In [1]:

```
import pickle
import pickle
file=open('/home/ece/Music/features FINAL','rb')
featuresdf=pickle.load(file)
file.close()
```

In [2]:

```
featuresdf
```

Out[2]:

	feature	class_label
0	[-95.34758, 131.61076, -80.91215, 95.421646, -...	KAWASAKI
1	[-44.43844, 75.582085, -28.329184, 23.416933, ...	HARLEY
2	[-8.866261, 49.40681, 21.061068, 28.81571, 6.4...	HARLEY
3	[32.28357, 65.85346, -18.080599, 18.747538, -1...	KTM
4	[-144.04903, 93.13892, -16.7343, 41.496807, -1...	KTM
...	...	...
235	[-227.46637, 116.28378, -11.652334, 43.611366,...	KTM
236	[-146.03716, 92.09145, -12.007631, 40.169983, ...	KTM
237	[-113.60618, 88.692184, -10.085182, 40.715332,...	KTM
238	[-150.85689, 89.39332, -12.109208, 40.25608, -...	KTM
239	[-205.87419, 121.9227, -10.783024, 42.32616, -...	KTM

240 rows × 2 columns

In [3]:

```
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical
import numpy as np
# Convert features and corresponding classification labels into numpy arrays
X = np.array(featuresdf.feature.tolist())
y = np.array(featuresdf.class_label.tolist())

# Encode the classification labels
le = LabelEncoder()
yy = to_categorical(le.fit_transform(y))
```

In [7]:

```
# split the dataset
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, yy, test_size=0.2, random_state = 42)
```

In [8]:

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D, MaxPooling2D
from keras.optimizers import Adam
from keras.utils import np_utils
from sklearn import metrics
```

In [9]:

```
import tensorflow as tf
from tensorflow import keras
```

In [11]:

```
model=keras.models.load_model('/home/ece/Music/weigh FINALfeatures.hdf5')
model.summary()
```

```
Model: "sequential"
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense (Dense)                (None, 256)               10496     
_________________________________________________________________
activation (Activation)      (None, 256)               0         
_________________________________________________________________
dropout (Dropout)            (None, 256)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 256)               65792     
_________________________________________________________________
activation_1 (Activation)    (None, 256)               0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 256)               65792     
_________________________________________________________________
activation_2 (Activation)    (None, 256)               0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_3 (Dense)              (None, 256)               65792     
_________________________________________________________________
activation_3 (Activation)    (None, 256)               0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 1542      
_________________________________________________________________
activation_4 (Activation)    (None, 6)                 0         
=================================================================
Total params: 209,414
Trainable params: 209,414
Non-trainable params: 0
_________________________________________________________________
```

In [12]:

```
loss,acc=model.evaluate(x_test,y_test)
print("accuray:{:5.2f}%".format(100*acc))
```

```
2/2 [==============================] - 0s 4ms/step - loss: 0.2101 - accuracy: 0.9792
accuray:97.92%
```

In [13]:

```
import librosa
import numpy as np

def extract_feature(file_name):

    try:
        audio_data, sample_rate = librosa.load(file_name, res_type='kaiser_fast')
        mfccs = librosa.feature.mfcc(y=audio_data, sr=sample_rate, n_mfcc=40)
        mfccsscaled = np.mean(mfccs.T,axis=0)

    except Exception as e:
        print("Error encountered while parsing file: ", file)
        return None, None

    return np.array([mfccsscaled])
```

In [14]:

```
category=[]
predicted_proba = []
pro=[]
y=[]
def print_prediction(file_name):
    prediction_feature = extract_feature(file_name)

    predicted_vector = np.argmax(model.predict(prediction_feature), axis=-1)
    print( predicted_vector)
    predicted_class = le.inverse_transform(predicted_vector)
    print("The predicted class is:", predicted_class[0], '\n')
    predicted_proba_vector = model.predict(prediction_feature)
    predicted_proba = predicted_proba_vector[0]
    for i in range(len(predicted_proba)):
        pro.append(format(predicted_proba[i], '.5f'))
        category = le.inverse_transform(np.array([i]))
        print(category[0], "\t\t : ", format(predicted_proba[i], '.5f') )
        y.append(pro[i])
```

In [15]:

```
filename = '/home/ece/Videos/bike sp/5harley5.wav'
print_prediction(filename)
```

```
[2]
The predicted class is: HARLEY

APACHE           :  0.00000
ENFIELD                  :  0.00000
HARLEY           :  1.00000
KAWASAKI                 :  0.00000
KTM              :  0.00000
PULSAR           :  0.00000
```

In [16]:

```
category=[]
predicted_proba = []
pro=[]
def print_prediction(file_name):
        prediction_feature = extract_feature(file_name)
        predicted_vector = np.argmax(model.predict(prediction_feature), axis=-1)
        predicted_class = le.inverse_transform(predicted_vector)
        print("The predicted class is:", predicted_class[0], '\n')
        global cur_labelb
        cur_labelb=ttk.Label(canvas,text = str("The predicted class is:")+str( predicted_class[0]), style='sp.TLabel')
        cur_labelb.place(x=790,y=900)
        #label1 = tk.Label(root, text = str("The predicted class is:")+str( predicted_class[0])).place(x=600,y=900)
        #canvas.create_text(400,200, text = str("The predicted class is:")+str( predicted_class[0]),font =("Helvetica",15),fill="white")
        predicted_proba_vector = model.predict(prediction_feature)
        predicted_proba = predicted_proba_vector[0]

        for i in range(len(predicted_proba)):
            pro.append(format(predicted_proba[i], '.5f'))
            category = le.inverse_transform(np.array([i]))
            print(category[0], "\t\t : ", format(predicted_proba[i], '.5f') )
            cur_label='Label'+str(i)
            cur_label=ttk.Label(canvas,text = str(category[0])+" = "+str(format(predicted_proba[i], '.3f')),style='green/black.TLabel')
            cur_label.grid(column=100,row=i+300,sticky='')
            #label = tkinter.Label(canvas, text = str(category[0])+"\t\t : "+str(format(predicted_proba[i], '.5f'))).grid(x=i+100,y=i+200)
```

In [17]:

```
def record():
    FORMAT = pyaudio.paInt16
    CHANNELS = 2
    RATE = 44100
    CHUNK = 1024
    RECORD_SECONDS = 5
    WAVE_OUTPUT_FILENAME = "file2.wav"

    audio = pyaudio.PyAudio()
    # start Recording
    stream = audio.open(format=FORMAT, channels=CHANNELS,
                rate=RATE, input=True,
                frames_per_buffer=CHUNK)
    print ("recording...")
    frames = []
    for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
        data = stream.read(CHUNK)
        frames.append(data)
    print ("finished recording")
    global cur_labela
    cur_labela=ttk.Label(canvas,text = "finished recording",style='green/black.TLabel')
    cur_labela.place(x=790,y=850)

    # stop Recording
    stream.stop_stream()
    stream.close()
    audio.terminate()

    waveFile = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
    waveFile.setnchannels(CHANNELS)
    waveFile.setsampwidth(audio.get_sample_size(FORMAT))
    waveFile.setframerate(RATE)
    waveFile.writeframes(b''.join(frames))
    waveFile.close()
def prd():
    global filepath
    filepath = '/home/ece/Music/file2.wav'
    print_prediction(filepath)

def restart():
    cur_labela=ttk.Label(canvas,text = "                        ",style='green/black.TLabel')
    cur_labela.place(x=790,y=850)
    cur_labelb=ttk.Label(canvas,text = "                                     ", style='sp.TLabel')
    cur_labelb.place(x=790,y=900)


def play():
    pygame.mixer.init()
    pygame.mixer.music.load(filepath)
    pygame.mixer.music.play(loops=0)

def play1():
    pygame.mixer.init()
    pygame.mixer.music.load(filepath1)
    pygame.mixer.music.play(loops=0)


def openfile():
    global filepath1
    filepath1=filedialog.askopenfilename()
    print(filepath1)
    print_prediction(filepath1)
```

In [18]:

```
import pygame
import os
import pyaudio
import wave
from tkinter import ttk
```

```
pygame 2.0.1 (SDL 2.0.14, Python 3.8.5)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

In [20]:

```
from tkinter.ttk import *
from tkinter import *
import tkinter
from tkinter import filedialog
canvas = Tk()
canvas.geometry('1920x1080')
```

```
canvas = Canvas(width=1920, height=1080, bg='black')

canvas.pack(expand=YES, fill=BOTH)

gif1 = PhotoImage(file='fn.png')

canvas.create_image(0, 0, image=gif1, anchor=NW)
style = ttk.Style()
ttk.Style().configure('green/black.TLabel',font=('Helvetica', 20,'bold'), foreground='#e9d10a', background='#020613')
ttk.Style().configure('green/black.TButton',font=('Helvetica', 20,'bold'), foreground='#1164e8', background='#010101')
ttk.Style().configure('sp.TLabel',font=('ariel', 25,'bold'), foreground='#e25d12', background='#020613')
ttk.Style().configure('spW.TLabel',font=('ariel', 35,'bold'), foreground='#e5ce66', background='#020613')

cur_labeld=ttk.Label(canvas,text = 'ACOUSTIC TARGET CLASSIFICATION ', style='spW.TLabel')
cur_labeld.place(x=500,y=30)

cur_labeld=ttk.Label(canvas,text = 'BASED ON MACHINE LEARNING', style='spW.TLabel')
cur_labeld.place(x=550,y=80)

button_rec = ttk.Button(canvas, text='START' ,style='green/black.TButton',command=record)
button_rec.place(x=500,y=750)

button_rec = ttk.Button(canvas, text='CLASSIFY RECORDED FILE',style='green/black.TButton',command=prd)
button_rec.place(x=1280,y=750)

button_rec = ttk.Button(canvas, text='CLASSIFY FROM FILE',style='green/black.TButton',command=openfile)
button_rec.place(x=1280,y=800)

button_rec = ttk.Button(canvas, text='RESET',style='green/black.TButton',command=restart)
button_rec.place(x=500,y=800)

play_button = ttk.Button(canvas, text='PLAY RECORDED FILE',style='green/black.TButton', command=play)
play_button.place(x=700,y=950)

play_button = ttk.Button(canvas, text='PLAY AUDIO FROM FILE',style='green/black.TButton', command=play1)
play_button.place(x=980,y=950)
mainloop()
```

In [ ]: