In [ ]:
```
%tensorflow_version 2.x
import tensorflow
tensorflow.__version__
```

Out[ ]:
```
'2.3.0'
```

In [ ]:

```
# Install the required libraries
!pip install numpy requests nlpaug
!pip install googletrans
!pip install fuzzywuzzy
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages
(1.18.5)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packag
es (2.23.0)
Requirement already satisfied: nlpaug in /usr/local/lib/python3.6/dist-packages
(1.0.1)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/di
st-packages (from requests) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/
local/lib/python3.6/dist-packages (from requests) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/d
ist-packages (from requests) (2020.6.20)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-pa
ckages (from requests) (2.10)
Requirement already satisfied: googletrans in /usr/local/lib/python3.6/dist-pac
kages (3.0.0)
Requirement already satisfied: httpx==0.13.3 in /usr/local/lib/python3.6/dist-p
ackages (from googletrans) (0.13.3)
Requirement already satisfied: hstspreload in /usr/local/lib/python3.6/dist-pac
kages (from httpx==0.13.3->googletrans) (2020.10.6)
Requirement already satisfied: sniffio in /usr/local/lib/python3.6/dist-package
s (from httpx==0.13.3->googletrans) (1.1.0)
Requirement already satisfied: rfc3986<2,>=1.3 in /usr/local/lib/python3.6/dist
-packages (from httpx==0.13.3->googletrans) (1.4.0)
Requirement already satisfied: chardet==3.* in /usr/local/lib/python3.6/dist-pa
ckages (from httpx==0.13.3->googletrans) (3.0.4)
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist-package
s (from httpx==0.13.3->googletrans) (2020.6.20)
Requirement already satisfied: idna==2.* in /usr/local/lib/python3.6/dist-packa
ges (from httpx==0.13.3->googletrans) (2.10)
Requirement already satisfied: httpcore==0.9.* in /usr/local/lib/python3.6/dist
-packages (from httpx==0.13.3->googletrans) (0.9.1)
Requirement already satisfied: contextvars>=2.1; python_version < "3.7" in /us
r/local/lib/python3.6/dist-packages (from sniffio->httpx==0.13.3->googletrans)
(2.4)
Requirement already satisfied: h2==3.* in /usr/local/lib/python3.6/dist-package
s (from httpcore==0.9.*->httpx==0.13.3->googletrans) (3.2.0)
Requirement already satisfied: h11<0.10,>=0.8 in /usr/local/lib/python3.6/dist-
packages (from httpcore==0.9.*->httpx==0.13.3->googletrans) (0.9.0)
Requirement already satisfied: immutables>=0.9 in /usr/local/lib/python3.6/dist
-packages (from contextvars>=2.1; python_version < "3.7"->sniffio->httpx==0.13.
3->googletrans) (0.14)
Requirement already satisfied: hyperframe<6,>=5.2.0 in /usr/local/lib/python3.
6/dist-packages (from h2==3.*->httpcore==0.9.*->httpx==0.13.3->googletrans) (5.
2.0)
Requirement already satisfied: hpack<4,>=3.0 in /usr/local/lib/python3.6/dist-p
ackages (from h2==3.*->httpcore==0.9.*->httpx==0.13.3->googletrans) (3.0.0)
Requirement already satisfied: fuzzywuzzy in /usr/local/lib/python3.6/dist-pack
ages (0.18.0)
```

In [ ]:
```
import spacy
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer

import nlpaug.augmenter.char as nac
import nlpaug.augmenter.word as naw
import nlpaug.augmenter.sentence as nas
from googletrans import Translator
import time
import re
import string
import nltk
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import stopwords
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
import pickle
from tqdm import tqdm
import os
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, Flatten,LSTM, Bidirection
al, Dropout, Conv1D, MaxPool1D, LSTM, TimeDistributed, GlobalMaxPool1D, GRU
from tensorflow.keras import regularizers, optimizers
from tensorflow.keras.initializers import Constant
from tqdm import tqdm
import numpy as np
from bs4 import BeautifulSoup
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,
recall_score, f1_score, roc_curve, auc
from IPython.display import display, HTML
```

In [ ]:
```
nlp = spacy.load('en_core_web_sm')
```

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

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

In [ ]:
```
import os
%cd "/content/drive/My Drive/Data/"
!pwd
```

```
/content/drive/My Drive/Data
/content/drive/My Drive/Data
```

In [ ]:
```
raw_data = pd.read_excel('/content/drive/My Drive/Data/input_data.xlsx')
data = raw_data.copy()
data.head()
```

Out[ ]:

	Short description	Description	Caller	Assignment group
0	login issue	-verified user details.(employee# & manager na...	spxjnwir pjlcoqds	GRP_0
1	outlook	\r\n\r\nreceived from: hmjdrvpb.komuaywn@gmail...	hmjdrvpb komuaywn	GRP_0
2	cant log in to vpn	\r\n\r\nreceived from: eylqgodm.ybqkwiam@gmail...	eylqgodm ybqkwiam	GRP_0
3	unable to access hr_tool page	unable to access hr_tool page	xbkucsvz gcpydteq	GRP_0
4	skype error	skype error	owlgqjme qhcozdfx	GRP_0

Data Preprocessing

In [ ]:
```
#Caller is unique and encrypted column, sicnce it will not add any value to our
model we can drop it.
data = data.drop(columns='Caller')
data['Description'] = data['Description'].replace(to_replace=[r"\\t|\\n|\\r", "
\t|\n|\r"], value=[" "," "], regex=True)
data['Short description'] = data['Short description'].replace(to_replace=[r"\\t
|\\n|\\r", "\t|\n|\r"], value=[" "," "], regex=True)
```

In [ ]:
```
duplicate = data[data.duplicated(keep=False)]
print('Preview of some dulplicate values in data\n')
display(duplicate.sort_values(by=['Short description']).head(10))
duplicate = data[data.duplicated(keep='first')]
print(f'\n\nTotal number of duplicate rows in data {duplicate.shape[0]}')
print(f'Duplicate rows droped from data.')
print(f'Number of rows in data before dropping duplicates rows: {data.shape[0]}
')
data = data.drop_duplicates( keep='first')
print(f'Number of rows in data after dropping duplicates rows: {data.shape[0]}'
)
data = data.reset_index(drop=True)
```

```
Preview of some dulplicate values in data
```

	Short description	Description	Assignment group
899	HostName_1030 is currently experiencing high c...	HostName_1030 is currently experiencing high c...	GRP_12
474	HostName_1030 is currently experiencing high c...	HostName_1030 is currently experiencing high c...	GRP_12
2701	account got locked	account got locked	GRP_0
2387	account got locked	account got locked	GRP_0
1988	account got locked	account got locked	GRP_0
7058	account is locked	account is locked	GRP_0
7170	account is locked	account is locked	GRP_0
4688	account locked	account locked	GRP_0
3800	account locked	account locked	GRP_0
3396	account locked	account locked	GRP_0

```
Total number of duplicate rows in data 591
Duplicate rows droped from data.
Number of rows in data before dropping duplicates rows: 8500
Number of rows in data after dropping duplicates rows: 7909
```

In [ ]:
```
data['Description'] = data['Description'].astype(str)
data['Short description'] = data['Short description'].astype(str)
```

In [ ]:

```
from fuzzywuzzy import fuzz
from fuzzywuzzy import process

data['short full desc similarity'] = 0
for index in data.index:
   data['short full desc similarity'][index] = fuzz.partial_ratio(data['Short d
escription'][index].lower(),data['Description'][index].lower())
print(' \nAdded column for similatiry score between short and long description'
)
display(data.head())
for index in data.index:
  if data['short full desc similarity'][index] < 100 :
    data['Description'][index] = data['Short description'][index] + ' ' +  data
['Description'][index]
print('\n\n\nConcatnated short description to description if similarity is less
then 100')
display(data.head())
print('\n\n\nDropped Description and short description columns')
data = data.drop(columns=[ 'Short description' , 'short full desc similarity'])
display(data.head())
```

```
Added column for similatiry score between short and long description
```

	Short description	Description	Assignment group	short full desc similarity
0	login issue	-verified user details.(employee# & manager na...	GRP_0	27
1	outlook	received from: hmjdrvpb.komuaywn@gmail.com...	GRP_0	100
2	cant log in to vpn	received from: eylqgodm.ybqkwiam@gmail.com...	GRP_0	83
3	unable to access hr_tool page	unable to access hr_tool page	GRP_0	100
4	skype error	skype error	GRP_0	100

```
Concatnated short description to description if similarity is less then 100
```

	Short description	Description	Assignment group	short full desc similarity
0	login issue	login issue -verified user details.(employee# ...	GRP_0	27
1	outlook	received from: hmjdrvpb.komuaywn@gmail.com...	GRP_0	100
2	cant log in to vpn	cant log in to vpn received from: eylqgodm...	GRP_0	83
3	unable to access hr_tool page	unable to access hr_tool page	GRP_0	100
4	skype error	skype error	GRP_0	100

```
Dropped Description and short description columns
```

	Description	Assignment group
0	login issue -verified user details.(employee# ...	GRP_0
1	received from: hmjdrvpb.komuaywn@gmail.com...	GRP_0
2	cant log in to vpn received from: eylqgodm...	GRP_0
3	unable to access hr_tool page	GRP_0
4	skype error	GRP_0

In [ ]:

```
# Drop any null values from data
data = data.dropna()
data = data.reset_index(drop=True)
```

In [ ]:

```
# for index in data.index:
#   if len(data['Description'][index]) > 5000:
#     data = data.drop(index=[index])
#     print(f'Row droped from index position {index}')

# data = data.reset_index(drop=True)
```

In [ ]:

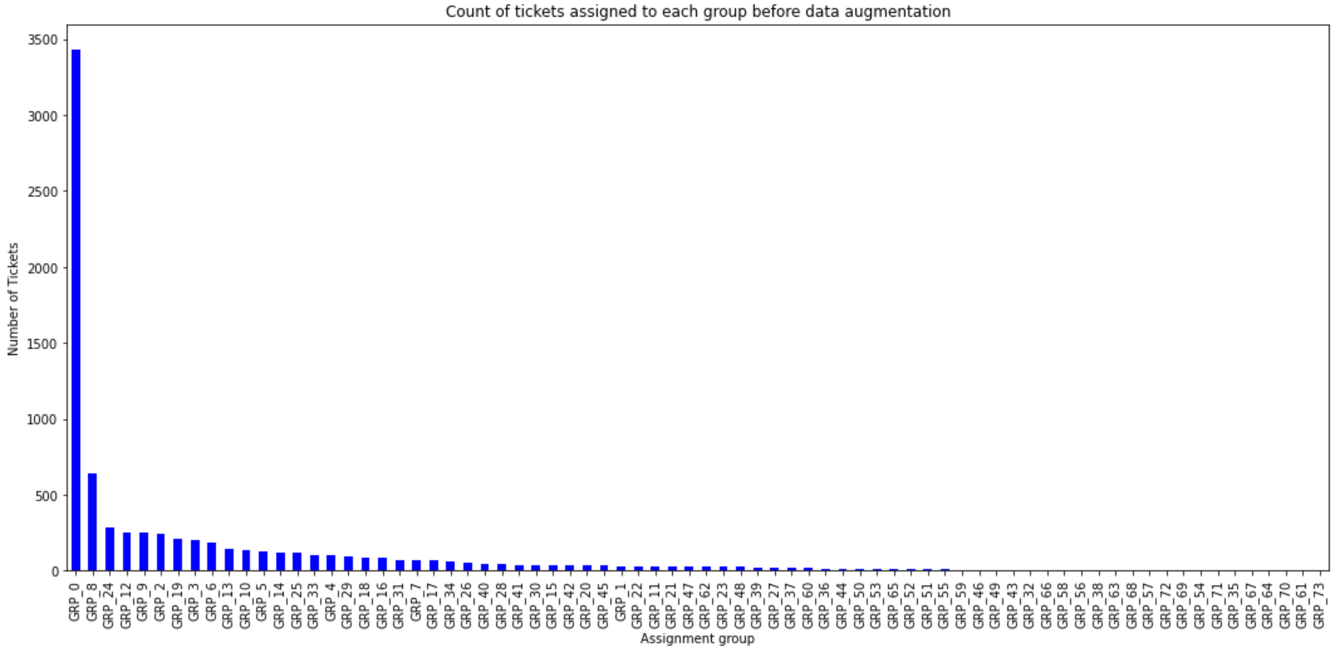
```
# Scale down description to 2000 character for more efficient traslation, Augme
ntaion, tokenisation.
data['Description'] = data['Description'].apply( lambda desc : desc[:2000])

translator = Translator()

def synonymAug(desc):
  lang_det = translator.detect(desc)
  if lang_det.lang != 'en':
    eng_translate = translator.translate(desc,dest='en')
    return eng_translate.text
  return desc


start = time.time()
data['Description'] = data['Description'].apply(synonymAug)
end = time.time()

print(f"Runtime for description translation is {end - start}")
```

```
Runtime for description translation is 1706.3084893226624
```

In [ ]:

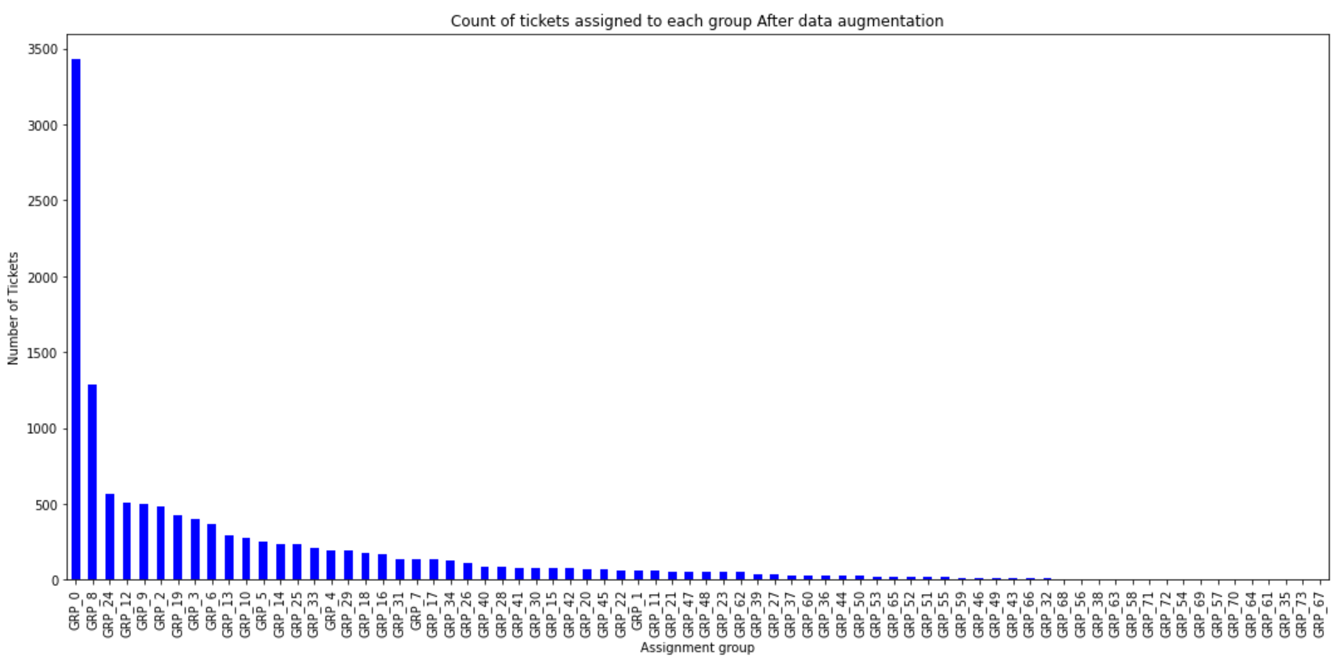
```
print(f'Shape of data before augmentation : {data.shape}')
plt.figure(figsize=(18,8))
plt.xticks(rotation=90)
plt.title('Count of tickets assigned to each group before data augmentation')
plt.xlabel("Assignment group")
plt.ylabel("Number of Tickets")
data['Assignment group'].value_counts().plot.bar(color ='blue')
plt.show()


aug = naw.SynonymAug()
au_data = aug.augment(data[data['Assignment group'] != 'GRP_0']['Description'].
tolist())
y_au_data =  data[data['Assignment group'] != 'GRP_0']['Assignment group'].toli
st()
xnew = data['Description'].tolist()
xnew.extend(au_data)
ynew = data['Assignment group'].tolist()
ynew.extend(y_au_data)
val = {'Description':xnew, 'Assignment group':ynew}
data = pd.DataFrame(val)


print(f'\n\n\nShape of data after augmentation : {data.shape}')
plt.figure(figsize=(18,8))
plt.xticks(rotation=90)
plt.title('Count of tickets assigned to each group After data augmentation')
plt.xlabel("Assignment group")
plt.ylabel("Number of Tickets")
data['Assignment group'].value_counts().plot.bar(color ='blue')
plt.show()
```

```
Shape of data before augmentation : (7909, 2)
```




```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
```

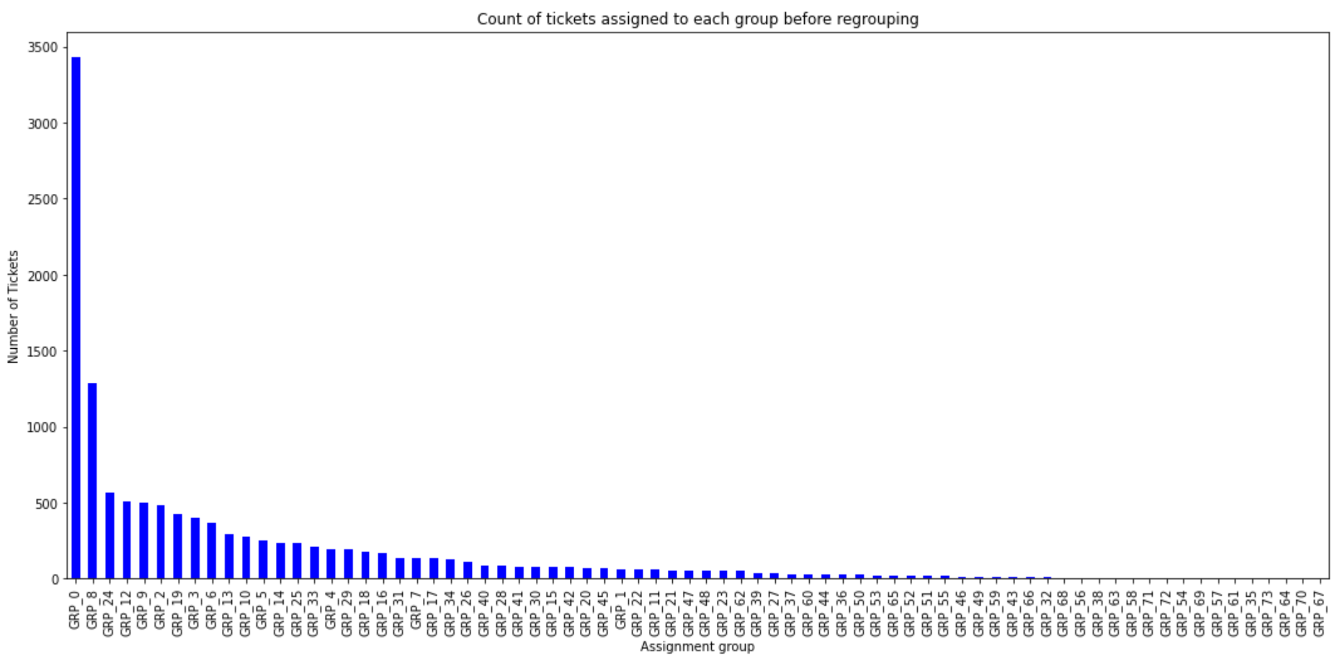
```
Shape of data after augmentation : (12389, 2)
```



In [ ]:
```python
data_groups = (data['Assignment group'].unique())
print(data_groups)
```

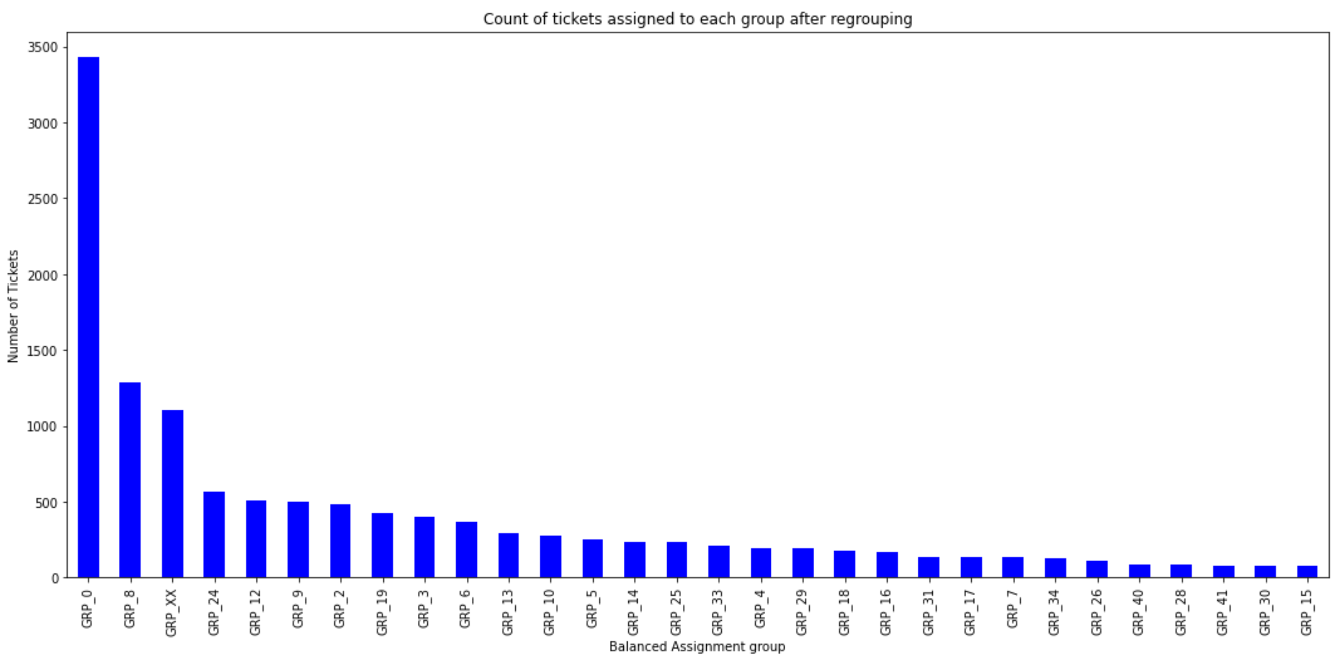
```
['GRP_0' 'GRP_1' 'GRP_3' 'GRP_4' 'GRP_5' 'GRP_6' 'GRP_7' 'GRP_8' 'GRP_9'
 'GRP_10' 'GRP_11' 'GRP_12' 'GRP_13' 'GRP_14' 'GRP_15' 'GRP_16' 'GRP_17'
 'GRP_18' 'GRP_19' 'GRP_2' 'GRP_20' 'GRP_21' 'GRP_22' 'GRP_23' 'GRP_24'
 'GRP_25' 'GRP_26' 'GRP_27' 'GRP_28' 'GRP_29' 'GRP_30' 'GRP_31' 'GRP_33'
 'GRP_34' 'GRP_35' 'GRP_36' 'GRP_37' 'GRP_38' 'GRP_39' 'GRP_40' 'GRP_41'
 'GRP_42' 'GRP_43' 'GRP_44' 'GRP_45' 'GRP_46' 'GRP_47' 'GRP_48' 'GRP_49'
 'GRP_50' 'GRP_51' 'GRP_52' 'GRP_53' 'GRP_54' 'GRP_55' 'GRP_56' 'GRP_57'
 'GRP_58' 'GRP_59' 'GRP_60' 'GRP_61' 'GRP_32' 'GRP_62' 'GRP_63' 'GRP_64'
 'GRP_65' 'GRP_66' 'GRP_67' 'GRP_68' 'GRP_69' 'GRP_70' 'GRP_71' 'GRP_72'
 'GRP_73']
```

In [ ]:
```python
counts = (data['Assignment group'].value_counts())
plt.figure(figsize=(18,8))
counts.sort_values(ascending=False).plot.bar(color ='blue')
plt.xticks(rotation=90)
plt.title('Count of tickets assigned to each group before regrouping')
plt.xlabel("Assignment group")
plt.ylabel("Number of Tickets")
plt.show()
```



In [ ]:

```python
number_of_classes = 30
def regroup_labels(data1): #first 1-22 groups and all other together
  counts = (data['Assignment group'].value_counts())
  grouplist=list(counts.index[(number_of_classes-1):])
  if  data1 in grouplist:
    return 'GRP_XX'
  else: return data1

data['Balanced Assignment Group'] = [regroup_labels(x) for x in data['Assignmen
t group']]

counts = (data['Balanced Assignment Group'].value_counts())
plt.figure(figsize=(18,8))
counts.sort_values(ascending=False).plot.bar(color ='blue')
plt.xticks(rotation=90)
plt.title('Count of tickets assigned to each group after regrouping')
plt.xlabel("Balanced Assignment group")
plt.ylabel("Number of Tickets")
plt.show()

data = data.drop(columns='Assignment group')
```



In [ ]:

```python
data['Raw Desc Word Count'] = 0
for index in data.index:
  data['Raw Desc Word Count'][index] = len(data['Description'][index].split())
data.head()
```

Out[ ]:

