Springboard Data Science Career Track Capstone Project

Classifying Urban sounds *using* Deep Learning

Milestone Report 1

Github Link:
https://github.com/xwu0223/Urban-Sound-Classification-using-Deep-Learning/blob/master/Data_Exploration_visualization_preprocessing.ipynb

Xingkai Wu

Sept 27 2019

Problem Statement:

The objective of this project will be to use Deep Learning techniques to classify urban sounds.

When given an audio sample in a wav format of a few seconds duration, we want to be able to determine if it contains one of the target urban sounds with a corresponding likelihood score. Conversely, if none of the target sounds were detected, we will be presented with an unknown score.

To achieve this, we plan on using different neural network architectures such as Multilayer Per- ceptrons (MLPs) and Convolutional Neural Networks (CNNs).

Analysis:

Data Exploration and Visualisation

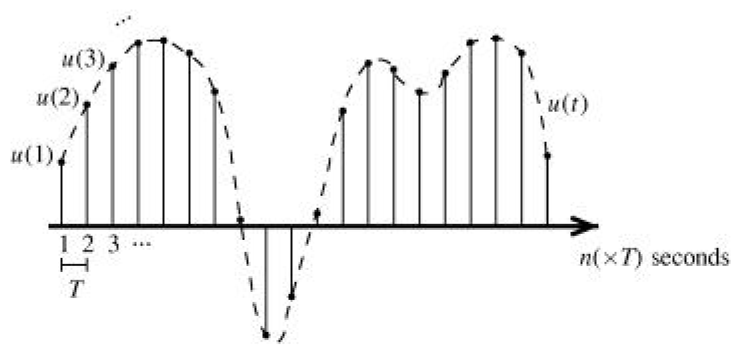
The Dataset:

This dataset contains 8732 labeled sound excerpts (<=4s) of urban sounds from 10 classes:

air_conditioner	car_horn	Children_playing	dog_bark	drilling
engine_idling	gun_shot	jackhammer	Siren	street_music

Wav file is stored in a way such that the sampling rate is 44.1kHz(44100 times per second).
Each sample is the amplitude of the wave at 1/44100 second=22.67us of interval, where the bit depth determines how detailed the sample will be also known as the dynamic range of the signal. For example, 16bit range means a sample's has 2^16=65536 amplitude values with equal amplitude interval.

The wav sound signal is discrete time signal, the following image is an example of discrete time signal

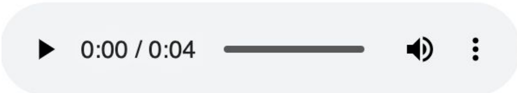


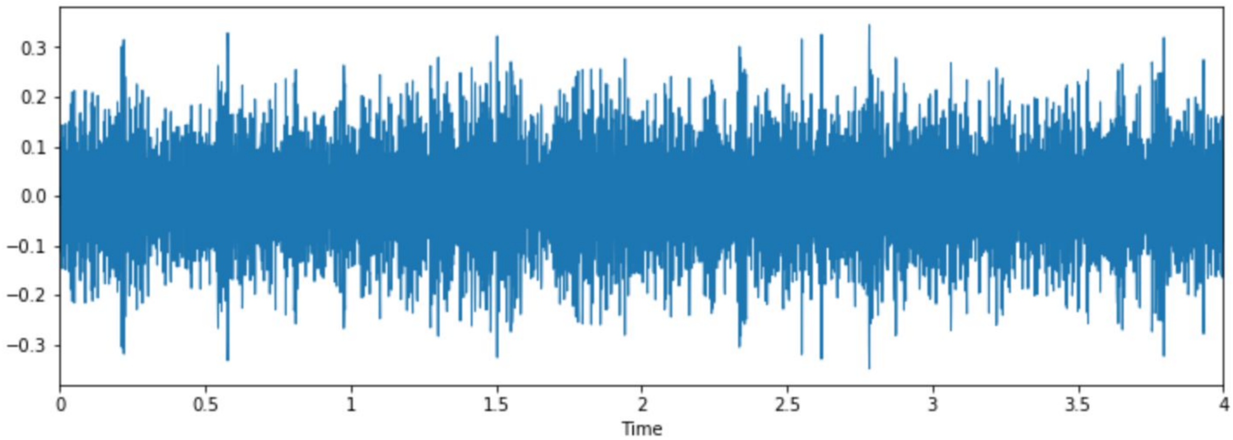
Therefore, the sound signal can be analyzed in one- dimensional array or vector of amplitude value.

