DEEP LEARNING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/DEEP-LEARNING/)

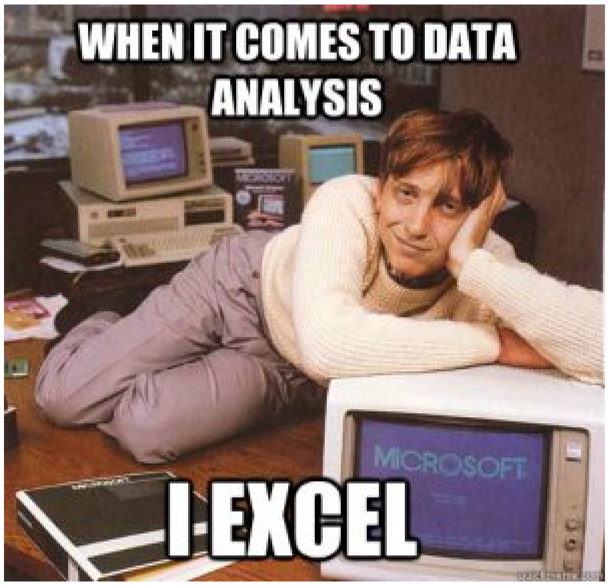
PYTHON (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/PYTHON-2/)

Getting Started with Audio Data Analysis using Deep Learning (with case study)

FAIZAN SHAIKH (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/AUTHOR/JALFAIZY/), AUGUST 24, 2017

Introduction

When you get started with data science, you start simple. You go through simple projects like Loan Prediction problem (https://datahack.analyticsvidhya.com/contest/practice-problem-loan-prediction-iii/) or Big Mart Sales Prediction (https://datahack.analyticsvidhya.com/contest/practice-problem-big-mart-sales-iii/). These problems have structured data arranged neatly in a tabular format. In other words, you are spoon-fed the hardest part in data science pipeline.



The datasets in real life are much more complex.

You first have to understand it, collect it from various sources and arrange it in a format which is ready for processing. This is even more difficult when the data is in an unstructured format such as image or audio. This is so because you would have to represent image/audio data in a standard way for it to be useful for analysis.

The abundance on unstructured data

Interestingly, unstructured data represents huge under-exploited opportunity. It is closer to how we communicate and interact as humans. It also contains a lot of useful & powerful information. For example, if a person speaks; you not only get what he / she says but also what were the emotions of the person from the voice.

Also the body language of the person can show you many more features about a person, because actions speak louder than words! So in short, unstructured data is complex but processing it can reap easy rewards.

In this article, I intend to cover an overview of audio / voice processing with a case study so that you would get a hands-on introduction to solving audio processing problems.

Let's get on with it!

Table of Contents

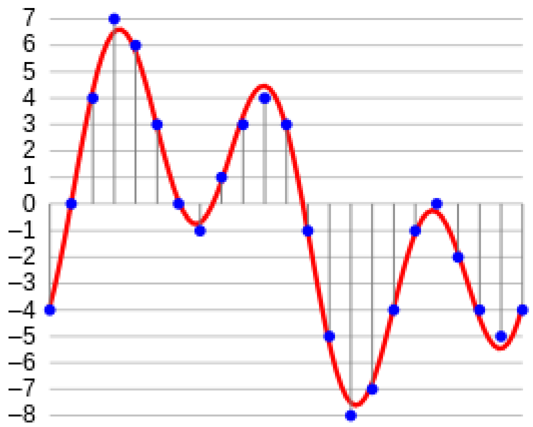
What do you mean by Audio data?

Directly or indirectly, you are always in contact with audio. Your brain is continuously processing and understanding audio data and giving you information about the environment. A simple example can be your conversations with people which you do daily. This speech is discerned by the other person to carry on the discussions. Even when you think you are in a quiet environment, you tend to catch much more subtle sounds, like the rustling of leaves or the splatter of rain. This is the extent of your connection with audio.

So can you somehow catch this audio floating all around you to do something constructive? Yes, of course! There are devices built which help you catch these sounds and represent it in computer readable format. Examples of these formats are

- **wav (Waveform Audio File) format**
- **mp3 (MPEG-1 Audio Layer 3) format**
- **WMA (Windows Media Audio) format**

If you give a thought on what an audio looks like, it is nothing but a wave like format of data, where the amplitude of audio change with respect to time. This can be pictorial represented as follows.



