**Final Project Report**

CSC 535 – Deep Learning

“Classifying Sounds with a Feedforward, Convolutional and Recurrent Neural Network”

By

Zainab Murtaza

675103141

**Link to Colaboratory Notebook (Unable to upload it simultaneously to my deliverables)**

https://colab.research.google.com/drive/1RlOJ3xCeVmbY2IWGioFI8sgIdsLp3gjR?usp=sharing

# Abstract

# This project is based on the multi-class classification of urban sounds present in daily life. Noise pollution happens to serve as the epitome of quality of life problems for urban residents; 9 out of 10 adults in metropolitan cities are subjected to excessive levels of noise, beyond what the Environmental Protection Agency considers to be harmful [1]. As humans, we perceive a plethora of sounds every second and instinctively strive to develop an efficient way of classifying said sounds into their respective categories, but in this project we use deep learning models to do the task for us instead!

# In conventional models, data is fluently used in a vectorized format. For example, textual datasets use corpus words that translate to tf-idf scores and features,alternatively, anomaly detection datasets use throughput and latencies in outlier prediction- as input for classifiers. Feature extraction is rather complex when audio is involved. I have explored the various facets of a library called “Librosa” which aides feature extraction in audio files. I have then approached the task of classifying said sounds with a Feedforward Neural Network (FNN), Convolutional Neural Network (CNN) and finally, a Recurrent Neural Network (RNN), using 10-Fold Cross Validation to train and evaluate three models, resulting in averaging accuracies in FNN as 54%, the best performing CNN at 61% and RNN at 54%.

# Problem Definition and Project Goals

The purpose of this project is to delve into a novel feature-extraction methodology in audio-based datasets and use that to predict averaged class-label classification accuracy.

**Dataset:** This open-sourced annotated dataset can be found at: <https://urbansounddataset.weebly.com/> under “UrbanSound8K”. It is around 5.6 GB in size and was uploaded to my personal Google Drive, after which it was then mounted to Colaboratory for use and in-notebook extraction. This dataset is divided into 10-folds by default. It contains 8,732 sound clips with a length of 4 seconds each, from ten classes:

*air\_conditioner (1000 audio files)*

*car\_horn (429 audio files)*

*children\_playing (1000 audio files)*

*dog\_bark (1000 audio files)*

*drilling (1000 audio files)*

*enginge\_idling (1000 audio files)*

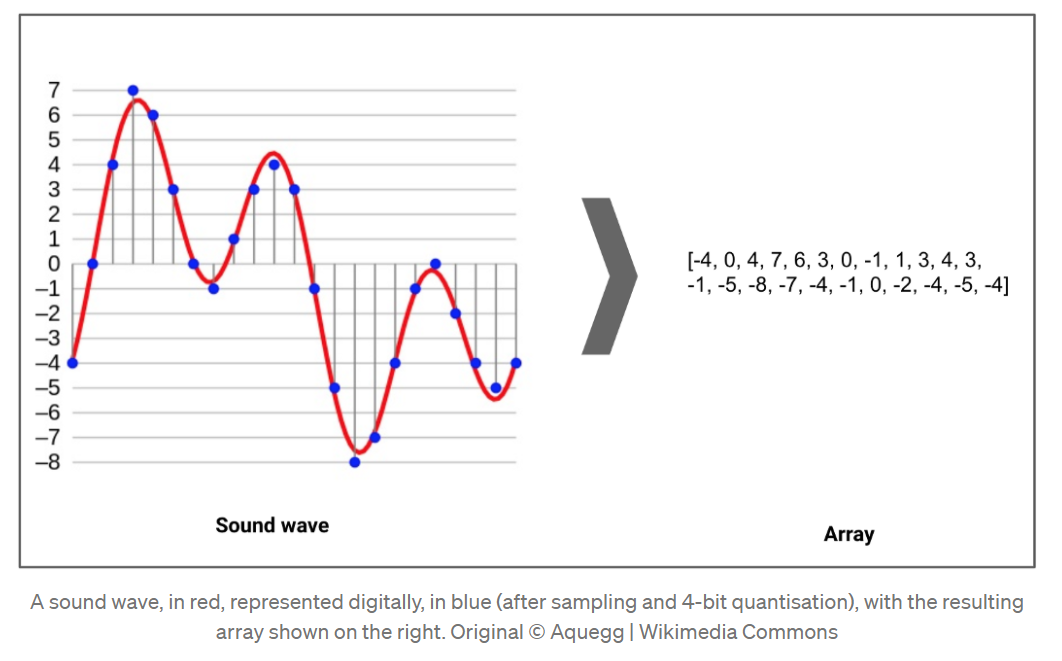
*gun\_shot (374 audio files)*

*jackhammer (1000 audio files)*

*siren (929 audio files)*

*street\_music (1000 audio files)*

In the project, I’ve pre-processed the sound files so that they can be stored as numpy arrays for ease of use. The dataset has files of format .wav and .mp3. To digitize sound waves, we sample them at discrete intervals defined commonly as sampling rates (44.1kHz for CD quality audio connoting samples are taken 44,100 times per second).