The following libraries are used to explore the audio signal in python:
<u>Ipython.display.Audio</u>: This package allows us to play audio directly in the Jupyter Notebook.
<u>Librosa</u>: This package allows us to load audio to notebook as a numpy array for analysis and manipulation.

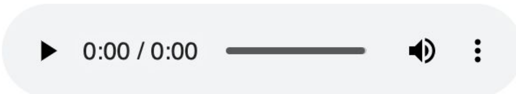
The following images are the 10 different classes in the file directory.

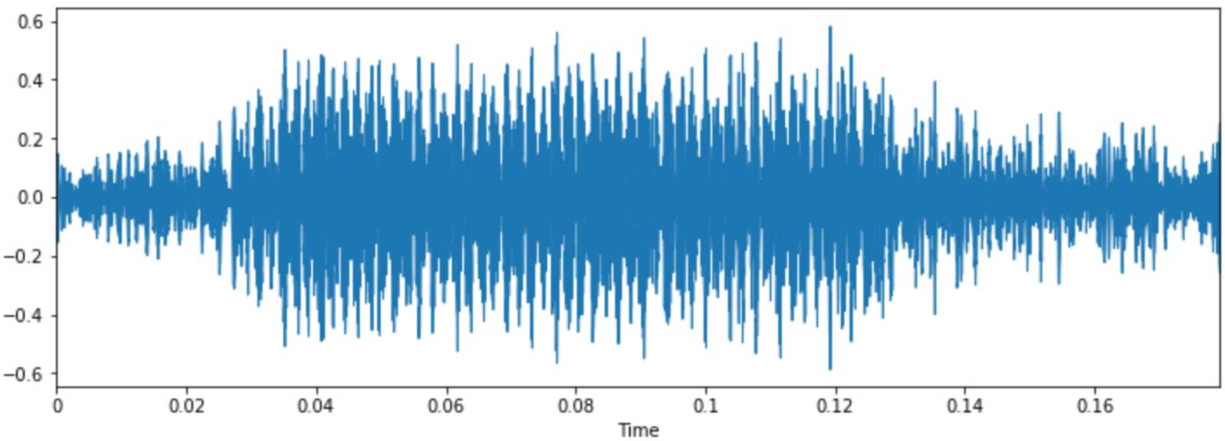
```
# Class: Air Conditioner

filename = '../Capstone_Project_2/train/Train/24.wav'
plt.figure(figsize=(12,4))
data,sample_rate = librosa.load(filename)
_ = librosa.display.waveplot(data,sr=sample_rate)
ipd.Audio(filename)
```



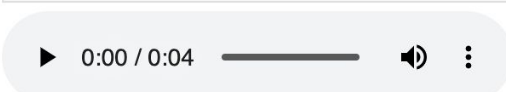


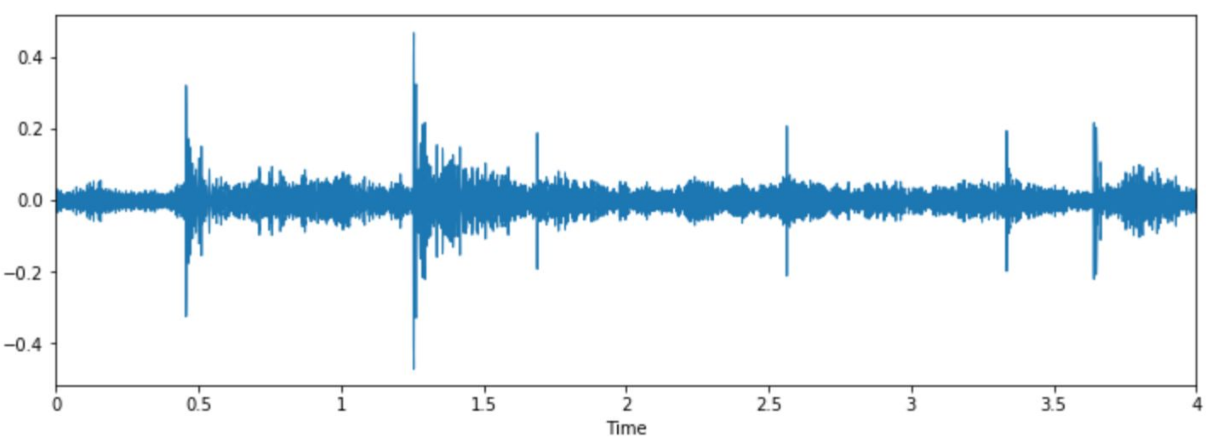
```python
# Class: Car Horn

filename = '../Capstone_Project_2/train/Train/48.wav'
plt.figure(figsize=(12,4))
data,sample_rate = librosa.load(filename)
_ = librosa.display.waveplot(data,sr=sample_rate)
ipd.Audio(filename)
```