Applications of Audio Processing

Although we discussed that audio data can be useful for analysis. But what are the potential applications of audio processing? Here I would list a few of them

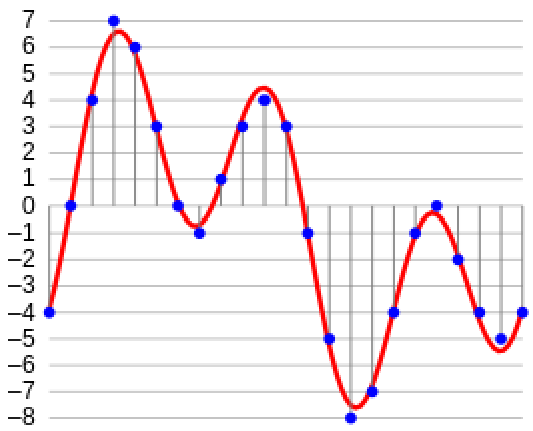
- Indexing music collections according to their audio features.
- Recommending music for radio channels
- Similarity search for audio files (aka Shazam)
- Speech processing and synthesis – generating artificial voice for conversational agents

Here's an exercise for you; can you think of an application of audio processing that can potentially help thousands of lives?

Data Handling in Audio domain

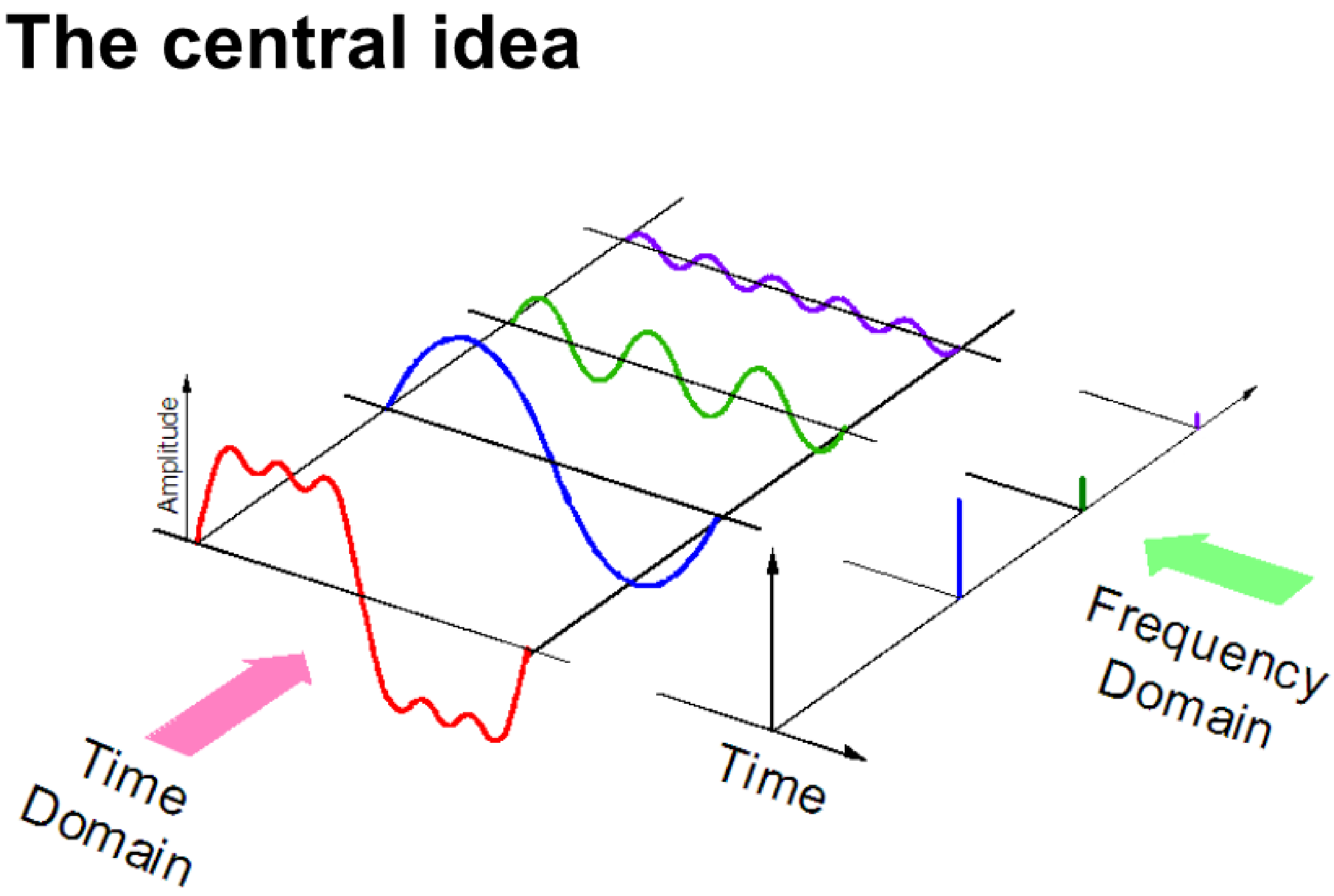
As with all unstructured data formats, audio data has a couple of preprocessing steps which have to be followed before it is presented for analysis.. We will cover this in detail in later article, here we will get an intuition on why this is done.