	Description	Balanced Assignment Group	Raw Desc Word Count
0	login issue -verified user details.(employee# ...	GRP_0	35
1	received from: hmjdrvpb.komuaywn@gmail.com...	GRP_0	25
2	cant log in to vpn received from: eylqgodm...	GRP_0	16
3	unable to access hr_tool page	GRP_0	5
4	skype error	GRP_0	2

In [ ]:
```
def custom_replacement(phrase):
    # Actual URL's are encrypted in data, so it is good to remove them from inp
ut data.
    phrase = re.sub(r'https?:\/\/.\/\w', ' ', phrase)
    phrase = re.sub(r'[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+', ' ', phr
ase)
    return phrase
```

In [ ]:
```
data['Processed Description'] = data.apply(lambda _: '', axis=1)
for index, row in data.iterrows():
  row['Description'] = BeautifulSoup(row['Description'], 'lxml').get_text()
  row['Description'] = custom_replacement(row['Description'])
  row['Description'] = re.sub('[!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n]+', ' ', r
ow['Description'])
  data['Processed Description'][index] = row['Description']

data.head()
```

Out[ ]:

	Description	Balanced Assignment Group	Raw Desc Word Count	Processed Description
0	login issue -verified user details. (employee# ...	GRP_0	35	login issue verified user details employee ...
1	received from: hmjdrvpb.komuaywn@gmail.com...	GRP_0	25	received from hello team my meetings...
2	cant log in to vpn received from: eylqgodm...	GRP_0	16	cant log in to vpn received from hi ...
3	unable to access hr_tool page	GRP_0	5	unable to access hr tool page
4	skype error	GRP_0	2	skype error

In [ ]:
```
# add custom words to spacy stopwords
custom_stopwords = {'hello','hi','ic','ic:','cc','cc:','bcc','bcc:','to:','subj
ect','subject:','sent:','received','from:',
                    'received from:','etc','com'}
nlp.Defaults.stop_words |= custom_stopwords
```

In [ ]:
```
from spacy.tokens import Doc

desc = list(data['Processed Description'])
docs = [nlp.make_doc(text) for text in desc]

def remove_tokens_on_match(doc):
    indexes = []
    for index, token in enumerate(doc):
        if ((token.is_stop) or (token.is_punct) or (token.like_email) or (token
.is_space) or (token.like_url)):
            indexes.append(index)
    doc2 = Doc(doc.vocab, words=[t.lemma_ for i, t in enumerate(doc) if i not i
n indexes])
    return doc2
```

In [ ]:
```
data['Number of Tokens'] = 0

for doc, ind in zip(docs, range(len(docs))):
  doc2 = remove_tokens_on_match(doc)
  data['Number of Tokens'][ind] = len(doc2)
  if len(doc2):
    data['Processed Description'][ind] =  ' '.join([t.text for t in doc2 ])
  else:
    data['Processed Description'][ind] = ' '

data = data[data['Number of Tokens']!=0]
data.head()
```

Out[ ]:

	Description	Balanced Assignment Group	Raw Desc Word Count	Processed Description	Number of Tokens
0	login issue -verified user details. (employee# ...	GRP_0	35	login issue verify user detail employee manage...	22
1	received from: hmjdrvpb.komuaywn@gmail.com...	GRP_0	25	team meeting skype meeting appear outlook cale...	11
2	cant log in to vpn received from: eylqgodm...	GRP_0	16	not log vpn log vpn well	6
3	unable to access hr_tool page	GRP_0	5	unable access hr tool page	5
4	skype error	GRP_0	2	skype error	2

In [ ]:
```
data.to_csv('/content/drive/My Drive/Data/processed_data_csv_10_10_2020')
```

In [ ]:

```
for i in range(0,5):
  print("Ticket Description:")
  print(data['Description'][i])
  print("\n\nProcessed Description:")
  print(data['Processed Description'][i])
  print("\n---------------------\n")
```

```
Ticket Description:
login issue -verified user details.(employee# & manager name)  -checked the use
r name in ad and reset the password.  -advised the user to login and check.  -c
aller confirmed that he was able to login.  -issue resolved.


Processed Description:
login issue verify user detail employee manager check user ad reset password ad
vise user login check caller confirm able login issue resolve

---------------------

Ticket Description:
    received from: hmjdrvpb.komuaywn@gmail.com    hello team,    my meetings/sk
ype meetings etc are not appearing in my outlook calendar, can somebody please
advise how to correct this?    kind


Processed Description:
team meeting skype meeting appear outlook calendar somebody advise correct kind

---------------------

Ticket Description:
cant log in to vpn     received from: eylqgodm.ybqkwiam@gmail.com    hi    i ca
nnot log on to vpn    best


Processed Description:
not log vpn log vpn well

---------------------

Ticket Description:
unable to access hr_tool page


Processed Description:
unable access hr tool page

---------------------

Ticket Description:
skype error


Processed Description:
skype error

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

Base Line Traditional Models

In [ ]:

```
import time
def model_fn(algo, train, test, algo_text, y_train, y_test, features, i):
  if algo == 'a':
    start = time.time()
    model = SVC()
    model.fit(train, y_train)
    end = time.time()
    print(f"model training time of {algo_text} is {end - start} seconds")
  elif algo == 'b':
    start = time.time()
    model = RandomForestClassifier()
    model.fit(train, y_train)
    end = time.time()
    print(f"model training time of {algo_text} is {end - start} seconds")
  elif algo == 'c':
    start = time.time()
    model = GaussianNB()
    model.fit(train, y_train)
    end = time.time()
    print(f"model training time of {algo_text} is {end - start} seconds")

  y_pred_train = model.predict(train)
  start_pred = time.time()
  y_pred = model.predict(test)
  end_pred = time.time()
  print(f"model predicting time of {algo_text} is {end - start} seconds")
  tr_ac = accuracy_score(y_train,y_pred_train)
  te_ac = accuracy_score(y_test,y_pred)
  print('The train accuracy of ' + features + ' with ' + algo_text + ' is: ', accuracy_score(y_train,y_pred_train))
  print('The test accuracy of ' + features + ' with ' + algo_text + ' is: ', accuracy_score(y_test,y_pred))
  results = pd.DataFrame({'Method':[algo_text], 'Features':[features], 'train accuracy': [tr_ac], 'test accuracy':[te_ac],'F1':[f1_score(y_test, y_pred, average='macro')],'Precesion':[precision_score(y_test, y_pred, average='macro')], 'Recall':[recall_score(y_test, y_pred, average='macro')],'Training time':[end - start],'Predicting time':[end_pred - start_pred]},index={i})
  results = results[['Method', 'Features', 'train accuracy','test accuracy','F1','Precesion','Recall','Training time','Predicting time']]
  return results
```

Glove

In [ ]:

```
embeddings_index_glove = {}
f = open('/content/drive/My Drive/Glove /glove.6B.300d.txt')
for line in tqdm(f):
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index_glove[word] = coefs
f.close()

# print('Found %s word vectors.' % len(embeddings_index_glove))
re_tok = re.compile(u'([{string.punctuation}“”¨«»®´·º½¾¿¡§£₤‘’])')

def tokenize(s):
    return re_tok.sub(r' \1 ', s).split()

nltk.download("stopwords")
stop_words = set(stopwords.words('english'))

def sent2vec(s, embeddings_index):
    words = str(s)
    words = tokenize(words)
    words = [w for w in words if not w in stop_words]
    words = [w for w in words if w.isalpha()]
    M = []
    for w in words:
        try:
            M.append(embeddings_index[w])
        except:
            continue
    M = np.array(M)
    v = M.sum(axis=0)
    if type(v) != np.ndarray:
        return np.zeros(300)
    return v / np.sqrt((v ** 2).sum())


X_train, X_test, y_train, y_test = train_test_split(data['Processed Descriptio
n'], data['Balanced Assignment Group'], test_size = 0.1, random_state = 42, shu
ffle = True)
le = LabelEncoder()
le.fit(y_train)
y_train=le.transform(y_train)
le = LabelEncoder()
le.fit(y_test)
y_test=le.transform(y_test)


X_train_glove = [sent2vec(x, embeddings_index_glove) for x in (X_train)]
X_test_glove = [sent2vec(x, embeddings_index_glove) for x in (X_test)]

X_train_glove = np.array(X_train_glove)
X_test_glove = np.array(X_test_glove)
model_svm_glove = model_fn('a', X_train_glove, X_test_glove, 'SVM', y_train, y_
test, 'Glove', 1)
model_rf_glove = model_fn('b', X_train_glove, X_test_glove, 'Random Forest', y_
train,y_test,'Glove', 2)
model_nb_glove = model_fn('c', X_train_glove, X_test_glove, 'Naive Bayes', y_tr
ain,y_test,'Glove', 3)

all = pd.concat([model_svm_glove,model_rf_glove])
```

```
all = pd.concat([all,model_nb_glove])
all
```

```
400000it [00:37, 10599.92it/s]

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
model training time of SVM is 73.68653059005737 seconds
model predicting time of SVM is 73.68653059005737 seconds
The train accuracy of Glove with SVM is:  0.3923573735199139
The test accuracy of Glove with SVM is:  0.35996771589991927
model training time of Random Forest is 32.57555294036865 seconds
model predicting time of Random Forest is 32.57555294036865 seconds
The train accuracy of Glove with Random Forest is:  0.9633118048080374
The test accuracy of Glove with Random Forest is:  0.5326876513317191
model training time of Naive Bayes is 0.04852938652038574 seconds
model predicting time of Naive Bayes is 0.04852938652038574 seconds
The train accuracy of Glove with Naive Bayes is:  0.20676354503049874
The test accuracy of Glove with Naive Bayes is:  0.19693301049233253
```

Out[ ]:

	Method	Features	train accuracy	test accuracy	F1	Precesion	Recall	Training time	Predicting time
1	SVM	Glove	0.392357	0.359968	0.099234	0.193111	0.111677	73.686531	6.797709
2	Random Forest	Glove	0.963312	0.532688	0.417793	0.782143	0.333869	32.575553	0.071652
3	Naive Bayes	Glove	0.206764	0.196933	0.172459	0.196671	0.248195	0.048529	0.048351

In [ ]:

```
# Selecting best parameters for Random forrest classifier

## Commented the code for default run, Please uncomment if require to run hyper
parameter tuning

# from sklearn.model_selection import GridSearchCV
# from sklearn.ensemble import RandomForestClassifier
# from scipy.stats import uniform
# def rf_opt_params(X_train,y_train,X_test,y_test):
#     model_rf = RandomForestClassifier()
#     distributions = dict(n_estimators=[50,200,1000],
#                          max_features= [ 'sqrt', 'log2'],
#                          max_depth= [10,100,None],
#                          criterion=['gini', 'entropy'])
#     param_grid = {
#         'n_estimators': [50,200,1000],
#         'max_features': ['sqrt', 'log2'],
#         'max_depth' : [10,100,None],
#         'criterion' :['gini', 'entropy']
#     }
#     clf = GridSearchCV(model_rf, param_grid,verbose=5,n_jobs=2,cv=3)
#         search = clf.fit(X_train, y_train)
#     print(search.best_params_)
#     return search
```

In [ ]:

```
x = data['Processed Description']
y = data['Balanced Assignment Group']
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random
_state=0)
```

In [ ]:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

tfidfconverter_m = TfidfVectorizer(decode_error='replace', encoding='utf-8')
X_tfidf_m = tfidfconverter_m.fit_transform(x.values.astype('U'))


from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import MultinomialNB

text_clf = Pipeline([('vect', TfidfVectorizer(ngram_range=(1,2),stop_words=stop
words.words('english'))), ('tfidf', TfidfTransformer()), ('clf', MultinomialNB
()),])
text_clf.fit(x, y)
```

Out[ ]:

```
Pipeline(memory=None,
         steps=[('vect',
                 TfidfVectorizer(analyzer='word', binary=False,
                                 decode_error='strict',
                                 dtype=<class 'numpy.float64'>,
                                 encoding='utf-8', input='content',
                                 lowercase=True, max_df=1.0, max_features=None,
                                 min_df=1, ngram_range=(1, 2), norm='l2',
                                 preprocessor=None, smooth_idf=True,
                                 stop_words=['i', 'me', 'my', 'myself', 'we',
                                             'our', 'ours', 'ourselves', 'yo
u'...
                                             'him', 'his', 'himself', 'she',
                                             "she's", 'her', 'hers', 'herself',
                                             'it', "it's", 'its', 'itself',
...],
                                 strip_accents=None, sublinear_tf=False,
                                 token_pattern='(?u)\\b\\w\\w+\\b',
                                 tokenizer=None, use_idf=True,
                                 vocabulary=None)),
                ('tfidf',
                 TfidfTransformer(norm='l2', smooth_idf=True,
                                  sublinear_tf=False, use_idf=True)),
                ('clf',
                 MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True))],
         verbose=False)
```

In [ ]:

```
#hyperparameter tuning
import time
start= time.time()
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.model_selection import train_test_split, GridSearchCV
#Defining pipeline
text_clf = Pipeline([('vect', TfidfVectorizer()),
                     ('tfidf', TfidfTransformer()),
                     ('clf', MultinomialNB())])

#defining tned_parameters
tuned_parameters = {
    'vect__ngram_range': [(1, 1), (1, 2), (2, 2)],
    'tfidf__use_idf': (True, False),
    'tfidf__norm': ('l1', 'l2'),

    'clf__alpha': [1, 1e-1, 1e-2]
}
scores = ['precision', 'recall']


x = data['Processed Description']
y = data['Balanced Assignment Group']
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, rando
m_state=42)

from sklearn.metrics import classification_report
clf = GridSearchCV(text_clf, tuned_parameters, cv=10,)
clf.fit(x_train, y_train)

print(classification_report(y_test, clf.predict(x_test), digits=4))
end=time.time()

print(f"Runtime of the program is {end - start}"+" seconds")
```

```
              precision    recall  f1-score   support

       GRP_0     0.8405    0.9004    0.8694      1124
      GRP_10     0.9733    0.8391    0.9012        87
      GRP_12     0.7798    0.8344    0.8062       157
      GRP_13     0.8605    0.8315    0.8457        89
      GRP_14     0.7333    0.8049    0.7674        82
      GRP_15     0.9048    0.8261    0.8636        23
      GRP_16     0.8235    0.6774    0.7434        62
      GRP_17     0.9318    0.9762    0.9535        42
      GRP_18     0.8929    0.8621    0.8772        58
      GRP_19     0.8175    0.7000    0.7542       160
       GRP_2     0.7961    0.7202    0.7562       168
      GRP_24     0.9639    0.9639    0.9639       194
      GRP_25     0.7917    0.7808    0.7862        73
      GRP_26     0.8571    0.6486    0.7385        37
      GRP_28     0.8800    0.6667    0.7586        33
      GRP_29     0.8475    0.7937    0.8197        63
       GRP_3     0.7252    0.7983    0.7600       119
      GRP_30     0.7917    0.8636    0.8261        22
      GRP_31     0.8378    0.7045    0.7654        44
      GRP_33     0.6575    0.9231    0.7680        52
      GRP_34     0.8696    0.8000    0.8333        50
       GRP_4     0.8333    0.7143    0.7692        70
      GRP_40     0.8214    0.9583    0.8846        24
      GRP_41     1.0000    0.9032    0.9492        31
       GRP_5     0.8313    0.8214    0.8263        84
       GRP_6     0.9018    0.8707    0.8860       116
       GRP_7     0.9500    0.7600    0.8444        50
       GRP_8     0.8628    0.9263    0.8934       448
       GRP_9     0.9618    0.9618    0.9618       157
      GRP_XX     0.7791    0.6883    0.7309       369

    accuracy                         0.8422      4088
   macro avg     0.8506    0.8173    0.8301      4088
weighted avg     0.8437    0.8422    0.8408      4088

Runtime of the program is 188.92419123649597 seconds
```

In [ ]:
```python
print("Best parameters set found on development set:\n")
print(clf.best_params_)
```

```
Best parameters set found on development set:

{'clf__alpha': 0.01, 'tfidf__norm': 'l2', 'tfidf__use_idf': True, 'vect__ngram_
range': (1, 2)}
```

In [ ]:
```
# accuracy after hyperparameter tuning
import time
start= time.time()

tfidfconverter_m1 = TfidfVectorizer(decode_error='replace', encoding='utf-8',ng
ram_range=(1, 2),norm= 'l2')

X_tfidf= tfidfconverter_m1.fit_transform(x.values.astype('U'))
#y_tfidf= tfidfconverter_m1.fit_transform(y.values.astype('U'))


x_train, x_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2,
random_state=0)
clf_nb_m1=MultinomialNB(alpha=0.01)
clf_nb_m1.fit(x_train, y_train)
#text_clf_nb.fit(x,y)
end=time.time()

print(f"Runtime of the program is {end - start}"+" seconds")
```

```
Runtime of the program is 0.19823336601257324 seconds
```

In [ ]:
```
pred = clf_nb_m1.predict(x_test)

#print(f"Runtime of the program is {end - start}"+" seconds")
acc = accuracy_score(y_test, pred)
print("test=",acc)
pred_train = clf_nb_m1.predict(x_train)
print("acc_train = ", accuracy_score(y_train, pred_train))
```

```
test= 0.854317998385795
acc_train =  0.978100716520335
```

In [ ]:
```
#pre-tuning
tfidfconverter_m1 = TfidfVectorizer(decode_error='replace', encoding='utf-8')

X_tfidf_pre= tfidfconverter_m1.fit_transform(x.values.astype('U'))
#y_tfidf= tfidfconverter_m1.fit_transform(y.values.astype('U'))


x_train, x_test, y_train, y_test = train_test_split(X_tfidf_pre, y, test_size=
0.2, random_state=0)
start= time.time()
clf_nb_m1_pre=MultinomialNB()
clf_nb_m1_pre.fit(x_train, y_train)
#text_clf_nb.fit(x,y)
end=time.time()

print(f"Runtime of the program is {end - start}"+" seconds")
```

```
Runtime of the program is 0.05823111534118652 seconds
```

In [ ]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
#x = v.fit_transform(df['Review'].values.astype('U'))
start=time.time()
pred = clf_nb_m1_pre.predict(x_test)
end=time.time()
print(f"Runtime of the program is {end - start}"+" seconds")
acc = accuracy_score(y_test, pred)
print(classification_report(y_test, clf_nb_m1_pre.predict(x_test), digits=4))
```

```
Runtime of the program is 0.011475324630737305 seconds
              precision    recall  f1-score   support

       GRP_0     0.3907    0.9986    0.5616       705
      GRP_10     0.0000    0.0000    0.0000        49
      GRP_12     0.8462    0.2018    0.3259       109
      GRP_13     1.0000    0.0169    0.0333        59
      GRP_14     1.0000    0.0800    0.1481        50
      GRP_15     0.0000    0.0000    0.0000        10
      GRP_16     0.0000    0.0000    0.0000        34
      GRP_17     0.0000    0.0000    0.0000        18
      GRP_18     0.0000    0.0000    0.0000        35
      GRP_19     0.0000    0.0000    0.0000        86
       GRP_2     0.7059    0.1237    0.2105        97
      GRP_24     0.9740    0.6148    0.7538       122
      GRP_25     0.0000    0.0000    0.0000        44
      GRP_26     0.0000    0.0000    0.0000        24
      GRP_28     0.0000    0.0000    0.0000        19
      GRP_29     0.0000    0.0000    0.0000        37
       GRP_3     0.0000    0.0000    0.0000        84
      GRP_30     0.0000    0.0000    0.0000        10
      GRP_31     1.0000    0.0345    0.0667        29
      GRP_33     1.0000    0.0328    0.0635        61
      GRP_34     0.0000    0.0000    0.0000        25
       GRP_4     0.0000    0.0000    0.0000        35
      GRP_40     0.0000    0.0000    0.0000        21
      GRP_41     0.0000    0.0000    0.0000        23
       GRP_5     0.0000    0.0000    0.0000        39
       GRP_6     1.0000    0.0469    0.0896        64
       GRP_7     0.0000    0.0000    0.0000        24
       GRP_8     0.5065    0.9671    0.6648       243
       GRP_9     0.0000    0.0000    0.0000       107
      GRP_XX     0.6296    0.2372    0.3446       215

    accuracy                         0.4479      2478
   macro avg     0.3018    0.1118    0.1087      2478
weighted avg     0.4344    0.4479    0.3230      2478
```

Neural Networks Model

In [ ]:
```
max_features = 10000
maxlen = 25
embedding_size = 300
```

In [ ]:
```
X = list(data['Processed Description'])
tokenizer = Tokenizer(num_words=max_features , split=' ')
tokenizer.fit_on_texts(X)
X = tokenizer.texts_to_sequences(X)
```

In [ ]:
```
X = pad_sequences(maxlen=maxlen, sequences=X, padding="post")
```

In [ ]:
```
EMBEDDING_FILE = '/content/drive/My Drive/Glove /glove.6B.300d.txt'

num_words = len(tokenizer.word_index) + 1
print(num_words)

embeddings = {}
for o in open(EMBEDDING_FILE):
    word = o.split(" ")[0]
    # print(word)
    embd = o.split(" ")[1:]
    embd = np.asarray(embd, dtype='float32')
    # print(embd)
    embeddings[word] = embd

# create a weight matrix for words in training docs
embedding_matrix = np.zeros((num_words, embedding_size))

for word, i in tokenizer.word_index.items():
        embedding_vector = embeddings.get(word)
        if embedding_vector is not None:
                embedding_matrix[i] = embedding_vector
```

```
19523
```

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

Out[ ]:
```
(19523, 300)
```

In [ ]:
```
num_words = len(tokenizer.word_index) + 1
print(num_words)
```

```
19523
```

In [ ]:
```
y = data['Balanced Assignment Group']
le = LabelEncoder()
y = le.fit_transform(y)
y = tensorflow.keras.utils.to_categorical(y, num_classes=number_of_classes)
```

Vanilla Neural Network Model

(only fully connected dense layers)

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, rand
om_state = 42, shuffle = True)
```

In [ ]:
```
number_of_classes = 30
hub_layer = None
learning_rate=0.1
decay_rate=.2
log_folder="logs"
max_features = 10000
maxlen = 25
embedding_size=300
EMBEDDING_FILE = 'glove.6B.300d.txt'
EMBEDDING_FILE = '/content/drive/My Drive/Glove /glove.6B.300d.txt'
TRAIN_DATA='/content/drive/My Drive/Data/processed_data_csv_10_10_2020'
# best run param: python text_classifer.py --input pdata.csv --train --epoch 10
0 --batchsize 56 --lr .001 --decay_rate .02
```

Utility Functions

- Exp Decay: For learning rate
- Load Data: Utility to help with loading data

In [ ]:

```
def exp_decay(epoch):
    # a eponential decay function that can be used to monitor loss
    lrate = learning_rate * np.exp(-decay_rate*epoch)
    return learning_rate

def load_data(path):
    # load the data from a csv
    if path.endswith("xlsx"):
      logging.info("Loading data")
      data= pd.read_excel(path)
      logging.info("Got data with shape {}".format(data.shape))
      x = data['Description']
      y = data['Assignment group']
      return x.astype('str'),y
    else:
      logging.info("Loading data")
      data= pd.read_csv(path)
      logging.info("Got data with shape {}".format(data.shape))
      x = data['Processed Description']
      y = data['Balanced Assignment Group']
      return x,y

def show_history(history):
    import matplotlib.pyplot as plt
    fig, axes = plt.subplots(1, 2, figsize=(14,6))
    ax = axes[0]
    print(history.history)
    ax.plot(np.sqrt(history.history['accuracy']), 'r', label='train_acc')
    ax.plot(np.sqrt(history.history['val_accuracy']), 'b' ,label='val_acc')
    ax.set_xlabel(r'Epoch', fontsize=20)
    ax.set_ylabel(r'Accuracy', fontsize=20)
    ax.legend()
    ax.tick_params(labelsize=20)

    ax = axes[1]
    ax.plot(np.sqrt(history.history['loss']), 'r', label='train')
    ax.plot(np.sqrt(history.history['val_loss']), 'b' ,label='val')
    ax.set_xlabel(r'Epoch', fontsize=20)
    ax.set_ylabel(r'Loss', fontsize=20)
    ax.legend()
    ax.tick_params(labelsize=20)
    plt.tight_layout()
    plt.show()

def encode_labels(labels):
    logging.info("Hot encoding labels")
    lb = LabelEncoder()
    dy_train = lb.fit_transform(labels)
    dy_train = np_utils.to_categorical(dy_train) # hot encoding
    return dy_train
```

Preprocess Steps

- Tokenizing
- Padding
- GLOVE

In [ ]:
```
def preprocess(X):
    # tokenizes, pads and prepocesses the data
    tokenizer = Tokenizer(num_words=max_features , split=' ')
    tokenizer.fit_on_texts(X)
    X = tokenizer.texts_to_sequences(X)
    X = pad_sequences(maxlen=maxlen, sequences=X, padding="post")
    return tokenizer, X

def make_embedding(X, tokenizer):
    # make an embedding using GLOVE
    # this is a 300 dimensinoal embedding
    num_words = len(tokenizer.word_index) + 1
    embeddings = {}
    for o in open(EMBEDDING_FILE):
        word = o.split(" ")[0]
        embd = o.split(" ")[1:]
        embd = np.asarray(embd, dtype='float32')
        embeddings[word] = embd
    embedding_matrix = np.zeros((num_words, embedding_size))
    for word, i in tokenizer.word_index.items():
            embedding_vector = embeddings.get(word)
            if embedding_vector is not None:
                embedding_matrix[i] = embedding_vector
    num_words = len(tokenizer.word_index) + 1
    return num_words, embedding_matrix

# TODO: Try New Embedding
```