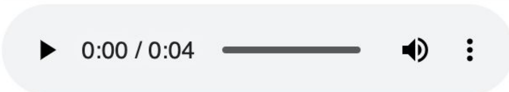


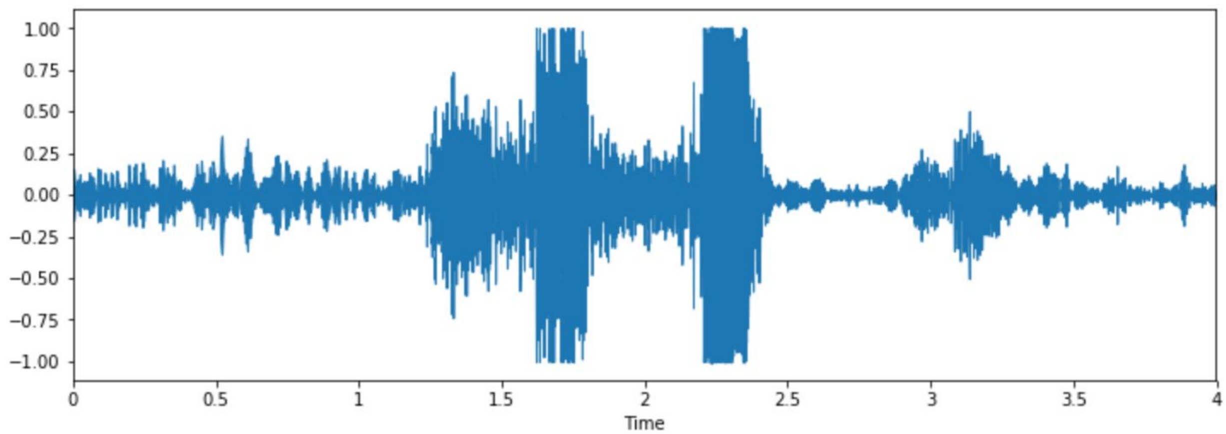

```python
# Class:Children Playing

filename = '../Capstone_Project_2/train/Train/6.wav'
plt.figure(figsize=(12,4))
data,sample_rate = librosa.load(filename)
_ = librosa.display.waveplot(data,sr=sample_rate)
ipd.Audio(filename)
```



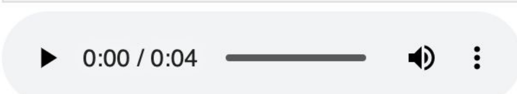


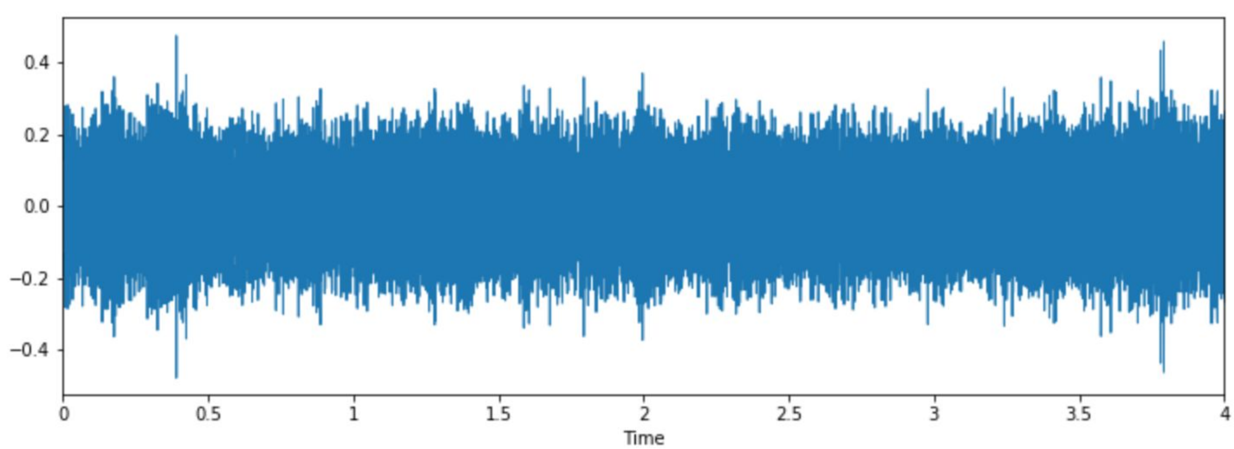
```
# Class: Dog Bark

filename = '../Capstone_Project_2/train/Train/4.wav'
plt.figure(figsize=(12,4))
data,sample_rate = librosa.load(filename)
_ = librosa.display.waveplot(data,sr=sample_rate)
ipd.Audio(filename)
```