The first step is to actually load the data into a machine understandable format. For this, we simply take values after every specific time steps. For example; in a 2 second audio file, we extract values at half a second. This is called **sampling of audio data,** and the rate at which it is sampled is called the **sampling rate.**



Another way of representing audio data is by converting it into a different domain of data representation, namely the frequency domain. When we sample an audio data, we require much more data points to represent the whole data and also, the sampling rate should be as high as possible.

On the other hand, if we represent audio data in **frequency domain**, much less computational space is required. To get an intuition, take a look at the image below

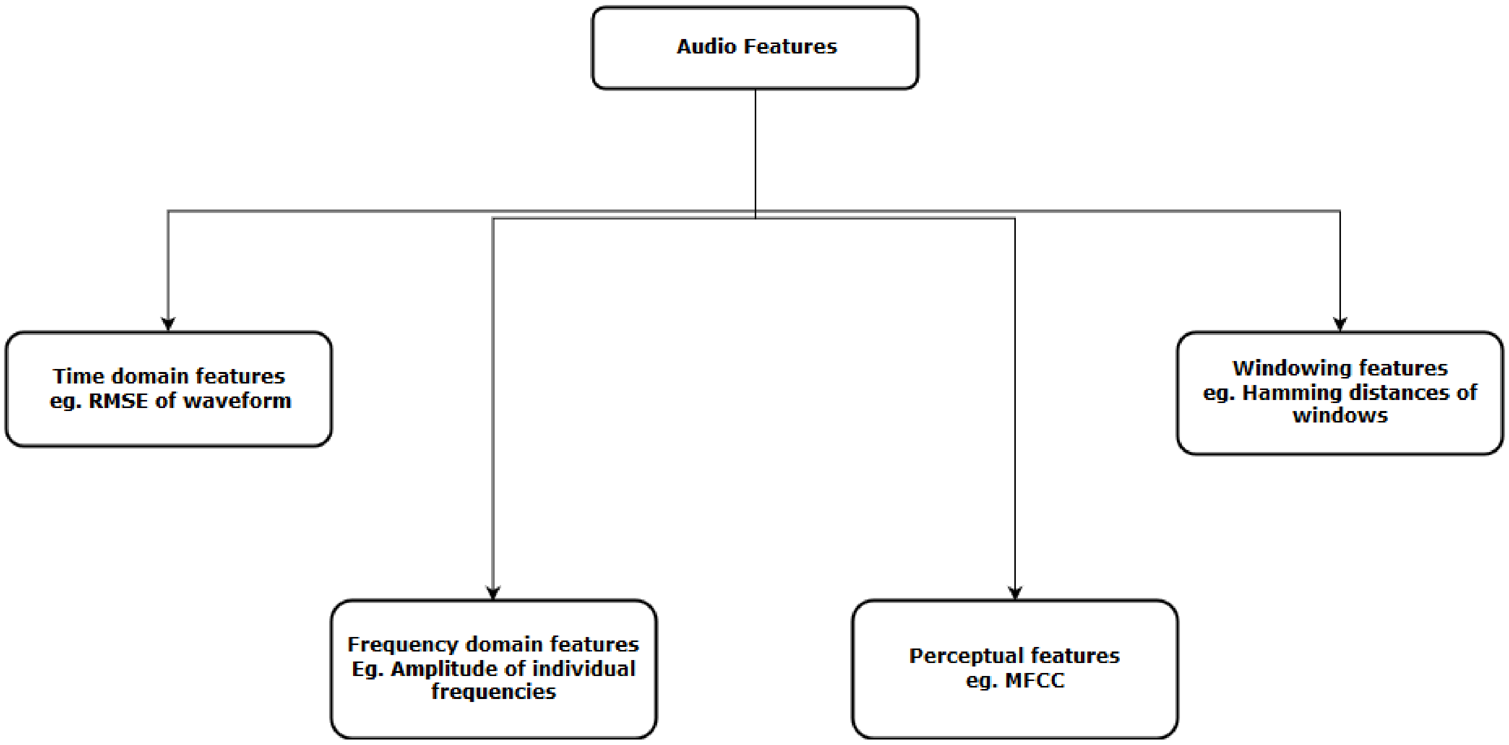


[Source (https://docs.google.com/presentation/d/1zzgNu_HbKL2iPkHS8-qhtDV20QfWt9lC3ZwPVZo8Rw0/pub?start=false&loop=false&delayms=3000#slide=id.g5a7a9806e_0_84)](https://docs.google.com/presentation/d/1zzgNu_HbKL2iPkHS8-qhtDV20QfWt9lC3ZwPVZo8Rw0/pub?start=false&loop=false&delayms=3000#slide=id.g5a7a9806e_0_84)

Here, we separate one audio signal into 3 different pure signals, which can now be represented as three unique values in frequency domain.

There are a few more ways in which audio data can be represented, for example. using [MFCs (Mel-Frequency cepstrums. (https://en.wikipedia.org/wiki/Mel-frequency_cepstrum)](https://en.wikipedia.org/wiki/Mel-frequency_cepstrum) PS: We will cover this in the later article). These are nothing but different ways to represent the data.