Model Builder

In [ ]:
```
def gen_model(embedding_layer):
    model = tf.keras.Sequential()
    model.add(embedding_layer)
    model.add(Flatten())
    model.add(Dense(500, input_shape=((embedding_size * maxlen),), activation=
'relu'))
    model.add(Dense(100, activation="relu"))
    model.add(Dense(number_of_classes, activation='softmax'))
    adam = optimizers.Adam(lr=.0001)
    model.compile(optimizer=adam,
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

Training Functions

- Single Training
- Hyperparam Training

In [ ]:
```
def train(ipath, batch_size=512, epochs=50):

    X,y = load_data(ipath) # load the data from the path

    model, X = preprocess(X) # this will tokenize and pad the text
    logging.info("Making embedding matrix")
    num_of_words, embedding_mat = make_embedding(X, model) # this uses glove mo
del. Creates a nx300 matrix.
    embedding_layer = Embedding(num_of_words, embedding_size, embeddings_initia
lizer = Constant(embedding_mat), input_length = maxlen, trainable = False)
    logging.info("Embedding shape is {}. Number of words: {}".format(embedding_
mat.shape, num_of_words))

    logging.info("Splitting model into train and test")
    y = encode_labels(y)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1,
random_state = 42, shuffle = True) # swap to cross validation here

    # tensorflow callbacks (for monitoring and augmentation)
    tt = time.time()

    loss_history = tkc.History()
    lr_rate = tkc.LearningRateScheduler(exp_decay)
    #stop_early = tkc.EarlyStopping(monitor='val_loss', patience=20) # uncommen
t this to top the training early if it isn't converging anymore. used to help p
revent overfitting.
    callbacks_list = [loss_history]#h, stop_early]#, lr_rate]

    logging.info("Generating model")
    model = gen_model(embedding_layer)
    print(gen_model(embedding_layer).summary()) #check console for a model summ
ary!

    logging.info("Fitting model")
    y_train = y_train
    y_test = y_test
    start = time.time()
    history = model.fit(X_train, y_train,
                        batch_size=(batch_size),
                        validation_data=(X_test, y_test),
                        callbacks=callbacks_list,
                        epochs=epochs,
                        verbose=1)

    end = time.time()
    training_time = end - start
    y_pred_train = np.argmax(model.predict(X_train), axis=-1)

    start = time.time()
    y_pred = np.argmax(model.predict(X_test), axis=-1)
    end = time.time()
    print(f"model prediction time is {end - start} seconds")

    y_train = np.argmax(y_train, axis=-1)
    y_test = np.argmax(y_test, axis=-1)
    print("Number of training records: {}".format(len(X_train)))
    print("Number of testing records: {}".format(len(X_test)))

    print("Training time {:.2f}".format(training_time))
    print(f'train accuracy : {accuracy_score(y_train,y_pred_train)}')
    print(f'test accuracy :{accuracy_score(y_test,y_pred)}')
```

```python
    fsc = f1_score(y_test, y_pred, average='macro')
    pres = precision_score(y_test, y_pred, average='macro')
    rec = recall_score(y_test, y_pred, average='macro')
    print(f'F1 score : {fsc}')
    print(f'Recall Score : {rec}')
    print(f'Precision Score : {pres}')
```

HyperParameter Tuning

In [ ]:
```
def hyper_train(ipath, batch_size=512, epochs=10):
    '''Train 2 evaluated hyperparam'''
    X,y = load_data(ipath) # load the data from the path
    model, X = preprocess(X) # this will tokenize and pad the text
    #logging.info("Making embedding matrix")
    num_of_words, embedding_mat = make_embedding(X, model) # this uses glove mo
del. Creates a nx300 matrix.
    embedding_layer = Embedding(num_of_words, embedding_size, embeddings_initia
lizer = Constant(embedding_mat), input_length = maxlen, trainable = False)
    #logging.info("Embedding shape is {}. Number of words: {}".format(embedding
_mat.shape, num_of_words))

    #logging.info("Splitting model into train and test")
    y = encode_labels(y)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1,
random_state = 42, shuffle = True) # swap to cross validation here

    DROPOUT_CHOICES = np.arange(0.0, 0.5, 0.1)
    DENSE_UNIT_CHOICES = np.arange(60, 1000, 30, dtype=int)
    DENSE_UNIT_CHOICES2 = np.arange(60, 500, 30, dtype=int)
    BATCH_SIZE_CHOICES = np.arange(64, 512, 64, dtype=int)
    BETA1_CHOICES = np.arange(.6, 1, .1)
    LEARNING_RATE_CHOICES = np.arange(.001, .1, .1)

    space = {
        'spatial_dropout': hp.choice('spatial_dropout', DROPOUT_CHOICES),
        'dense_units':  hp.choice('dense_units', DENSE_UNIT_CHOICES),
        'dense_units2':  hp.choice('dense_units2', DENSE_UNIT_CHOICES2),
        'batch_size':  hp.choice('batch_size', BATCH_SIZE_CHOICES),
        'learning_rate':  hp.choice('learning_rate', LEARNING_RATE_CHOICES),
        'beta1':  hp.choice('beta1', BETA1_CHOICES)
    }

    def objective(params, verbose=0, epochs=50):
        model = tf.keras.Sequential()
        model.add(embedding_layer)
        model.add(Flatten())
        model.add(Dense(params['dense_units'], input_shape=((embedding_size * m
axlen),), activation='relu'))
        model.add(Dropout(params['spatial_dropout']))
        model.add(Dense(params['dense_units2'], activation="relu"))
        model.add(Dense(number_of_classes, activation='sigmoid'))
        adam = optimizers.Adam(lr=params['learning_rate'], decay=decay_rate, be
ta_1=params['beta1'], beta_2=0.999, epsilon=None, amsgrad=False)
        model.compile(optimizer=adam, loss='categorical_crossentropy', metrics=
['accuracy'])

        model.fit(X_train, y_train,
                  batch_size=params['batch_size'],
                  validation_data=(X_test, y_test),
                  callbacks=[tkc.EarlyStopping(patience = 5, monitor='val_accur
acy'), TqdmCallback(verbose=0)],
                  epochs=epochs)

        predictions = model.predict(X_test, verbose=2)
        acc = (predictions.argmax(axis = 1) == y_test.argmax(axis = 1)).mean()
        score_train = model.evaluate(X_train, y_train, verbose=0)
        score_test = model.evaluate(X_test, y_test, verbose=0)
        return {'loss': -acc, 'status': STATUS_OK}

    #logging.info("Fitting model")
```

```
    trials = Trials()
    best = fmin(objective, space, algo=rand.suggest, trials=trials, max_evals=4
, rstate=np.random.RandomState(99))
    return best, space_eval(space, best)
```

Prepocessed Data Analysis

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
%cd "/content/drive/My Drive/Data/"
TRAIN_DATA='/content/drive/My Drive/Data/processed_data_csv_10_10_2020'
train(TRAIN_DATA, batch_size=128, epochs=50)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call d
rive.mount("/content/drive", force_remount=True).
/content/drive/My Drive/Data
19523
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 25, 300)           5856900
_________________________________________________________________
flatten_1 (Flatten)          (None, 7500)              0
_________________________________________________________________
dense_3 (Dense)              (None, 500)               3750500
_________________________________________________________________
dense_4 (Dense)              (None, 100)               50100
_________________________________________________________________
dense_5 (Dense)              (None, 30)                3030
=================================================================
Total params: 9,660,530
Trainable params: 3,803,630
Non-trainable params: 5,856,900
_________________________________________________________________
None
Epoch 1/50
88/88 [==============================] - 5s 58ms/step - loss: 2.4693 - accurac
y: 0.3878 - val_loss: 2.1764 - val_accuracy: 0.4027
Epoch 2/50
88/88 [==============================] - 5s 55ms/step - loss: 1.7769 - accurac
y: 0.5291 - val_loss: 1.8919 - val_accuracy: 0.4746
Epoch 3/50
88/88 [==============================] - 5s 55ms/step - loss: 1.3496 - accurac
y: 0.6519 - val_loss: 1.7007 - val_accuracy: 0.5262
Epoch 4/50
88/88 [==============================] - 5s 55ms/step - loss: 1.0215 - accurac
y: 0.7574 - val_loss: 1.5732 - val_accuracy: 0.5577
Epoch 5/50
88/88 [==============================] - 5s 55ms/step - loss: 0.7846 - accurac
y: 0.8224 - val_loss: 1.4935 - val_accuracy: 0.5892
Epoch 6/50
88/88 [==============================] - 5s 55ms/step - loss: 0.6242 - accurac
y: 0.8651 - val_loss: 1.4364 - val_accuracy: 0.5964
Epoch 7/50
88/88 [==============================] - 5s 55ms/step - loss: 0.5066 - accurac
y: 0.8933 - val_loss: 1.3931 - val_accuracy: 0.6174
Epoch 8/50
88/88 [==============================] - 5s 55ms/step - loss: 0.4204 - accurac
y: 0.9129 - val_loss: 1.4068 - val_accuracy: 0.6077
Epoch 9/50
88/88 [==============================] - 5s 55ms/step - loss: 0.3546 - accurac
y: 0.9271 - val_loss: 1.4007 - val_accuracy: 0.6110
Epoch 10/50
88/88 [==============================] - 5s 54ms/step - loss: 0.3055 - accurac
y: 0.9372 - val_loss: 1.3944 - val_accuracy: 0.6223
Epoch 11/50
88/88 [==============================] - 5s 55ms/step - loss: 0.2662 - accurac
y: 0.9456 - val_loss: 1.4092 - val_accuracy: 0.6239
Epoch 12/50
88/88 [==============================] - 5s 55ms/step - loss: 0.2332 - accurac
y: 0.9527 - val_loss: 1.4058 - val_accuracy: 0.6352
Epoch 13/50
88/88 [==============================] - 5s 55ms/step - loss: 0.2159 - accurac
```

```
y: 0.9553 - val_loss: 1.4068 - val_accuracy: 0.6384
Epoch 14/50
88/88 [==============================] - 5s 55ms/step - loss: 0.1894 - accurac
y: 0.9629 - val_loss: 1.4117 - val_accuracy: 0.6392
Epoch 15/50
88/88 [==============================] - 5s 55ms/step - loss: 0.1693 - accurac
y: 0.9658 - val_loss: 1.4450 - val_accuracy: 0.6368
Epoch 16/50
88/88 [==============================] - 5s 55ms/step - loss: 0.1589 - accurac
y: 0.9682 - val_loss: 1.4606 - val_accuracy: 0.6489
Epoch 17/50
88/88 [==============================] - 5s 55ms/step - loss: 0.1483 - accurac
y: 0.9716 - val_loss: 1.4409 - val_accuracy: 0.6384
Epoch 18/50
88/88 [==============================] - 5s 55ms/step - loss: 0.1324 - accurac
y: 0.9736 - val_loss: 1.4704 - val_accuracy: 0.6449
Epoch 19/50
88/88 [==============================] - 5s 55ms/step - loss: 0.1240 - accurac
y: 0.9740 - val_loss: 1.5122 - val_accuracy: 0.6433
Epoch 20/50
88/88 [==============================] - 5s 55ms/step - loss: 0.1207 - accurac
y: 0.9745 - val_loss: 1.4971 - val_accuracy: 0.6408
Epoch 21/50
88/88 [==============================] - 5s 55ms/step - loss: 0.1151 - accurac
y: 0.9756 - val_loss: 1.5214 - val_accuracy: 0.6473
Epoch 22/50
88/88 [==============================] - 5s 55ms/step - loss: 0.1082 - accurac
y: 0.9777 - val_loss: 1.5482 - val_accuracy: 0.6538
Epoch 23/50
88/88 [==============================] - 5s 55ms/step - loss: 0.1014 - accurac
y: 0.9781 - val_loss: 1.5620 - val_accuracy: 0.6457
Epoch 24/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0973 - accurac
y: 0.9786 - val_loss: 1.5548 - val_accuracy: 0.6441
Epoch 25/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0970 - accurac
y: 0.9787 - val_loss: 1.6117 - val_accuracy: 0.6473
Epoch 26/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0928 - accurac
y: 0.9795 - val_loss: 1.5870 - val_accuracy: 0.6425
Epoch 27/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0948 - accurac
y: 0.9775 - val_loss: 1.6278 - val_accuracy: 0.6441
Epoch 28/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0854 - accurac
y: 0.9795 - val_loss: 1.6469 - val_accuracy: 0.6449
Epoch 29/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0830 - accurac
y: 0.9800 - val_loss: 1.6247 - val_accuracy: 0.6449
Epoch 30/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0774 - accurac
y: 0.9816 - val_loss: 1.6456 - val_accuracy: 0.6449
Epoch 31/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0749 - accurac
y: 0.9830 - val_loss: 1.6677 - val_accuracy: 0.6481
Epoch 32/50
88/88 [==============================] - 5s 56ms/step - loss: 0.0742 - accurac
y: 0.9826 - val_loss: 1.6870 - val_accuracy: 0.6392
Epoch 33/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0772 - accurac
y: 0.9822 - val_loss: 1.6689 - val_accuracy: 0.6416
Epoch 34/50
```

```
88/88 [==============================] - 5s 55ms/step - loss: 0.0743 - accurac
y: 0.9834 - val_loss: 1.7262 - val_accuracy: 0.6425
Epoch 35/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0712 - accurac
y: 0.9833 - val_loss: 1.7010 - val_accuracy: 0.6384
Epoch 36/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0673 - accurac
y: 0.9835 - val_loss: 1.7071 - val_accuracy: 0.6392
Epoch 37/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0714 - accurac
y: 0.9828 - val_loss: 1.7667 - val_accuracy: 0.6425
Epoch 38/50
88/88 [==============================] - 5s 62ms/step - loss: 0.0710 - accurac
y: 0.9827 - val_loss: 1.7274 - val_accuracy: 0.6408
Epoch 39/50
88/88 [==============================] - 5s 59ms/step - loss: 0.0684 - accurac
y: 0.9819 - val_loss: 1.7257 - val_accuracy: 0.6408
Epoch 40/50
88/88 [==============================] - 5s 61ms/step - loss: 0.0695 - accurac
y: 0.9830 - val_loss: 1.7514 - val_accuracy: 0.6263
Epoch 41/50
88/88 [==============================] - 8s 88ms/step - loss: 0.0789 - accurac
y: 0.9794 - val_loss: 1.7408 - val_accuracy: 0.6384
Epoch 42/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0627 - accurac
y: 0.9843 - val_loss: 1.7660 - val_accuracy: 0.6481
Epoch 43/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0626 - accurac
y: 0.9848 - val_loss: 1.7563 - val_accuracy: 0.6368
Epoch 44/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0607 - accurac
y: 0.9848 - val_loss: 1.8495 - val_accuracy: 0.6433
Epoch 45/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0589 - accurac
y: 0.9839 - val_loss: 1.8635 - val_accuracy: 0.6392
Epoch 46/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0608 - accurac
y: 0.9854 - val_loss: 1.8312 - val_accuracy: 0.6384
Epoch 47/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0587 - accurac
y: 0.9850 - val_loss: 1.7944 - val_accuracy: 0.6473
Epoch 48/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0576 - accurac
y: 0.9856 - val_loss: 1.9134 - val_accuracy: 0.6408
Epoch 49/50
88/88 [==============================] - 5s 55ms/step - loss: 0.0586 - accurac
y: 0.9859 - val_loss: 1.8236 - val_accuracy: 0.6416
Epoch 50/50
88/88 [==============================] - 5s 56ms/step - loss: 0.0570 - accurac
y: 0.9849 - val_loss: 1.8448 - val_accuracy: 0.6465
model prediction time is 0.3434257507324219 seconds
Number of training records: 11148
Number of testing records: 1239
Training time 250.28
train accuracy : 0.986275565123789
test accuracy :0.6464891041162227
F1 score : 0.5690549189891663
Recall Score : 0.5279558390166799
Precision Score : 0.6603429870095937
```

Raw Analysis

In [ ]:

```
from google.colab import drive
drive.mount('/content/drive')
%cd "/content/drive/My Drive/Data/"
TRAIN_DATA='/content/drive/My Drive/Data/input_data.xlsx'
number_of_classes = 74
train(TRAIN_DATA, batch_size=128)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call d
rive.mount("/content/drive", force_remount=True).
/content/drive/My Drive/Data
22463
Model: "sequential_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_7 (Embedding)      (None, 25, 300)           6738900   
_________________________________________________________________
flatten_15 (Flatten)         (None, 7500)              0         
_________________________________________________________________
dense_45 (Dense)             (None, 500)               3750500   
_________________________________________________________________
dense_46 (Dense)             (None, 100)               50100     
_________________________________________________________________
dense_47 (Dense)             (None, 74)                7474      
=================================================================
Total params: 10,546,974
Trainable params: 3,808,074
Non-trainable params: 6,738,900
_________________________________________________________________
None
Epoch 1/50
60/60 [==============================] - 4s 59ms/step - loss: 2.9744 - accurac
y: 0.4984 - val_loss: 2.4699 - val_accuracy: 0.5047
Epoch 2/50
60/60 [==============================] - 3s 55ms/step - loss: 2.0806 - accurac
y: 0.5762 - val_loss: 2.1981 - val_accuracy: 0.5282
Epoch 3/50
60/60 [==============================] - 3s 55ms/step - loss: 1.7444 - accurac
y: 0.6136 - val_loss: 2.0337 - val_accuracy: 0.5671
Epoch 4/50
60/60 [==============================] - 3s 55ms/step - loss: 1.4762 - accurac
y: 0.6535 - val_loss: 1.9349 - val_accuracy: 0.5824
Epoch 5/50
60/60 [==============================] - 3s 54ms/step - loss: 1.2446 - accurac
y: 0.7038 - val_loss: 1.8722 - val_accuracy: 0.5824
Epoch 6/50
60/60 [==============================] - 3s 57ms/step - loss: 1.0445 - accurac
y: 0.7614 - val_loss: 1.8482 - val_accuracy: 0.5871
Epoch 7/50
60/60 [==============================] - 4s 59ms/step - loss: 0.8733 - accurac
y: 0.8076 - val_loss: 1.8169 - val_accuracy: 0.5929
Epoch 8/50
60/60 [==============================] - 4s 61ms/step - loss: 0.7313 - accurac
y: 0.8443 - val_loss: 1.8068 - val_accuracy: 0.5988
Epoch 9/50
60/60 [==============================] - 3s 56ms/step - loss: 0.6131 - accurac
y: 0.8715 - val_loss: 1.8113 - val_accuracy: 0.5953
Epoch 10/50
60/60 [==============================] - 4s 60ms/step - loss: 0.5208 - accurac
y: 0.8932 - val_loss: 1.8217 - val_accuracy: 0.6047
Epoch 11/50
60/60 [==============================] - 4s 61ms/step - loss: 0.4500 - accurac
y: 0.9125 - val_loss: 1.8454 - val_accuracy: 0.6000
Epoch 12/50
60/60 [==============================] - 3s 56ms/step - loss: 0.3944 - accurac
y: 0.9180 - val_loss: 1.8890 - val_accuracy: 0.5918
Epoch 13/50
60/60 [==============================] - 3s 55ms/step - loss: 0.3505 - accurac
```

```
y: 0.9278 - val_loss: 1.8952 - val_accuracy: 0.5906
Epoch 14/50
60/60 [==============================] - 3s 54ms/step - loss: 0.3160 - accurac
y: 0.9339 - val_loss: 1.9149 - val_accuracy: 0.5953
Epoch 15/50
60/60 [==============================] - 3s 54ms/step - loss: 0.2879 - accurac
y: 0.9388 - val_loss: 1.9476 - val_accuracy: 0.6000
Epoch 16/50
60/60 [==============================] - 3s 54ms/step - loss: 0.2660 - accurac
y: 0.9437 - val_loss: 1.9603 - val_accuracy: 0.6000
Epoch 17/50
60/60 [==============================] - 3s 55ms/step - loss: 0.2466 - accurac
y: 0.9451 - val_loss: 1.9867 - val_accuracy: 0.5988
Epoch 18/50
60/60 [==============================] - 3s 54ms/step - loss: 0.2310 - accurac
y: 0.9486 - val_loss: 2.0304 - val_accuracy: 0.5894
Epoch 19/50
60/60 [==============================] - 3s 55ms/step - loss: 0.2169 - accurac
y: 0.9501 - val_loss: 2.0456 - val_accuracy: 0.5941
Epoch 20/50
60/60 [==============================] - 3s 55ms/step - loss: 0.2030 - accurac
y: 0.9545 - val_loss: 2.0623 - val_accuracy: 0.5871
Epoch 21/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1912 - accurac
y: 0.9563 - val_loss: 2.0846 - val_accuracy: 0.5929
Epoch 22/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1848 - accurac
y: 0.9579 - val_loss: 2.0984 - val_accuracy: 0.5871
Epoch 23/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1768 - accurac
y: 0.9596 - val_loss: 2.1232 - val_accuracy: 0.5929
Epoch 24/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1695 - accurac
y: 0.9601 - val_loss: 2.1452 - val_accuracy: 0.5812
Epoch 25/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1642 - accurac
y: 0.9624 - val_loss: 2.1578 - val_accuracy: 0.5835
Epoch 26/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1577 - accurac
y: 0.9626 - val_loss: 2.1743 - val_accuracy: 0.5812
Epoch 27/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1556 - accurac
y: 0.9627 - val_loss: 2.1685 - val_accuracy: 0.5882
Epoch 28/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1522 - accurac
y: 0.9635 - val_loss: 2.1858 - val_accuracy: 0.5871
Epoch 29/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1473 - accurac
y: 0.9637 - val_loss: 2.2216 - val_accuracy: 0.5847
Epoch 30/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1395 - accurac
y: 0.9665 - val_loss: 2.2869 - val_accuracy: 0.5882
Epoch 31/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1366 - accurac
y: 0.9663 - val_loss: 2.2357 - val_accuracy: 0.5824
Epoch 32/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1375 - accurac
y: 0.9664 - val_loss: 2.2325 - val_accuracy: 0.5906
Epoch 33/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1290 - accurac
y: 0.9671 - val_loss: 2.2986 - val_accuracy: 0.5941
Epoch 34/50
```

```
60/60 [==============================] - 3s 57ms/step - loss: 0.1300 - accurac
y: 0.9672 - val_loss: 2.3398 - val_accuracy: 0.5918
Epoch 35/50
60/60 [==============================] - 4s 62ms/step - loss: 0.1256 - accurac
y: 0.9680 - val_loss: 2.3207 - val_accuracy: 0.5835
Epoch 36/50
60/60 [==============================] - 4s 62ms/step - loss: 0.1204 - accurac
y: 0.9692 - val_loss: 2.2933 - val_accuracy: 0.5859
Epoch 37/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1242 - accurac
y: 0.9681 - val_loss: 2.3448 - val_accuracy: 0.5812
Epoch 38/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1175 - accurac
y: 0.9707 - val_loss: 2.4170 - val_accuracy: 0.5953
Epoch 39/50
60/60 [==============================] - 3s 56ms/step - loss: 0.1168 - accurac
y: 0.9673 - val_loss: 2.4096 - val_accuracy: 0.5929
Epoch 40/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1146 - accurac
y: 0.9697 - val_loss: 2.4707 - val_accuracy: 0.5882
Epoch 41/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1099 - accurac
y: 0.9725 - val_loss: 2.3494 - val_accuracy: 0.5871
Epoch 42/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1101 - accurac
y: 0.9703 - val_loss: 2.4214 - val_accuracy: 0.5882
Epoch 43/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1135 - accurac
y: 0.9694 - val_loss: 2.4120 - val_accuracy: 0.5847
Epoch 44/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1089 - accurac
y: 0.9697 - val_loss: 2.3898 - val_accuracy: 0.5847
Epoch 45/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1053 - accurac
y: 0.9718 - val_loss: 2.4668 - val_accuracy: 0.5824
Epoch 46/50
60/60 [==============================] - 3s 56ms/step - loss: 0.1031 - accurac
y: 0.9722 - val_loss: 2.4855 - val_accuracy: 0.5824
Epoch 47/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1066 - accurac
y: 0.9712 - val_loss: 2.4802 - val_accuracy: 0.5859
Epoch 48/50
60/60 [==============================] - 3s 55ms/step - loss: 0.1011 - accurac
y: 0.9744 - val_loss: 2.4538 - val_accuracy: 0.5871
Epoch 49/50
60/60 [==============================] - 3s 55ms/step - loss: 0.0987 - accurac
y: 0.9737 - val_loss: 2.5297 - val_accuracy: 0.5835
Epoch 50/50
60/60 [==============================] - 3s 56ms/step - loss: 0.1006 - accurac
y: 0.9732 - val_loss: 2.4579 - val_accuracy: 0.5894
model prediction time is 0.24892354011535645 seconds
Number of training records: 7650
Number of testing records: 850
Training time 174.67
train accuracy : 0.9769934640522876
test accuracy :0.5894117647058823
F1 score : 0.20197395275037586
Recall Score : 0.19673882725657532
Precision Score : 0.25022535968898674
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Hyperparam Analysis

In [ ]:

```
number_of_classes = 30
tf.autograph.set_verbosity(0) #issue with logging
TRAIN_DATA='/content/drive/My Drive/Data/processed_data_csv_10_10_2020'
hyper_train(TRAIN_DATA)
```

```
Training model on hyperparameters
Epoch 1/50
25/25 - 5s - loss: nan - accuracy: 0.2689 - val_loss: nan - val_accuracy: 0.258
3

Epoch 2/50
25/25 - 5s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 3/50
25/25 - 6s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 4/50
25/25 - 6s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 5/50
25/25 - 5s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 6/50
25/25 - 5s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 7/50
25/25 - 5s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 8/50
25/25 - 5s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 9/50
25/25 - 5s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 00009: early stopping
39/39 - 0s

