Capstone Project Report

Name: Sumit Patel

Course: AI and ML (Batch – AUG 2020)

Duration: 10 months

Speech – Music Recognition

Problem Statement:

Use GTZAN music genre dataset for classifying the song genre. Train a RNN that takes a song as an input and predicts the corresponding genre for the song. For implementation purposes, you can just use a specific time duration of the song as input (eg: just the first 10-20 seconds of each song). The dataset has a total of 10 classes, you may train for any 5 or all the classes using 90% of available data for training and rest for the test dataset.

Prerequisites

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has latest version of python. The following url https://www.python.org/downloads/ can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-externalcommand/. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic. Second and easier option is to download anaconda and use its anaconda prompt to run the commands.

To install anaconda check this url https://www.anaconda.com/download/ You will also need to download and install below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run below commands in command prompt/terminal to install these packages:

pip install numpy

pip install pandas

pip install sklearn

pip install tensorflow

If you have chosen to install anaconda then run below commands in anaconda prompt to install these packages:

conda install -c anaconda numpy

conda install -c anaconda pandas

conda install -c anaconda sklearn

conda install -c anaconda tensorflow

Importing the libraries and loading dataset.

```
# Import Libraries
import pandas as pd
import numpy as np
from sklearn import preprocessing
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.metrics import classification_report
```

Loading and reading data

```
# Import Dataset
data = pd.read_csv('../21. Speech Music Recognition/Data/features_30_sec.csv')
data.head()
```

	filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	s
0	blues.00000.wav	661794	0.350088	0.088757	0.130228	0.002827	1784.165850	129774.064525	2002.449060	
1	blues.00001.wav	661794	0.340914	0.094980	0.095948	0.002373	1530.176679	375850.073649	2039.036516	
2	blues.00002.wav	661794	0.363637	0.085275	0.175570	0.002746	1552.811865	156467.643368	1747.702312	
3	blues.00003.wav	661794	0.404785	0.093999	0.141093	0.006346	1070.106615	184355.942417	1596.412872	
4	blues.00004.wav	661794	0.308526	0.087841	0.091529	0.002303	1835.004266	343399.939274	1748.172116	

Data pre-processing

```
# Normalize the data
min_max = preprocessing.MinMaxScaler()
scaled_df = min_max.fit_transform(X.values)
final_df = pd.DataFrame(scaled_df,columns=X.columns)
final_df.head()
```

	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	spectral_bandwidth_var	rc
0	0.362279	0.695468	0.318188	0.101983	0.314117	0.040233	0.422879	0.109789	
1	0.343622	0.793392	0.230894	0.085580	0.248405	0.121475	0.436889	0.296867	
2	0.389832	0.640692	0.433652	0.099064	0.254261	0.049046	0.325334	0.095712	
3	0.473508	0.777954	0.345856	0.229160	0.129376	0.058253	0.267404	0.227566	
4	0.277759	0.681062	0.219641	0.083075	0.327270	0.110761	0.325514	0.113536	

```
# Convert Labels to one-Hot codes
cols = Y['label'].unique()
Y_encode = np.array(Y)
for i, item in enumerate(cols):
    ind = np.where(Y_encode == item)[0]
    Y_encode[ind] = i
Y_encode = pd.DataFrame(to_categorical(Y_encode))
Y_encode.head()
```

	0	1	2	3	4	5	6	7	8	9
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Training the model

Evaluating the model

```
y_preds_rnn = np.round(model_rnn.predict(x_test_rnn))
y_preds_rnn[:5]
```

```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 0., 0., 0., 0., 1., 0.]], dtype=float32)
```

```
print(classification_report(y_test, np.array(y_preds_rnn), target_names = Y['label'].unique()))
```

```
              precision    recall  f1-score   support

       blues       0.80      0.40      0.53        10
   classical       0.89      0.80      0.84        10
     country       0.40      0.25      0.31         8
       disco       0.50      0.08      0.14        12
      hiphop       0.88      0.47      0.61        15
        jazz       0.79      0.92      0.85        12
       metal       1.00      0.88      0.93         8
         pop       0.67      0.33      0.44         6
      reggae       0.54      0.70      0.61        10
        rock       1.00      0.33      0.50         9

   micro avg       0.75      0.52      0.62       100
   macro avg       0.75      0.52      0.58       100
weighted avg       0.75      0.52      0.58       100
 samples avg       0.52      0.52      0.52       100
```