Now the next step is to extract features from this audio representations, so that our algorithm can work on these features and perform the task it is designed for. Here's a visual representation of the categories of audio features that can be extracted.


Audio Features
Time domain features eg. RMSE of waveform
Windowing features eg. Hamming distances of windows
Frequency domain features Eg. Amplitude of individual frequencies
Perceptual features eg. MFCC

After extracting these features, it is then sent to the machine learning model for further analysis.

Let’s solve the UrbanSound challenge!

Let us have a better practical overview in a real life project, the Urban Sound challenge (https://datahack.analyticsvidhya.com/contest/practice-problem-urban-sound-classification/). This practice problem is meant to introduce you to audio processing in the usual classification scenario.

The dataset contains 8732 sound excerpts (<=4s) of urban sounds from 10 classes, namely:

- air conditioner,
- car horn,
- children playing,
- dog bark,
- drilling,
- engine idling,
- gun shot,
- jackhammer,
- siren, and
- street music

Here's a sound excerpt from the dataset. Can you guess which class does it belong to?

00:00 00:00

To play this in the jupyter notebook, you can simply follow along with the code.

```
import IPython.display as ipd

ipd.Audio('../data/Train/2022.wav')
```

Now let us load this audio in our notebook as a numpy array. For this, we will use librosa (https://github.com/librosa/librosa) library in python. To install librosa, just type this in command line

```
pip install librosa
```