Train Accuracy [nan, 0.2788841128349304] , Test Accuracy [nan, 0.25827279686927
795]
Epoch 1/50
44/44 - 7s - loss: nan - accuracy: 0.2732 - val_loss: nan - val_accuracy: 0.258
3

Epoch 2/50
44/44 - 7s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 3/50
44/44 - 7s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 4/50
44/44 - 7s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 5/50
44/44 - 7s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3
```

```
Epoch 6/50
44/44 - 7s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 7/50
44/44 - 7s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 8/50
44/44 - 8s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 9/50
44/44 - 7s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 00009: early stopping
39/39 - 1s

Train Accuracy [nan, 0.2788841128349304] , Test Accuracy [nan, 0.25827279686927
795]
Epoch 1/50
30/30 - 6s - loss: 2.5337 - accuracy: 0.2876 - val_loss: 2.1885 - val_accuracy:
0.2583

Epoch 2/50
30/30 - 6s - loss: 1.8636 - accuracy: 0.2827 - val_loss: 2.0523 - val_accuracy:
0.2583

Epoch 3/50
30/30 - 6s - loss: 1.6739 - accuracy: 0.2828 - val_loss: 1.9886 - val_accuracy:
0.2583

Epoch 4/50
30/30 - 6s - loss: 1.5744 - accuracy: 0.2831 - val_loss: 1.9497 - val_accuracy:
0.2583

Epoch 5/50
30/30 - 6s - loss: 1.5069 - accuracy: 0.2831 - val_loss: 1.9260 - val_accuracy:
0.2583

Epoch 6/50
30/30 - 6s - loss: 1.4576 - accuracy: 0.2836 - val_loss: 1.9116 - val_accuracy:
0.2583

Epoch 7/50
30/30 - 6s - loss: 1.4193 - accuracy: 0.2834 - val_loss: 1.8964 - val_accuracy:
0.2583

Epoch 8/50
30/30 - 6s - loss: 1.3897 - accuracy: 0.2836 - val_loss: 1.8899 - val_accuracy:
0.2591

Epoch 9/50
30/30 - 6s - loss: 1.3626 - accuracy: 0.2838 - val_loss: 1.8794 - val_accuracy:
0.2591

Epoch 10/50
30/30 - 6s - loss: 1.3379 - accuracy: 0.2845 - val_loss: 1.8791 - val_accuracy:
0.2599
```

```
Epoch 11/50
30/30 - 6s - loss: 1.3206 - accuracy: 0.2842 - val_loss: 1.8659 - val_accuracy:
0.2599

Epoch 12/50
30/30 - 6s - loss: 1.3043 - accuracy: 0.2848 - val_loss: 1.8599 - val_accuracy:
0.2599

Epoch 13/50
30/30 - 6s - loss: 1.2880 - accuracy: 0.2860 - val_loss: 1.8568 - val_accuracy:
0.2599

Epoch 14/50
30/30 - 6s - loss: 1.2731 - accuracy: 0.2861 - val_loss: 1.8546 - val_accuracy:
0.2599

Epoch 15/50
30/30 - 6s - loss: 1.2602 - accuracy: 0.2870 - val_loss: 1.8482 - val_accuracy:
0.2599

Epoch 16/50
30/30 - 6s - loss: 1.2477 - accuracy: 0.2865 - val_loss: 1.8493 - val_accuracy:
0.2599

Epoch 17/50
30/30 - 6s - loss: 1.2377 - accuracy: 0.2869 - val_loss: 1.8451 - val_accuracy:
0.2599

Epoch 18/50
30/30 - 6s - loss: 1.2275 - accuracy: 0.2881 - val_loss: 1.8434 - val_accuracy:
0.2615

Epoch 19/50
30/30 - 6s - loss: 1.2189 - accuracy: 0.2907 - val_loss: 1.8407 - val_accuracy:
0.2631

Epoch 20/50
30/30 - 6s - loss: 1.2130 - accuracy: 0.2936 - val_loss: 1.8376 - val_accuracy:
0.2639

Epoch 21/50
30/30 - 6s - loss: 1.2028 - accuracy: 0.2949 - val_loss: 1.8380 - val_accuracy:
0.2639

Epoch 22/50
30/30 - 6s - loss: 1.1965 - accuracy: 0.2972 - val_loss: 1.8348 - val_accuracy:
0.2655

Epoch 23/50
30/30 - 6s - loss: 1.1870 - accuracy: 0.2983 - val_loss: 1.8348 - val_accuracy:
0.2655

Epoch 24/50
30/30 - 6s - loss: 1.1812 - accuracy: 0.2998 - val_loss: 1.8327 - val_accuracy:
0.2760

Epoch 25/50
30/30 - 6s - loss: 1.1752 - accuracy: 0.2998 - val_loss: 1.8323 - val_accuracy:
0.2776

Epoch 26/50
30/30 - 6s - loss: 1.1707 - accuracy: 0.3010 - val_loss: 1.8306 - val_accuracy:
```

```
0.2776

Epoch 27/50
30/30 - 6s - loss: 1.1649 - accuracy: 0.3021 - val_loss: 1.8283 - val_accuracy:
0.2817

Epoch 28/50
30/30 - 6s - loss: 1.1592 - accuracy: 0.3051 - val_loss: 1.8264 - val_accuracy:
0.2857

Epoch 29/50
30/30 - 6s - loss: 1.1539 - accuracy: 0.3057 - val_loss: 1.8265 - val_accuracy:
0.2857

Epoch 30/50
30/30 - 6s - loss: 1.1473 - accuracy: 0.3093 - val_loss: 1.8261 - val_accuracy:
0.2857

Epoch 31/50
30/30 - 7s - loss: 1.1437 - accuracy: 0.3105 - val_loss: 1.8257 - val_accuracy:
0.2857

Epoch 32/50
30/30 - 7s - loss: 1.1361 - accuracy: 0.3109 - val_loss: 1.8255 - val_accuracy:
0.2857

Epoch 33/50
30/30 - 7s - loss: 1.1338 - accuracy: 0.3140 - val_loss: 1.8242 - val_accuracy:
0.2873

Epoch 34/50
30/30 - 6s - loss: 1.1288 - accuracy: 0.3163 - val_loss: 1.8224 - val_accuracy:
0.2889

Epoch 35/50
30/30 - 6s - loss: 1.1261 - accuracy: 0.3208 - val_loss: 1.8216 - val_accuracy:
0.2889

Epoch 36/50
30/30 - 6s - loss: 1.1224 - accuracy: 0.3227 - val_loss: 1.8205 - val_accuracy:
0.2938

Epoch 37/50
30/30 - 6s - loss: 1.1184 - accuracy: 0.3253 - val_loss: 1.8203 - val_accuracy:
0.2962

Epoch 38/50
30/30 - 6s - loss: 1.1151 - accuracy: 0.3275 - val_loss: 1.8179 - val_accuracy:
0.2970

Epoch 39/50
30/30 - 6s - loss: 1.1104 - accuracy: 0.3297 - val_loss: 1.8180 - val_accuracy:
0.2986

Epoch 40/50
30/30 - 6s - loss: 1.1076 - accuracy: 0.3327 - val_loss: 1.8183 - val_accuracy:
0.3002

Epoch 41/50
30/30 - 6s - loss: 1.1034 - accuracy: 0.3369 - val_loss: 1.8172 - val_accuracy:
0.3010
```

```
Epoch 42/50
30/30 - 6s - loss: 1.1018 - accuracy: 0.3376 - val_loss: 1.8163 - val_accuracy:
0.3067

Epoch 43/50
30/30 - 6s - loss: 1.0947 - accuracy: 0.3392 - val_loss: 1.8156 - val_accuracy:
0.3091

Epoch 44/50
30/30 - 6s - loss: 1.0947 - accuracy: 0.3435 - val_loss: 1.8154 - val_accuracy:
0.3164

Epoch 45/50
30/30 - 6s - loss: 1.0907 - accuracy: 0.3480 - val_loss: 1.8154 - val_accuracy:
0.3180

Epoch 46/50
30/30 - 6s - loss: 1.0885 - accuracy: 0.3497 - val_loss: 1.8125 - val_accuracy:
0.3212

Epoch 47/50
30/30 - 6s - loss: 1.0860 - accuracy: 0.3541 - val_loss: 1.8128 - val_accuracy:
0.3245

Epoch 48/50
30/30 - 6s - loss: 1.0841 - accuracy: 0.3557 - val_loss: 1.8143 - val_accuracy:
0.3269

Epoch 49/50
30/30 - 6s - loss: 1.0809 - accuracy: 0.3593 - val_loss: 1.8138 - val_accuracy:
0.3309

Epoch 50/50
30/30 - 6s - loss: 1.0777 - accuracy: 0.3611 - val_loss: 1.8147 - val_accuracy:
0.3317

39/39 - 1s

Train Accuracy [1.0578171014785767, 0.3658055365085602] , Test Accuracy [1.8146
969079971313, 0.33171913027763367]
Epoch 1/50
44/44 - 5s - loss: 2.2092 - accuracy: 0.4177 - val_loss: 1.9875 - val_accuracy:
0.4326

Epoch 2/50
44/44 - 5s - loss: 1.6189 - accuracy: 0.5587 - val_loss: 1.8432 - val_accuracy:
0.4625

Epoch 3/50
44/44 - 5s - loss: 1.4175 - accuracy: 0.6220 - val_loss: 1.7636 - val_accuracy:
0.4907

Epoch 4/50
44/44 - 5s - loss: 1.2885 - accuracy: 0.6712 - val_loss: 1.7131 - val_accuracy:
0.5077

Epoch 5/50
44/44 - 5s - loss: 1.1999 - accuracy: 0.6973 - val_loss: 1.6785 - val_accuracy:
0.5206

Epoch 6/50
44/44 - 5s - loss: 1.1325 - accuracy: 0.7174 - val_loss: 1.6557 - val_accuracy:
```

```
0.5262

Epoch 7/50
44/44 - 5s - loss: 1.0832 - accuracy: 0.7326 - val_loss: 1.6287 - val_accuracy:
0.5383

Epoch 8/50
44/44 - 5s - loss: 1.0346 - accuracy: 0.7491 - val_loss: 1.6110 - val_accuracy:
0.5416

Epoch 9/50
44/44 - 5s - loss: 0.9982 - accuracy: 0.7606 - val_loss: 1.5969 - val_accuracy:
0.5440

Epoch 10/50
44/44 - 5s - loss: 0.9676 - accuracy: 0.7696 - val_loss: 1.5820 - val_accuracy:
0.5504

Epoch 11/50
44/44 - 5s - loss: 0.9405 - accuracy: 0.7806 - val_loss: 1.5743 - val_accuracy:
0.5472

Epoch 12/50
44/44 - 5s - loss: 0.9134 - accuracy: 0.7882 - val_loss: 1.5627 - val_accuracy:
0.5504

Epoch 13/50
44/44 - 5s - loss: 0.8946 - accuracy: 0.7943 - val_loss: 1.5549 - val_accuracy:
0.5504

Epoch 14/50
44/44 - 5s - loss: 0.8747 - accuracy: 0.7984 - val_loss: 1.5464 - val_accuracy:
0.5569

Epoch 15/50
44/44 - 5s - loss: 0.8585 - accuracy: 0.8015 - val_loss: 1.5384 - val_accuracy:
0.5593

Epoch 16/50
44/44 - 5s - loss: 0.8435 - accuracy: 0.8094 - val_loss: 1.5333 - val_accuracy:
0.5626

Epoch 17/50
44/44 - 5s - loss: 0.8275 - accuracy: 0.8128 - val_loss: 1.5265 - val_accuracy:
0.5650

Epoch 18/50
44/44 - 5s - loss: 0.8143 - accuracy: 0.8165 - val_loss: 1.5205 - val_accuracy:
0.5658

Epoch 19/50
44/44 - 5s - loss: 0.8005 - accuracy: 0.8183 - val_loss: 1.5147 - val_accuracy:
0.5658

Epoch 20/50
44/44 - 5s - loss: 0.7926 - accuracy: 0.8201 - val_loss: 1.5114 - val_accuracy:
0.5666

Epoch 21/50
44/44 - 5s - loss: 0.7778 - accuracy: 0.8241 - val_loss: 1.5080 - val_accuracy:
0.5674
```

```
Epoch 22/50
44/44 - 5s - loss: 0.7672 - accuracy: 0.8280 - val_loss: 1.5021 - val_accuracy:
0.5698

Epoch 23/50
44/44 - 5s - loss: 0.7615 - accuracy: 0.8287 - val_loss: 1.4987 - val_accuracy:
0.5706

Epoch 24/50
44/44 - 5s - loss: 0.7496 - accuracy: 0.8343 - val_loss: 1.4955 - val_accuracy:
0.5730

Epoch 25/50
44/44 - 5s - loss: 0.7404 - accuracy: 0.8367 - val_loss: 1.4933 - val_accuracy:
0.5738

Epoch 26/50
44/44 - 5s - loss: 0.7319 - accuracy: 0.8422 - val_loss: 1.4901 - val_accuracy:
0.5738

Epoch 27/50
44/44 - 5s - loss: 0.7263 - accuracy: 0.8393 - val_loss: 1.4872 - val_accuracy:
0.5747

Epoch 28/50
44/44 - 5s - loss: 0.7161 - accuracy: 0.8419 - val_loss: 1.4828 - val_accuracy:
0.5755

Epoch 29/50
44/44 - 5s - loss: 0.7103 - accuracy: 0.8437 - val_loss: 1.4811 - val_accuracy:
0.5747

Epoch 30/50
44/44 - 5s - loss: 0.7042 - accuracy: 0.8447 - val_loss: 1.4772 - val_accuracy:
0.5779

Epoch 31/50
44/44 - 5s - loss: 0.6970 - accuracy: 0.8497 - val_loss: 1.4756 - val_accuracy:
0.5787

Epoch 32/50
44/44 - 5s - loss: 0.6891 - accuracy: 0.8507 - val_loss: 1.4721 - val_accuracy:
0.5827

Epoch 33/50
44/44 - 5s - loss: 0.6839 - accuracy: 0.8530 - val_loss: 1.4702 - val_accuracy:
0.5811

Epoch 34/50
44/44 - 5s - loss: 0.6812 - accuracy: 0.8536 - val_loss: 1.4682 - val_accuracy:
0.5811

Epoch 35/50
44/44 - 5s - loss: 0.6760 - accuracy: 0.8511 - val_loss: 1.4661 - val_accuracy:
0.5811

Epoch 36/50
44/44 - 5s - loss: 0.6692 - accuracy: 0.8547 - val_loss: 1.4647 - val_accuracy:
0.5819

Epoch 37/50
44/44 - 5s - loss: 0.6640 - accuracy: 0.8559 - val_loss: 1.4621 - val_accuracy:
```

```
0.5811

Epoch 38/50
44/44 - 5s - loss: 0.6598 - accuracy: 0.8574 - val_loss: 1.4602 - val_accuracy:
0.5827

Epoch 39/50
44/44 - 5s - loss: 0.6536 - accuracy: 0.8586 - val_loss: 1.4580 - val_accuracy:
0.5827

Epoch 40/50
44/44 - 5s - loss: 0.6505 - accuracy: 0.8591 - val_loss: 1.4564 - val_accuracy:
0.5827

Epoch 00040: early stopping
39/39 - 0s

Train Accuracy [0.6182645559310913, 0.8693935871124268] , Test Accuracy [1.4563
502073287964, 0.5827280282974243]
Epoch 1/50
35/35 - 6s - loss: 2.4613 - accuracy: 0.2827 - val_loss: 2.1781 - val_accuracy:
0.2583

Epoch 2/50
35/35 - 6s - loss: 1.8165 - accuracy: 0.2789 - val_loss: 2.0332 - val_accuracy:
0.2583

Epoch 3/50
35/35 - 6s - loss: 1.6306 - accuracy: 0.2789 - val_loss: 1.9690 - val_accuracy:
0.2583

Epoch 4/50
35/35 - 6s - loss: 1.5283 - accuracy: 0.2789 - val_loss: 1.9329 - val_accuracy:
0.2583

Epoch 5/50
35/35 - 6s - loss: 1.4573 - accuracy: 0.2789 - val_loss: 1.9083 - val_accuracy:
0.2583

Epoch 6/50
35/35 - 5s - loss: 1.4059 - accuracy: 0.2789 - val_loss: 1.8936 - val_accuracy:
0.2583

Epoch 7/50
35/35 - 6s - loss: 1.3667 - accuracy: 0.2789 - val_loss: 1.8797 - val_accuracy:
0.2583

Epoch 8/50
35/35 - 5s - loss: 1.3330 - accuracy: 0.2789 - val_loss: 1.8689 - val_accuracy:
0.2583

Epoch 9/50
35/35 - 5s - loss: 1.3072 - accuracy: 0.2789 - val_loss: 1.8612 - val_accuracy:
0.2583

Epoch 00009: early stopping
39/39 - 0s

Train Accuracy [1.268911600112915, 0.2788841128349304] , Test Accuracy [1.86122
4889755249, 0.25827279686927795]
Epoch 1/50
30/30 - 5s - loss: 2.2295 - accuracy: 0.4126 - val_loss: 1.9997 - val_accuracy:
```

```
0.4350

Epoch 2/50
30/30 - 5s - loss: 1.6247 - accuracy: 0.5536 - val_loss: 1.8580 - val_accuracy:
0.4778

Epoch 3/50
30/30 - 6s - loss: 1.4220 - accuracy: 0.6186 - val_loss: 1.7843 - val_accuracy:
0.4972

Epoch 4/50
30/30 - 6s - loss: 1.2973 - accuracy: 0.6610 - val_loss: 1.7366 - val_accuracy:
0.5052

Epoch 5/50
30/30 - 5s - loss: 1.2068 - accuracy: 0.6909 - val_loss: 1.7001 - val_accuracy:
0.5149

Epoch 6/50
30/30 - 5s - loss: 1.1394 - accuracy: 0.7076 - val_loss: 1.6741 - val_accuracy:
0.5246

Epoch 7/50
30/30 - 5s - loss: 1.0854 - accuracy: 0.7306 - val_loss: 1.6533 - val_accuracy:
0.5295

Epoch 8/50
30/30 - 5s - loss: 1.0449 - accuracy: 0.7464 - val_loss: 1.6346 - val_accuracy:
0.5424

Epoch 9/50
30/30 - 5s - loss: 1.0061 - accuracy: 0.7568 - val_loss: 1.6215 - val_accuracy:
0.5448

Epoch 10/50
30/30 - 5s - loss: 0.9720 - accuracy: 0.7649 - val_loss: 1.6073 - val_accuracy:
0.5440

Epoch 11/50
30/30 - 5s - loss: 0.9451 - accuracy: 0.7760 - val_loss: 1.5996 - val_accuracy:
0.5545

Epoch 12/50
30/30 - 5s - loss: 0.9192 - accuracy: 0.7816 - val_loss: 1.5849 - val_accuracy:
0.5545

Epoch 13/50
30/30 - 5s - loss: 0.8977 - accuracy: 0.7920 - val_loss: 1.5757 - val_accuracy:
0.5569

Epoch 14/50
30/30 - 5s - loss: 0.8781 - accuracy: 0.7955 - val_loss: 1.5700 - val_accuracy:
0.5617

Epoch 15/50
30/30 - 5s - loss: 0.8635 - accuracy: 0.7983 - val_loss: 1.5607 - val_accuracy:
0.5634

Epoch 16/50
30/30 - 5s - loss: 0.8457 - accuracy: 0.8042 - val_loss: 1.5531 - val_accuracy:
0.5642
```

```
Epoch 17/50
30/30 - 5s - loss: 0.8279 - accuracy: 0.8113 - val_loss: 1.5456 - val_accuracy:
0.5658

Epoch 18/50
30/30 - 5s - loss: 0.8165 - accuracy: 0.8148 - val_loss: 1.5395 - val_accuracy:
0.5698

Epoch 19/50
30/30 - 5s - loss: 0.8039 - accuracy: 0.8152 - val_loss: 1.5356 - val_accuracy:
0.5706

Epoch 20/50
30/30 - 5s - loss: 0.7911 - accuracy: 0.8200 - val_loss: 1.5338 - val_accuracy:
0.5747

Epoch 21/50
30/30 - 5s - loss: 0.7807 - accuracy: 0.8222 - val_loss: 1.5290 - val_accuracy:
0.5722

Epoch 22/50
30/30 - 5s - loss: 0.7695 - accuracy: 0.8274 - val_loss: 1.5252 - val_accuracy:
0.5714

Epoch 23/50
30/30 - 5s - loss: 0.7621 - accuracy: 0.8299 - val_loss: 1.5207 - val_accuracy:
0.5698

Epoch 24/50
30/30 - 5s - loss: 0.7504 - accuracy: 0.8333 - val_loss: 1.5166 - val_accuracy:
0.5714

Epoch 25/50
30/30 - 5s - loss: 0.7414 - accuracy: 0.8313 - val_loss: 1.5137 - val_accuracy:
0.5706

Epoch 26/50
30/30 - 5s - loss: 0.7347 - accuracy: 0.8364 - val_loss: 1.5105 - val_accuracy:
0.5747

Epoch 27/50
30/30 - 5s - loss: 0.7258 - accuracy: 0.8403 - val_loss: 1.5073 - val_accuracy:
0.5722

Epoch 28/50
30/30 - 5s - loss: 0.7188 - accuracy: 0.8405 - val_loss: 1.5038 - val_accuracy:
0.5747

Epoch 00028: early stopping
39/39 - 0s

Train Accuracy [0.6867393851280212, 0.8514531850814819] , Test Accuracy [1.5038
059949874878, 0.5746569633483887]
Epoch 1/50
88/88 - 8s - loss: 2.0811 - accuracy: 0.4440 - val_loss: 1.9173 - val_accuracy:
0.4689

Epoch 2/50
88/88 - 8s - loss: 1.4789 - accuracy: 0.5978 - val_loss: 1.7985 - val_accuracy:
0.4923

Epoch 3/50
```

```
88/88 - 8s - loss: 1.3009 - accuracy: 0.6604 - val_loss: 1.7462 - val_accuracy:
0.5020

Epoch 4/50
88/88 - 8s - loss: 1.1963 - accuracy: 0.6943 - val_loss: 1.6986 - val_accuracy:
0.5133

Epoch 5/50
88/88 - 8s - loss: 1.1231 - accuracy: 0.7209 - val_loss: 1.6716 - val_accuracy:
0.5238

Epoch 6/50
88/88 - 8s - loss: 1.0673 - accuracy: 0.7392 - val_loss: 1.6511 - val_accuracy:
0.5254

Epoch 7/50
88/88 - 8s - loss: 1.0234 - accuracy: 0.7553 - val_loss: 1.6325 - val_accuracy:
0.5303

Epoch 8/50
88/88 - 8s - loss: 0.9873 - accuracy: 0.7650 - val_loss: 1.6187 - val_accuracy:
0.5303

Epoch 9/50
88/88 - 8s - loss: 0.9563 - accuracy: 0.7757 - val_loss: 1.6057 - val_accuracy:
0.5351

Epoch 10/50
88/88 - 8s - loss: 0.9300 - accuracy: 0.7819 - val_loss: 1.5936 - val_accuracy:
0.5416

Epoch 11/50
88/88 - 8s - loss: 0.9068 - accuracy: 0.7903 - val_loss: 1.5857 - val_accuracy:
0.5400

Epoch 12/50
88/88 - 9s - loss: 0.8864 - accuracy: 0.7966 - val_loss: 1.5753 - val_accuracy:
0.5504

Epoch 13/50
88/88 - 9s - loss: 0.8679 - accuracy: 0.8018 - val_loss: 1.5693 - val_accuracy:
0.5537

Epoch 14/50
88/88 - 9s - loss: 0.8516 - accuracy: 0.8060 - val_loss: 1.5619 - val_accuracy:
0.5545

Epoch 15/50
88/88 - 8s - loss: 0.8365 - accuracy: 0.8096 - val_loss: 1.5554 - val_accuracy:
0.5601

Epoch 16/50
88/88 - 8s - loss: 0.8227 - accuracy: 0.8148 - val_loss: 1.5499 - val_accuracy:
0.5617

Epoch 17/50
88/88 - 8s - loss: 0.8099 - accuracy: 0.8185 - val_loss: 1.5439 - val_accuracy:
0.5626

Epoch 18/50
88/88 - 8s - loss: 0.7981 - accuracy: 0.8228 - val_loss: 1.5400 - val_accuracy:
0.5642
```

```
Epoch 19/50
88/88 - 8s - loss: 0.7871 - accuracy: 0.8252 - val_loss: 1.5353 - val_accuracy:
0.5634

Epoch 20/50
88/88 - 9s - loss: 0.7769 - accuracy: 0.8279 - val_loss: 1.5311 - val_accuracy:
0.5666

Epoch 21/50
88/88 - 8s - loss: 0.7673 - accuracy: 0.8314 - val_loss: 1.5269 - val_accuracy:
0.5682

Epoch 22/50
88/88 - 8s - loss: 0.7583 - accuracy: 0.8333 - val_loss: 1.5228 - val_accuracy:
0.5698

Epoch 23/50
88/88 - 8s - loss: 0.7499 - accuracy: 0.8364 - val_loss: 1.5195 - val_accuracy:
0.5722

Epoch 24/50
88/88 - 8s - loss: 0.7419 - accuracy: 0.8392 - val_loss: 1.5157 - val_accuracy:
0.5714

Epoch 25/50
88/88 - 8s - loss: 0.7343 - accuracy: 0.8410 - val_loss: 1.5124 - val_accuracy:
0.5738

Epoch 26/50
88/88 - 8s - loss: 0.7271 - accuracy: 0.8434 - val_loss: 1.5099 - val_accuracy:
0.5763

Epoch 27/50
88/88 - 8s - loss: 0.7202 - accuracy: 0.8445 - val_loss: 1.5062 - val_accuracy:
0.5763