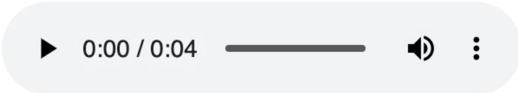


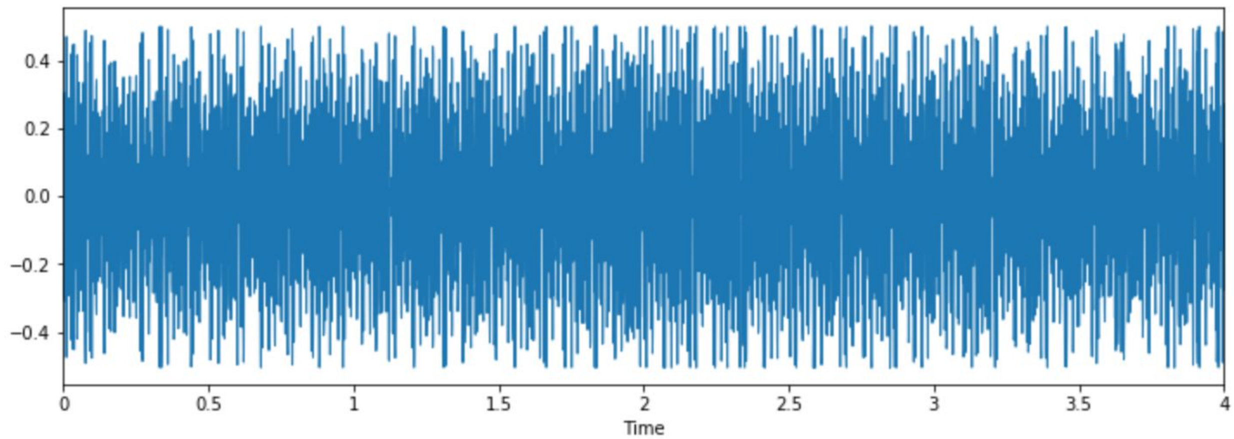

```
# Class: Drilling

filename = '../Capstone_Project_2/train/Train/11.wav'
plt.figure(figsize=(12,4))
data,sample_rate = librosa.load(filename)
_ = librosa.display.waveplot(data,sr=sample_rate)
ipd.Audio(filename)
```



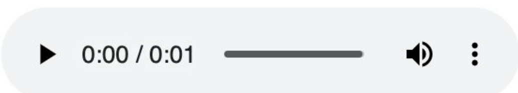


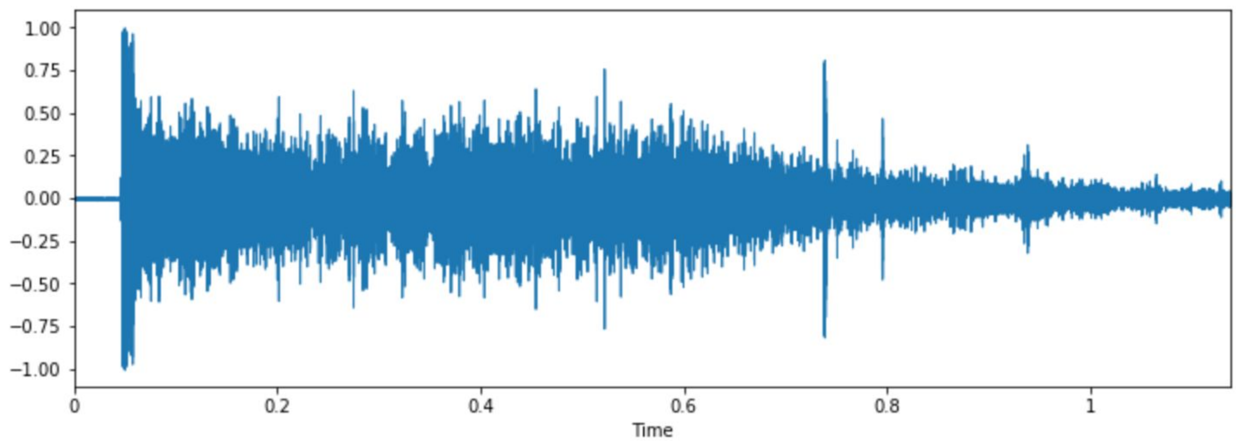
```python
# Class: Engine Idling

filename = '../Capstone_Project_2/train/Train/17.wav'
plt.figure(figsize=(12,4))
data,sample_rate = librosa.load(filename)
_ = librosa.display.waveplot(data,sr=sample_rate)
ipd.Audio(filename)
```