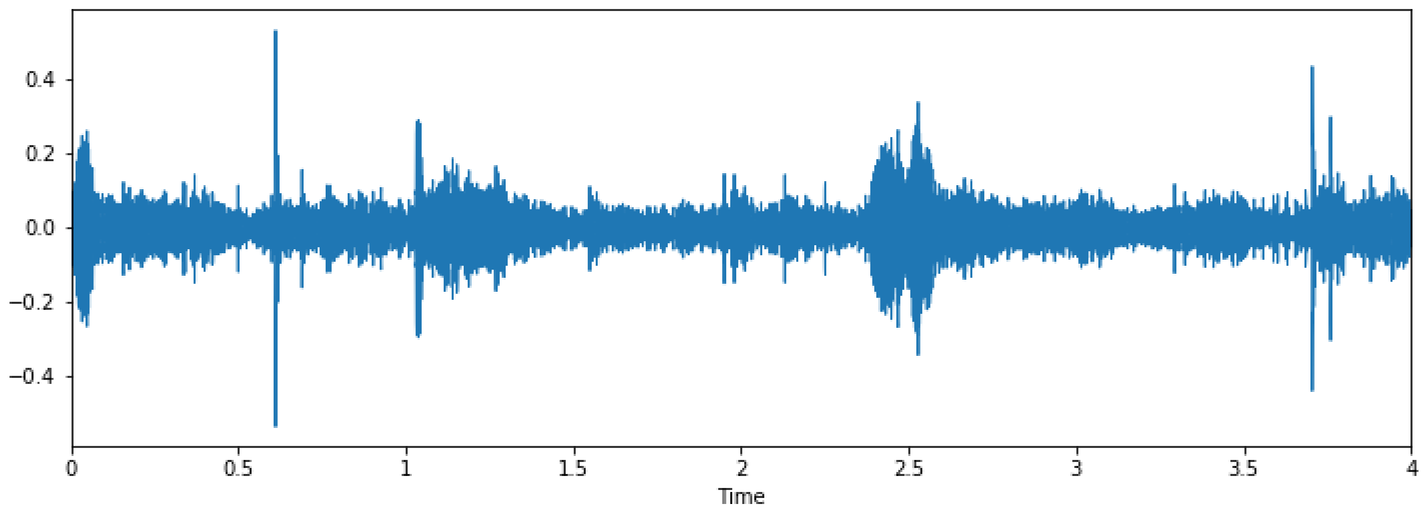
Now we can run the following code to load the data

```
data, sampling_rate = librosa.load('../data/Train/2022.wav')
```

When you load the data, it gives you two objects; a numpy array of an audio file and the corresponding sampling rate by which it was extracted. Now to represent this as a waveform (which it originally is), use the following code

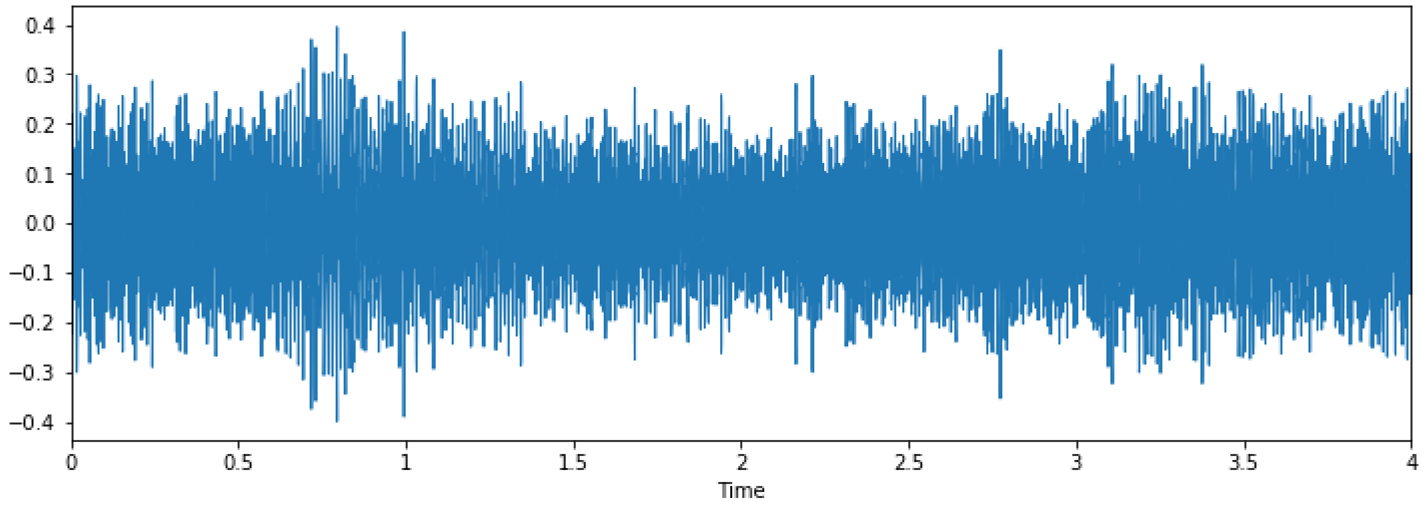
```
% pylab inline

import os

import pandas as pd

import librosa

import glob


plt.figure(figsize=(12, 4))

librosa.display.waveplot(data, sr=sampling_rate)
```