Epoch 28/50
88/88 - 8s - loss: 0.7137 - accuracy: 0.8463 - val_loss: 1.5038 - val_accuracy:
0.5755

Epoch 29/50
88/88 - 8s - loss: 0.7074 - accuracy: 0.8480 - val_loss: 1.5010 - val_accuracy:
0.5779

Epoch 30/50
88/88 - 8s - loss: 0.7014 - accuracy: 0.8489 - val_loss: 1.4987 - val_accuracy:
0.5795

Epoch 31/50
88/88 - 8s - loss: 0.6957 - accuracy: 0.8505 - val_loss: 1.4958 - val_accuracy:
0.5803

Epoch 32/50
88/88 - 8s - loss: 0.6903 - accuracy: 0.8522 - val_loss: 1.4932 - val_accuracy:
0.5787

Epoch 33/50
88/88 - 8s - loss: 0.6849 - accuracy: 0.8531 - val_loss: 1.4909 - val_accuracy:
0.5795

Epoch 34/50
```

```
88/88 - 8s - loss: 0.6798 - accuracy: 0.8544 - val_loss: 1.4895 - val_accuracy:
0.5803

Epoch 35/50
88/88 - 8s - loss: 0.6748 - accuracy: 0.8557 - val_loss: 1.4869 - val_accuracy:
0.5795

Epoch 36/50
88/88 - 8s - loss: 0.6701 - accuracy: 0.8571 - val_loss: 1.4854 - val_accuracy:
0.5803

Epoch 37/50
88/88 - 8s - loss: 0.6655 - accuracy: 0.8581 - val_loss: 1.4836 - val_accuracy:
0.5803

Epoch 38/50
88/88 - 8s - loss: 0.6611 - accuracy: 0.8590 - val_loss: 1.4818 - val_accuracy:
0.5803

Epoch 39/50
88/88 - 8s - loss: 0.6568 - accuracy: 0.8599 - val_loss: 1.4801 - val_accuracy:
0.5819

Epoch 40/50
88/88 - 8s - loss: 0.6526 - accuracy: 0.8613 - val_loss: 1.4776 - val_accuracy:
0.5819

Epoch 41/50
88/88 - 8s - loss: 0.6485 - accuracy: 0.8617 - val_loss: 1.4768 - val_accuracy:
0.5819

Epoch 42/50
88/88 - 8s - loss: 0.6446 - accuracy: 0.8623 - val_loss: 1.4749 - val_accuracy:
0.5851

Epoch 43/50
88/88 - 9s - loss: 0.6408 - accuracy: 0.8629 - val_loss: 1.4730 - val_accuracy:
0.5843

Epoch 44/50
88/88 - 8s - loss: 0.6371 - accuracy: 0.8638 - val_loss: 1.4717 - val_accuracy:
0.5843

Epoch 45/50
88/88 - 8s - loss: 0.6335 - accuracy: 0.8646 - val_loss: 1.4700 - val_accuracy:
0.5860

Epoch 46/50
88/88 - 8s - loss: 0.6300 - accuracy: 0.8657 - val_loss: 1.4685 - val_accuracy:
0.5868

Epoch 47/50
88/88 - 8s - loss: 0.6266 - accuracy: 0.8668 - val_loss: 1.4673 - val_accuracy:
0.5884

Epoch 48/50
88/88 - 8s - loss: 0.6232 - accuracy: 0.8678 - val_loss: 1.4657 - val_accuracy:
0.5884

Epoch 49/50
88/88 - 8s - loss: 0.6200 - accuracy: 0.8689 - val_loss: 1.4646 - val_accuracy:
0.5876
```

```
Epoch 50/50
88/88 - 8s - loss: 0.6168 - accuracy: 0.8695 - val_loss: 1.4635 - val_accuracy:
0.5876

39/39 - 1s

Train Accuracy [0.6137406826019287, 0.8707391619682312] , Test Accuracy [1.4635
034799575806, 0.5875706076622009]
Epoch 1/50
25/25 - 6s - loss: 2.2371 - accuracy: 0.4065 - val_loss: 1.9835 - val_accuracy:
0.4350

Epoch 2/50
25/25 - 5s - loss: 1.5534 - accuracy: 0.5775 - val_loss: 1.8161 - val_accuracy:
0.4843

Epoch 3/50
25/25 - 5s - loss: 1.2934 - accuracy: 0.6594 - val_loss: 1.7175 - val_accuracy:
0.5085

Epoch 4/50
25/25 - 5s - loss: 1.1328 - accuracy: 0.7147 - val_loss: 1.6569 - val_accuracy:
0.5270

Epoch 5/50
25/25 - 5s - loss: 1.0238 - accuracy: 0.7520 - val_loss: 1.6122 - val_accuracy:
0.5400

Epoch 6/50
25/25 - 5s - loss: 0.9408 - accuracy: 0.7791 - val_loss: 1.5820 - val_accuracy:
0.5496

Epoch 7/50
25/25 - 5s - loss: 0.8806 - accuracy: 0.7991 - val_loss: 1.5605 - val_accuracy:
0.5513

Epoch 8/50
25/25 - 5s - loss: 0.8300 - accuracy: 0.8117 - val_loss: 1.5389 - val_accuracy:
0.5658

Epoch 9/50
25/25 - 7s - loss: 0.7889 - accuracy: 0.8235 - val_loss: 1.5234 - val_accuracy:
0.5714

Epoch 10/50
25/25 - 5s - loss: 0.7530 - accuracy: 0.8371 - val_loss: 1.5107 - val_accuracy:
0.5779

Epoch 11/50
25/25 - 5s - loss: 0.7266 - accuracy: 0.8414 - val_loss: 1.5015 - val_accuracy:
0.5851

Epoch 12/50
25/25 - 5s - loss: 0.7002 - accuracy: 0.8499 - val_loss: 1.4886 - val_accuracy:
0.5876

Epoch 13/50
25/25 - 5s - loss: 0.6785 - accuracy: 0.8539 - val_loss: 1.4787 - val_accuracy:
0.5884

Epoch 14/50
```

```
25/25 - 5s - loss: 0.6545 - accuracy: 0.8623 - val_loss: 1.4732 - val_accuracy: 0.5876

Epoch 15/50
25/25 - 5s - loss: 0.6387 - accuracy: 0.8654 - val_loss: 1.4654 - val_accuracy: 0.5916

Epoch 16/50
25/25 - 5s - loss: 0.6249 - accuracy: 0.8682 - val_loss: 1.4601 - val_accuracy: 0.5932

Epoch 17/50
25/25 - 5s - loss: 0.6096 - accuracy: 0.8701 - val_loss: 1.4566 - val_accuracy: 0.5989

Epoch 18/50
25/25 - 5s - loss: 0.5963 - accuracy: 0.8728 - val_loss: 1.4512 - val_accuracy: 0.5989

Epoch 19/50
25/25 - 5s - loss: 0.5829 - accuracy: 0.8787 - val_loss: 1.4459 - val_accuracy: 0.6005

Epoch 20/50
25/25 - 5s - loss: 0.5719 - accuracy: 0.8802 - val_loss: 1.4409 - val_accuracy: 0.6021

Epoch 21/50
25/25 - 5s - loss: 0.5625 - accuracy: 0.8837 - val_loss: 1.4383 - val_accuracy: 0.6037

Epoch 22/50
25/25 - 6s - loss: 0.5493 - accuracy: 0.8867 - val_loss: 1.4344 - val_accuracy: 0.6013

Epoch 23/50
25/25 - 5s - loss: 0.5450 - accuracy: 0.8840 - val_loss: 1.4318 - val_accuracy: 0.6045

Epoch 24/50
25/25 - 5s - loss: 0.5353 - accuracy: 0.8893 - val_loss: 1.4277 - val_accuracy: 0.6037

Epoch 25/50
25/25 - 5s - loss: 0.5244 - accuracy: 0.8907 - val_loss: 1.4232 - val_accuracy: 0.6029

Epoch 26/50
25/25 - 5s - loss: 0.5185 - accuracy: 0.8923 - val_loss: 1.4215 - val_accuracy: 0.6053

Epoch 27/50
25/25 - 5s - loss: 0.5129 - accuracy: 0.8928 - val_loss: 1.4186 - val_accuracy: 0.6053

Epoch 28/50
25/25 - 5s - loss: 0.5053 - accuracy: 0.8952 - val_loss: 1.4165 - val_accuracy: 0.6061

Epoch 29/50
25/25 - 5s - loss: 0.4992 - accuracy: 0.8955 - val_loss: 1.4160 - val_accuracy: 0.6094
```

```
Epoch 30/50
25/25 - 5s - loss: 0.4939 - accuracy: 0.8974 - val_loss: 1.4138 - val_accuracy:
0.6069

Epoch 31/50
25/25 - 5s - loss: 0.4863 - accuracy: 0.9007 - val_loss: 1.4110 - val_accuracy:
0.6077

Epoch 32/50
25/25 - 5s - loss: 0.4817 - accuracy: 0.8992 - val_loss: 1.4090 - val_accuracy:
0.6077

Epoch 33/50
25/25 - 5s - loss: 0.4775 - accuracy: 0.9019 - val_loss: 1.4085 - val_accuracy:
0.6086

Epoch 34/50
25/25 - 5s - loss: 0.4698 - accuracy: 0.9055 - val_loss: 1.4057 - val_accuracy:
0.6118

Epoch 35/50
25/25 - 5s - loss: 0.4660 - accuracy: 0.9060 - val_loss: 1.4038 - val_accuracy:
0.6094

Epoch 36/50
25/25 - 5s - loss: 0.4619 - accuracy: 0.9056 - val_loss: 1.4025 - val_accuracy:
0.6134

Epoch 37/50
25/25 - 5s - loss: 0.4566 - accuracy: 0.9064 - val_loss: 1.4015 - val_accuracy:
0.6102

Epoch 38/50
25/25 - 5s - loss: 0.4522 - accuracy: 0.9076 - val_loss: 1.3999 - val_accuracy:
0.6126

Epoch 39/50
25/25 - 5s - loss: 0.4471 - accuracy: 0.9083 - val_loss: 1.3987 - val_accuracy:
0.6134

Epoch 40/50
25/25 - 5s - loss: 0.4451 - accuracy: 0.9079 - val_loss: 1.3971 - val_accuracy:
0.6126

Epoch 41/50
25/25 - 5s - loss: 0.4408 - accuracy: 0.9094 - val_loss: 1.3956 - val_accuracy:
0.6150

Epoch 42/50
25/25 - 5s - loss: 0.4356 - accuracy: 0.9129 - val_loss: 1.3955 - val_accuracy:
0.6142

Epoch 43/50
25/25 - 5s - loss: 0.4340 - accuracy: 0.9103 - val_loss: 1.3934 - val_accuracy:
0.6166

Epoch 44/50
25/25 - 5s - loss: 0.4290 - accuracy: 0.9113 - val_loss: 1.3925 - val_accuracy:
0.6166

Epoch 45/50
```

```
25/25 - 5s - loss: 0.4251 - accuracy: 0.9144 - val_loss: 1.3902 - val_accuracy:
0.6174

Epoch 46/50
25/25 - 5s - loss: 0.4237 - accuracy: 0.9138 - val_loss: 1.3895 - val_accuracy:
0.6182

Epoch 47/50
25/25 - 5s - loss: 0.4216 - accuracy: 0.9151 - val_loss: 1.3890 - val_accuracy:
0.6215

Epoch 48/50
25/25 - 5s - loss: 0.4165 - accuracy: 0.9162 - val_loss: 1.3891 - val_accuracy:
0.6190

Epoch 49/50
25/25 - 5s - loss: 0.4153 - accuracy: 0.9141 - val_loss: 1.3876 - val_accuracy:
0.6207

Epoch 50/50
25/25 - 5s - loss: 0.4105 - accuracy: 0.9172 - val_loss: 1.3870 - val_accuracy:
0.6199

39/39 - 0s

Train Accuracy [0.3862849473953247, 0.9217796921730042] , Test Accuracy [1.3870
229721069336, 0.619854748249054]
Epoch 1/50
88/88 - 8s - loss: 2.3514 - accuracy: 0.2910 - val_loss: 2.1103 - val_accuracy:
0.2607

Epoch 2/50
88/88 - 8s - loss: 1.7571 - accuracy: 0.2871 - val_loss: 2.0088 - val_accuracy:
0.2623

Epoch 3/50
88/88 - 10s - loss: 1.6135 - accuracy: 0.2880 - val_loss: 1.9673 - val_accurac
y: 0.2631

Epoch 4/50
88/88 - 9s - loss: 1.5353 - accuracy: 0.2893 - val_loss: 1.9418 - val_accuracy:
0.2631

Epoch 5/50
88/88 - 8s - loss: 1.4826 - accuracy: 0.2905 - val_loss: 1.9254 - val_accuracy:
0.2631

Epoch 6/50
88/88 - 8s - loss: 1.4429 - accuracy: 0.2917 - val_loss: 1.9114 - val_accuracy:
0.2647

Epoch 7/50
88/88 - 8s - loss: 1.4115 - accuracy: 0.3035 - val_loss: 1.9005 - val_accuracy:
0.2801

Epoch 8/50
88/88 - 9s - loss: 1.3857 - accuracy: 0.3133 - val_loss: 1.8946 - val_accuracy:
0.2857

Epoch 9/50
88/88 - 8s - loss: 1.3636 - accuracy: 0.3216 - val_loss: 1.8880 - val_accuracy:
0.2922
```

```
Epoch 10/50
88/88 - 8s - loss: 1.3443 - accuracy: 0.3306 - val_loss: 1.8799 - val_accuracy:
0.3035

Epoch 11/50
88/88 - 8s - loss: 1.3273 - accuracy: 0.3453 - val_loss: 1.8749 - val_accuracy:
0.3180

Epoch 12/50
88/88 - 8s - loss: 1.3119 - accuracy: 0.3586 - val_loss: 1.8683 - val_accuracy:
0.3277

Epoch 13/50
88/88 - 8s - loss: 1.2971 - accuracy: 0.3688 - val_loss: 1.8623 - val_accuracy:
0.3366

Epoch 14/50
88/88 - 8s - loss: 1.2767 - accuracy: 0.3848 - val_loss: 1.8449 - val_accuracy:
0.3559

Epoch 15/50
88/88 - 8s - loss: 1.2471 - accuracy: 0.4081 - val_loss: 1.8154 - val_accuracy:
0.3713

Epoch 16/50
88/88 - 8s - loss: 1.2012 - accuracy: 0.4341 - val_loss: 1.7839 - val_accuracy:
0.3801

Epoch 17/50
88/88 - 8s - loss: 1.1715 - accuracy: 0.4533 - val_loss: 1.7708 - val_accuracy:
0.3923

Epoch 18/50
88/88 - 8s - loss: 1.1501 - accuracy: 0.4743 - val_loss: 1.7566 - val_accuracy:
0.4019

Epoch 19/50
88/88 - 8s - loss: 1.1276 - accuracy: 0.5032 - val_loss: 1.7380 - val_accuracy:
0.4197

Epoch 20/50
88/88 - 8s - loss: 1.1004 - accuracy: 0.5439 - val_loss: 1.7138 - val_accuracy:
0.4407

Epoch 21/50
88/88 - 8s - loss: 1.0643 - accuracy: 0.6020 - val_loss: 1.6811 - val_accuracy:
0.4617

Epoch 22/50
88/88 - 8s - loss: 1.0244 - accuracy: 0.6527 - val_loss: 1.6520 - val_accuracy:
0.4778

Epoch 23/50
88/88 - 8s - loss: 0.9905 - accuracy: 0.6918 - val_loss: 1.6370 - val_accuracy:
0.4907

Epoch 24/50
88/88 - 8s - loss: 0.9639 - accuracy: 0.7084 - val_loss: 1.6231 - val_accuracy:
0.5012

Epoch 25/50
```

```
88/88 - 8s - loss: 0.9410 - accuracy: 0.7236 - val_loss: 1.6129 - val_accuracy:
0.5044

Epoch 26/50
88/88 - 8s - loss: 0.9206 - accuracy: 0.7332 - val_loss: 1.6017 - val_accuracy:
0.5117

Epoch 27/50
88/88 - 8s - loss: 0.9016 - accuracy: 0.7464 - val_loss: 1.5904 - val_accuracy:
0.5190

Epoch 28/50
88/88 - 8s - loss: 0.8835 - accuracy: 0.7607 - val_loss: 1.5814 - val_accuracy:
0.5343

Epoch 29/50
88/88 - 8s - loss: 0.8666 - accuracy: 0.7712 - val_loss: 1.5710 - val_accuracy:
0.5416

Epoch 30/50
88/88 - 8s - loss: 0.8512 - accuracy: 0.7783 - val_loss: 1.5612 - val_accuracy:
0.5440

Epoch 31/50
88/88 - 8s - loss: 0.8365 - accuracy: 0.7878 - val_loss: 1.5545 - val_accuracy:
0.5480

Epoch 32/50
88/88 - 8s - loss: 0.8233 - accuracy: 0.7921 - val_loss: 1.5472 - val_accuracy:
0.5464

Epoch 33/50
88/88 - 8s - loss: 0.8113 - accuracy: 0.7983 - val_loss: 1.5421 - val_accuracy:
0.5448

Epoch 34/50
88/88 - 9s - loss: 0.8005 - accuracy: 0.8028 - val_loss: 1.5368 - val_accuracy:
0.5448

Epoch 35/50
88/88 - 8s - loss: 0.7905 - accuracy: 0.8088 - val_loss: 1.5328 - val_accuracy:
0.5472

Epoch 36/50
88/88 - 8s - loss: 0.7814 - accuracy: 0.8138 - val_loss: 1.5290 - val_accuracy:
0.5480

Epoch 37/50
88/88 - 8s - loss: 0.7730 - accuracy: 0.8161 - val_loss: 1.5240 - val_accuracy:
0.5488

Epoch 38/50
88/88 - 8s - loss: 0.7652 - accuracy: 0.8201 - val_loss: 1.5198 - val_accuracy:
0.5488

Epoch 39/50
88/88 - 8s - loss: 0.7579 - accuracy: 0.8216 - val_loss: 1.5160 - val_accuracy:
0.5513

Epoch 40/50
88/88 - 8s - loss: 0.7510 - accuracy: 0.8239 - val_loss: 1.5138 - val_accuracy:
0.5537
```

```
Epoch 41/50
88/88 - 9s - loss: 0.7445 - accuracy: 0.8255 - val_loss: 1.5122 - val_accuracy:
0.5553

Epoch 42/50
88/88 - 9s - loss: 0.7384 - accuracy: 0.8267 - val_loss: 1.5090 - val_accuracy:
0.5545

Epoch 43/50
88/88 - 9s - loss: 0.7326 - accuracy: 0.8289 - val_loss: 1.5058 - val_accuracy:
0.5569

Epoch 44/50
88/88 - 8s - loss: 0.7271 - accuracy: 0.8303 - val_loss: 1.5025 - val_accuracy:
0.5577

Epoch 45/50
88/88 - 9s - loss: 0.7219 - accuracy: 0.8308 - val_loss: 1.4999 - val_accuracy:
0.5593

Epoch 46/50
88/88 - 8s - loss: 0.7169 - accuracy: 0.8323 - val_loss: 1.4974 - val_accuracy:
0.5593

Epoch 47/50
88/88 - 8s - loss: 0.7122 - accuracy: 0.8342 - val_loss: 1.4953 - val_accuracy:
0.5609

Epoch 48/50
88/88 - 8s - loss: 0.7077 - accuracy: 0.8357 - val_loss: 1.4937 - val_accuracy:
0.5609

Epoch 49/50
88/88 - 8s - loss: 0.7032 - accuracy: 0.8381 - val_loss: 1.4931 - val_accuracy:
0.5593

Epoch 50/50
88/88 - 8s - loss: 0.6990 - accuracy: 0.8397 - val_loss: 1.4909 - val_accuracy:
0.5617

39/39 - 1s

Train Accuracy [0.6951443552970886, 0.8404198288917542] , Test Accuracy [1.4909
29365158081, 0.5617433190345764]
Epoch 1/50
44/44 - 7s - loss: 2.1442 - accuracy: 0.4290 - val_loss: 1.9435 - val_accuracy:
0.4592

Epoch 2/50
44/44 - 7s - loss: 1.4932 - accuracy: 0.5978 - val_loss: 1.8026 - val_accuracy:
0.4907

Epoch 3/50
44/44 - 7s - loss: 1.2787 - accuracy: 0.6651 - val_loss: 1.7306 - val_accuracy:
0.5117

Epoch 4/50
44/44 - 7s - loss: 1.1496 - accuracy: 0.7041 - val_loss: 1.6831 - val_accuracy:
0.5198

Epoch 5/50
```

```
44/44 - 7s - loss: 1.0566 - accuracy: 0.7356 - val_loss: 1.6477 - val_accuracy:
0.5262

Epoch 6/50
44/44 - 7s - loss: 0.9878 - accuracy: 0.7611 - val_loss: 1.6246 - val_accuracy:
0.5408

Epoch 7/50
44/44 - 7s - loss: 0.9367 - accuracy: 0.7758 - val_loss: 1.6087 - val_accuracy:
0.5440

Epoch 8/50
44/44 - 7s - loss: 0.8904 - accuracy: 0.7905 - val_loss: 1.5894 - val_accuracy:
0.5472

Epoch 9/50
44/44 - 7s - loss: 0.8560 - accuracy: 0.8002 - val_loss: 1.5744 - val_accuracy:
0.5593

Epoch 10/50
44/44 - 7s - loss: 0.8239 - accuracy: 0.8089 - val_loss: 1.5592 - val_accuracy:
0.5569

Epoch 11/50
44/44 - 7s - loss: 0.7994 - accuracy: 0.8174 - val_loss: 1.5484 - val_accuracy:
0.5601

Epoch 12/50
44/44 - 7s - loss: 0.7710 - accuracy: 0.8251 - val_loss: 1.5401 - val_accuracy:
0.5593

Epoch 13/50
44/44 - 7s - loss: 0.7523 - accuracy: 0.8311 - val_loss: 1.5309 - val_accuracy:
0.5666

Epoch 14/50
44/44 - 7s - loss: 0.7343 - accuracy: 0.8345 - val_loss: 1.5232 - val_accuracy:
0.5658

Epoch 15/50
44/44 - 7s - loss: 0.7162 - accuracy: 0.8425 - val_loss: 1.5177 - val_accuracy:
0.5698

Epoch 16/50
44/44 - 7s - loss: 0.6990 - accuracy: 0.8455 - val_loss: 1.5111 - val_accuracy:
0.5714

Epoch 17/50
44/44 - 7s - loss: 0.6861 - accuracy: 0.8471 - val_loss: 1.5041 - val_accuracy:
0.5763

Epoch 18/50
44/44 - 7s - loss: 0.6717 - accuracy: 0.8540 - val_loss: 1.4997 - val_accuracy:
0.5787

Epoch 19/50
44/44 - 7s - loss: 0.6595 - accuracy: 0.8573 - val_loss: 1.4946 - val_accuracy:
0.5819

Epoch 20/50
44/44 - 7s - loss: 0.6510 - accuracy: 0.8578 - val_loss: 1.4904 - val_accuracy:
0.5835
```

```
Epoch 21/50
44/44 - 7s - loss: 0.6387 - accuracy: 0.8596 - val_loss: 1.4866 - val_accuracy:
0.5811

Epoch 22/50
44/44 - 7s - loss: 0.6308 - accuracy: 0.8649 - val_loss: 1.4804 - val_accuracy:
0.5835

Epoch 23/50
44/44 - 7s - loss: 0.6186 - accuracy: 0.8676 - val_loss: 1.4792 - val_accuracy:
0.5819

Epoch 24/50
44/44 - 7s - loss: 0.6121 - accuracy: 0.8659 - val_loss: 1.4740 - val_accuracy:
0.5868

Epoch 25/50
44/44 - 7s - loss: 0.6020 - accuracy: 0.8710 - val_loss: 1.4713 - val_accuracy:
0.5892

Epoch 26/50
44/44 - 7s - loss: 0.5933 - accuracy: 0.8733 - val_loss: 1.4679 - val_accuracy:
0.5884

Epoch 27/50
44/44 - 7s - loss: 0.5890 - accuracy: 0.8753 - val_loss: 1.4652 - val_accuracy:
0.5884

Epoch 28/50
44/44 - 7s - loss: 0.5813 - accuracy: 0.8755 - val_loss: 1.4614 - val_accuracy:
0.5884

Epoch 29/50
44/44 - 7s - loss: 0.5740 - accuracy: 0.8767 - val_loss: 1.4597 - val_accuracy:
0.5892

Epoch 30/50
44/44 - 7s - loss: 0.5679 - accuracy: 0.8809 - val_loss: 1.4575 - val_accuracy:
0.5900

Epoch 31/50
44/44 - 7s - loss: 0.5629 - accuracy: 0.8791 - val_loss: 1.4542 - val_accuracy:
0.5916

Epoch 32/50
44/44 - 7s - loss: 0.5574 - accuracy: 0.8819 - val_loss: 1.4533 - val_accuracy:
0.5908

Epoch 33/50
44/44 - 7s - loss: 0.5504 - accuracy: 0.8834 - val_loss: 1.4509 - val_accuracy:
0.5932

Epoch 34/50
44/44 - 8s - loss: 0.5452 - accuracy: 0.8841 - val_loss: 1.4490 - val_accuracy:
0.5940

Epoch 35/50
44/44 - 8s - loss: 0.5402 - accuracy: 0.8848 - val_loss: 1.4474 - val_accuracy:
0.5948

Epoch 36/50
```

```
44/44 - 10s - loss: 0.5353 - accuracy: 0.8861 - val_loss: 1.4447 - val_accurac
y: 0.5948