The diagram above illustrates how an audio excerpt can be interpreted from a waveform and be morphed into an array or a vector of amplitude values.

In the section “**Data Exploration and Pre-processing”,** I have cogently defined the technical features of this dataset.

# Related Work

There are a myriad works that have addressed sound classification. My work is heavily inspired by:

1. Urban Sound Classification by satyen95 on Github [2]

They used the same dataset, but used varying model methodologies in order to achieve their results. (different number of hidden layers and a different optimizer) I didn’t run their code to see what their result was (that would have taken a substantial amount of time), but the core premise of referenced project is related to mine.

1. Let’s Build An Audio Classifier by David Glavas [3]

They used the same dataset, and achieved accuracies of 70.3% across all 10-folds. This project was referred to in order to get an in-depth understanding of the intricacies behind audio feature extraction, the different techniques used to represent audio signals and computing the terms under which these features worked as input to the classifier modalities used in my own project. Working with audio was entirely new to me and went beyond the scope of this course, and this project served as a cogent, foundational interpretation of Mel-Frequency Cepstrums and Mel-frequency Cepstral Coefficients.

They also mentioned that the efficacy of sound classification lies primarily in the conversion of an audio file to an image, such as CRP, MFCC (Mel-Frequency Cepstral Coefficient), or spectrogram, proceeding with a image recognition based-convolutional deep learning network (AlexNet, Google LeNet) to classify said image [4].

1. Urban Sound Classification by Aqib Saeed [5]

Aforementioned guide, in conjunction with [3] guided many pitfalls I dealt with, and is focused around the research paper that provided the dataset [6]. With inspiration derived from exploring features, delving into visualization, laying down the bases for neural networks, I spent some time troubleshooting a substantial number of issues with this project and in the ones mentioned earlier in this report with missing data/incorrect formatting(dealing with lossy formats), deprecated functions and library functionality, along with persisting indexing input errors.

# Data Exploration and Pre-Processing

## General Information

This dataset comprises of 8,732 sound clips from ten classes, which are drawn from Urban Sound Taxonomy [6].

**air conditioner, car horn, children playing, dog bark, drilling, engine idling, gunshot, jackhammer, siren, and street music.**

By default, the dataset is divided into 10-folds, and to avoid overfitting during 10-Fold Cross Validation, attribute selection is profoundly correlation based. For each network used in this project, FNN, CNN and RNN, an average accuracy of around 10-Folds is calculated.

Audio format for these files is .wav. Coincidentally, this is the best format to work with as .wav files are inherently CD-quality music files; lossless and uncompressed. Two classes, namely “children playing” and “gunshot” were solely used for variety whilst others are rendered to be high frequency and typically appear in urban noise complaints.

As mentioned previously, for feature extraction, Mel-Frequency Cepstral Coefficients (MFCC’s) are a popular choice for sound analysis and are served to provide a competitive baseline to benchmark novel techniques.

## Visualization

This project begins with visualizing sound files via Waveplots and Spectrograms with the objective to coherently establish the difference between each type of audio.

