In [1]:

```
import pandas as pd
import joblib
import numpy as np
```

Loading data

In [2]:

```
X = joblib.load("Inputs")
X.shape
```

Out[2]:

```
(8732, 40)
```

In [3]:

```
Y = joblib.load("Outputs")
Y.shape
```

Out[3]:

```
(8732,)
```

Encoding data

In [4]:

```
from sklearn.preprocessing import LabelEncoder
LE = LabelEncoder()
```

In [5]:

```
Y = LE.fit_transform(Y)
Y
```

Out[5]:

```
array([3, 2, 2, ..., 1, 1, 1], dtype=int64)
```

Splitting data

In [6]:

```
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_stat
X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=0.1, r
```

In [7]:

```
X_train.shape
```

Out[7]:

```
(6286, 40)
```

Model Building

In [8]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout,Activation
from tensorflow.keras.callbacks import EarlyStopping
```

In [9]:

```
model=Sequential()

###first layer
model.add(Dense(256,input_shape=(40,)))
model.add(Activation('relu'))
model.add(Dropout(0.5))

###second layer
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.4))

###third layer
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.3))

###final layer
model.add(Dense(len(set(Y))))
model.add(Activation('softmax'))
```

In [10]:

```
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, 128)               32896

 activation_1 (Activation)   (None, 128)               0

 dropout_1 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 64)                8256

 activation_2 (Activation)   (None, 64)                0

 dropout_2 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 10)                650

 activation_3 (Activation)   (None, 10)                0

=================================================================
Total params: 52,298
Trainable params: 52,298
Non-trainable params: 0
_________________________________________________________________
```

In [11]:

```
# compiling the model
model.compile(loss='sparse_categorical_crossentropy',metrics=['accuracy'],optimizer=
```

In [12]:

```
# EarlyStopping to save compute resources
ES = EarlyStopping(monitor='val_loss',mode='min',patience=8)
```

In [13]:

```
# fitting the model
model.fit(X_train, Y_train, batch_size = 32, epochs = 200, validation_data=(X_val,Y_
```

```
accuracy: 0.2911 - val_loss: 1.7194 - val_accuracy: 0.4192
Epoch 8/200
197/197 [==============================] - 0s 2ms/step - loss: 1.7980 -
accuracy: 0.3474 - val_loss: 1.6405 - val_accuracy: 0.4320
Epoch 9/200
197/197 [==============================] - 0s 2ms/step - loss: 1.7376 -
accuracy: 0.3785 - val_loss: 1.5639 - val_accuracy: 0.4793
Epoch 10/200
197/197 [==============================] - 0s 2ms/step - loss: 1.6811 -
accuracy: 0.3968 - val_loss: 1.4992 - val_accuracy: 0.4778
Epoch 11/200
197/197 [==============================] - 0s 2ms/step - loss: 1.6234 -
accuracy: 0.4236 - val_loss: 1.4518 - val_accuracy: 0.5107
Epoch 12/200
197/197 [==============================] - 0s 2ms/step - loss: 1.5752 -
accuracy: 0.4480 - val_loss: 1.4154 - val_accuracy: 0.5422
Epoch 13/200
197/197 [==============================] - 0s 2ms/step - loss: 1.5342 -
accuracy: 0.4605 - val_loss: 1.3458 - val_accuracy: 0.5608
Epoch 14/200
```

Evaluation

In [14]:

```
test_accuracy=model.evaluate(X_test,Y_test,verbose=0)
print(test_accuracy[1])
```

```
0.8277046084403992
```

In [15]:

```
Y_pred = model.predict(X_test)
Y_pred_argmax = np.argmax(Y_pred, axis=1)

```

```
55/55 [==============================] - 0s 1ms/step
```

In [16]:

```
from sklearn.metrics import classification_report
print(classification_report(Y_test,Y_pred_argmax))
```

```
              precision    recall  f1-score   support

           0       0.77      0.93      0.84       203
           1       0.96      0.82      0.88        89
           2       0.59      0.80      0.67       176
           3       0.93      0.69      0.79       241
           4       0.93      0.83      0.88       206
           5       0.94      0.88      0.91       227
           6       0.79      0.56      0.66        82
           7       0.88      0.97      0.92       189
           8       0.91      0.92      0.91       168
           9       0.70      0.75      0.72       166

    accuracy                           0.83      1747
   macro avg       0.84      0.81      0.82      1747
weighted avg       0.84      0.83      0.83      1747
```

Dumping Model

In [17]:

```
joblib.dump(model,"model_85acc")
```

...

Testing

In [18]:

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

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

    return mfccsscaled
```

In [19]:

```
def get_my_predictions(filename):
    prediction_feature=get_features(filename)
    prediction_feature=prediction_feature.reshape(1,-1)
    op = model.predict(prediction_feature)
    return LE.inverse_transform([np.argmax(op)])
```

In [20]:

```
pd.Series(LE.classes_)
```

Out[20]:

```
0     air_conditioner
1            car_horn
2    children_playing
3            dog_bark
4            drilling
5       engine_idling
6            gun_shot
7          jackhammer
8               siren
9        street_music
dtype: object
```

In [29]:

```
# Dog Bark
print(get_my_predictions(r"C:\Users\Admin\Downloads\dog-barking-70772.mp3"))
print(get_my_predictions(r"C:\Users\Admin\Downloads\barking-156375.mp3"))
```

```
1/1 [==============================] - 0s 16ms/step
['dog_bark']
1/1 [==============================] - 0s 16ms/step
['children_playing']
```

In [30]:

```
# Jackhammer
print(get_my_predictions(r"C:\Users\Admin\Downloads\construction_site-19522.mp3"))
print(get_my_predictions(r"C:\Users\Admin\Downloads\jackhammer-01-62270.mp3"))
```

```
1/1 [==============================] - 0s 16ms/step
['jackhammer']
1/1 [==============================] - 0s 17ms/step
['street_music']
```

In [31]:

```
# carhorn
print(get_my_predictions(r"C:\Users\Admin\Downloads\car-horn-6408.mp3"))
print(get_my_predictions(r"C:\Users\Admin\Downloads\car-horn-beep-beep-two-beeps-honl
```

```
1/1 [==============================] - 0s 17ms/step
['children_playing']
1/1 [==============================] - 0s 17ms/step
['car_horn']
```

In [ ]:

```
1
```