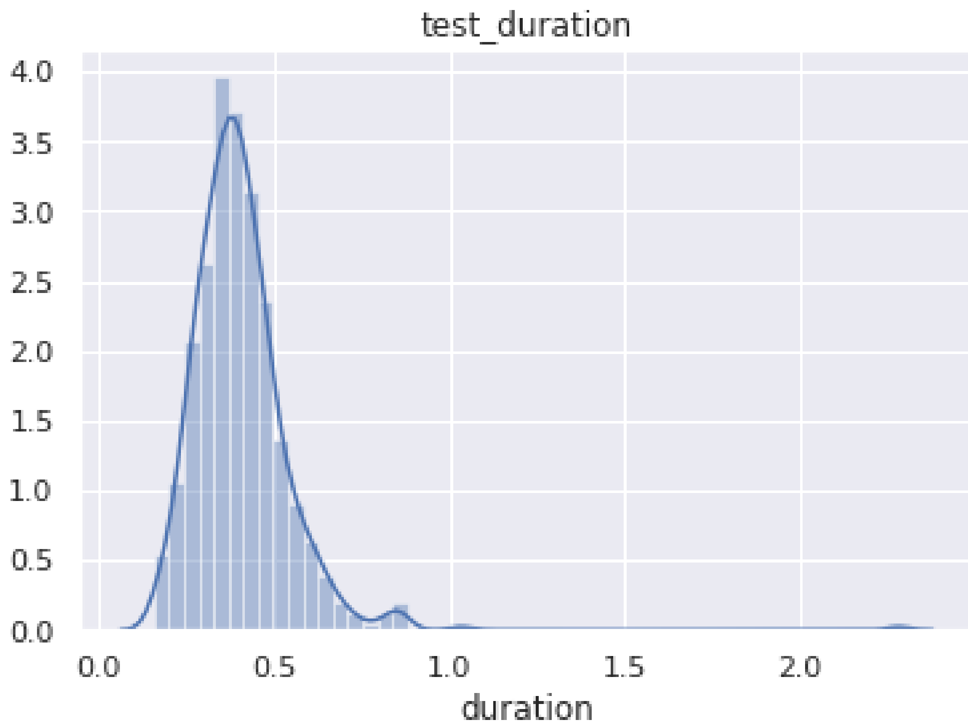
[ ]:
```
[Text(0.5, 1.0, 'train_duration')]
```



Here the maximum number of audio file has duration length 0.25 to 0.5

[ ]:
```
#plot the histogram of the duration for test
sns.set()
sns.distplot(X_test_processed['duration']).set(title='test_duration')
```

[ ]: [Text(0.5, 1.0, 'test_duration')]



Here also the maximum number of test_audio file has duration between 0.25 to 0.5

[ ]:
```
#print 0 to 100 percentile values with step size of 10 for train data duration.
for i in range(0,101,10):
  print(f'{i}th percentile is {np.percentile(X_train_processed.duration.
↪values,i)}')
```

```
0th percentile is 0.1435374149659864
10th percentile is 0.25824943310657594
20th percentile is 0.29960997732426303
30th percentile is 0.3310204081632653
40th percentile is 0.3581587301587302
50th percentile is 0.38988662131519275
60th percentile is 0.41941043083900226
70th percentile is 0.4481904761904762
80th percentile is 0.4849251700680272
```

```
90th percentile is 0.5554421768707483
100th percentile is 2.195918367346939
```

[ ]:
```
##print 90 to 100 percentile values with step size of 1.
for i in range(90,101,1):
  print(f'{i}th percentile is {np.percentile(X_train_processed.duration.
↪values,i)}')
```

```
90th percentile is 0.5554421768707483
91th percentile is 0.5739142857142857
92th percentile is 0.5829315192743765
93th percentile is 0.6025904761904762
94th percentile is 0.6150358276643991
95th percentile is 0.6253424036281179
96th percentile is 0.6430421768707483
97th percentile is 0.6584195011337869
98th percentile is 0.6838820861678004
99th percentile is 0.7831392290249433
100th percentile is 2.195918367346939
```

[ ]:
```
## as discussed above, Pad with Zero if length of sequence is less than 17640␣
↪else Truncate the number.
## save in the X_train_pad_seq, X_test_pad_seq
## also Create masking vector X_train_mask, X_test_mask

## all the X_train_pad_seq, X_test_pad_seq, X_train_mask, X_test_mask will be␣
↪numpy arrays mask vector dtype must be bool.
```

[3]:
```
def pad_mask(data,max_length):

    pad_data,mask_data=[],[]
    if(len(data)<max_length):
        mask_data=[1]*len(data)
        pad_data = [0]*(max_length - len(data))
        data = list(data)
        data.extend(pad_data)
        mask_data.extend(pad_data)
    else:
        data = list(data)
        data = data[:max_length]
        mask_data = [1]*max_length
    return data,mask_data
```

[ ]:
```
 ## X_train_pad_seq , X_train_mask
X_train_pad_seq,X_train_mask = [],[]

for x in tqdm(X_train_processed['raw_data']):
```

```
    pad_data,mask_data=pad_mask(x,17640)
    X_train_pad_seq.append(pad_data)
    X_train_mask.append(mask_data)

X_train_pad_seq = np.array(X_train_pad_seq)
X_train_mask = np.array(X_train_mask)
```

```
HBox(children=(FloatProgress(value=0.0, max=1400.0), HTML(value='')))
```

[ ]:
```
 ## X_test_pad_seq , X_test_mask

X_test_pad_seq,X_test_mask = [],[]

for x in tqdm(X_test_processed['raw_data']):
    pad_data,mask_data=pad_mask(x,17640)
    X_test_pad_seq.append(pad_data)
    X_test_mask.append(mask_data)

X_test_pad_seq = np.array(X_test_pad_seq)
X_test_mask = np.array(X_test_mask)
```

```
HBox(children=(FloatProgress(value=0.0, max=600.0), HTML(value='')))
```

[ ]:
```
X_train_mask = X_train_mask.astype('bool')
X_test_mask = X_test_mask.astype('bool')
```

0.0.1 1. Giving Raw data directly.