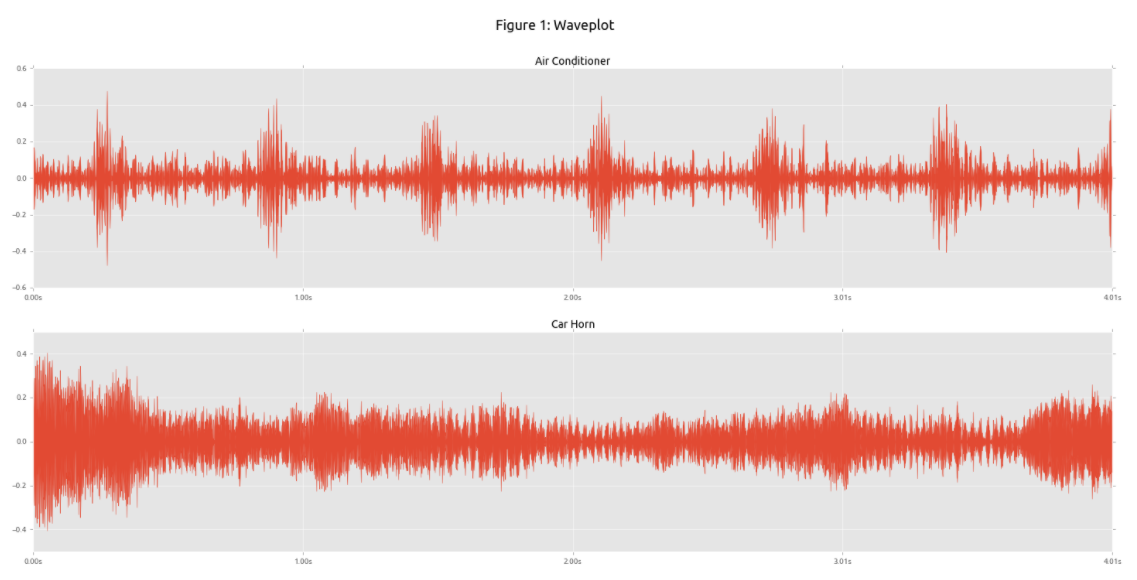
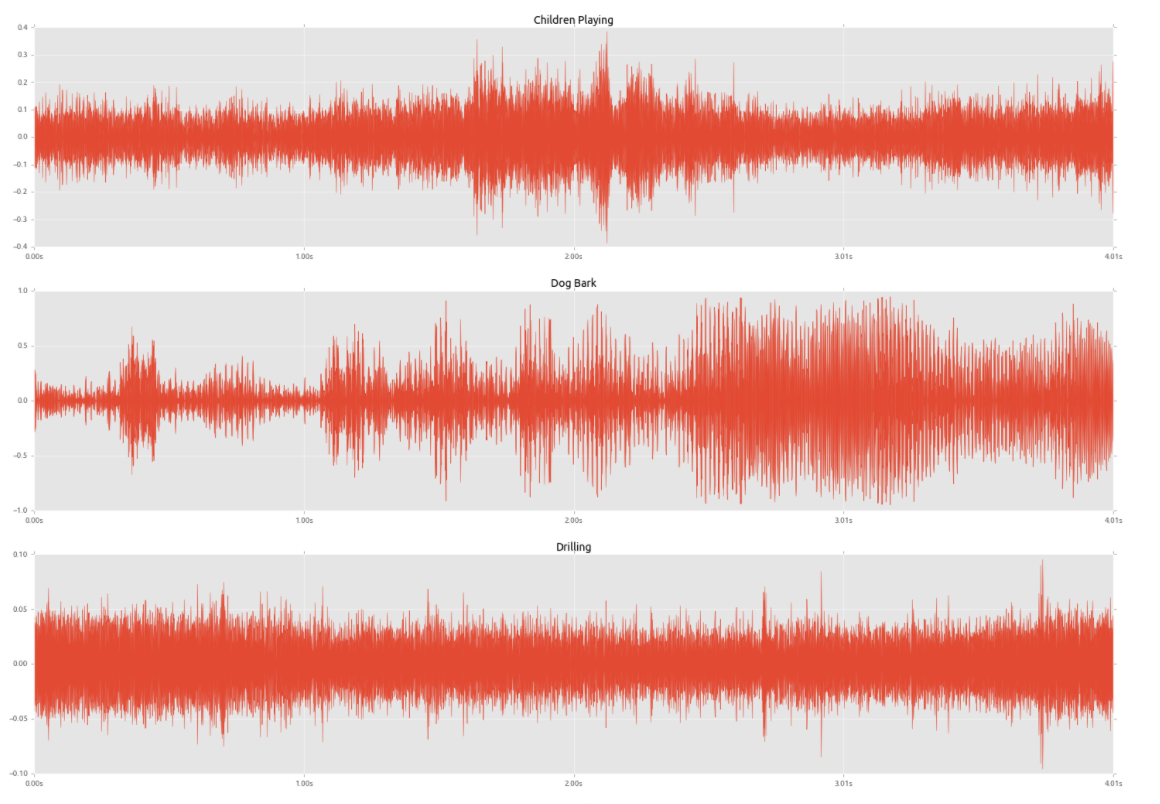
Waveplots are used to plot waveforms of amplitude vs time where the first axis is an amplitude and second axis is time.