The output comes out as follows

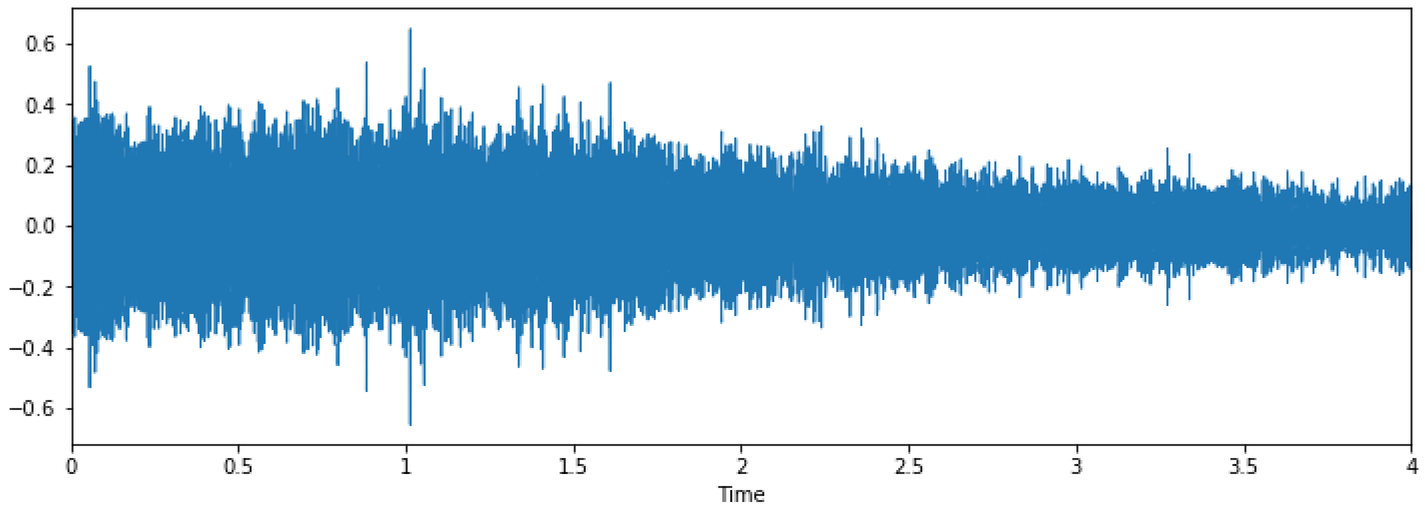


Let us now visually inspect our data and see if we can find patterns in the data

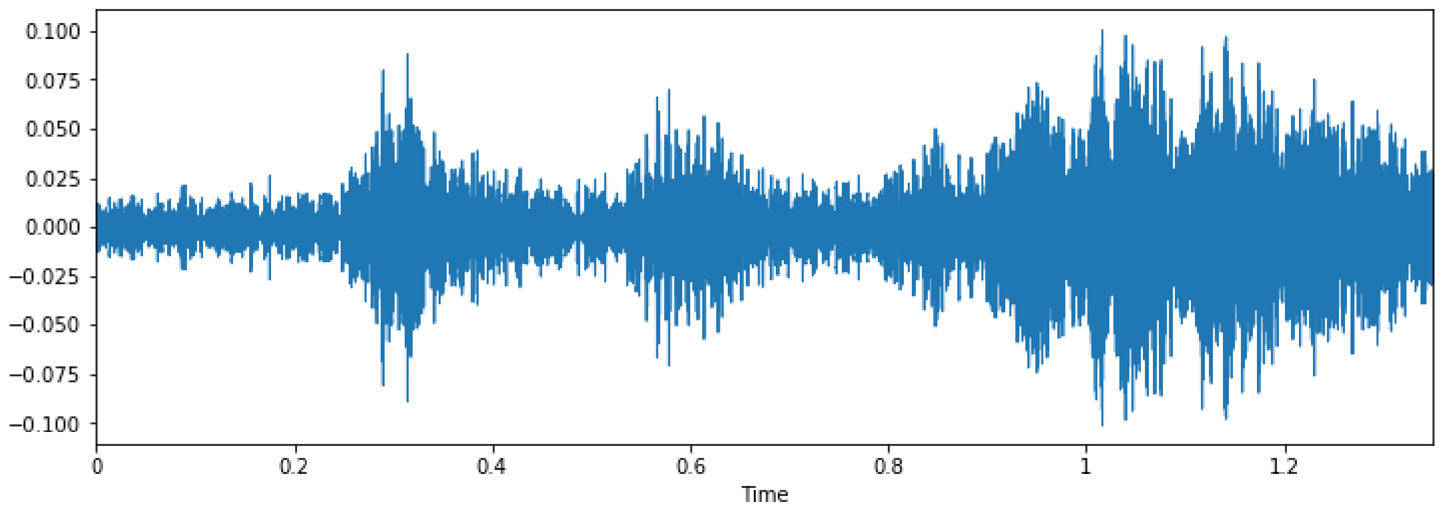
```
Class:  jackhammer
```



```
Class: drilling
```



```
Class: dog_barking
```



We can see that it may be difficult to differentiate between jackhammer and drilling, but it is still easy to discern between dog_barking and drilling. To see more such examples, you can use this code

```
i = random.choice(train.index)


audio_name = train.ID[i]
path = os.path.join(data_dir, 'Train', str(audio_name) + '.wav')


print('Class: ', train.Class[i])
x, sr = librosa.load('../data/Train/' + str(train.ID[i]) + '.wav')


plt.figure(figsize=(12, 4))
librosa.display.waveplot(x, sr=sr)
```

Intermission: Our first submission

We will do a similar approach as we did for Age detection (https://www.analyticsvidhya.com/blog/2017/06/hands-on-with-deep-learning-solution-for-age-detection-practice-problem/) problem, to see the class distributions and just predict the max occurrence of all test cases as that class.

Let us see the distributions for this problem.

```
train.Class.value_counts()
```

```
Out[10]:


jackhammer 0.122907

engine_idling 0.114811

siren 0.111684

dog_bark 0.110396

air_conditioner 0.110396

children_playing 0.110396

street_music 0.110396

drilling 0.110396

car_horn 0.056302

gun_shot 0.042318
```

We see that jackhammer class has more values than any other class. So let us create our first submission with this idea.

```
test = pd.read_csv('../data/test.csv')

test['Class'] = 'jackhammer'

test.to_csv(‘sub01.csv’, index=False)