[4]:
```
from tensorflow.keras.layers import Input, LSTM, Dense,concatenate
from tensorflow.keras.models import Model


from tensorflow.keras.callbacks import TensorBoard , EarlyStopping ,␣
↪ReduceLROnPlateau
import tensorflow as tf
```

[ ]:
```
## as discussed above, please write the LSTM
input_pad = Input(shape=(max_length,1)) ## Padded input
input_mask = Input(shape=(max_length),dtype='bool') ## Masked input
l = tf.keras.layers.LSTM(50)(input_pad,mask=input_mask)
l = tf.keras.layers.Dense(50,kernel_regularizer=tf.keras.regularizers.l2())(l)
```

```
l = tf.keras.layers.Dense(10,activation='softmax')(l)

model = Model([input_pad,input_mask],l)
model.summary()
```

```
Model: "functional_3"
__________________________________________________________________________________
________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
================
input_5 (InputLayer)            [(None, 17640, 1)]   0

__________________________________________________________________________________
________________
input_6 (InputLayer)            [(None, 17640)]      0

__________________________________________________________________________________
________________
lstm_2 (LSTM)                   (None, 50)           10400       input_5[0][0]
                                                                 input_6[0][0]

__________________________________________________________________________________
________________
dense_3 (Dense)                 (None, 50)           2550        lstm_2[0][0]

__________________________________________________________________________________
________________
dense_4 (Dense)                 (None, 10)           510         dense_3[0][0]
==================================================================================
================
Total params: 13,460
Trainable params: 13,460
Non-trainable params: 0
__________________________________________________________________________________
________________
```

[5]:
```
from sklearn.metrics import f1_score

def f1_score_micro(y_true,y_pred):
     return f1_score(y_true,y_pred,average = 'micro')

def micro_f1(y_true, y_proba):
    y_pred = tf.math.argmax(y_proba,axis=1)
    return tf.py_function(f1_score_micro,(y_true,y_pred),tf.double)
```

[ ]:
```
#train your model

!mkdir '/content/drive/My Drive/30) spoken digit assignement/logs'
log_dir = '/content/drive/My Drive/30) spoken digit assignement/logs'
tensorboard = TensorBoard(log_dir,histogram_freq=1)
```

```
reduce_lr_plateau = ReduceLROnPlateau(monitor = 'loss',verbose=1)

callbacks = [tensorboard,reduce_lr_plateau]

model.compile(optimizer='Adam',loss =␣
 ↪'sparse_categorical_crossentropy',metrics=[micro_f1])
```

```
mkdir: cannot create directory '/content/drive/My Drive/30) spoken digit
assignement/logs': File exists
```

[ ]:
```
X_train_pad_seq.shape, X_train_mask.shape
```

[ ]:
```
((1400, 17640), (1400, 17640))
```

[ ]:
```
y_train = y_train.astype('int')
y_test = y_test.astype('int')
```

[ ]:
```
model.fit([X_train_pad_seq,X_train_mask],y_train,
          epochs=10,verbose=1,
          validation_data=([X_test_pad_seq,X_test_mask],y_test),
          callbacks= callbacks)
```

```
Epoch 1/10
 1/44 […] - ETA: 0s - loss: 2.7900 - micro_f1:
0.0938WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow/python/ops/summary_ops_v2.py:1277: stop (from
tensorflow.python.eager.profiler) is deprecated and will be removed after
2020-07-01.
Instructions for updating:
use `tf.profiler.experimental.stop` instead.
 2/44 [>…] - ETA: 1:10 - loss: 2.7878 - micro_f1:
0.0469WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared
to the batch time (batch time: 0.7858s vs `on_train_batch_end` time: 2.5598s).
Check your callbacks.
44/44 [==============================] - 36s 824ms/step - loss: 2.6784 -
micro_f1: 0.0907 - val_loss: 2.5756 - val_micro_f1: 0.0992
Epoch 2/10
44/44 [==============================] - 33s 742ms/step - loss: 2.5095 -
micro_f1: 0.0812 - val_loss: 2.4500 - val_micro_f1: 0.0998
Epoch 3/10
44/44 [==============================] - 33s 750ms/step - loss: 2.4130 -
micro_f1: 0.0859 - val_loss: 2.3796 - val_micro_f1: 0.0927
Epoch 4/10
44/44 [==============================] - 33s 755ms/step - loss: 2.3596 -
micro_f1: 0.0933 - val_loss: 2.3412 - val_micro_f1: 0.0992
Epoch 5/10
```

```
44/44 [==============================] - 33s 745ms/step - loss: 2.3310 -
micro_f1: 0.0909 - val_loss: 2.3211 - val_micro_f1: 0.1003
Epoch 6/10
44/44 [==============================] - 32s 736ms/step - loss: 2.3161 -
micro_f1: 0.0961 - val_loss: 2.3110 - val_micro_f1: 0.0998
Epoch 7/10
44/44 [==============================] - 33s 743ms/step - loss: 2.3089 -
micro_f1: 0.0895 - val_loss: 2.3063 - val_micro_f1: 0.0998
Epoch 8/10
44/44 [==============================] - 33s 759ms/step - loss: 2.3057 -
micro_f1: 0.0874 - val_loss: 2.3041 - val_micro_f1: 0.1014
Epoch 9/10
44/44 [==============================] - 33s 745ms/step - loss: 2.3041 -
micro_f1: 0.0798 - val_loss: 2.3032 - val_micro_f1: 0.1003
Epoch 10/10
44/44 [==============================] - 32s 737ms/step - loss: 2.3034 -
micro_f1: 0.0817 - val_loss: 2.3028 - val_micro_f1: 0.0992
```