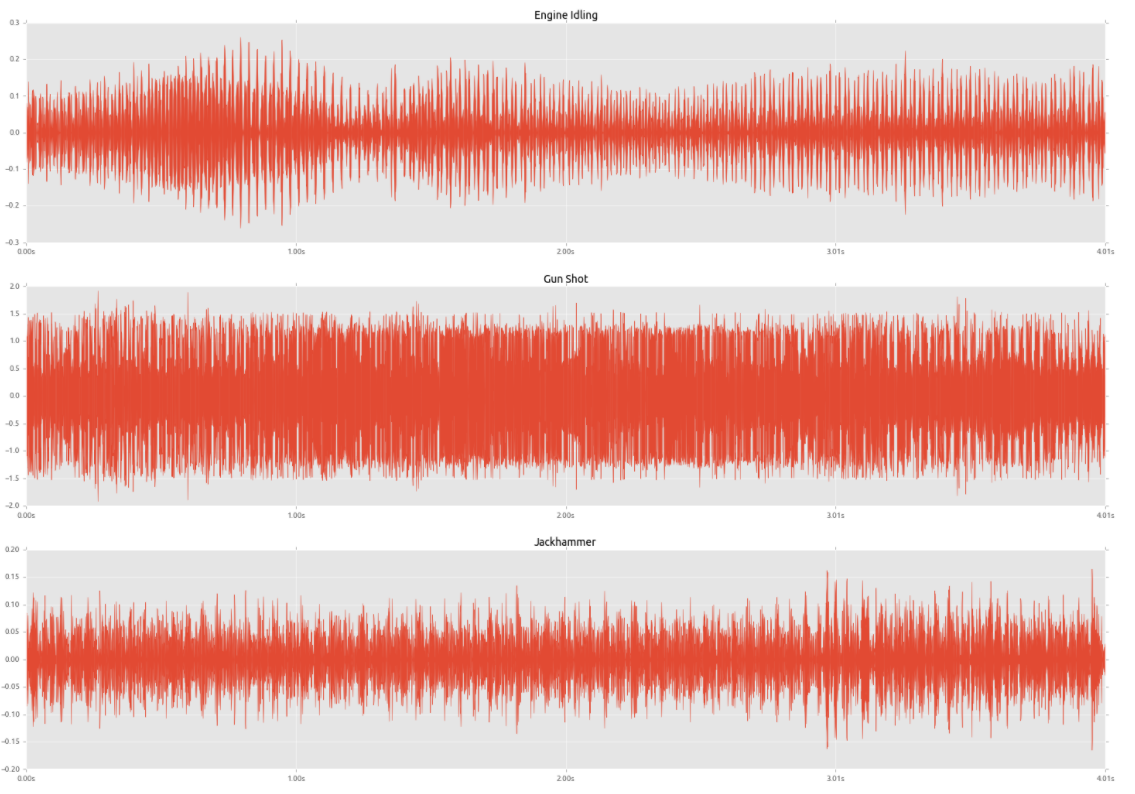
Spectrograms are useful for visualising the spectrum of frequencies of a sound and how they vary during a very short period of time.The **specgram** method from Matplotlib serves as the foundation for the computation behind spectral plotting. Librosa, in turn, aides log and wave power spectrograms.

