Name : Umesh Kumar

Roll number : 210020146

SOC, IIT Bombay' 24 FINAL REPORT

Speech Emotion Recognition

AIM

In this project we are going to recognize the emotion contained in particular voice or mp3 file by combining techniques of Data Science and Deep learning

DATASET

- TESS - this dataset contains 2800 voices of 200 different words which are fitted in the statement "Say the word ____" by two people aged 24 and 64 years, containing 7 unique emotions in whole dataset.
- RAVEDESS dataset contains 1440 files, 60 trials per actor ad their are 24 actors recording voices in different emotions.

PROBLEM STATEMENT

We need to classify emotions in one of the eight types:

```
emotions
happy        592
disgust      592
fear         592
angry        592
surprised    592
sad          592
neutral      496
calm         192
Name: count, dtype: int64
```

APPROACH AND CODE

After pre-processing the audio files and creating a dataset we get total 4240 audios (2800 from TESS and 1440 from RAVEDESS) with their labels that what is the emotion

corresponding to that audio file.

Used **Librosa** and **Pyaudio** library to extract MEL () and MFCC (Mel-frequency cepstral coefficients) features out of Audio files and then appended them horizontally so that total features extracted from 1 audio file becomes vector of 141 length (13 MEL features and 128 MFCC features).

Finally total datashape we got is (4240, 141)

Now using scikit-learn made Train data (80% of total data) and Test data (20% of total data)

Training data length we got : 3392 and Testing data length we got : 848

Used label_encoder from scikit learn to convert labels into a vector of length 8. In which index corresponding to value 1 represents the particular emotion which corresponds to that label.

```
from tensorflow.keras.utils import to_categorical
lb= LabelEncoder()

y_train = to_categorical(lb.fit_transform(y_train))
y_test = to_categorical(lb.transform(y_test))

print(y_train.shape)
print(y_test.shape)
print(y_train[0:5])
print(y_test[0:5])
```

```
(3392, 8)
(848, 8)
[[0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0.]]
[[0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0.]]
```

CNN MODEL

```
model_cnn = Sequential([
    Conv1D(64, kernel_size=(8), activation='relu', input_shape=(X_train.shape[1], 1)),
    Conv1D(128, kernel_size=(8), activation='relu'),
    MaxPooling1D(pool_size=(4)),
    Dropout(0.2),

    Conv1D(128, kernel_size=(8), activation='relu'),
    MaxPooling1D(pool_size=(4)),
    Dropout(0.2),
    BatchNormalization(),
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.2),
    Dense(8, activation='softmax')
])
```

CNN model would be inputed with a vector of (141, 1) and outputs a vector of (8, 1) which corresponds to probability of each index (emotion)

After that we'll select the index corresponding to the maximum probability and labeling it.

This model is giving an F1 accuracy score of 0.81 on 30 epochs

LSTM MODEL

```
input_array= keras.Input(shape=(X_train.shape[1],1), name = 'input_array')
lstm_layer = LSTM(512, name = 'lstm_layer')(input_array)
dense_1 = Dense(256, activation= 'relu', kernel_initializer='he_normal', name= 'dense_1')(lstm_layer)
dropout_1 = Dropout(rate= 0.2,  name= 'dropout_1')(dense_1)
dense_2 = Dense(128, activation= 'relu', kernel_initializer='he_normal',  name= 'dense_2')(dropout_1)
bn= BatchNormalization()(dense_2)
output_layer = Dense(8, activation= 'softmax', name= 'output_layer')(bn)

model_lstm = Model(inputs=input_array, outputs= output_layer)
model_lstm.summary()
```

Here we used LSTM layer to predict the outputs and this is giving an F1 score of 0.73 on 200 epochs.

CONCLUSION

CNN model is pretty good and fast as compared to LSTM in this task.