[ ]:
```
<tensorflow.python.keras.callbacks.History at 0x7f9f9b093908>
```

[2]:
```
%load_ext tensorboard
```

[ ]:
```
%tensorboard --logdir '/content/drive/My Drive/30) spoken digit assignement/
 ↪logs'
```

```
<IPython.core.display.Javascript object>
```

Observation

Using Raw data is not a good option because loss and micro-f1 was not not imporving

0.0.2 2. Converting into spectrogram and giving spectrogram data as input

[6]:
```
def convert_to_spectrogram(raw_data):
    '''converting to spectrogram'''
    spectrum = librosa.feature.melspectrogram(y=raw_data, sr=22050, n_mels=64)
    logmel_spectrum = librosa.power_to_db(S=spectrum, ref=np.max)
    return logmel_spectrum
```

[ ]:
```
##use convert_to_spectrogram and convert every raw sequence in X_train_pad_seq␣
 ↪and X_test_pad-seq.
## save those all in the X_train_spectrogram and X_test_spectrogram ( These two␣
 ↪arrays must be numpy arrays)
```

[ ]:
```
X_train_spectrogram = []
X_test_spectrogram = []
X_train_pad_seq = X_train_pad_seq.reshape(1400,17640)
```

```
X_test_pad_seq = X_test_pad_seq.reshape(600,17640)

for i in range(X_train_pad_seq.shape[0]):
    X_train_spectrogram1 = convert_to_spectrogram(X_train_pad_seq[i])
    X_train_spectrogram.append(X_train_spectrogram1)

for i in range(X_test_pad_seq.shape[0]):
    X_test_spectrogram1 = convert_to_spectrogram(X_test_pad_seq[i])
    X_test_spectrogram.append(X_test_spectrogram1)

X_train_spectrogram = np.array(X_train_spectrogram)
X_test_spectrogram = np.array(X_test_spectrogram)
```

[ ]:
```
X_train_spectrogram.shape
```

[ ]:
```
(1400, 64, 35)
```

[ ]:
```
input = Input(shape = (64,35,))
l1 = tf.keras.layers.LSTM(25,return_sequences=True)(input)
l1 = tf.keras.layers.GlobalAveragePooling1D()(l1)

output = tf.keras.layers.Dense(10,activation= 'softmax')(l1)

model = Model(inputs =input,outputs = output)

model.summary()
```

```
Model: "functional_21"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_15 (InputLayer)        [(None, 64, 35)]          0
_________________________________________________________________
lstm_11 (LSTM)               (None, 64, 25)            6100
_________________________________________________________________
global_average_pooling1d_8 ( (None, 25)                0
_________________________________________________________________
dense_14 (Dense)             (None, 10)                260
=================================================================
Total params: 6,360
Trainable params: 6,360
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
from sklearn.metrics import f1_score

def f1_score_micro(y_true,y_pred):
```

```
    return f1_score(y_true,y_pred,average = 'micro')

def micro_f1(y_true, y_proba):
    y_pred = tf.math.argmax(y_proba,axis=1)
    return tf.py_function(f1_score_micro,(y_true,y_pred),tf.double)
```