### Features:

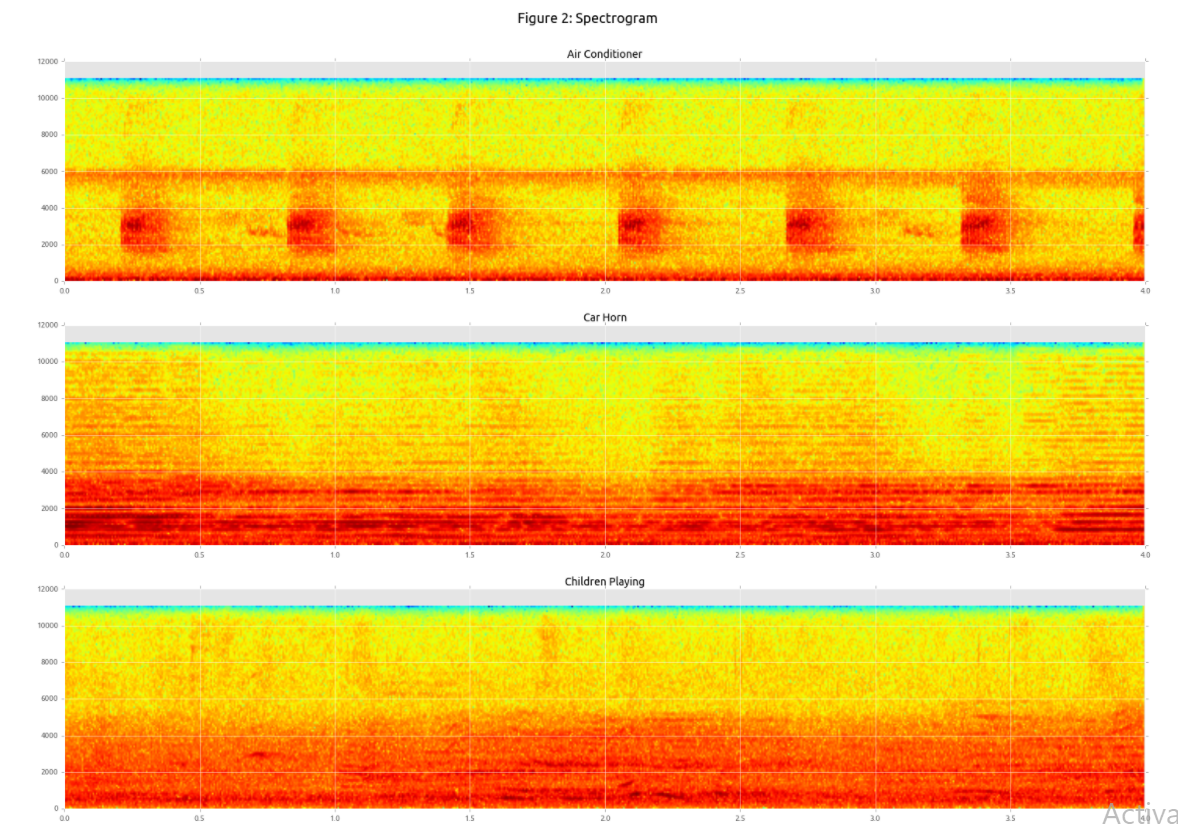
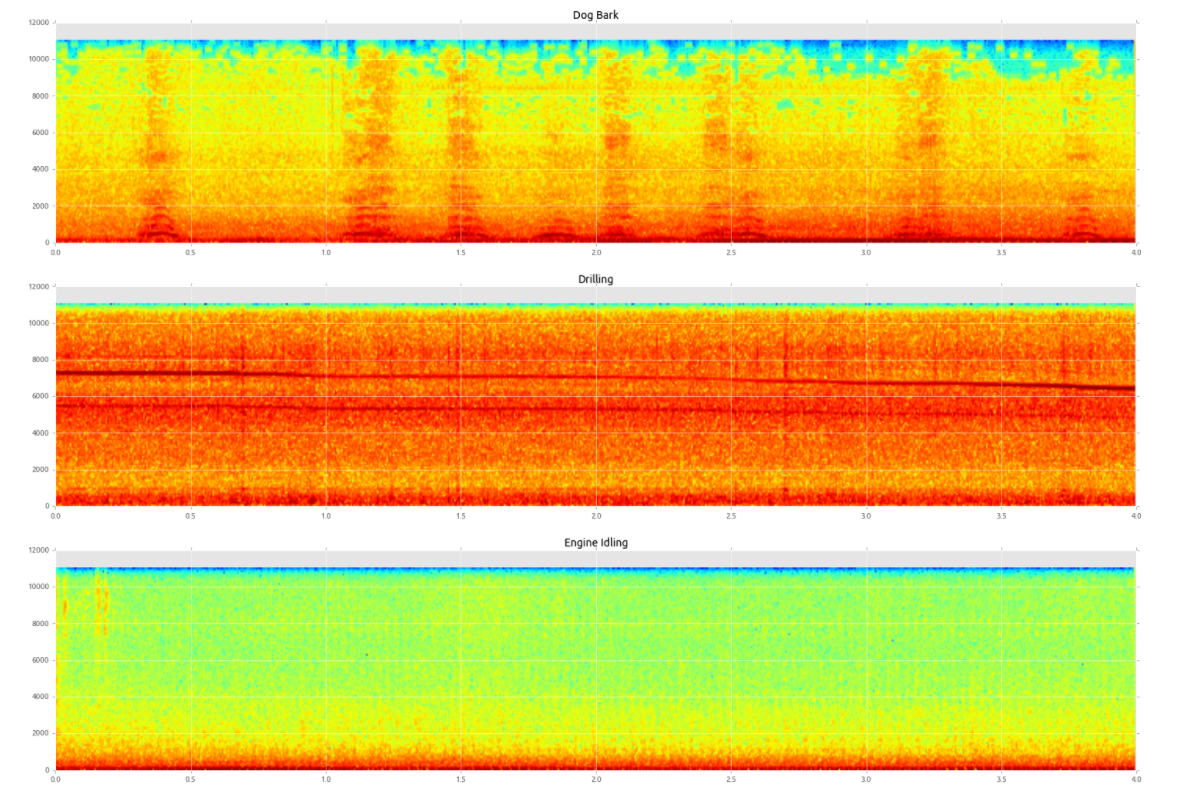
* *melspectrogram*: Mel-Scaled Power Spectrogram
* *mfcc*: Mel-Frequency Cepstral Coefficients
* *chorma-stft*: Chromagram from a Waveform or Power Spectrogram
* *spectral\_contrast*: Spectral Contrast referred to from: [7]
* *tonnetz*: Tonal Centroid features (tonnetz) referred to from: [8]