Epoch 37/50
44/44 - 7s - loss: 0.5301 - accuracy: 0.8879 - val_loss: 1.4433 - val_accuracy:
0.5948

Epoch 38/50
44/44 - 7s - loss: 0.5258 - accuracy: 0.8886 - val_loss: 1.4407 - val_accuracy:
0.5948

Epoch 39/50
44/44 - 7s - loss: 0.5211 - accuracy: 0.8922 - val_loss: 1.4393 - val_accuracy:
0.5956

Epoch 40/50
44/44 - 7s - loss: 0.5183 - accuracy: 0.8902 - val_loss: 1.4376 - val_accuracy:
0.5989

Epoch 41/50
44/44 - 7s - loss: 0.5129 - accuracy: 0.8927 - val_loss: 1.4355 - val_accuracy:
0.5981

Epoch 42/50
44/44 - 7s - loss: 0.5082 - accuracy: 0.8941 - val_loss: 1.4345 - val_accuracy:
0.5997

Epoch 43/50
44/44 - 7s - loss: 0.5061 - accuracy: 0.8939 - val_loss: 1.4324 - val_accuracy:
0.6005

Epoch 44/50
44/44 - 7s - loss: 0.5027 - accuracy: 0.8938 - val_loss: 1.4310 - val_accuracy:
0.6029

Epoch 45/50
44/44 - 7s - loss: 0.4987 - accuracy: 0.8957 - val_loss: 1.4300 - val_accuracy:
0.6029

Epoch 46/50
44/44 - 7s - loss: 0.4957 - accuracy: 0.8960 - val_loss: 1.4279 - val_accuracy:
0.6045

Epoch 47/50
44/44 - 7s - loss: 0.4921 - accuracy: 0.8981 - val_loss: 1.4281 - val_accuracy:
0.6029

Epoch 48/50
44/44 - 7s - loss: 0.4897 - accuracy: 0.8954 - val_loss: 1.4258 - val_accuracy:
0.6061

Epoch 49/50
44/44 - 7s - loss: 0.4846 - accuracy: 0.8993 - val_loss: 1.4244 - val_accuracy:
0.6045

Epoch 50/50
44/44 - 7s - loss: 0.4818 - accuracy: 0.8984 - val_loss: 1.4237 - val_accuracy:
0.6061

39/39 - 1s

Train Accuracy [0.45788705348968506, 0.9046465754508972] , Test Accuracy [1.423
```

```
7335920333862, 0.6061339974403381]
Epoch 1/50
88/88 - 6s - loss: 2.1497 - accuracy: 0.4344 - val_loss: 1.9972 - val_accuracy:
0.4487

Epoch 2/50
88/88 - 6s - loss: 1.5951 - accuracy: 0.5683 - val_loss: 1.8910 - val_accuracy:
0.4649

Epoch 3/50
88/88 - 6s - loss: 1.4379 - accuracy: 0.6133 - val_loss: 1.8358 - val_accuracy:
0.4802

Epoch 4/50
88/88 - 6s - loss: 1.3437 - accuracy: 0.6472 - val_loss: 1.7971 - val_accuracy:
0.4883

Epoch 5/50
88/88 - 6s - loss: 1.2777 - accuracy: 0.6699 - val_loss: 1.7679 - val_accuracy:
0.4996

Epoch 6/50
88/88 - 6s - loss: 1.2275 - accuracy: 0.6868 - val_loss: 1.7468 - val_accuracy:
0.5077

Epoch 7/50
88/88 - 6s - loss: 1.1869 - accuracy: 0.7010 - val_loss: 1.7291 - val_accuracy:
0.5165

Epoch 8/50
88/88 - 6s - loss: 1.1534 - accuracy: 0.7130 - val_loss: 1.7141 - val_accuracy:
0.5206

Epoch 9/50
88/88 - 6s - loss: 1.1247 - accuracy: 0.7227 - val_loss: 1.7033 - val_accuracy:
0.5270

Epoch 10/50
88/88 - 6s - loss: 1.1002 - accuracy: 0.7297 - val_loss: 1.6922 - val_accuracy:
0.5278

Epoch 11/50
88/88 - 6s - loss: 1.0784 - accuracy: 0.7370 - val_loss: 1.6831 - val_accuracy:
0.5287

Epoch 12/50
88/88 - 6s - loss: 1.0591 - accuracy: 0.7437 - val_loss: 1.6736 - val_accuracy:
0.5335

Epoch 13/50
88/88 - 6s - loss: 1.0413 - accuracy: 0.7496 - val_loss: 1.6661 - val_accuracy:
0.5351

Epoch 14/50
88/88 - 6s - loss: 1.0254 - accuracy: 0.7541 - val_loss: 1.6581 - val_accuracy:
0.5408

Epoch 15/50
88/88 - 6s - loss: 1.0109 - accuracy: 0.7590 - val_loss: 1.6527 - val_accuracy:
0.5424

Epoch 16/50
```

```
88/88 - 6s - loss: 0.9974 - accuracy: 0.7624 - val_loss: 1.6470 - val_accuracy:
0.5464

Epoch 17/50
88/88 - 6s - loss: 0.9852 - accuracy: 0.7652 - val_loss: 1.6415 - val_accuracy:
0.5456

Epoch 18/50
88/88 - 6s - loss: 0.9738 - accuracy: 0.7690 - val_loss: 1.6365 - val_accuracy:
0.5504

Epoch 19/50
88/88 - 6s - loss: 0.9630 - accuracy: 0.7714 - val_loss: 1.6320 - val_accuracy:
0.5496

Epoch 20/50
88/88 - 6s - loss: 0.9530 - accuracy: 0.7747 - val_loss: 1.6279 - val_accuracy:
0.5529

Epoch 21/50
88/88 - 6s - loss: 0.9436 - accuracy: 0.7782 - val_loss: 1.6243 - val_accuracy:
0.5537

Epoch 22/50
88/88 - 6s - loss: 0.9347 - accuracy: 0.7801 - val_loss: 1.6198 - val_accuracy:
0.5561

Epoch 23/50
88/88 - 6s - loss: 0.9263 - accuracy: 0.7835 - val_loss: 1.6167 - val_accuracy:
0.5577

Epoch 24/50
88/88 - 6s - loss: 0.9183 - accuracy: 0.7861 - val_loss: 1.6129 - val_accuracy:
0.5585

Epoch 25/50
88/88 - 6s - loss: 0.9107 - accuracy: 0.7890 - val_loss: 1.6097 - val_accuracy:
0.5593

Epoch 26/50
88/88 - 6s - loss: 0.9035 - accuracy: 0.7918 - val_loss: 1.6062 - val_accuracy:
0.5585

Epoch 27/50
88/88 - 6s - loss: 0.8966 - accuracy: 0.7947 - val_loss: 1.6034 - val_accuracy:
0.5593

Epoch 28/50
88/88 - 6s - loss: 0.8901 - accuracy: 0.7969 - val_loss: 1.6006 - val_accuracy:
0.5601

Epoch 29/50
88/88 - 6s - loss: 0.8837 - accuracy: 0.7989 - val_loss: 1.5982 - val_accuracy:
0.5609

Epoch 30/50
88/88 - 6s - loss: 0.8777 - accuracy: 0.8010 - val_loss: 1.5952 - val_accuracy:
0.5626

Epoch 31/50
88/88 - 6s - loss: 0.8718 - accuracy: 0.8028 - val_loss: 1.5927 - val_accuracy:
0.5609
```

```
Epoch 32/50
88/88 - 6s - loss: 0.8663 - accuracy: 0.8055 - val_loss: 1.5904 - val_accuracy:
0.5609

Epoch 33/50
88/88 - 7s - loss: 0.8610 - accuracy: 0.8074 - val_loss: 1.5884 - val_accuracy:
0.5609

Epoch 34/50
88/88 - 9s - loss: 0.8558 - accuracy: 0.8089 - val_loss: 1.5860 - val_accuracy:
0.5609

Epoch 35/50
88/88 - 6s - loss: 0.8507 - accuracy: 0.8100 - val_loss: 1.5838 - val_accuracy:
0.5617

Epoch 36/50
88/88 - 6s - loss: 0.8459 - accuracy: 0.8124 - val_loss: 1.5817 - val_accuracy:
0.5634

Epoch 37/50
88/88 - 6s - loss: 0.8413 - accuracy: 0.8132 - val_loss: 1.5797 - val_accuracy:
0.5634

Epoch 38/50
88/88 - 6s - loss: 0.8367 - accuracy: 0.8144 - val_loss: 1.5780 - val_accuracy:
0.5634

Epoch 39/50
88/88 - 6s - loss: 0.8324 - accuracy: 0.8154 - val_loss: 1.5758 - val_accuracy:
0.5642

Epoch 40/50
88/88 - 6s - loss: 0.8281 - accuracy: 0.8166 - val_loss: 1.5741 - val_accuracy:
0.5650

Epoch 41/50
88/88 - 6s - loss: 0.8240 - accuracy: 0.8182 - val_loss: 1.5725 - val_accuracy:
0.5650

Epoch 42/50
88/88 - 6s - loss: 0.8200 - accuracy: 0.8189 - val_loss: 1.5704 - val_accuracy:
0.5650

Epoch 43/50
88/88 - 6s - loss: 0.8160 - accuracy: 0.8198 - val_loss: 1.5688 - val_accuracy:
0.5658

Epoch 44/50
88/88 - 6s - loss: 0.8122 - accuracy: 0.8203 - val_loss: 1.5677 - val_accuracy:
0.5642

Epoch 45/50
88/88 - 6s - loss: 0.8086 - accuracy: 0.8219 - val_loss: 1.5657 - val_accuracy:
0.5658

Epoch 46/50
88/88 - 6s - loss: 0.8049 - accuracy: 0.8228 - val_loss: 1.5641 - val_accuracy:
0.5650

Epoch 47/50
```

```
88/88 - 6s - loss: 0.8014 - accuracy: 0.8236 - val_loss: 1.5624 - val_accuracy:
0.5658

Epoch 48/50
88/88 - 6s - loss: 0.7980 - accuracy: 0.8239 - val_loss: 1.5605 - val_accuracy:
0.5650

Epoch 49/50
88/88 - 6s - loss: 0.7947 - accuracy: 0.8262 - val_loss: 1.5594 - val_accuracy:
0.5650

Epoch 50/50
88/88 - 6s - loss: 0.7914 - accuracy: 0.8271 - val_loss: 1.5581 - val_accuracy:
0.5650

39/39 - 0s

Train Accuracy [0.7884023189544678, 0.8276820778846741] , Test Accuracy [1.5581
315755844116, 0.5649717450141907]
Epoch 1/50
30/30 - 5s - loss: 2.6160 - accuracy: 0.3274 - val_loss: 2.3312 - val_accuracy:
0.2825

Epoch 2/50
30/30 - 5s - loss: 2.0088 - accuracy: 0.3147 - val_loss: 2.1285 - val_accuracy:
0.2825

Epoch 3/50
30/30 - 5s - loss: 1.8046 - accuracy: 0.3150 - val_loss: 2.0609 - val_accuracy:
0.2841

Epoch 4/50
30/30 - 5s - loss: 1.7040 - accuracy: 0.3127 - val_loss: 2.0224 - val_accuracy:
0.2841

Epoch 5/50
30/30 - 5s - loss: 1.6406 - accuracy: 0.3128 - val_loss: 1.9965 - val_accuracy:
0.2849

Epoch 6/50
30/30 - 5s - loss: 1.5916 - accuracy: 0.3124 - val_loss: 1.9720 - val_accuracy:
0.2849

Epoch 7/50
30/30 - 5s - loss: 1.5521 - accuracy: 0.3120 - val_loss: 1.9597 - val_accuracy:
0.2865

Epoch 8/50
30/30 - 5s - loss: 1.5190 - accuracy: 0.3133 - val_loss: 1.9486 - val_accuracy:
0.2881

Epoch 9/50
30/30 - 5s - loss: 1.4947 - accuracy: 0.3173 - val_loss: 1.9366 - val_accuracy:
0.2889

Epoch 10/50
30/30 - 5s - loss: 1.4722 - accuracy: 0.3177 - val_loss: 1.9279 - val_accuracy:
0.2897

Epoch 11/50
30/30 - 5s - loss: 1.4535 - accuracy: 0.3202 - val_loss: 1.9196 - val_accuracy:
0.2938
```

```
Epoch 12/50
30/30 - 5s - loss: 1.4387 - accuracy: 0.3236 - val_loss: 1.9155 - val_accuracy:
0.3091

Epoch 13/50
30/30 - 5s - loss: 1.4211 - accuracy: 0.3271 - val_loss: 1.9078 - val_accuracy:
0.3140

Epoch 14/50
30/30 - 5s - loss: 1.4080 - accuracy: 0.3324 - val_loss: 1.9006 - val_accuracy:
0.3140

Epoch 15/50
30/30 - 5s - loss: 1.3927 - accuracy: 0.3381 - val_loss: 1.8916 - val_accuracy:
0.3188

Epoch 16/50
30/30 - 5s - loss: 1.3771 - accuracy: 0.3475 - val_loss: 1.8825 - val_accuracy:
0.3212

Epoch 17/50
30/30 - 5s - loss: 1.3628 - accuracy: 0.3534 - val_loss: 1.8782 - val_accuracy:
0.3220

Epoch 18/50
30/30 - 5s - loss: 1.3518 - accuracy: 0.3563 - val_loss: 1.8726 - val_accuracy:
0.3236

Epoch 19/50
30/30 - 5s - loss: 1.3407 - accuracy: 0.3599 - val_loss: 1.8654 - val_accuracy:
0.3253

Epoch 20/50
30/30 - 5s - loss: 1.3261 - accuracy: 0.3632 - val_loss: 1.8549 - val_accuracy:
0.3277

Epoch 21/50
30/30 - 5s - loss: 1.3103 - accuracy: 0.3654 - val_loss: 1.8432 - val_accuracy:
0.3293

Epoch 22/50
30/30 - 5s - loss: 1.2947 - accuracy: 0.3681 - val_loss: 1.8360 - val_accuracy:
0.3309

Epoch 23/50
30/30 - 5s - loss: 1.2840 - accuracy: 0.3714 - val_loss: 1.8318 - val_accuracy:
0.3333

Epoch 24/50
30/30 - 5s - loss: 1.2744 - accuracy: 0.3732 - val_loss: 1.8289 - val_accuracy:
0.3358

Epoch 25/50
30/30 - 5s - loss: 1.2637 - accuracy: 0.3747 - val_loss: 1.8269 - val_accuracy:
0.3374

Epoch 26/50
30/30 - 5s - loss: 1.2592 - accuracy: 0.3767 - val_loss: 1.8252 - val_accuracy:
0.3374

Epoch 27/50
```

```
30/30 - 5s - loss: 1.2528 - accuracy: 0.3791 - val_loss: 1.8219 - val_accuracy:
0.3374

Epoch 28/50
30/30 - 5s - loss: 1.2457 - accuracy: 0.3810 - val_loss: 1.8202 - val_accuracy:
0.3422

Epoch 29/50
30/30 - 5s - loss: 1.2387 - accuracy: 0.3821 - val_loss: 1.8168 - val_accuracy:
0.3430

Epoch 30/50
30/30 - 5s - loss: 1.2319 - accuracy: 0.3851 - val_loss: 1.8126 - val_accuracy:
0.3438

Epoch 31/50
30/30 - 5s - loss: 1.2230 - accuracy: 0.3892 - val_loss: 1.8040 - val_accuracy:
0.3479

Epoch 32/50
30/30 - 5s - loss: 1.2109 - accuracy: 0.3933 - val_loss: 1.7994 - val_accuracy:
0.3495

Epoch 33/50
30/30 - 5s - loss: 1.2044 - accuracy: 0.3951 - val_loss: 1.7968 - val_accuracy:
0.3495

Epoch 34/50
30/30 - 5s - loss: 1.1981 - accuracy: 0.3985 - val_loss: 1.7948 - val_accuracy:
0.3535

Epoch 35/50
30/30 - 5s - loss: 1.1921 - accuracy: 0.4023 - val_loss: 1.7938 - val_accuracy:
0.3567

Epoch 36/50
30/30 - 5s - loss: 1.1880 - accuracy: 0.4053 - val_loss: 1.7918 - val_accuracy:
0.3567

Epoch 37/50
30/30 - 5s - loss: 1.1831 - accuracy: 0.4090 - val_loss: 1.7900 - val_accuracy:
0.3567

Epoch 38/50
30/30 - 5s - loss: 1.1748 - accuracy: 0.4125 - val_loss: 1.7873 - val_accuracy:
0.3575

Epoch 39/50
30/30 - 8s - loss: 1.1728 - accuracy: 0.4163 - val_loss: 1.7844 - val_accuracy:
0.3616

Epoch 40/50
30/30 - 7s - loss: 1.1682 - accuracy: 0.4216 - val_loss: 1.7824 - val_accuracy:
0.3632

Epoch 41/50
30/30 - 6s - loss: 1.1609 - accuracy: 0.4241 - val_loss: 1.7794 - val_accuracy:
0.3656

Epoch 42/50
30/30 - 5s - loss: 1.1539 - accuracy: 0.4314 - val_loss: 1.7786 - val_accuracy:
0.3664
```

```
Epoch 43/50
30/30 - 5s - loss: 1.1494 - accuracy: 0.4357 - val_loss: 1.7743 - val_accuracy:
0.3705

Epoch 44/50
30/30 - 5s - loss: 1.1464 - accuracy: 0.4431 - val_loss: 1.7716 - val_accuracy:
0.3737

Epoch 45/50
30/30 - 5s - loss: 1.1410 - accuracy: 0.4476 - val_loss: 1.7691 - val_accuracy:
0.3801

Epoch 46/50
30/30 - 5s - loss: 1.1327 - accuracy: 0.4553 - val_loss: 1.7670 - val_accuracy:
0.3810

Epoch 47/50
30/30 - 5s - loss: 1.1273 - accuracy: 0.4635 - val_loss: 1.7633 - val_accuracy:
0.3842

Epoch 48/50
30/30 - 5s - loss: 1.1214 - accuracy: 0.4717 - val_loss: 1.7594 - val_accuracy:
0.3898

Epoch 49/50
30/30 - 5s - loss: 1.1157 - accuracy: 0.4795 - val_loss: 1.7556 - val_accuracy:
0.3963

Epoch 50/50
30/30 - 5s - loss: 1.1098 - accuracy: 0.4884 - val_loss: 1.7523 - val_accuracy:
0.4003

39/39 - 0s

Train Accuracy [1.0840171575546265, 0.49686041474342346] , Test Accuracy [1.752
2926330566406, 0.4003228545188904]
Epoch 1/50
30/30 - 6s - loss: 2.5328 - accuracy: 0.2910 - val_loss: 2.2158 - val_accuracy:
0.2583

Epoch 2/50
30/30 - 6s - loss: 1.8407 - accuracy: 0.2793 - val_loss: 2.0590 - val_accuracy:
0.2583

Epoch 3/50
30/30 - 6s - loss: 1.6447 - accuracy: 0.2791 - val_loss: 1.9953 - val_accuracy:
0.2583

Epoch 4/50
30/30 - 6s - loss: 1.5468 - accuracy: 0.2791 - val_loss: 1.9675 - val_accuracy:
0.2583

Epoch 5/50
30/30 - 6s - loss: 1.4781 - accuracy: 0.2791 - val_loss: 1.9464 - val_accuracy:
0.2583

Epoch 6/50
30/30 - 6s - loss: 1.4297 - accuracy: 0.2791 - val_loss: 1.9276 - val_accuracy:
0.2583

Epoch 7/50
```

```
30/30 - 6s - loss: 1.3941 - accuracy: 0.2791 - val_loss: 1.9136 - val_accuracy: 
0.2583

Epoch 8/50
30/30 - 6s - loss: 1.3633 - accuracy: 0.2791 - val_loss: 1.9048 - val_accuracy: 
0.2583

Epoch 9/50
30/30 - 6s - loss: 1.3370 - accuracy: 0.2791 - val_loss: 1.8945 - val_accuracy: 
0.2583

Epoch 00009: early stopping
39/39 - 0s

Train Accuracy [1.3183469772338867, 0.27906352281570435] , Test Accuracy [1.894
500494003296, 0.25827279686927795]
Epoch 1/50
59/59 - 7s - loss: nan - accuracy: 0.2746 - val_loss: nan - val_accuracy: 0.258
3

Epoch 2/50
59/59 - 7s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 3/50
59/59 - 7s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 4/50
59/59 - 7s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 5/50
59/59 - 7s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 6/50
59/59 - 7s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 7/50
59/59 - 7s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 8/50
59/59 - 7s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 9/50
59/59 - 7s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 00009: early stopping
39/39 - 1s

Train Accuracy [nan, 0.2788841128349304] , Test Accuracy [nan, 0.25827279686927
795]
Epoch 1/50
88/88 - 6s - loss: 2.4836 - accuracy: 0.3234 - val_loss: 2.2994 - val_accuracy: 
0.2889

Epoch 2/50
```

```
88/88 - 6s - loss: 2.0022 - accuracy: 0.3160 - val_loss: 2.1298 - val_accuracy:
0.2889

Epoch 3/50
88/88 - 6s - loss: 1.7893 - accuracy: 0.3149 - val_loss: 2.0648 - val_accuracy:
0.2889

Epoch 4/50
88/88 - 6s - loss: 1.6998 - accuracy: 0.3149 - val_loss: 2.0341 - val_accuracy:
0.2889

Epoch 5/50
88/88 - 6s - loss: 1.6430 - accuracy: 0.3149 - val_loss: 2.0100 - val_accuracy:
0.2889

Epoch 6/50
88/88 - 6s - loss: 1.6014 - accuracy: 0.3149 - val_loss: 1.9961 - val_accuracy:
0.2889

Epoch 7/50
88/88 - 6s - loss: 1.5688 - accuracy: 0.3149 - val_loss: 1.9832 - val_accuracy:
0.2881

Epoch 8/50
88/88 - 6s - loss: 1.5424 - accuracy: 0.3148 - val_loss: 1.9746 - val_accuracy:
0.2881

Epoch 9/50
88/88 - 6s - loss: 1.5204 - accuracy: 0.3148 - val_loss: 1.9654 - val_accuracy:
0.2881

Epoch 00009: early stopping
39/39 - 0s

Train Accuracy [1.5049084424972534, 0.3148546814918518] , Test Accuracy [1.9653
936624526978, 0.2881355881690979]
Epoch 1/50
88/88 - 7s - loss: 2.1161 - accuracy: 0.4302 - val_loss: 2.0040 - val_accuracy:
0.4302

Epoch 2/50
88/88 - 7s - loss: 1.5813 - accuracy: 0.5638 - val_loss: 1.8984 - val_accuracy:
0.4633

Epoch 3/50
88/88 - 7s - loss: 1.4255 - accuracy: 0.6143 - val_loss: 1.8396 - val_accuracy:
0.4859

Epoch 4/50
88/88 - 7s - loss: 1.3310 - accuracy: 0.6498 - val_loss: 1.8021 - val_accuracy:
0.4899

Epoch 5/50
88/88 - 7s - loss: 1.2645 - accuracy: 0.6735 - val_loss: 1.7745 - val_accuracy:
0.4948

Epoch 6/50
88/88 - 7s - loss: 1.2140 - accuracy: 0.6900 - val_loss: 1.7536 - val_accuracy:
0.4996

Epoch 7/50
88/88 - 8s - loss: 1.1729 - accuracy: 0.7037 - val_loss: 1.7366 - val_accuracy:
```

```
0.5085

Epoch 8/50
88/88 - 7s - loss: 1.1389 - accuracy: 0.7166 - val_loss: 1.7215 - val_accuracy:
0.5117

Epoch 9/50
88/88 - 8s - loss: 1.1099 - accuracy: 0.7244 - val_loss: 1.7079 - val_accuracy:
0.5190

Epoch 10/50
88/88 - 8s - loss: 1.0849 - accuracy: 0.7365 - val_loss: 1.6972 - val_accuracy:
0.5174

Epoch 11/50
88/88 - 8s - loss: 1.0625 - accuracy: 0.7427 - val_loss: 1.6874 - val_accuracy:
0.5190

Epoch 12/50
88/88 - 7s - loss: 1.0428 - accuracy: 0.7488 - val_loss: 1.6790 - val_accuracy:
0.5198

Epoch 13/50
88/88 - 7s - loss: 1.0250 - accuracy: 0.7548 - val_loss: 1.6706 - val_accuracy:
0.5174

Epoch 14/50
88/88 - 7s - loss: 1.0088 - accuracy: 0.7590 - val_loss: 1.6637 - val_accuracy:
0.5182

Epoch 15/50
88/88 - 7s - loss: 0.9940 - accuracy: 0.7649 - val_loss: 1.6570 - val_accuracy:
0.5174

Epoch 16/50
88/88 - 7s - loss: 0.9804 - accuracy: 0.7687 - val_loss: 1.6512 - val_accuracy:
0.5206

Epoch 17/50
88/88 - 7s - loss: 0.9679 - accuracy: 0.7732 - val_loss: 1.6453 - val_accuracy:
0.5222

Epoch 18/50
88/88 - 7s - loss: 0.9562 - accuracy: 0.7782 - val_loss: 1.6401 - val_accuracy:
0.5238

Epoch 19/50
88/88 - 7s - loss: 0.9453 - accuracy: 0.7819 - val_loss: 1.6352 - val_accuracy:
0.5254

Epoch 20/50
88/88 - 7s - loss: 0.9350 - accuracy: 0.7852 - val_loss: 1.6309 - val_accuracy:
0.5262