```

This seems like a good idea as a benchmark for any challenge, but for this problem, it seems a bit unfair. This is so because the dataset is not much imbalanced.

Let’s solve the challenge! Part 2: Building better models

Now let us see how we can leverage the concepts we learned above to solve the problem. We will follow these steps to solve the problem.

Step 1: Load audio files
Step 2: Extract features from audio
Step 3: Convert the data to pass it in our deep learning model
Step 4: Run a deep learning model and get results

Below is a code of how I implemented these steps

Step 1 and 2 combined: Load audio files and extract features

```
def parser(row):
   # function to load files and extract features
   file_name = os.path.join(os.path.abspath(data_dir), 'Train', str(row.ID) + '.wav')

   # handle exception to check if there isn't a file which is corrupted
   try:
      # here kaiser_fast is a technique used for faster extraction
      X, sample_rate = librosa.load(file_name, res_type='kaiser_fast')
      # we extract mfcc feature from data
      mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T,axis=0)
   except Exception as e:
      print("Error encountered while parsing file: ", file)
      return None, None

   feature = mfccs
   label = row.Class

   return [feature, label]

temp = train.apply(parser, axis=1)
temp.columns = ['feature', 'label']
```

Step 3: Convert the data to pass it in our deep learning model

```
from sklearn.preprocessing import LabelEncoder


X = np.array(temp.feature.tolist())
y = np.array(temp.label.tolist())


lb = LabelEncoder()


y = np_utils.to_categorical(lb.fit_transform(y))
```