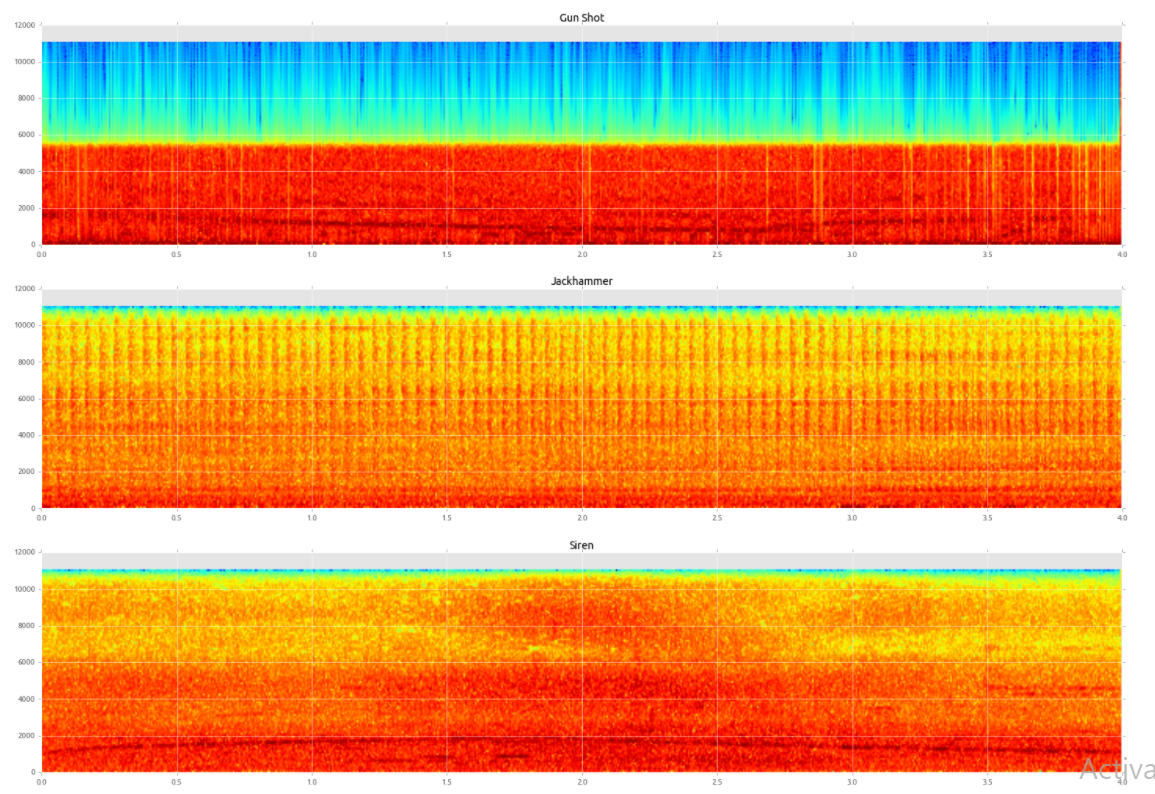
**Peruse the Waveplot and Spectrogram graphs on the next page:** (please note, the graphic may look a little cramped in the output of my project due to Colaboratory visualization restraints)

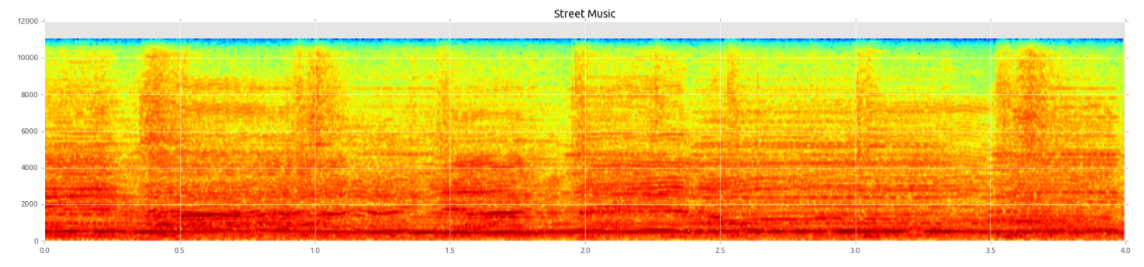












I encountered an issue with some lossy/buggy files whilst unarchiving my .tar.gz file (you may see a “trying to tune from empty frequency set” warning in the code output) and to solve this, I used try-except to void/eliminate null units (missing data) and wrapped it around the feature extraction method resulting in no warnings after.

(As far as what I have done with the data, please refer to Colaboratory notebook, this section is provided there and will adhere to provide a more cogent and coherent explanation for the code.)

As input, **parse\_audio\_files** retrieves parent directory name and the subdirectory lying within along with the file extension (.wav). This is iterated within all files in the sub-directory and invokes extract\_feature. **extract\_feature** then perceives its input as the file path, uses Librosa.load to read, return and extract features in the bullet points above.

The two methods mentioned above earlier ensure that raw sound clips are morphed into features, with a class label for individual clips that are used for classification learning. Each sound clip has it's class label in the file name, and therefore splitting strings by computing the array's second item gives us the class label. For instance:

103258-5-0-10.wav is a file, then the class label is 5.

Each file within a fold undergoes recursive iteration to extract corresponding labels and features, and the features are thus ultimately saved as numpy arrays.

# Data Analysis and Results

In order to accentuate model-accuracy percentage, this project incorporates three neural networks; FNN, CNN and RNN.

**FNN:**

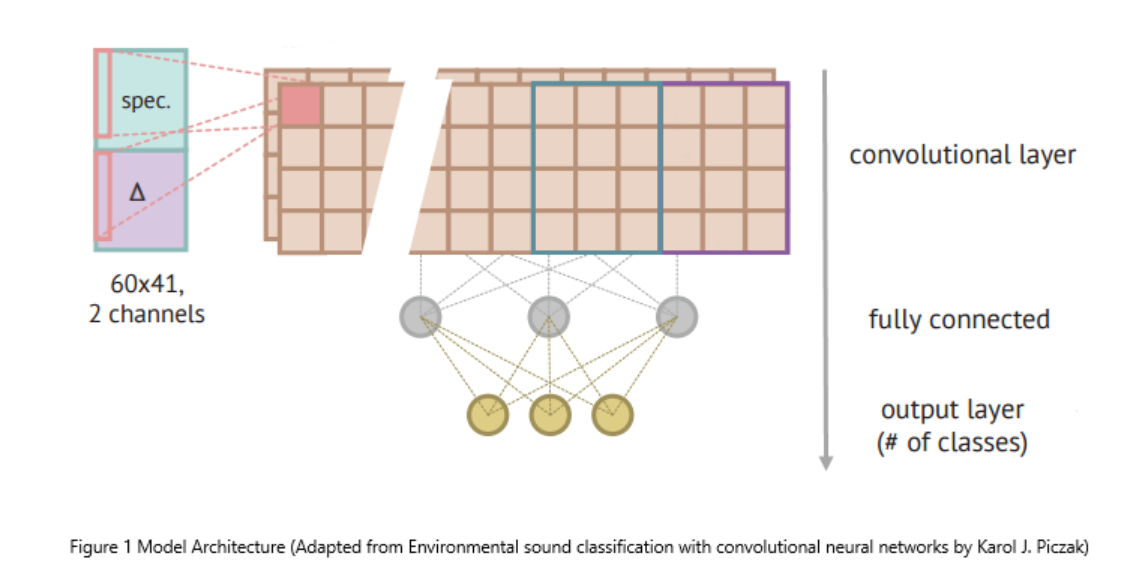
The FNN uses Keras (Tensorflow) sequential model is comprised of three layers; 256, 128 and 64 neurons in each. It also uses Softmax Activation and an Adam Optimizer.

Training and evaluation with extracted features are carried out by 10-Fold Cross Validation. Training fold data is read and amalgamated in order to train the network. Performance evaluation is then carried out on the holdout fold. This process is incremented for every fold in order to compute averaged accuracies and achieve a performance estimate.