Epoch 21/50
88/88 - 7s - loss: 0.9254 - accuracy: 0.7880 - val_loss: 1.6264 - val_accuracy:
0.5270

Epoch 22/50
88/88 - 7s - loss: 0.9162 - accuracy: 0.7899 - val_loss: 1.6224 - val_accuracy:
0.5287
```

```
Epoch 23/50
88/88 - 7s - loss: 0.9076 - accuracy: 0.7924 - val_loss: 1.6184 - val_accuracy:
0.5295

Epoch 24/50
88/88 - 7s - loss: 0.8995 - accuracy: 0.7946 - val_loss: 1.6144 - val_accuracy:
0.5303

Epoch 25/50
88/88 - 7s - loss: 0.8917 - accuracy: 0.7966 - val_loss: 1.6114 - val_accuracy:
0.5319

Epoch 26/50
88/88 - 7s - loss: 0.8842 - accuracy: 0.7990 - val_loss: 1.6074 - val_accuracy:
0.5327

Epoch 27/50
88/88 - 7s - loss: 0.8772 - accuracy: 0.8015 - val_loss: 1.6040 - val_accuracy:
0.5343

Epoch 28/50
88/88 - 7s - loss: 0.8703 - accuracy: 0.8026 - val_loss: 1.6012 - val_accuracy:
0.5359

Epoch 29/50
88/88 - 7s - loss: 0.8639 - accuracy: 0.8057 - val_loss: 1.5986 - val_accuracy:
0.5359

Epoch 30/50
88/88 - 7s - loss: 0.8576 - accuracy: 0.8083 - val_loss: 1.5958 - val_accuracy:
0.5367

Epoch 31/50
88/88 - 7s - loss: 0.8517 - accuracy: 0.8086 - val_loss: 1.5931 - val_accuracy:
0.5359

Epoch 32/50
88/88 - 7s - loss: 0.8460 - accuracy: 0.8097 - val_loss: 1.5908 - val_accuracy:
0.5375

Epoch 33/50
88/88 - 7s - loss: 0.8405 - accuracy: 0.8119 - val_loss: 1.5883 - val_accuracy:
0.5367

Epoch 34/50
88/88 - 7s - loss: 0.8351 - accuracy: 0.8128 - val_loss: 1.5859 - val_accuracy:
0.5375

Epoch 35/50
88/88 - 7s - loss: 0.8299 - accuracy: 0.8145 - val_loss: 1.5837 - val_accuracy:
0.5351

Epoch 36/50
88/88 - 7s - loss: 0.8250 - accuracy: 0.8156 - val_loss: 1.5812 - val_accuracy:
0.5359

Epoch 37/50
88/88 - 7s - loss: 0.8203 - accuracy: 0.8177 - val_loss: 1.5791 - val_accuracy:
0.5375

Epoch 38/50
88/88 - 7s - loss: 0.8157 - accuracy: 0.8184 - val_loss: 1.5770 - val_accuracy:
```

```
0.5383

Epoch 39/50
88/88 - 7s - loss: 0.8111 - accuracy: 0.8196 - val_loss: 1.5752 - val_accuracy:
0.5383

Epoch 40/50
88/88 - 7s - loss: 0.8068 - accuracy: 0.8205 - val_loss: 1.5731 - val_accuracy:
0.5391

Epoch 41/50
88/88 - 7s - loss: 0.8026 - accuracy: 0.8212 - val_loss: 1.5711 - val_accuracy:
0.5408

Epoch 42/50
88/88 - 7s - loss: 0.7985 - accuracy: 0.8222 - val_loss: 1.5694 - val_accuracy:
0.5400

Epoch 43/50
88/88 - 7s - loss: 0.7945 - accuracy: 0.8232 - val_loss: 1.5677 - val_accuracy:
0.5416

Epoch 44/50
88/88 - 7s - loss: 0.7906 - accuracy: 0.8238 - val_loss: 1.5658 - val_accuracy:
0.5416

Epoch 45/50
88/88 - 7s - loss: 0.7868 - accuracy: 0.8253 - val_loss: 1.5637 - val_accuracy:
0.5424

Epoch 46/50
88/88 - 7s - loss: 0.7831 - accuracy: 0.8268 - val_loss: 1.5620 - val_accuracy:
0.5424

Epoch 47/50
88/88 - 7s - loss: 0.7795 - accuracy: 0.8275 - val_loss: 1.5605 - val_accuracy:
0.5416

Epoch 48/50
88/88 - 7s - loss: 0.7760 - accuracy: 0.8286 - val_loss: 1.5589 - val_accuracy:
0.5416

Epoch 49/50
88/88 - 9s - loss: 0.7726 - accuracy: 0.8290 - val_loss: 1.5573 - val_accuracy:
0.5432

Epoch 50/50
88/88 - 8s - loss: 0.7693 - accuracy: 0.8297 - val_loss: 1.5561 - val_accuracy:
0.5432

39/39 - 1s

Train Accuracy [0.7663925290107727, 0.8305525779724121] , Test Accuracy [1.5560
564994812012, 0.543179988861084]
Epoch 1/50
59/59 - 6s - loss: nan - accuracy: 0.2742 - val_loss: nan - val_accuracy: 0.258
3

Epoch 2/50
59/59 - 6s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3
```

```
Epoch 3/50
59/59 - 6s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 4/50
59/59 - 6s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 5/50
59/59 - 6s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 6/50
59/59 - 5s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 7/50
59/59 - 5s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 8/50
59/59 - 5s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 9/50
59/59 - 5s - loss: nan - accuracy: 0.2789 - val_loss: nan - val_accuracy: 0.258
3

Epoch 00009: early stopping
39/39 - 0s

Train Accuracy [nan, 0.2788841128349304] , Test Accuracy [nan, 0.25827279686927
795]
Epoch 1/50
35/35 - 7s - loss: 2.1393 - accuracy: 0.4274 - val_loss: 1.9072 - val_accuracy:
0.4681

Epoch 2/50
35/35 - 7s - loss: 1.4477 - accuracy: 0.6076 - val_loss: 1.7394 - val_accuracy:
0.4980

Epoch 3/50
35/35 - 7s - loss: 1.2010 - accuracy: 0.6921 - val_loss: 1.6580 - val_accuracy:
0.5182

Epoch 4/50
35/35 - 7s - loss: 1.0542 - accuracy: 0.7406 - val_loss: 1.6072 - val_accuracy:
0.5391

Epoch 5/50
35/35 - 7s - loss: 0.9538 - accuracy: 0.7724 - val_loss: 1.5688 - val_accuracy:
0.5464

Epoch 6/50
35/35 - 7s - loss: 0.8836 - accuracy: 0.7939 - val_loss: 1.5476 - val_accuracy:
0.5553

Epoch 7/50
35/35 - 7s - loss: 0.8260 - accuracy: 0.8097 - val_loss: 1.5251 - val_accuracy:
0.5609

Epoch 8/50
```

```
35/35 - 7s - loss: 0.7774 - accuracy: 0.8225 - val_loss: 1.5050 - val_accuracy:
0.5690

Epoch 9/50
35/35 - 7s - loss: 0.7414 - accuracy: 0.8371 - val_loss: 1.4944 - val_accuracy:
0.5682

Epoch 10/50
35/35 - 7s - loss: 0.7100 - accuracy: 0.8445 - val_loss: 1.4822 - val_accuracy:
0.5730

Epoch 11/50
35/35 - 7s - loss: 0.6824 - accuracy: 0.8529 - val_loss: 1.4715 - val_accuracy:
0.5755

Epoch 12/50
35/35 - 7s - loss: 0.6617 - accuracy: 0.8569 - val_loss: 1.4619 - val_accuracy:
0.5835

Epoch 13/50
35/35 - 7s - loss: 0.6433 - accuracy: 0.8616 - val_loss: 1.4572 - val_accuracy:
0.5811

Epoch 14/50
35/35 - 7s - loss: 0.6229 - accuracy: 0.8688 - val_loss: 1.4511 - val_accuracy:
0.5860

Epoch 15/50
35/35 - 7s - loss: 0.6064 - accuracy: 0.8695 - val_loss: 1.4454 - val_accuracy:
0.5876

Epoch 16/50
35/35 - 7s - loss: 0.5908 - accuracy: 0.8736 - val_loss: 1.4375 - val_accuracy:
0.5892

Epoch 17/50
35/35 - 7s - loss: 0.5767 - accuracy: 0.8777 - val_loss: 1.4305 - val_accuracy:
0.5924

Epoch 18/50
35/35 - 7s - loss: 0.5623 - accuracy: 0.8803 - val_loss: 1.4281 - val_accuracy:
0.5940

Epoch 19/50
35/35 - 7s - loss: 0.5542 - accuracy: 0.8828 - val_loss: 1.4228 - val_accuracy:
0.5973

Epoch 20/50
35/35 - 7s - loss: 0.5444 - accuracy: 0.8867 - val_loss: 1.4202 - val_accuracy:
0.5964

Epoch 21/50
35/35 - 7s - loss: 0.5329 - accuracy: 0.8891 - val_loss: 1.4158 - val_accuracy:
0.5981

Epoch 22/50
35/35 - 7s - loss: 0.5258 - accuracy: 0.8905 - val_loss: 1.4141 - val_accuracy:
0.5973

Epoch 23/50
35/35 - 7s - loss: 0.5163 - accuracy: 0.8915 - val_loss: 1.4097 - val_accuracy:
0.5997
```

```
Epoch 24/50
35/35 - 7s - loss: 0.5099 - accuracy: 0.8935 - val_loss: 1.4081 - val_accuracy:
0.5989

Epoch 25/50
35/35 - 7s - loss: 0.4998 - accuracy: 0.8945 - val_loss: 1.4056 - val_accuracy:
0.6045

Epoch 26/50
35/35 - 7s - loss: 0.4936 - accuracy: 0.8983 - val_loss: 1.4025 - val_accuracy:
0.6061

Epoch 27/50
35/35 - 7s - loss: 0.4875 - accuracy: 0.8962 - val_loss: 1.3995 - val_accuracy:
0.6061

Epoch 28/50
35/35 - 7s - loss: 0.4799 - accuracy: 0.9006 - val_loss: 1.3960 - val_accuracy:
0.6069

Epoch 29/50
35/35 - 7s - loss: 0.4765 - accuracy: 0.9001 - val_loss: 1.3957 - val_accuracy:
0.6077

Epoch 30/50
35/35 - 7s - loss: 0.4696 - accuracy: 0.9029 - val_loss: 1.3934 - val_accuracy:
0.6069

Epoch 31/50
35/35 - 8s - loss: 0.4654 - accuracy: 0.9046 - val_loss: 1.3917 - val_accuracy:
0.6102

Epoch 32/50
35/35 - 7s - loss: 0.4599 - accuracy: 0.9075 - val_loss: 1.3902 - val_accuracy:
0.6094

Epoch 33/50
35/35 - 7s - loss: 0.4521 - accuracy: 0.9077 - val_loss: 1.3897 - val_accuracy:
0.6094

Epoch 34/50
35/35 - 7s - loss: 0.4480 - accuracy: 0.9085 - val_loss: 1.3873 - val_accuracy:
0.6094

Epoch 35/50
35/35 - 8s - loss: 0.4428 - accuracy: 0.9093 - val_loss: 1.3855 - val_accuracy:
0.6094

Epoch 36/50
35/35 - 8s - loss: 0.4393 - accuracy: 0.9095 - val_loss: 1.3848 - val_accuracy:
0.6110

Epoch 37/50
35/35 - 8s - loss: 0.4354 - accuracy: 0.9109 - val_loss: 1.3836 - val_accuracy:
0.6118

Epoch 38/50
35/35 - 7s - loss: 0.4310 - accuracy: 0.9123 - val_loss: 1.3821 - val_accuracy:
0.6110

Epoch 39/50
```

```
35/35 - 7s - loss: 0.4284 - accuracy: 0.9109 - val_loss: 1.3808 - val_accuracy:
0.6150

Epoch 40/50
35/35 - 7s - loss: 0.4239 - accuracy: 0.9134 - val_loss: 1.3788 - val_accuracy:
0.6118

Epoch 41/50
35/35 - 7s - loss: 0.4189 - accuracy: 0.9163 - val_loss: 1.3769 - val_accuracy:
0.6150

Epoch 42/50
35/35 - 8s - loss: 0.4152 - accuracy: 0.9166 - val_loss: 1.3773 - val_accuracy:
0.6126

Epoch 43/50
35/35 - 7s - loss: 0.4136 - accuracy: 0.9161 - val_loss: 1.3753 - val_accuracy:
0.6126

Epoch 44/50
35/35 - 7s - loss: 0.4080 - accuracy: 0.9168 - val_loss: 1.3737 - val_accuracy:
0.6142

Epoch 45/50
35/35 - 7s - loss: 0.4082 - accuracy: 0.9163 - val_loss: 1.3737 - val_accuracy:
0.6158

Epoch 46/50
35/35 - 7s - loss: 0.4020 - accuracy: 0.9186 - val_loss: 1.3729 - val_accuracy:
0.6158

Epoch 47/50
35/35 - 7s - loss: 0.3999 - accuracy: 0.9176 - val_loss: 1.3714 - val_accuracy:
0.6158

Epoch 48/50
35/35 - 7s - loss: 0.3975 - accuracy: 0.9183 - val_loss: 1.3707 - val_accuracy:
0.6166

Epoch 49/50
35/35 - 7s - loss: 0.3935 - accuracy: 0.9208 - val_loss: 1.3696 - val_accuracy:
0.6182

Epoch 50/50
35/35 - 7s - loss: 0.3938 - accuracy: 0.9217 - val_loss: 1.3687 - val_accuracy:
0.6174

39/39 - 1s

Train Accuracy [0.37031805515289307, 0.9266235828399658] , Test Accuracy [1.368
7020540237427, 0.6174334287643433]
Epoch 1/50
44/44 - 8s - loss: 2.4755 - accuracy: 0.3639 - val_loss: 2.1934 - val_accuracy:
0.3438

Epoch 2/50
44/44 - 7s - loss: 1.8171 - accuracy: 0.3842 - val_loss: 2.0101 - val_accuracy:
0.3503

Epoch 3/50
44/44 - 7s - loss: 1.6097 - accuracy: 0.3965 - val_loss: 1.9292 - val_accuracy:
0.3584
```

```
Epoch 4/50
44/44 - 8s - loss: 1.4994 - accuracy: 0.4082 - val_loss: 1.8881 - val_accuracy:
0.3737

Epoch 5/50
44/44 - 7s - loss: 1.4220 - accuracy: 0.4231 - val_loss: 1.8535 - val_accuracy:
0.3826

Epoch 6/50
44/44 - 7s - loss: 1.3640 - accuracy: 0.4405 - val_loss: 1.8318 - val_accuracy:
0.3914

Epoch 7/50
44/44 - 8s - loss: 1.3115 - accuracy: 0.4569 - val_loss: 1.8107 - val_accuracy:
0.3995

Epoch 8/50
44/44 - 7s - loss: 1.2692 - accuracy: 0.4795 - val_loss: 1.7921 - val_accuracy:
0.4100

Epoch 9/50
44/44 - 8s - loss: 1.2247 - accuracy: 0.5090 - val_loss: 1.7692 - val_accuracy:
0.4286

Epoch 10/50
44/44 - 7s - loss: 1.1825 - accuracy: 0.5530 - val_loss: 1.7506 - val_accuracy:
0.4407

Epoch 11/50
44/44 - 7s - loss: 1.1329 - accuracy: 0.6028 - val_loss: 1.7245 - val_accuracy:
0.4625

Epoch 12/50
44/44 - 7s - loss: 1.0845 - accuracy: 0.6477 - val_loss: 1.6971 - val_accuracy:
0.4778

Epoch 13/50
44/44 - 7s - loss: 1.0449 - accuracy: 0.6777 - val_loss: 1.6785 - val_accuracy:
0.4883

Epoch 14/50
44/44 - 7s - loss: 1.0022 - accuracy: 0.6990 - val_loss: 1.6540 - val_accuracy:
0.5004

Epoch 15/50
44/44 - 7s - loss: 0.9651 - accuracy: 0.7155 - val_loss: 1.6322 - val_accuracy:
0.5044

Epoch 16/50
44/44 - 7s - loss: 0.9331 - accuracy: 0.7299 - val_loss: 1.6217 - val_accuracy:
0.5101

Epoch 17/50
44/44 - 7s - loss: 0.9059 - accuracy: 0.7390 - val_loss: 1.6041 - val_accuracy:
0.5133

Epoch 18/50
44/44 - 7s - loss: 0.8821 - accuracy: 0.7480 - val_loss: 1.5933 - val_accuracy:
0.5157

Epoch 19/50
```

```
44/44 - 7s - loss: 0.8618 - accuracy: 0.7533 - val_loss: 1.5778 - val_accuracy:
0.5214

Epoch 20/50
44/44 - 7s - loss: 0.8455 - accuracy: 0.7628 - val_loss: 1.5733 - val_accuracy:
0.5246

Epoch 21/50
44/44 - 7s - loss: 0.8232 - accuracy: 0.7719 - val_loss: 1.5669 - val_accuracy:
0.5254

Epoch 22/50
44/44 - 7s - loss: 0.8111 - accuracy: 0.7748 - val_loss: 1.5564 - val_accuracy:
0.5303

Epoch 23/50
44/44 - 7s - loss: 0.7957 - accuracy: 0.7818 - val_loss: 1.5506 - val_accuracy:
0.5311

Epoch 24/50
44/44 - 7s - loss: 0.7831 - accuracy: 0.7881 - val_loss: 1.5434 - val_accuracy:
0.5367

Epoch 25/50
44/44 - 7s - loss: 0.7682 - accuracy: 0.7939 - val_loss: 1.5406 - val_accuracy:
0.5359

Epoch 26/50
44/44 - 7s - loss: 0.7593 - accuracy: 0.8009 - val_loss: 1.5318 - val_accuracy:
0.5416

Epoch 27/50
44/44 - 8s - loss: 0.7487 - accuracy: 0.8042 - val_loss: 1.5266 - val_accuracy:
0.5464

Epoch 28/50
44/44 - 7s - loss: 0.7380 - accuracy: 0.8106 - val_loss: 1.5209 - val_accuracy:
0.5488

Epoch 29/50
44/44 - 7s - loss: 0.7253 - accuracy: 0.8154 - val_loss: 1.5166 - val_accuracy:
0.5521

Epoch 30/50
44/44 - 7s - loss: 0.7166 - accuracy: 0.8198 - val_loss: 1.5127 - val_accuracy:
0.5537

Epoch 31/50
44/44 - 7s - loss: 0.7087 - accuracy: 0.8228 - val_loss: 1.5070 - val_accuracy:
0.5601

Epoch 32/50
44/44 - 7s - loss: 0.7010 - accuracy: 0.8269 - val_loss: 1.5050 - val_accuracy:
0.5617

Epoch 33/50
44/44 - 7s - loss: 0.6891 - accuracy: 0.8300 - val_loss: 1.5003 - val_accuracy:
0.5634

Epoch 34/50
44/44 - 8s - loss: 0.6814 - accuracy: 0.8331 - val_loss: 1.4979 - val_accuracy:
0.5666
```

```
Epoch 35/50
44/44 - 8s - loss: 0.6759 - accuracy: 0.8346 - val_loss: 1.4948 - val_accuracy:
0.5666

Epoch 36/50
44/44 - 8s - loss: 0.6691 - accuracy: 0.8360 - val_loss: 1.4896 - val_accuracy:
0.5674

Epoch 37/50
44/44 - 7s - loss: 0.6631 - accuracy: 0.8379 - val_loss: 1.4891 - val_accuracy:
0.5674

Epoch 38/50
44/44 - 7s - loss: 0.6560 - accuracy: 0.8406 - val_loss: 1.4866 - val_accuracy:
0.5714

Epoch 39/50
44/44 - 8s - loss: 0.6501 - accuracy: 0.8402 - val_loss: 1.4865 - val_accuracy:
0.5698

Epoch 40/50
44/44 - 8s - loss: 0.6447 - accuracy: 0.8410 - val_loss: 1.4818 - val_accuracy:
0.5730

Epoch 41/50
44/44 - 7s - loss: 0.6398 - accuracy: 0.8434 - val_loss: 1.4813 - val_accuracy:
0.5738

Epoch 42/50
44/44 - 7s - loss: 0.6337 - accuracy: 0.8457 - val_loss: 1.4799 - val_accuracy:
0.5747

Epoch 43/50
44/44 - 7s - loss: 0.6296 - accuracy: 0.8503 - val_loss: 1.4771 - val_accuracy:
0.5755

Epoch 44/50
44/44 - 8s - loss: 0.6295 - accuracy: 0.8477 - val_loss: 1.4744 - val_accuracy:
0.5771

Epoch 45/50
44/44 - 7s - loss: 0.6228 - accuracy: 0.8490 - val_loss: 1.4726 - val_accuracy:
0.5787

Epoch 46/50
44/44 - 7s - loss: 0.6190 - accuracy: 0.8508 - val_loss: 1.4724 - val_accuracy:
0.5763

Epoch 47/50
44/44 - 7s - loss: 0.6130 - accuracy: 0.8532 - val_loss: 1.4704 - val_accuracy:
0.5787

Epoch 48/50
44/44 - 7s - loss: 0.6107 - accuracy: 0.8515 - val_loss: 1.4689 - val_accuracy:
0.5803

Epoch 49/50
44/44 - 7s - loss: 0.6015 - accuracy: 0.8553 - val_loss: 1.4665 - val_accuracy:
0.5811

Epoch 50/50
```

```
44/44 - 7s - loss: 0.6035 - accuracy: 0.8573 - val_loss: 1.4654 - val_accuracy:
0.5819

39/39 - 1s

Train Accuracy [0.5718443989753723, 0.8658952116966248] , Test Accuracy [1.4654
066562652588, 0.5819209218025208]
Epoch 1/50
30/30 - 4s - loss: 2.2637 - accuracy: 0.4026 - val_loss: 2.0660 - val_accuracy:
0.4286

Epoch 2/50
30/30 - 4s - loss: 1.6963 - accuracy: 0.5327 - val_loss: 1.9432 - val_accuracy:
0.4504

Epoch 3/50
30/30 - 4s - loss: 1.5168 - accuracy: 0.5930 - val_loss: 1.8694 - val_accuracy:
0.4705

Epoch 4/50
30/30 - 4s - loss: 1.4019 - accuracy: 0.6195 - val_loss: 1.8157 - val_accuracy:
0.4899

Epoch 5/50
30/30 - 4s - loss: 1.3206 - accuracy: 0.6516 - val_loss: 1.7812 - val_accuracy:
0.4939

Epoch 6/50
30/30 - 4s - loss: 1.2611 - accuracy: 0.6728 - val_loss: 1.7538 - val_accuracy:
0.4939

Epoch 7/50
30/30 - 4s - loss: 1.2100 - accuracy: 0.6912 - val_loss: 1.7353 - val_accuracy:
0.5004

Epoch 8/50
30/30 - 4s - loss: 1.1694 - accuracy: 0.7041 - val_loss: 1.7182 - val_accuracy:
0.5061

Epoch 9/50
30/30 - 4s - loss: 1.1369 - accuracy: 0.7131 - val_loss: 1.7049 - val_accuracy:
0.5061

Epoch 10/50
30/30 - 4s - loss: 1.1050 - accuracy: 0.7237 - val_loss: 1.6927 - val_accuracy:
0.5133

Epoch 11/50
30/30 - 4s - loss: 1.0834 - accuracy: 0.7292 - val_loss: 1.6828 - val_accuracy:
0.5157

Epoch 12/50
30/30 - 4s - loss: 1.0576 - accuracy: 0.7379 - val_loss: 1.6695 - val_accuracy:
0.5214

Epoch 13/50
30/30 - 4s - loss: 1.0363 - accuracy: 0.7473 - val_loss: 1.6615 - val_accuracy:
0.5270

Epoch 14/50
30/30 - 4s - loss: 1.0177 - accuracy: 0.7517 - val_loss: 1.6522 - val_accuracy:
0.5295
```

```
Epoch 15/50
30/30 - 4s - loss: 1.0017 - accuracy: 0.7581 - val_loss: 1.6448 - val_accuracy:
0.5311

Epoch 16/50
30/30 - 4s - loss: 0.9839 - accuracy: 0.7636 - val_loss: 1.6392 - val_accuracy:
0.5303

Epoch 17/50
30/30 - 4s - loss: 0.9719 - accuracy: 0.7650 - val_loss: 1.6338 - val_accuracy:
0.5327

Epoch 18/50
30/30 - 4s - loss: 0.9573 - accuracy: 0.7715 - val_loss: 1.6262 - val_accuracy:
0.5343

Epoch 19/50
30/30 - 4s - loss: 0.9447 - accuracy: 0.7739 - val_loss: 1.6215 - val_accuracy:
0.5335

Epoch 20/50
30/30 - 4s - loss: 0.9312 - accuracy: 0.7776 - val_loss: 1.6172 - val_accuracy:
0.5351

Epoch 21/50
30/30 - 4s - loss: 0.9222 - accuracy: 0.7806 - val_loss: 1.6121 - val_accuracy:
0.5343

Epoch 22/50
30/30 - 4s - loss: 0.9123 - accuracy: 0.7837 - val_loss: 1.6064 - val_accuracy:
0.5391

Epoch 23/50
30/30 - 4s - loss: 0.9012 - accuracy: 0.7878 - val_loss: 1.6023 - val_accuracy:
0.5400

Epoch 24/50
30/30 - 4s - loss: 0.8918 - accuracy: 0.7907 - val_loss: 1.5989 - val_accuracy:
0.5408

Epoch 25/50
30/30 - 4s - loss: 0.8848 - accuracy: 0.7922 - val_loss: 1.5936 - val_accuracy:
0.5432