Step 4: Run a deep learning model and get results

```
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


num_labels = y.shape[1]

filter_size = 2


# build model

model = Sequential()


model.add(Dense(256, input_shape=(40,)))

model.add(Activation('relu'))

model.add(Dropout(0.5))


model.add(Dense(256))

model.add(Activation('relu'))

model.add(Dropout(0.5))


model.add(Dense(num_labels))

model.add(Activation('softmax'))


model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')
```

Now let us train our model

```
model.fit(X, y, batch_size=32, epochs=5, validation_data=(val_x, val_y))
```

This is the result I got on training for 5 epochs

```
Train on 5435 samples, validate on 1359 samples
Epoch 1/10
5435/5435 [==============================] - 2s - loss: 12.0145 - acc: 0.1799 - val_loss: 8.3553 - va
l_acc: 0.2958
Epoch 2/10
5435/5435 [==============================] - 0s - loss: 7.6847 - acc: 0.2925 - val_loss: 2.1265 - val
_acc: 0.5026
Epoch 3/10
5435/5435 [==============================] - 0s - loss: 2.5338 - acc: 0.3553 - val_loss: 1.7296 - val
_acc: 0.5033
Epoch 4/10
5435/5435 [==============================] - 0s - loss: 1.8101 - acc: 0.4039 - val_loss: 1.4127 - val
_acc: 0.6144
Epoch 5/10
5435/5435 [==============================] - 0s - loss: 1.5522 - acc: 0.4822 - val_loss: 1.2489 - val
_acc: 0.6637
```

Seems ok, but the score can be increased obviously. (PS: I could get an accuracy of 80% on my validation dataset). Now its your turn, can you increase on this score? If you do, let me know in the comments below!

Future steps to explore

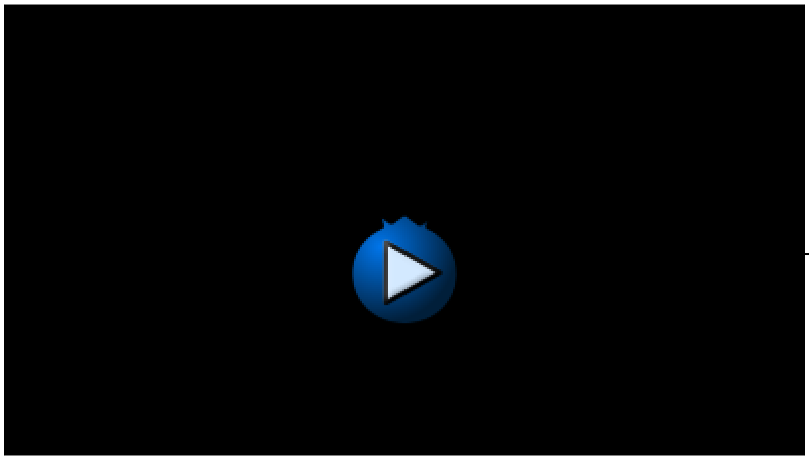
Now that we saw a simple applications, we can ideate a few more methods which can help us improve our score

1. We applied a simple neural network model to the problem. Our immediate next step should be to **understand where does the model fail and why**. By this, we want to conceptualize our understanding of the failures of algorithm so that the next time we build a model, it does not do the same mistakes
2. We can build **more efficient models** that our "better models", such as convolutional neural networks or recurrent neural networks. These models have be proven to solve such problems with greater ease.
3. We touched the concept of **data augmentation**, but we did not apply them here. You could try it to see if it works for the problem.

End Notes

In this article, I have given a brief overview of audio processing with an case study on UrbanSound challenge. I have also shown the steps you perform when dealing with audio data in python with librosa package. Giving this "shastra" in your hand, I hope you could try your own algorithms in Urban Sound challenge, or try solving your own audio problems in daily life. If you have any suggestions/ideas, do let me know in the comments below!

Learn (https://www.analyticsvidhya.com/blog), engage (http://discuss.analyticsvidhya.com/), hack (https://datahack.analyticsvidhya.com/) and get hired (https://www.analyticsvidhya.com/jobs/#/user/)!


(https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2017/08/23192318/temp.wav)
Podcast: Play in new window (https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2017/08/23192318/temp.wav) | Download (https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2017/08/23192318/temp.wav)