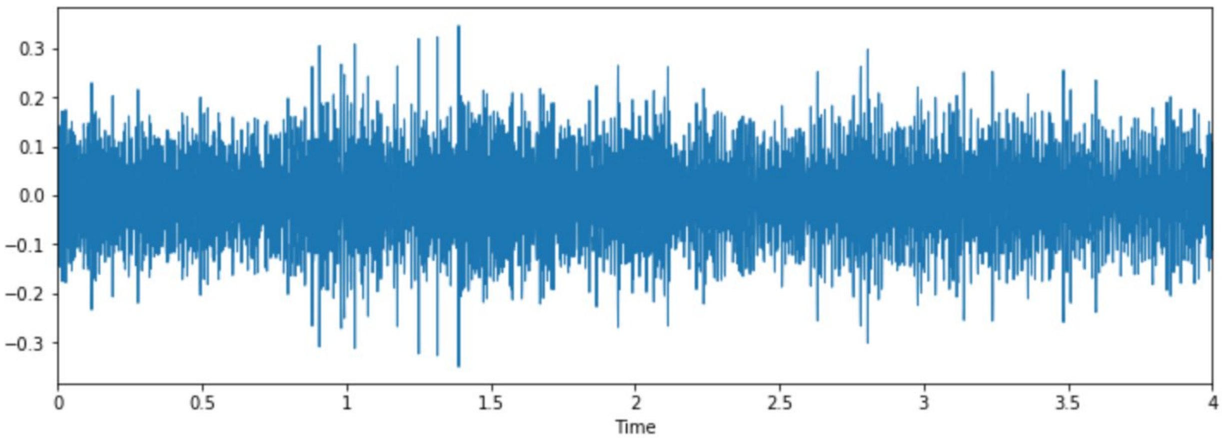



```python
# Class: Gun Shot

filename = '../Capstone_Project_2/train/Train/12.wav'
plt.figure(figsize=(12,4))
data,sample_rate = librosa.load(filename)
_ = librosa.display.waveplot(data,sr=sample_rate)
ipd.Audio(filename)
```

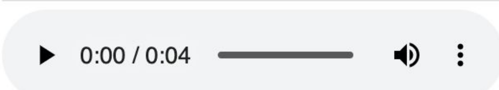


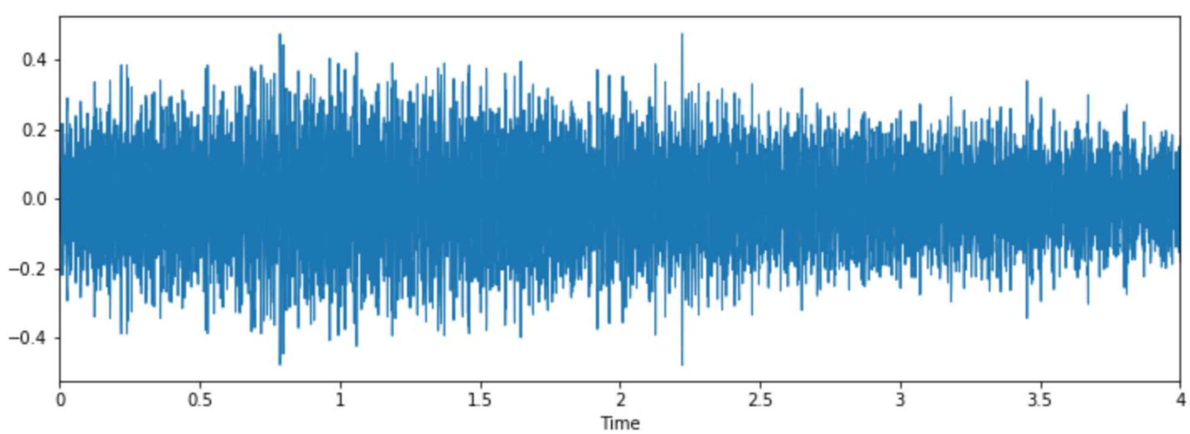


```python
# Class: Jackhammer

filename = '../Capstone_Project_2/train/Train/33.wav'
plt.figure(figsize=(12,4))
data,sample_rate = librosa.load(filename)
_ = librosa.display.waveplot(data,sr=sample_rate)
ipd.Audio(filename)
```