Epoch 26/50
30/30 - 4s - loss: 0.8741 - accuracy: 0.7946 - val_loss: 1.5897 - val_accuracy:
0.5448

Epoch 27/50
30/30 - 4s - loss: 0.8681 - accuracy: 0.7979 - val_loss: 1.5883 - val_accuracy:
0.5440

Epoch 28/50
30/30 - 4s - loss: 0.8597 - accuracy: 0.7975 - val_loss: 1.5826 - val_accuracy:
0.5432

Epoch 29/50
30/30 - 4s - loss: 0.8551 - accuracy: 0.7989 - val_loss: 1.5794 - val_accuracy:
0.5448

Epoch 30/50
```

```
30/30 - 4s - loss: 0.8448 - accuracy: 0.8031 - val_loss: 1.5780 - val_accuracy:
0.5488

Epoch 31/50
30/30 - 5s - loss: 0.8369 - accuracy: 0.8061 - val_loss: 1.5751 - val_accuracy:
0.5488

Epoch 32/50
30/30 - 5s - loss: 0.8364 - accuracy: 0.8045 - val_loss: 1.5711 - val_accuracy:
0.5504

Epoch 33/50
30/30 - 4s - loss: 0.8293 - accuracy: 0.8082 - val_loss: 1.5686 - val_accuracy:
0.5513

Epoch 34/50
30/30 - 4s - loss: 0.8218 - accuracy: 0.8110 - val_loss: 1.5672 - val_accuracy:
0.5521

Epoch 35/50
30/30 - 4s - loss: 0.8169 - accuracy: 0.8097 - val_loss: 1.5645 - val_accuracy:
0.5521

Epoch 36/50
30/30 - 4s - loss: 0.8128 - accuracy: 0.8117 - val_loss: 1.5615 - val_accuracy:
0.5529

Epoch 37/50
30/30 - 4s - loss: 0.8038 - accuracy: 0.8147 - val_loss: 1.5581 - val_accuracy:
0.5529

Epoch 38/50
30/30 - 4s - loss: 0.8006 - accuracy: 0.8152 - val_loss: 1.5576 - val_accuracy:
0.5545

Epoch 39/50
30/30 - 4s - loss: 0.7959 - accuracy: 0.8161 - val_loss: 1.5564 - val_accuracy:
0.5577

Epoch 40/50
30/30 - 4s - loss: 0.7920 - accuracy: 0.8192 - val_loss: 1.5540 - val_accuracy:
0.5577

Epoch 41/50
30/30 - 4s - loss: 0.7852 - accuracy: 0.8229 - val_loss: 1.5508 - val_accuracy:
0.5577

Epoch 42/50
30/30 - 4s - loss: 0.7817 - accuracy: 0.8235 - val_loss: 1.5487 - val_accuracy:
0.5593

Epoch 43/50
30/30 - 4s - loss: 0.7772 - accuracy: 0.8264 - val_loss: 1.5466 - val_accuracy:
0.5569

Epoch 44/50
30/30 - 4s - loss: 0.7722 - accuracy: 0.8265 - val_loss: 1.5464 - val_accuracy:
0.5577

Epoch 45/50
30/30 - 4s - loss: 0.7676 - accuracy: 0.8257 - val_loss: 1.5448 - val_accuracy:
0.5617
```

```
Epoch 46/50
30/30 - 4s - loss: 0.7669 - accuracy: 0.8273 - val_loss: 1.5439 - val_accuracy:
0.5626

Epoch 47/50
30/30 - 4s - loss: 0.7620 - accuracy: 0.8276 - val_loss: 1.5414 - val_accuracy:
0.5626

Epoch 48/50
30/30 - 4s - loss: 0.7604 - accuracy: 0.8287 - val_loss: 1.5401 - val_accuracy:
0.5617

Epoch 49/50
30/30 - 4s - loss: 0.7540 - accuracy: 0.8297 - val_loss: 1.5387 - val_accuracy:
0.5617

Epoch 50/50
30/30 - 4s - loss: 0.7498 - accuracy: 0.8337 - val_loss: 1.5382 - val_accuracy:
0.5634

39/39 - 0s

Train Accuracy [0.7194824814796448, 0.8421241641044617] , Test Accuracy [1.5382
425785064697, 0.5633575320243835]
100%|██████████| 20/20 [1:16:36<00:00, 229.84s/it, best loss: -0.61985472154963
68]
It took 4635.30 seconds
```

In [ ]:

```
model = Sequential()
model.add(Embedding(num_words, embedding_size, embeddings_initializer = Constan
t(embedding_matrix), input_length = maxlen, trainable = False))
model.add(Flatten())
model.add(Dense(500, input_shape=((embedding_size*maxlen),), activation='relu'
))
model.add(Dense(100, activation='relu'))
model.add(Dense(number_of_classes, activation='softmax'))

adam = optimizers.Adam(lr=0.0001)
# Compile model
model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accura
cy'])
```

In [ ]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 25, 300)           5856900
_________________________________________________________________
flatten (Flatten)            (None, 7500)              0
_________________________________________________________________
dense (Dense)                (None, 500)               3750500
_________________________________________________________________
dense_1 (Dense)              (None, 100)               50100
_________________________________________________________________
dense_2 (Dense)              (None, 30)                3030
=================================================================
Total params: 9,660,530
Trainable params: 3,803,630
Non-trainable params: 5,856,900
_________________________________________________________________
```

In [ ]:
```
# Fit the model and evaluate score
start = time.time()
history = model.fit(X_train, y_train, epochs=50, batch_size=128, verbose= 0)
end = time.time()
print(f"model training time is {end - start} seconds")

score_train = model.evaluate(X_train, y_train, verbose=0)
score_test = model.evaluate(X_test, y_test, verbose=0)

print(f'Train Accuracy {score_train} , Test Accuracy {score_test} ')
```

```
model training time is 204.15734028816223 seconds
Train Accuracy [0.04331778734922409, 0.9883387088775635] , Test Accuracy [1.828
0701637268066, 0.6263115406036377]
```

In [ ]:
```
y_pred_train = np.argmax(model.predict(X_train), axis=-1)

start = time.time()
y_pred = np.argmax(model.predict(X_test), axis=-1)
end = time.time()
print(f"model prediction time is {end - start} seconds")

y_train = np.argmax(y_train, axis=-1)
y_test = np.argmax(y_test, axis=-1)
print(f'train accuracy : {accuracy_score(y_train,y_pred_train)}')
print(f'test accuracy :{accuracy_score(y_test,y_pred)}')
fsc = f1_score(y_test, y_pred, average='macro')
pres = precision_score(y_test, y_pred, average='macro')
rec = recall_score(y_test, y_pred, average='macro')
print(f'F1 score : {fsc}')
print(f'Recall Score : {rec}')
print(f'Precision Score : {pres}')
```

```
model prediction time is 0.2893257141113281 seconds
train accuracy : 0.9883387154646573
test accuracy :0.6263115415657788
F1 score : 0.560401006467488
Recall Score : 0.5258883521464058
Precision Score : 0.6369960167654113
```

Neural network - Bidirectional Lstm

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, rand
om_state = 42, shuffle = True)
```

In [ ]:
```
from hyperopt import hp, fmin, tpe, hp, STATUS_OK, Trials, space_eval,rand

DROPOUT_CHOICES = np.arange(0.0, 0.4, 0.1)
LSTM_UNIT_CHOICES = np.arange(120, 600, 60, dtype=int)
DENSE_UNIT_CHOICES = np.arange(60, 300, 30, dtype=int)
BATCH_SIZE_CHOICES = np.arange(64, 512, 64, dtype=int)
space = {

    'spatial_dropout': hp.choice('spatial_dropout', DROPOUT_CHOICES),
    'lstm_units': hp.choice('lstm_units', LSTM_UNIT_CHOICES),
    'lstm_dropout':  hp.choice('lstm_dropout', DROPOUT_CHOICES),
    'lstm_rec_dropout':  hp.choice('lstm_rec_dropout', DROPOUT_CHOICES),
    'dense_units':  hp.choice('dense_units', DENSE_UNIT_CHOICES),
    'batch_size':  hp.choice('batch_size', BATCH_SIZE_CHOICES)
}
```

In [ ]:
```
def objective(params, verbose=1, checkpoint_path = '/content/drive/My Drive/Trial code/Kapil/Models/model_2.hdf5'):

    if verbose > 0:
        print ('Params testing: ', params)
        print ('\n ')

    model = Sequential()
    model.add(Embedding(num_words, output_dim=embedding_size, embeddings_initializer = Constant(embedding_matrix), input_length = maxlen, trainable = True))
    model.add(SpatialDropout1D(params['spatial_dropout']))
    model.add(Bidirectional(LSTM(params['lstm_units'], dropout=params['lstm_dropout'], recurrent_dropout=params['lstm_rec_dropout'], return_sequences=True)))
    model.add(Bidirectional(LSTM(params['lstm_units'], dropout=params['lstm_dropout'], recurrent_dropout=params['lstm_rec_dropout'])))
    model.add(Dense(params['dense_units'], activation='relu'))
    model.add(Dense(number_of_classes, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

    model.fit(
        X_train,
        y_train,
        validation_data=(X_test, y_test),
        epochs=30,
        batch_size=params['batch_size'],
        callbacks= [
            EarlyStopping(patience = 2, min_delta=0.01, verbose=2,  monitor='val_accuracy')
        ],
        verbose=2
    )

    predictions = model.predict(X_test, verbose=2)
    acc = (predictions.argmax(axis = 1) == y_test.argmax(axis = 1)).mean()
    return {'loss': -acc, 'status': STATUS_OK}
```

In [ ]:
```
trials = Trials()
best = fmin(objective, space, algo=rand.suggest, trials=trials, max_evals=4, rs
tate=np.random.RandomState(99))
```

```
Params testing:
{'batch_size': 384, 'dense_units': 270, 'lstm_dropout': 0.30000000000000004, 'l
stm_rec_dropout': 0.1, 'lstm_units': 240, 'spatial_dropout': 0.1}
Epoch 1/30
30/30 - 126s - loss: 2.3950 - accuracy: 0.3774 - val_loss: 2.0486 - val_accurac
y: 0.3979

Epoch 2/30
30/30 - 126s - loss: 1.8015 - accuracy: 0.4694 - val_loss: 1.7171 - val_accurac
y: 0.4576

Epoch 3/30
30/30 - 124s - loss: 1.5257 - accuracy: 0.5358 - val_loss: 1.6503 - val_accurac
y: 0.5303

Epoch 4/30
30/30 - 123s - loss: 1.2949 - accuracy: 0.5958 - val_loss: 1.3150 - val_accurac
y: 0.5973

Epoch 5/30
30/30 - 123s - loss: 1.0887 - accuracy: 0.6667 - val_loss: 1.1597 - val_accurac
y: 0.6465

Epoch 6/30
30/30 - 123s - loss: 0.9230 - accuracy: 0.7123 - val_loss: 1.0430 - val_accurac
y: 0.6901

Epoch 7/30
30/30 - 124s - loss: 0.7514 - accuracy: 0.7696 - val_loss: 0.9394 - val_accurac
y: 0.7078

Epoch 8/30
30/30 - 123s - loss: 0.6478 - accuracy: 0.8031 - val_loss: 0.9024 - val_accurac
y: 0.7312

Epoch 9/30
30/30 - 124s - loss: 0.5577 - accuracy: 0.8274 - val_loss: 0.7959 - val_accurac
y: 0.7676

Epoch 10/30
30/30 - 124s - loss: 0.4544 - accuracy: 0.8569 - val_loss: 0.7277 - val_accurac
y: 0.7845

Epoch 11/30
30/30 - 124s - loss: 0.3976 - accuracy: 0.8745 - val_loss: 0.7608 - val_accurac
y: 0.7958

Epoch 12/30
30/30 - 125s - loss: 0.3818 - accuracy: 0.8795 - val_loss: 0.6798 - val_accurac
y: 0.8015

Epoch 13/30
30/30 - 124s - loss: 0.3033 - accuracy: 0.9034 - val_loss: 0.6581 - val_accurac
y: 0.8152

Epoch 14/30
30/30 - 123s - loss: 0.2874 - accuracy: 0.9069 - val_loss: 0.6209 - val_accurac
y: 0.8200

Epoch 15/30
30/30 - 123s - loss: 0.2247 - accuracy: 0.9283 - val_loss: 0.6229 - val_accurac
```

```
y: 0.8345

Epoch 16/30
30/30 - 124s - loss: 0.2156 - accuracy: 0.9310 - val_loss: 0.6602 - val_accurac
y: 0.8329

Epoch 17/30
30/30 - 126s - loss: 0.1907 - accuracy: 0.9374 - val_loss: 0.6303 - val_accurac
y: 0.8273

Epoch 00017: early stopping
39/39 - 5s

Params testing:
{'batch_size': 384, 'dense_units': 120, 'lstm_dropout': 0.30000000000000004, 'l
stm_rec_dropout': 0.0, 'lstm_units': 480, 'spatial_dropout': 0.0}
Epoch 1/30
30/30 - 283s - loss: 2.3187 - accuracy: 0.3871 - val_loss: 2.0039 - val_accurac
y: 0.4149

Epoch 2/30
30/30 - 284s - loss: 1.6834 - accuracy: 0.4976 - val_loss: 1.6549 - val_accurac
y: 0.4996

Epoch 3/30
30/30 - 282s - loss: 1.3698 - accuracy: 0.5752 - val_loss: 1.3715 - val_accurac
y: 0.5714

Epoch 4/30
30/30 - 283s - loss: 1.1231 - accuracy: 0.6480 - val_loss: 1.2034 - val_accurac
y: 0.5989

Epoch 5/30
30/30 - 281s - loss: 0.9508 - accuracy: 0.7014 - val_loss: 1.1127 - val_accurac
y: 0.6473

Epoch 6/30
30/30 - 284s - loss: 0.7365 - accuracy: 0.7687 - val_loss: 0.9549 - val_accurac
y: 0.7183

Epoch 7/30
30/30 - 284s - loss: 0.6672 - accuracy: 0.7902 - val_loss: 0.9195 - val_accurac
y: 0.7127

Epoch 8/30
30/30 - 285s - loss: 0.5470 - accuracy: 0.8233 - val_loss: 0.7860 - val_accurac
y: 0.7692

Epoch 9/30
30/30 - 282s - loss: 0.4614 - accuracy: 0.8577 - val_loss: 0.7677 - val_accurac
y: 0.7797

Epoch 10/30
30/30 - 283s - loss: 0.3801 - accuracy: 0.8812 - val_loss: 0.6779 - val_accurac
y: 0.8087

Epoch 11/30
30/30 - 283s - loss: 0.2750 - accuracy: 0.9125 - val_loss: 0.6587 - val_accurac
y: 0.8095

Epoch 12/30
30/30 - 284s - loss: 0.2528 - accuracy: 0.9195 - val_loss: 0.6991 - val_accurac
```

```
y: 0.7934

Epoch 00012: early stopping
39/39 - 12s

Params testing:
{'batch_size': 448, 'dense_units': 60, 'lstm_dropout': 0.0, 'lstm_rec_dropout':
0.30000000000000004, 'lstm_units': 360, 'spatial_dropout': 0.30000000000000004}
Epoch 1/30
25/25 - 206s - loss: 2.4291 - accuracy: 0.3672 - val_loss: 2.1108 - val_accurac
y: 0.3914

Epoch 2/30
25/25 - 207s - loss: 1.8492 - accuracy: 0.4711 - val_loss: 1.7024 - val_accurac
y: 0.4835

Epoch 3/30
25/25 - 210s - loss: 1.5149 - accuracy: 0.5441 - val_loss: 1.4403 - val_accurac
y: 0.5488

Epoch 4/30
25/25 - 208s - loss: 1.2513 - accuracy: 0.6271 - val_loss: 1.2140 - val_accurac
y: 0.6215

Epoch 5/30
25/25 - 207s - loss: 1.0318 - accuracy: 0.6807 - val_loss: 1.0652 - val_accurac
y: 0.6723

Epoch 6/30
25/25 - 210s - loss: 0.8702 - accuracy: 0.7319 - val_loss: 1.0146 - val_accurac
y: 0.6973

Epoch 7/30
25/25 - 207s - loss: 0.7204 - accuracy: 0.7768 - val_loss: 0.9011 - val_accurac
y: 0.7353

Epoch 8/30
25/25 - 207s - loss: 0.6052 - accuracy: 0.8109 - val_loss: 0.8529 - val_accurac
y: 0.7571

Epoch 9/30
25/25 - 206s - loss: 0.5231 - accuracy: 0.8379 - val_loss: 0.7423 - val_accurac
y: 0.7805

Epoch 10/30
25/25 - 206s - loss: 0.4214 - accuracy: 0.8700 - val_loss: 0.7030 - val_accurac
y: 0.7885

Epoch 11/30
25/25 - 205s - loss: 0.3601 - accuracy: 0.8843 - val_loss: 0.6554 - val_accurac
y: 0.8128

Epoch 12/30
25/25 - 206s - loss: 0.3080 - accuracy: 0.9010 - val_loss: 0.6329 - val_accurac
y: 0.8241

Epoch 13/30
25/25 - 208s - loss: 0.2705 - accuracy: 0.9164 - val_loss: 0.6359 - val_accurac
y: 0.8152

Epoch 14/30
25/25 - 207s - loss: 0.2306 - accuracy: 0.9233 - val_loss: 0.6203 - val_accurac
```

```
y: 0.8345

Epoch 15/30
25/25 - 207s - loss: 0.2021 - accuracy: 0.9337 - val_loss: 0.5931 - val_accurac
y: 0.8402

Epoch 16/30
25/25 - 206s - loss: 0.1854 - accuracy: 0.9389 - val_loss: 0.6168 - val_accurac
y: 0.8483

Epoch 17/30
25/25 - 207s - loss: 0.1623 - accuracy: 0.9458 - val_loss: 0.5648 - val_accurac
y: 0.8507

Epoch 18/30
25/25 - 207s - loss: 0.1422 - accuracy: 0.9529 - val_loss: 0.5976 - val_accurac
y: 0.8418

Epoch 00018: early stopping
39/39 - 13s

Params testing:
{'batch_size': 128, 'dense_units': 120, 'lstm_dropout': 0.0, 'lstm_rec_dropou
t': 0.0, 'lstm_units': 540, 'spatial_dropout': 0.30000000000000004}
Epoch 1/30
88/88 - 414s - loss: 2.0473 - accuracy: 0.4325 - val_loss: 1.6819 - val_accurac
y: 0.4939

Epoch 2/30
88/88 - 411s - loss: 1.4570 - accuracy: 0.5575 - val_loss: 1.3288 - val_accurac
y: 0.5924

Epoch 3/30
88/88 - 411s - loss: 1.1302 - accuracy: 0.6510 - val_loss: 1.1429 - val_accurac
y: 0.6618

Epoch 4/30
88/88 - 412s - loss: 0.8675 - accuracy: 0.7276 - val_loss: 0.9962 - val_accurac
y: 0.6949

Epoch 5/30
88/88 - 411s - loss: 0.6769 - accuracy: 0.7876 - val_loss: 0.8425 - val_accurac
y: 0.7619

Epoch 6/30
88/88 - 412s - loss: 0.4899 - accuracy: 0.8428 - val_loss: 0.7809 - val_accurac
y: 0.7724

Epoch 7/30
88/88 - 413s - loss: 0.3621 - accuracy: 0.8852 - val_loss: 0.6193 - val_accurac
y: 0.8241

Epoch 8/30
88/88 - 411s - loss: 0.2908 - accuracy: 0.9074 - val_loss: 0.6827 - val_accurac
y: 0.8119

Epoch 9/30
88/88 - 415s - loss: 0.2151 - accuracy: 0.9323 - val_loss: 0.6421 - val_accurac
y: 0.8289

Epoch 00009: early stopping
39/39 - 15s
```

```
100%|██████████| 4/4 [3:43:29<00:00, 3352.44s/it, best loss: -0.841807909604519
8]
```

In [ ]:

```python
model = Sequential()
model.add(Embedding(num_words, output_dim=embedding_size, embeddings_initialize
r = Constant(embedding_matrix), input_length = maxlen, trainable = True))
model.add(SpatialDropout1D(0.3))
model.add(Bidirectional(LSTM(200, dropout=0.25, recurrent_dropout=0.25, return_
sequences=True)))
model.add(Bidirectional(LSTM(200, dropout=0.25, recurrent_dropout=0.25)))
model.add(Dense(100, activation='relu'))
model.add(Dense(number_of_classes, activation='softmax'))
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [ ]:

```python
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_1 (Embedding)      (None, 25, 300)           5856900   
_________________________________________________________________
bidirectional (Bidirectional (None, 25, 200)           320800    
_________________________________________________________________
time_distributed (TimeDistri (None, 25, 100)           20100     
_________________________________________________________________
flatten_1 (Flatten)          (None, 2500)              0         
_________________________________________________________________
dense_4 (Dense)              (None, 300)               750300    
_________________________________________________________________
dense_5 (Dense)              (None, 30)                9030      
=================================================================
Total params: 6,957,130
Trainable params: 1,100,230
Non-trainable params: 5,856,900
_________________________________________________________________
```

In [ ]:

```
# Fit the model and evaluate score
start = time.time()
# history = model.fit(X_train, y_train, epochs=30, batch_size=128, verbose= 0)
model.fit(
        X_train,
        y_train,
        validation_data=(X_test, y_test),
        epochs=30,
        batch_size=128,
        callbacks= [
            EarlyStopping(patience = 2, min_delta=0.01, verbose=1,  monitor='va
l_accuracy')
        ],
        verbose=1
    )
end = time.time()
print(f"model training time is {end - start} seconds")
```

```
Epoch 1/30
88/88 [==============================] - 143s 2s/step - loss: 2.2019 - accurac
y: 0.4003 - val_loss: 1.8570 - val_accuracy: 0.4326
Epoch 2/30
88/88 [==============================] - 145s 2s/step - loss: 1.7169 - accurac
y: 0.4872 - val_loss: 1.5685 - val_accuracy: 0.5311
Epoch 3/30
88/88 [==============================] - 143s 2s/step - loss: 1.4207 - accurac
y: 0.5702 - val_loss: 1.3645 - val_accuracy: 0.5827
Epoch 4/30
88/88 [==============================] - 144s 2s/step - loss: 1.2119 - accurac
y: 0.6218 - val_loss: 1.2044 - val_accuracy: 0.6416
Epoch 5/30
88/88 [==============================] - 143s 2s/step - loss: 1.0463 - accurac
y: 0.6788 - val_loss: 1.0744 - val_accuracy: 0.6691
Epoch 6/30
88/88 [==============================] - 146s 2s/step - loss: 0.8716 - accurac
y: 0.7266 - val_loss: 0.9268 - val_accuracy: 0.7264
Epoch 7/30
88/88 [==============================] - 143s 2s/step - loss: 0.7407 - accurac
y: 0.7671 - val_loss: 0.8358 - val_accuracy: 0.7538
Epoch 8/30
88/88 [==============================] - 145s 2s/step - loss: 0.6158 - accurac
y: 0.8083 - val_loss: 0.7850 - val_accuracy: 0.7684
Epoch 9/30
88/88 [==============================] - 143s 2s/step - loss: 0.5351 - accurac
y: 0.8349 - val_loss: 0.7625 - val_accuracy: 0.7797
Epoch 10/30
88/88 [==============================] - 145s 2s/step - loss: 0.4650 - accurac
y: 0.8538 - val_loss: 0.6943 - val_accuracy: 0.7974
Epoch 11/30
88/88 [==============================] - 143s 2s/step - loss: 0.3844 - accurac
y: 0.8776 - val_loss: 0.6882 - val_accuracy: 0.8087
Epoch 12/30
88/88 [==============================] - 146s 2s/step - loss: 0.3478 - accurac
y: 0.8893 - val_loss: 0.6661 - val_accuracy: 0.8095
Epoch 13/30
88/88 [==============================] - 143s 2s/step - loss: 0.3282 - accurac
y: 0.8972 - val_loss: 0.6457 - val_accuracy: 0.8200
Epoch 14/30
88/88 [==============================] - 147s 2s/step - loss: 0.2674 - accurac
y: 0.9141 - val_loss: 0.5996 - val_accuracy: 0.8434
Epoch 15/30
88/88 [==============================] - 143s 2s/step - loss: 0.2414 - accurac
y: 0.9223 - val_loss: 0.5994 - val_accuracy: 0.8345
Epoch 16/30
88/88 [==============================] - 147s 2s/step - loss: 0.2207 - accurac
y: 0.9264 - val_loss: 0.6148 - val_accuracy: 0.8329
Epoch 00016: early stopping
model training time is 2346.4287717342377 seconds
```

In [ ]:

```
y_pred_train = np.argmax(model.predict(X_train), axis=-1)
start = time.time()
y_pred = np.argmax(model.predict(X_test), axis=-1)
end = time.time()
print(f"model prediction time is {end - start} seconds")
y_train = np.argmax(y_train, axis=-1)
y_test = np.argmax(y_test, axis=-1)
print(f'train accuracy : {accuracy_score(y_train,y_pred_train)}')
print(f'test accuracy :{accuracy_score(y_test,y_pred)}')
fsc = f1_score(y_test, y_pred, average='macro')
pres = precision_score(y_test, y_pred, average='macro')
rec = recall_score(y_test, y_pred, average='macro')
print(f'F1 score : {fsc}')
print(f'Recall Score : {rec}')
print(f'Precision Score : {pres}')
```

```
model prediction time is 3.505152702331543 seconds
train accuracy : 0.9680660208109078
test accuracy :0.8329297820823245
F1 score : 0.8088428919963238
Recall Score : 0.8016101963222023
Precision Score : 0.8364069029227538
```