[ ]:
```
#train your model

!mkdir '/content/drive/My Drive/30) spoken digit assignement/log2'
log_dir = '/content/drive/My Drive/30) spoken digit assignement/log2'
tensorboard = TensorBoard(log_dir,histogram_freq=True)

reducelr = ReduceLROnPlateau(monitor='loss', factor=0.2, patience=2, min_lr=0.
↪0001, verbose = 1)


callbacks = [tensorboard,reducelr]



model.compile(optimizer=tf.optimizers.Adam(),loss =␣
↪'sparse_categorical_crossentropy',metrics=[micro_f1] )
model.fit(X_train_spectrogram,y_train,
          validation_data=(X_test_spectrogram,y_test),
          epochs=50,
          callbacks=callbacks)
```

```
Epoch 1/50
 2/44 [>…] - ETA: 3s - loss: 1.4242 - micro_f1:
0.5938WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared
to the batch time (batch time: 0.0141s vs `on_train_batch_end` time: 0.1303s).
Check your callbacks.
44/44 [==============================] - 1s 22ms/step - loss: 1.4753 - micro_f1:
0.5071 - val_loss: 1.5346 - val_micro_f1: 0.4792
Epoch 2/50
44/44 [==============================] - 0s 11ms/step - loss: 1.4323 - micro_f1:
0.5367 - val_loss: 1.5232 - val_micro_f1: 0.4616
Epoch 3/50
44/44 [==============================] - 0s 11ms/step - loss: 1.4249 - micro_f1:
0.5315 - val_loss: 1.5643 - val_micro_f1: 0.4682
Epoch 4/50
44/44 [==============================] - 0s 11ms/step - loss: 1.4270 - micro_f1:
0.5355 - val_loss: 1.5250 - val_micro_f1: 0.4934
Epoch 5/50
44/44 [==============================] - 0s 11ms/step - loss: 1.4067 - micro_f1:
0.5518 - val_loss: 1.4973 - val_micro_f1: 0.4962
Epoch 6/50
```

```
44/44 [==============================] - 0s 10ms/step - loss: 1.4036 - micro_f1:
0.5514 - val_loss: 1.5411 - val_micro_f1: 0.4836
Epoch 7/50
44/44 [==============================] - 0s 10ms/step - loss: 1.4322 - micro_f1:
0.5414 - val_loss: 1.5230 - val_micro_f1: 0.4923
Epoch 8/50
40/44 [==========================>…] - ETA: 0s - loss: 1.4159 - micro_f1:
0.5437
Epoch 00008: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.
44/44 [==============================] - 0s 11ms/step - loss: 1.4087 - micro_f1:
0.5523 - val_loss: 1.4872 - val_micro_f1: 0.5164
Epoch 9/50
44/44 [==============================] - 1s 12ms/step - loss: 1.3856 - micro_f1:
0.5658 - val_loss: 1.4848 - val_micro_f1: 0.5033
Epoch 10/50
44/44 [==============================] - 1s 12ms/step - loss: 1.3756 - micro_f1:
0.5760 - val_loss: 1.4845 - val_micro_f1: 0.5093
Epoch 11/50
44/44 [==============================] - 0s 10ms/step - loss: 1.3731 - micro_f1:
0.5753 - val_loss: 1.4954 - val_micro_f1: 0.5066
Epoch 12/50
44/44 [==============================] - 0s 11ms/step - loss: 1.3710 - micro_f1:
0.5750 - val_loss: 1.4859 - val_micro_f1: 0.5044
Epoch 13/50
44/44 [==============================] - 1s 12ms/step - loss: 1.3710 - micro_f1:
0.5769 - val_loss: 1.4743 - val_micro_f1: 0.5280
Epoch 14/50
44/44 [==============================] - 1s 12ms/step - loss: 1.3652 - micro_f1:
0.5675 - val_loss: 1.4749 - val_micro_f1: 0.5203
Epoch 15/50
44/44 [==============================] - 0s 11ms/step - loss: 1.3646 - micro_f1:
0.5734 - val_loss: 1.4795 - val_micro_f1: 0.5137
Epoch 16/50
44/44 [==============================] - 0s 11ms/step - loss: 1.3631 - micro_f1:
0.5786 - val_loss: 1.4949 - val_micro_f1: 0.5011
Epoch 17/50
44/44 [==============================] - 1s 12ms/step - loss: 1.3619 - micro_f1:
0.5715 - val_loss: 1.4800 - val_micro_f1: 0.5049
Epoch 18/50
44/44 [==============================] - 0s 11ms/step - loss: 1.3615 - micro_f1:
0.5758 - val_loss: 1.4754 - val_micro_f1: 0.5197
Epoch 19/50
44/44 [==============================] - 0s 10ms/step - loss: 1.3600 - micro_f1:
0.5748 - val_loss: 1.4883 - val_micro_f1: 0.5005
Epoch 20/50
44/44 [==============================] - 0s 11ms/step - loss: 1.3559 - micro_f1:
0.5798 - val_loss: 1.4642 - val_micro_f1: 0.5263
Epoch 21/50
```

```
44/44 [=============================] - 0s 11ms/step - loss: 1.3556 - micro_f1:
0.5703 - val_loss: 1.4797 - val_micro_f1: 0.5104
Epoch 22/50
44/44 [=============================] - 0s 11ms/step - loss: 1.3540 - micro_f1:
0.5769 - val_loss: 1.4631 - val_micro_f1: 0.5274
Epoch 23/50
44/44 [=============================] - 1s 12ms/step - loss: 1.3582 - micro_f1:
0.5760 - val_loss: 1.4690 - val_micro_f1: 0.5280
Epoch 24/50
44/44 [=============================] - 1s 13ms/step - loss: 1.3517 - micro_f1:
0.5843 - val_loss: 1.4704 - val_micro_f1: 0.5071
Epoch 25/50
44/44 [=============================] - 1s 12ms/step - loss: 1.3473 - micro_f1:
0.5741 - val_loss: 1.4606 - val_micro_f1: 0.5143
Epoch 26/50
44/44 [=============================] - 0s 11ms/step - loss: 1.3475 - micro_f1:
0.5831 - val_loss: 1.4701 - val_micro_f1: 0.5274
Epoch 27/50
44/44 [=============================] - 1s 12ms/step - loss: 1.3448 - micro_f1:
0.5857 - val_loss: 1.4609 - val_micro_f1: 0.5252
Epoch 28/50
44/44 [=============================] - 0s 11ms/step - loss: 1.3401 - micro_f1:
0.5786 - val_loss: 1.4740 - val_micro_f1: 0.4962
Epoch 29/50
44/44 [=============================] - 0s 11ms/step - loss: 1.3365 - micro_f1:
0.5862 - val_loss: 1.4616 - val_micro_f1: 0.5225
Epoch 30/50
44/44 [=============================] - 0s 11ms/step - loss: 1.3342 - micro_f1:
0.5836 - val_loss: 1.4593 - val_micro_f1: 0.5252
Epoch 31/50
44/44 [=============================] - 1s 12ms/step - loss: 1.3315 - micro_f1:
0.5902 - val_loss: 1.4602 - val_micro_f1: 0.5302
Epoch 32/50
44/44 [=============================] - 1s 12ms/step - loss: 1.3264 - micro_f1:
0.5850 - val_loss: 1.4492 - val_micro_f1: 0.5241
Epoch 33/50
44/44 [=============================] - 0s 11ms/step - loss: 1.3321 - micro_f1:
0.5938 - val_loss: 1.4583 - val_micro_f1: 0.5164
Epoch 34/50
44/44 [=============================] - 0s 11ms/step - loss: 1.3241 - micro_f1:
0.5982 - val_loss: 1.4574 - val_micro_f1: 0.5159
Epoch 35/50
44/44 [=============================] - 0s 10ms/step - loss: 1.3276 - micro_f1:
0.5964 - val_loss: 1.4465 - val_micro_f1: 0.5291
Epoch 36/50
39/44 [========================>…] - ETA: 0s - loss: 1.3128 - micro_f1:
0.5970
Epoch 00036: ReduceLROnPlateau reducing learning rate to 0.0001.
```

```
44/44 [==============================] - 0s 11ms/step - loss: 1.3243 - micro_f1:
0.5902 - val_loss: 1.4448 - val_micro_f1: 0.5197
Epoch 37/50
44/44 [==============================] - 1s 12ms/step - loss: 1.3173 - micro_f1:
0.5907 - val_loss: 1.4461 - val_micro_f1: 0.5367
Epoch 38/50
44/44 [==============================] - 0s 11ms/step - loss: 1.3147 - micro_f1:
0.5982 - val_loss: 1.4464 - val_micro_f1: 0.5400
Epoch 39/50
44/44 [==============================] - 1s 12ms/step - loss: 1.3151 - micro_f1:
0.5930 - val_loss: 1.4410 - val_micro_f1: 0.5395
Epoch 40/50
44/44 [==============================] - 1s 12ms/step - loss: 1.3129 - micro_f1:
0.5971 - val_loss: 1.4489 - val_micro_f1: 0.5334
Epoch 41/50
44/44 [==============================] - 1s 12ms/step - loss: 1.3122 - micro_f1:
0.5904 - val_loss: 1.4517 - val_micro_f1: 0.5367
Epoch 42/50
44/44 [==============================] - 0s 11ms/step - loss: 1.3122 - micro_f1:
0.5930 - val_loss: 1.4434 - val_micro_f1: 0.5318
Epoch 43/50
44/44 [==============================] - 1s 13ms/step - loss: 1.3090 - micro_f1:
0.5985 - val_loss: 1.4547 - val_micro_f1: 0.5269
Epoch 44/50
44/44 [==============================] - 0s 11ms/step - loss: 1.3098 - micro_f1:
0.6009 - val_loss: 1.4398 - val_micro_f1: 0.5400
Epoch 45/50
44/44 [==============================] - 0s 10ms/step - loss: 1.3098 - micro_f1:
0.5961 - val_loss: 1.4465 - val_micro_f1: 0.5334
Epoch 46/50
44/44 [==============================] - 0s 10ms/step - loss: 1.3067 - micro_f1:
0.5992 - val_loss: 1.4457 - val_micro_f1: 0.5236
Epoch 47/50
44/44 [==============================] - 0s 11ms/step - loss: 1.3056 - micro_f1:
0.5987 - val_loss: 1.4437 - val_micro_f1: 0.5334
Epoch 48/50
44/44 [==============================] - 0s 11ms/step - loss: 1.3048 - micro_f1:
0.5978 - val_loss: 1.4408 - val_micro_f1: 0.5345
Epoch 49/50
44/44 [==============================] - 0s 11ms/step - loss: 1.3046 - micro_f1:
0.6027 - val_loss: 1.4371 - val_micro_f1: 0.5395
Epoch 50/50
44/44 [==============================] - 0s 11ms/step - loss: 1.2983 - micro_f1:
0.5973 - val_loss: 1.4314 - val_micro_f1: 0.5099
```

```
[ ]: <tensorflow.python.keras.callbacks.History at 0x7f9ef536def0>
```

```
[ ]: !kill 30256
```

```
[ ]: %tensorboard --logdir 'log2'
```

```
Reusing TensorBoard on port 6006 (pid 30739), started 0:01:25 ago. (Use '!kill 30739' to kill i
```

```
<IPython.core.display.Javascript object>
```

```
[ ]: ## generating augmented data.
def generate_augmented_data(file_path):
    augmented_data = []
    samples = load_wav(file_path,get_duration=False)
    for time_value in [0.7, 1, 1.3]:
        for pitch_value in [-1, 0, 1]:
            time_stretch_data = librosa.effects.time_stretch(samples,␣
 ↪rate=time_value)
            final_data = librosa.effects.pitch_shift(time_stretch_data,␣
 ↪sr=sample_rate, n_steps=pitch_value)
            augmented_data.append(final_data)
    return augmented_data
```

```
[ ]: temp_path = df_audio.iloc[1999].path
print(temp_path)
aug_temp = generate_augmented_data(temp_path)
len(aug_temp)
```

```
/content/drive/My Drive/30) spoken digit assignement/recordings/0_jackson_49.wav
```

```
[ ]: 9
```

As discussed above, for one data point, we will get 9 augmented data points.
We have 2000 data points(train plus test) so, after augmentation we will get 18000 (train - 12600, test - 5400).

do the above steps i.e training with raw data and spectrogram data with augmentation.

```
[ ]: X_train_processed.columns
```

```
[ ]: Index(['raw_data', 'duration'], dtype='object')
```

```
[ ]: augmented_train_data = []
augmented_test_data = []
train_labels = []
test_labels = []

for i, j in tqdm(zip(X_train, X_train.index)):
    augmented_data = generate_augmented_data(i)
```

```
    for m in augmented_data:
        augmented_train_data.append(m)
        train_labels.append(j)

for i,j in tqdm(zip(X_test, X_test.index)):
    augmented_data = generate_augmented_data(i)
    for n in augmented_data:
        augmented_test_data.append(n)
        test_labels.append(j)

augmented_train_data = np.array(augmented_train_data)
augmented_test_data = np.array(augmented_test_data)
```

```
HBox(children=(FloatProgress(value=1.0, bar_style='info', max=1.0), HTML(value='')))
```

```
HBox(children=(FloatProgress(value=1.0, bar_style='info', max=1.0), HTML(value='')))
```

[ ]:

```
target_train = []
target_test = []

for i, j in zip(y_train.index, y_train):
    for o in train_labels:
        if o == i:
            target_train.append(j)

for i, j in zip(y_test.index, y_test):
    for o in test_labels:
        if o == i:
            target_test.append(j)

print(len(target_train))
print(len(target_test))
```

```
12600
5400
```

[ ]:

```
target_train = np.array(target_train)
target_test = np.array(target_test)
```

[9]:

```
augmented_train_data = np.load('/content/drive/My Drive/30) spoken digit␣
 ↪assignement/augmented_train_data.npy',allow_pickle=True)
augmented_test_data = np.load('/content/drive/My Drive/30) spoken digit␣
 ↪assignement/augmented_test_data.npy',allow_pickle=True)
```

```
target_train = np.load('/content/drive/My Drive/30) spoken digit assignement/
↪target_train.npy',allow_pickle=True)
target_test = np.load('/content/drive/My Drive/30) spoken digit assignement/
↪target_test.npy',allow_pickle=True)
```

[10]:
```
max_length = 17640
```

[12]:
```
X_train_pad_seq,X_train_mask = [],[]

for x in (augmented_train_data):
    pad_data,mask_data=pad_mask(x,max_length)
    X_train_pad_seq.append(pad_data)
    X_train_mask.append(mask_data)

X_train_pad_seq = np.array(X_train_pad_seq)
X_train_mask = np.array(X_train_mask)
```

[13]:
```
X_test_pad_seq,X_test_mask = [],[]

for x in (augmented_test_data):
    pad_data,mask_data=pad_mask(x,max_length)
    X_test_pad_seq.append(pad_data)
    X_test_mask.append(mask_data)

X_test_pad_seq = np.array(X_test_pad_seq)
X_test_mask = np.array(X_test_mask)
```

[ ]:
```
X_train_mask = X_train_mask.astype('bool')
X_test_mask = X_test_mask.astype('bool')
```

[ ]:
```
input_pad = Input(shape=(max_length,1)) ## Padded input
input_mask = Input(shape=(max_length),dtype='bool') ## Masked input
l = tf.keras.layers.LSTM(50)(input_pad,mask=input_mask)
l = Dense(50,activation='relu')(l)

l = tf.keras.layers.Dense(10,activation='softmax')(l)

model3 = Model([input_pad,input_mask],l)
model3.summary()
```

```
Model: "model"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape          Param #     Connected to
================================================================================
==================
input_1 (InputLayer)            [(None, 17640, 1)]    0
```

```
----------------------------------------------------------------------------------
------------------
input_2 (InputLayer)            [(None, 17640)]      0

----------------------------------------------------------------------------------
------------------
lstm (LSTM)                     (None, 50)           10400       input_1[0][0]

----------------------------------------------------------------------------------
------------------
dense (Dense)                   (None, 50)           2550        lstm[0][0]

----------------------------------------------------------------------------------
------------------
dense_1 (Dense)                 (None, 10)           510         dense[0][0]
==================================================================================
==================
Total params: 13,460
Trainable params: 13,460
Non-trainable params: 0

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

[ ]:
```
## Callbacks and comipiling

!mkdir 'log3_aug'
log_dir = 'log3_aug'
tensorboard = TensorBoard(log_dir=log_dir,histogram_freq=1)


reduce_lr_plateau = ReduceLROnPlateau(monitor = 'loss',verbose=2,mode = 'max')

callbacks = [tensorboard,reduce_lr_plateau]

model3.compile(optimizer = tf.keras.optimizers.Adam(0.001),
               loss = 'sparse_categorical_crossentropy',metrics = [micro_f1])
```

[ ]:
```
target_train = target_train.astype('int')
target_test = target_test.astype('int')
```

[ ]:
```
X_train_pad_seq = X_train_pad_seq.reshape(12600,17640,1)
X_test_pad_seq = X_test_pad_seq.reshape(5400,17640,1)
X_train_pad_seq.shape
```

[ ]:
```
(12600, 17640, 1)
```

[ ]:
```
model3.fit([X_train_pad_seq,X_train_mask],target_train,epochs=5,validation_data␣
↪= ([X_test_pad_seq,X_test_mask],target_test),callbacks= callbacks)
```