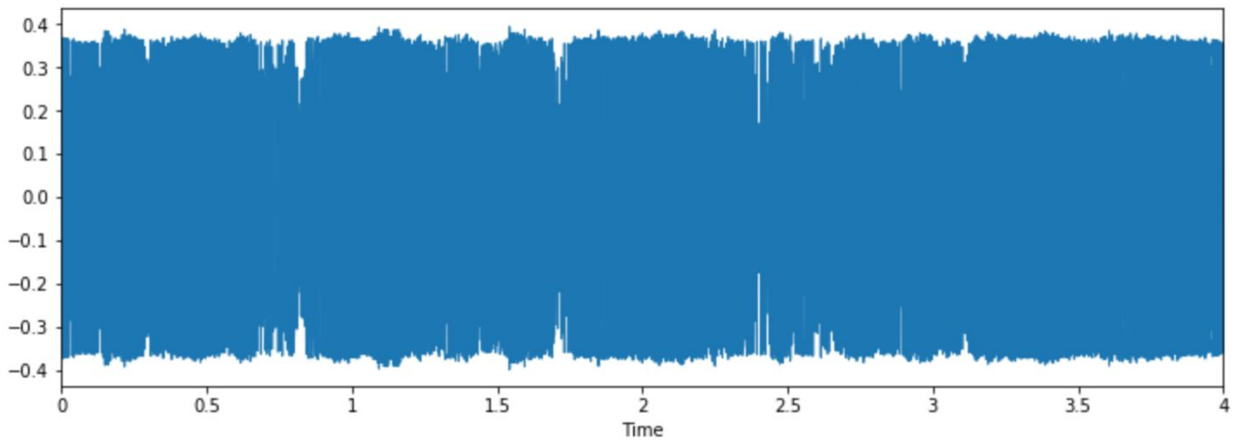

```python
# Class: Siren

filename = '../Capstone_Project_2/train/Train/0.wav'
plt.figure(figsize=(12,4))
data,sample_rate = librosa.load(filename)
_ = librosa.display.waveplot(data,sr=sample_rate)
ipd.Audio(filename)
```





```python
# Class: Street Music

filename = '../Capstone_Project_2/train/Train/1.wav'
plt.figure(figsize=(12,4))
data,sample_rate = librosa.load(filename)
_ = librosa.display.waveplot(data,sr=sample_rate)
ipd.Audio(filename)
```

0:00 / 0:04



Visually, it is very trick to tell the differences between the classes. Especially the air conditioner, drilling, engine idling and jackhammer are similar in shape.

Likewise, the dog bark and gunshot are similar as they both have have peaks. Children playing and street music are also very close in shape.

The Metadata Set:

```
import pandas as pd
metadata = pd.read_csv('../Capstone_Project_2/train/train.csv')
metadata.head()
```

	ID	Class
0	0	siren
1	1	street_music
2	2	drilling
3	3	siren
4	4	dog_bark

The following table shows the count for each class in the dataset. The dataset is **<u>not</u>** balanced.

```
print(metadata['Class'].value_counts())
```

```
jackhammer          668
engine_idling       624
siren               607
drilling            600
air_conditioner     600
children_playing    600
dog_bark            600
street_music        600
car_horn            306
gun_shot            230
Name: Class, dtype: int64
```

It's obvious to tell that the ID number is the prefix of the audio file name, let's merge the the audio file name and the metadata set.

```
df_merge.head()
```

	File_Name	ID	Class
0	4666.wav	4666	engine_idling
1	2217.wav	2217	engine_idling
2	7409.wav	7409	drilling
3	1078.wav	1078	car_horn
4	6717.wav	6717	children_playing

Audio file properties:

All the wav file has following 3 properties

1. Sample rate
2. Number of channels
3. Bit depth

```
from pathlib import Path
import soundfile as sf
data_folder = Path("../Capstone_Project_2/train/Train/")
sample_rate = []
num_channel = []
bit_depth = []
for name in df_merge['File_Name']:
    ob = sf.SoundFile(data_folder / name)
    sample_rate.append(ob.samplerate)
    num_channel.append(ob.channels)
    bit_depth.append(ob.subtype)
```

```
sound_file = pd.DataFrame(
    {'File_Name': file,
     'sample_rate': sample_rate,
     'num_channel': num_channel,
     'bit_depth':bit_depth
    })
```

```
combined = df_merge.merge(sound_file,on='File_Name')
```

```
combined.head()
```

	File_Name	ID	Class	sample_rate	num_channel	bit_depth
0	4666.wav	4666	engine_idling	44100	2	PCM_16
1	2217.wav	2217	engine_idling	48000	1	PCM_24
2	7409.wav	7409	drilling	44100	2	PCM_16
3	1078.wav	1078	car_horn	44100	2	PCM_16
4	6717.wav	6717	children_playing	44100	2	PCM_16

```
print(combined['bit_depth'].value_counts(normalize=True))
```

```
PCM_16        0.667341
PCM_24        0.302668
FLOAT         0.021711
PCM_U8        0.007912
MS_ADPCM      0.000184
IMA_ADPCM     0.000184
Name: bit_depth, dtype: float64
```

```
print(combined['sample_rate'].value_counts(normalize=True))
```

```
44100    0.598896
48000    0.305980
96000    0.066053
24000    0.009752
16000    0.007176
22050    0.006624
11025    0.003680
8000     0.001104
32000    0.000736
Name: sample_rate, dtype: float64
```

Algorithms and Techniques:
The proposed solution to this problem is to apply Deep Learning techniques that have proved to be highly successful in the field of image classification.

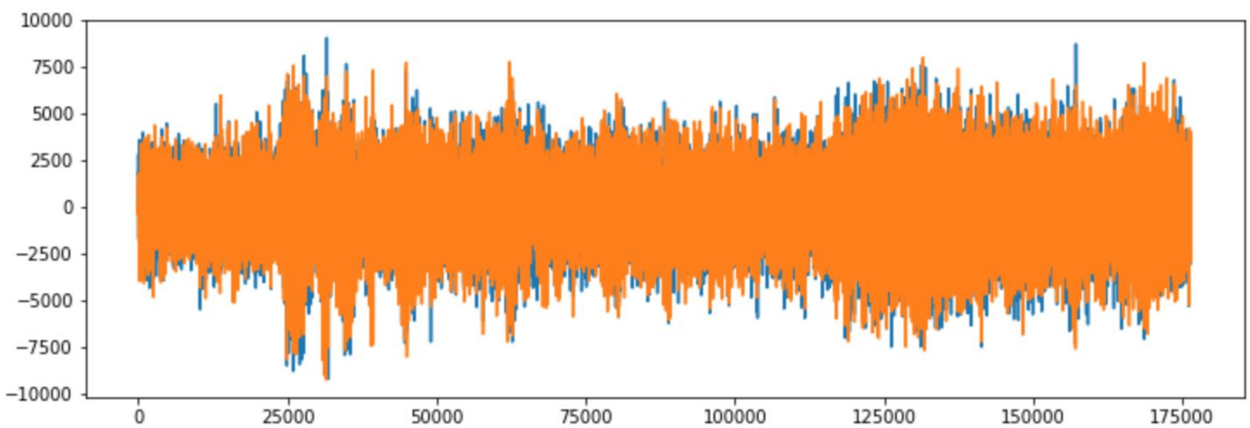
First we will extract Mel-Frequency Cepstral Coefficients (MFCC) from the audio samples on a per-frame basis with a window size of a few milliseconds. The MFCC summarises the frequency distribution across the window size, so it is possible to analyse both the frequency and time characteristics of the sound. These audio representations will allow us to identify features for classification.

The next step will be to train a Deep Neural Network with these data sets and make predictions. We will begin by using a simple neural network architectures, such as Multi-Layer Perceptron before experimenting with more complex architectures such as Convolutional Neural Networks.

```python
import librosa
from scipy.io import wavfile as wav
import numpy as np

filename = '../Capstone_Project_2/train/Train/4666.wav'

librosa_audio, librosa_sample_rate = librosa.load(filename)
scipy_sample_rate, scipy_audio = wav.read(filename)

print('Original sample rate:', scipy_sample_rate)
print('Librosa sample rate:', librosa_sample_rate)
```

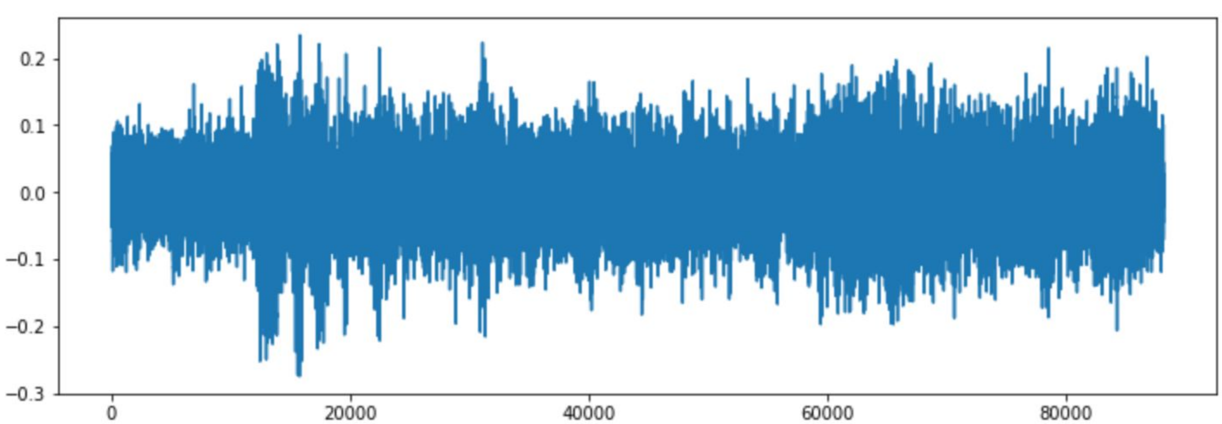
```
Original sample rate: 44100
Librosa sample rate: 22050
```

```
The original audio clip 4666.wav has 2 channel, the first plot illustrate the 2 seperate channels, the second plot
illustrated the merged channel using librosa.
Librosa converts the audio signal to mono, meaning the number of channels will always be 1.
```




```python
plt.figure(figsize=(12, 4))
plt.plot(librosa_audio)
```

```
[<matplotlib.lines.Line2D at 0x1c3c2cce48>]
```



We can't tell the difference much in term of shape of the signal of 2 channels and combined into 1.

MFCC(Mel- frequency Cepstrum Coefficient) is a function to extract the coefficients of Mel- Frequency Cepstrum which is a representation of a short term power spectrum of a sound based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

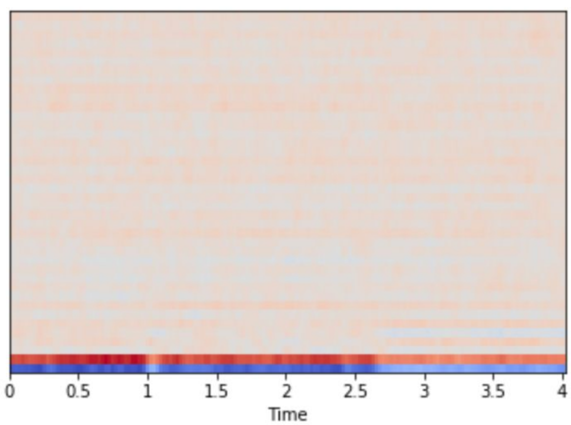
```
mfccs = librosa.feature.mfcc(y=librosa_audio, sr=librosa_sample_rate, n_mfcc=40)
print(mfccs.shape)
```

```
(40, 173)
```

```
This shows the merged channel clip of 4666.wav calculated a series of 40 MFCCs over 173 frames.
```

```
import librosa.display
librosa.display.specshow(mfccs, sr=librosa_sample_rate, x_axis='time')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c3c23bac8>
```




```
This plot illustrates the sectrum of the merged channel clip 4666.wav
```

Let us extract the 40 feature coefficients of each audio signal:

```
# Set the path to the full UrbanSound dataset
fulldatasetpath = '../Capstone_Project_2/train/Train/'

metadata = pd.read_csv('../Capstone_Project_2/train_detail.csv')

features = []

# Iterate through each sound file and extract the features
for index, row in metadata.iterrows():

    file_name = os.path.join(os.path.abspath(fulldatasetpath)+'/',str(row["File_Name"]))

    class_label = row["Class"]
    data = extract_features(file_name)

    features.append([data, class_label])

# Convert into a Panda dataframe
featuresdf = pd.DataFrame(features, columns=['feature','class_label'])

print('Finished feature extraction from ', len(featuresdf), ' files')
```

```
Finished feature extraction from  5435  files
```

```
featuresdf.head()
```

	feature	class_label
0	[-148.29195, 125.17131, -18.48167, 13.752467, ...	engine_idling
1	[-216.45773, 175.77246, 0.244275, 65.77827, 6....	engine_idling
2	[-193.82822, 105.66298, -39.473175, 32.525784,...	drilling
3	[-143.99443, 111.079796, -33.822388, 46.725525...	car_horn
4	[-261.05377, 119.62229, -55.061405, 31.013195,...	children_playing

Now, encode the categorical class label into one-hot numerical code by using sklearn preprocessing LabelEncoder so it can be understood when analysing.

```
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical

# 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))
```