```
Train on 12600 samples, validate on 5400 samples
Epoch 1/5
```

```
12600/12600 [==============================] - 240s 19ms/sample - loss: 2.3029 -
micro_f1: 0.0953 - val_loss: 2.3026 - val_micro_f1: 0.1002
Epoch 2/5
12600/12600 [==============================] - 242s 19ms/sample - loss: 2.3028 -
micro_f1: 0.0981 - val_loss: 2.3026 - val_micro_f1: 0.0999
Epoch 3/5
12600/12600 [==============================] - 244s 19ms/sample - loss: 2.3028 -
micro_f1: 0.0962 - val_loss: 2.3026 - val_micro_f1: 0.1002
Epoch 4/5
12600/12600 [==============================] - 249s 20ms/sample - loss: 2.3028 -
micro_f1: 0.0925 - val_loss: 2.3026 - val_micro_f1: 0.0999
Epoch 5/5
12600/12600 [==============================] - 243s 19ms/sample - loss: 2.3029 -
micro_f1: 0.0975 - val_loss: 2.3026 - val_micro_f1: 0.1000
```

[ ]:
```
<tensorflow.python.keras.callbacks.History at 0x20e4f477048>
```

Model 4

Converting the augmented data into spectrogram and giving spectrogram data as input

[7]:
```python
import tqdm as tqdm
```

[14]:
```python
X_train_spectrogram = []
X_test_spectrogram = []
X_train_pad_seq = X_train_pad_seq.reshape(12600,17640)
X_test_pad_seq = X_test_pad_seq.reshape(5400,17640)

for i in range(X_train_pad_seq.shape[0]):
    X_train_spectrogram1 = convert_to_spectrogram(X_train_pad_seq[i])
    X_train_spectrogram.append(X_train_spectrogram1)

for i in range(X_test_pad_seq.shape[0]):
    X_test_spectrogram1 = convert_to_spectrogram(X_test_pad_seq[i])
    X_test_spectrogram.append(X_test_spectrogram1)

X_train_spectrogram = np.array(X_train_spectrogram)
X_test_spectrogram = np.array(X_test_spectrogram)
```

[15]:
```python
input_layer= Input(shape=(64,35,),name='input_layer')

x=LSTM(25,return_sequences=True)(input_layer)
x= tf.keras.layers.GlobalAveragePooling1D()(x)
output = Dense(10,activation='softmax', kernel_regularizer=tf.keras.
 ↪regularizers.l2(0.001))(x)
model4 =Model(inputs=input_layer,outputs=output)
```

[16]:
```python
model4.summary()
```

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (InputLayer)     [(None, 64, 35)]          0
_________________________________________________________________
lstm (LSTM)                  (None, 64, 25)            6100
_________________________________________________________________
global_average_pooling1d (Gl (None, 25)                0
_________________________________________________________________
dense (Dense)                (None, 10)                260
=================================================================
Total params: 6,360
Trainable params: 6,360
Non-trainable params: 0
_________________________________________________________________
```

[25]:

```
!mkdir 'log4_aug'
log_dir = 'log4_aug'
tensorboard = TensorBoard(log_dir=log_dir,histogram_freq=1)



callbacks = [tensorboard]

model4.compile(optimizer = tf.keras.optimizers.Adam(),
               loss = 'sparse_categorical_crossentropy',metrics = [micro_f1])
```

```
mkdir: cannot create directory ‘log4_aug’: File exists
```

[23]:

```
target_train = target_train.astype('int')
target_test = target_test.astype('int')
```

[ ]:

```
model4.fit(X_train_spectrogram,target_train,steps_per_epoch=12600,␣
 ↪epochs=50,validation_data = (X_test_spectrogram,target_test),callbacks=␣
 ↪callbacks)
```

```
Epoch 1/50
    2/12600 […] - ETA: 7:13 - loss: 2.1623 -
micro_f1: 0.0000e+00WARNING:tensorflow:Callbacks method `on_train_batch_end` is
slow compared to the batch time (batch time: 0.0108s vs `on_train_batch_end`
time: 0.0573s). Check your callbacks.
12600/12600 [==============================] - 85s 7ms/step - loss: 2.3064 -
micro_f1: 0.1071 - val_loss: 2.3207 - val_micro_f1: 0.0982
Epoch 2/50
12600/12600 [==============================] - 85s 7ms/step - loss: 2.3057 -
micro_f1: 0.1089 - val_loss: 2.3267 - val_micro_f1: 0.0926
Epoch 3/50
```

```
12600/12600 [==============================] - 84s 7ms/step - loss: 2.3025 -
micro_f1: 0.1153 - val_loss: 2.3196 - val_micro_f1: 0.1028
Epoch 4/50
12600/12600 [==============================] - 85s 7ms/step - loss: 2.3033 -
micro_f1: 0.1106 - val_loss: 2.3147 - val_micro_f1: 0.0974
Epoch 5/50
12600/12600 [==============================] - 84s 7ms/step - loss: 2.3030 -
micro_f1: 0.1088 - val_loss: 2.3246 - val_micro_f1: 0.0975
Epoch 6/50
12600/12600 [==============================] - 85s 7ms/step - loss: 2.3028 -
micro_f1: 0.1099 - val_loss: 2.3187 - val_micro_f1: 0.0912
Epoch 7/50
12600/12600 [==============================] - 84s 7ms/step - loss: 2.3032 -
micro_f1: 0.1123 - val_loss: 2.3169 - val_micro_f1: 0.0913
Epoch 8/50
12600/12600 [==============================] - 85s 7ms/step - loss: 2.3034 -
micro_f1: 0.1145 - val_loss: 2.3168 - val_micro_f1: 0.0982
Epoch 9/50
12600/12600 [==============================] - 87s 7ms/step - loss: 2.3016 -
micro_f1: 0.1125 - val_loss: 2.3140 - val_micro_f1: 0.1017
Epoch 10/50
12600/12600 [==============================] - 86s 7ms/step - loss: 2.3006 -
micro_f1: 0.1147 - val_loss: 2.3163 - val_micro_f1: 0.0906
Epoch 11/50
12600/12600 [==============================] - 86s 7ms/step - loss: 2.3012 -
micro_f1: 0.1087 - val_loss: 2.3273 - val_micro_f1: 0.1043
Epoch 12/50
12600/12600 [==============================] - 85s 7ms/step - loss: 2.3005 -
micro_f1: 0.1114 - val_loss: 2.3161 - val_micro_f1: 0.0976
Epoch 13/50
12600/12600 [==============================] - 86s 7ms/step - loss: 2.3018 -
micro_f1: 0.1179 - val_loss: 2.3159 - val_micro_f1: 0.0934
Epoch 14/50
12600/12600 [==============================] - 84s 7ms/step - loss: 2.3000 -
micro_f1: 0.1119 - val_loss: 2.3133 - val_micro_f1: 0.0862
Epoch 15/50
12600/12600 [==============================] - 84s 7ms/step - loss: 2.2995 -
micro_f1: 0.1148 - val_loss: 2.3166 - val_micro_f1: 0.0976
Epoch 16/50
12600/12600 [==============================] - 83s 7ms/step - loss: 2.2987 -
micro_f1: 0.1144 - val_loss: 2.3168 - val_micro_f1: 0.0969
Epoch 17/50
12600/12600 [==============================] - 83s 7ms/step - loss: 2.2984 -
micro_f1: 0.1198 - val_loss: 2.3178 - val_micro_f1: 0.1063
Epoch 18/50
12600/12600 [==============================] - 87s 7ms/step - loss: 2.2995 -
micro_f1: 0.1171 - val_loss: 2.3178 - val_micro_f1: 0.0957
Epoch 19/50
```

```
12600/12600 [==============================] - 84s 7ms/step - loss: 2.3000 -
micro_f1: 0.1173 - val_loss: 2.3400 - val_micro_f1: 0.1002
Epoch 20/50
12600/12600 [==============================] - 84s 7ms/step - loss: 2.2994 -
micro_f1: 0.1161 - val_loss: 2.3141 - val_micro_f1: 0.0965
Epoch 21/50
12600/12600 [==============================] - 85s 7ms/step - loss: 2.2985 -
micro_f1: 0.1163 - val_loss: 2.3189 - val_micro_f1: 0.0971
Epoch 22/50
12600/12600 [==============================] - 85s 7ms/step - loss: 2.2988 -
micro_f1: 0.1163 - val_loss: 2.3125 - val_micro_f1: 0.0936
Epoch 23/50
12600/12600 [==============================] - 84s 7ms/step - loss: 2.2987 -
micro_f1: 0.1194 - val_loss: 2.3136 - val_micro_f1: 0.0997
Epoch 24/50
12600/12600 [==============================] - 85s 7ms/step - loss: 2.2986 -
micro_f1: 0.1191 - val_loss: 2.3221 - val_micro_f1: 0.0933
Epoch 25/50
12600/12600 [==============================] - 84s 7ms/step - loss: 2.2988 -
micro_f1: 0.1157 - val_loss: 2.3241 - val_micro_f1: 0.0928
Epoch 26/50
12600/12600 [==============================] - 84s 7ms/step - loss: 2.2990 -
micro_f1: 0.1152 - val_loss: 2.3162 - val_micro_f1: 0.0995
Epoch 27/50
12600/12600 [==============================] - 83s 7ms/step - loss: 2.2982 -
micro_f1: 0.1201 - val_loss: 2.3167 - val_micro_f1: 0.0934
Epoch 28/50
12600/12600 [==============================] - 84s 7ms/step - loss: 2.2978 -
micro_f1: 0.1171 - val_loss: 2.3143 - val_micro_f1: 0.1006
Epoch 29/50
12600/12600 [==============================] - 83s 7ms/step - loss: 2.2970 -
micro_f1: 0.1196 - val_loss: 2.3163 - val_micro_f1: 0.0978
Epoch 30/50
12600/12600 [==============================] - 82s 7ms/step - loss: 2.2976 -
micro_f1: 0.1190 - val_loss: 2.3163 - val_micro_f1: 0.0868
Epoch 31/50
12600/12600 [==============================] - 82s 7ms/step - loss: 2.2991 -
micro_f1: 0.1140 - val_loss: 2.3158 - val_micro_f1: 0.0904
Epoch 32/50
12600/12600 [==============================] - 83s 7ms/step - loss: 2.2993 -
micro_f1: 0.1145 - val_loss: 2.3218 - val_micro_f1: 0.0987
Epoch 33/50
12600/12600 [==============================] - 80s 6ms/step - loss: 2.3000 -
micro_f1: 0.1116 - val_loss: 2.3240 - val_micro_f1: 0.0839
Epoch 34/50
12600/12600 [==============================] - 82s 6ms/step - loss: 2.3006 -
micro_f1: 0.1163 - val_loss: 2.3209 - val_micro_f1: 0.1041
Epoch 35/50
```

```
12600/12600 [==============================] - 84s 7ms/step - loss: 2.2989 -
micro_f1: 0.1174 - val_loss: 2.3194 - val_micro_f1: 0.0901
Epoch 36/50
12600/12600 [==============================] - 82s 7ms/step - loss: 2.2987 -
micro_f1: 0.1160 - val_loss: 2.3189 - val_micro_f1: 0.0923
Epoch 37/50
 6619/12600 [==============>…] - ETA: 38s - loss: 2.2981 -
micro_f1: 0.1141Buffered data was truncated after reaching the output size
limit.
```

[5]:
```
!kill 236
```

[7]:
```
%tensorboard --logdir '/content/drive/My Drive/30) spoken digit assignement/
 ↪log2'
```

```
<IPython.core.display.Javascript object>
```

Observation

1) Micro F1 score is better for the spectogram data as compared to raw data .
2) Data augmentation has no affect on Micro F1 score , But converting it to spectogram data mattered a lot

[ ]:
```

```