Averaged 10-Fold Accuracy for this model is 0.54463.

**CNN:**

To preface work with CNN, a defined function computes Log-Scaled Mel-Spectrograms and its deltas from a certain sound clip. In order to ensure that the input retains a fixed size, said sound clips are then segmented into 60 rows and 41 columns. Therefore, the two channels that are fed into the CNN comprise of Log Scaled Mel-Spectrograms and its deltas. Please refer to figure below for clarity:

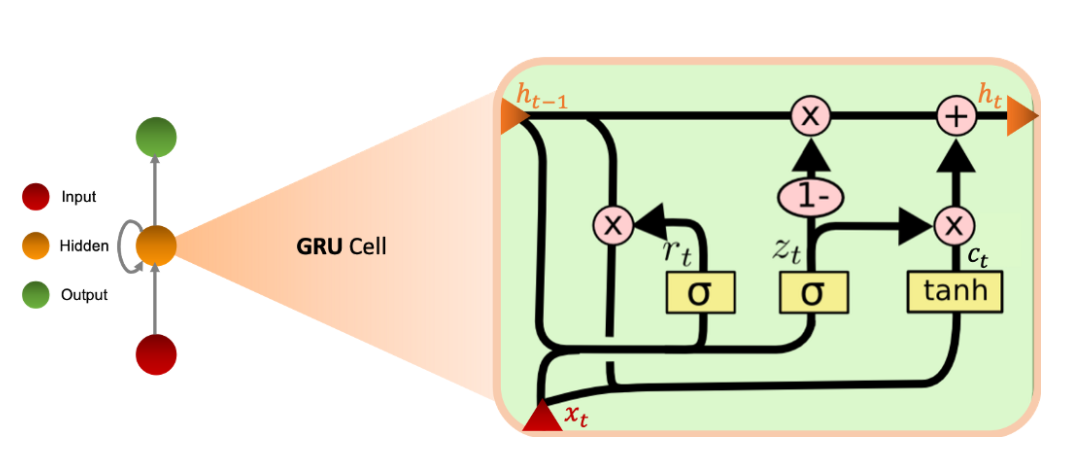


To prepare data for training the CNN, two functions, namely extract\_features and windows were used. Extract\_features iterates over all folds, calculates features and appends them to arrays using numpy.savez.

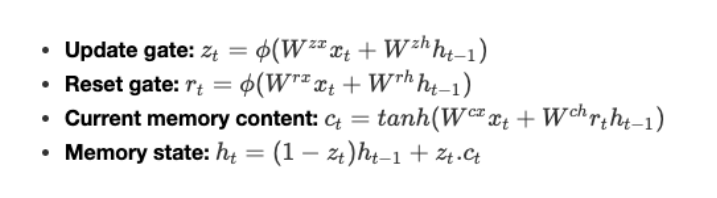
This CNN has 4 convolutional blocks, involving Batch Normalization and Max Pooling for each one. Global Pooling is added after the final convolutional dense layer for embedding extraction, which is in turn followed by a fully-connected output layer. Finally, softmax activation in a dense layer is introduced to output probabilities of audio classes. In order to train the network, Adam Optimizer is enforced to lower Sparse Categorical Crossentropy objectives. Follwing that, test sound clips undergo evaluation by averaging predictions encapsulating entire segments of audio, under the assumption that the clips are of unknown length. Finally, 10-fold accuracies are averaged to quantify network performance estimation. The CNN gives an accuracy of 0.6417 which is considered to be efficient, given the structure of our dataset and the intricacies behind the selection of features used for this type of classification.

**RNN:**

Striving to achieve better accuracy results, a Gated Recurrent Unit (GRU) was introduced to the model via a RNN. The incentive behind the use of GRU gating mechanisms is that it allows the recurrent network to save a larger amount of historical information and long-term sequence dependencies for enhanced prediction results. It consists of an update gate which encapsulates the quantity of information that is kept from the past, along with a reset gate which defines the quantity of information to forget. Please refer to the graph below for better visualization:



Equations in the GRU are as follows:



ϕ can be defined as a non-linear integer function and the parameters W are learned by the model.

This model has two hidden layers, one RNN and a dense layer, with 128 neurons. After evaluating and training the model with 10-Fold Cross Validation, this model achieved an accuracy of 0.5419.

More explanations for steps in all neural networks are given in detail in the Colaboratory notebook prior to code cells in order to establish a better flow of understanding.

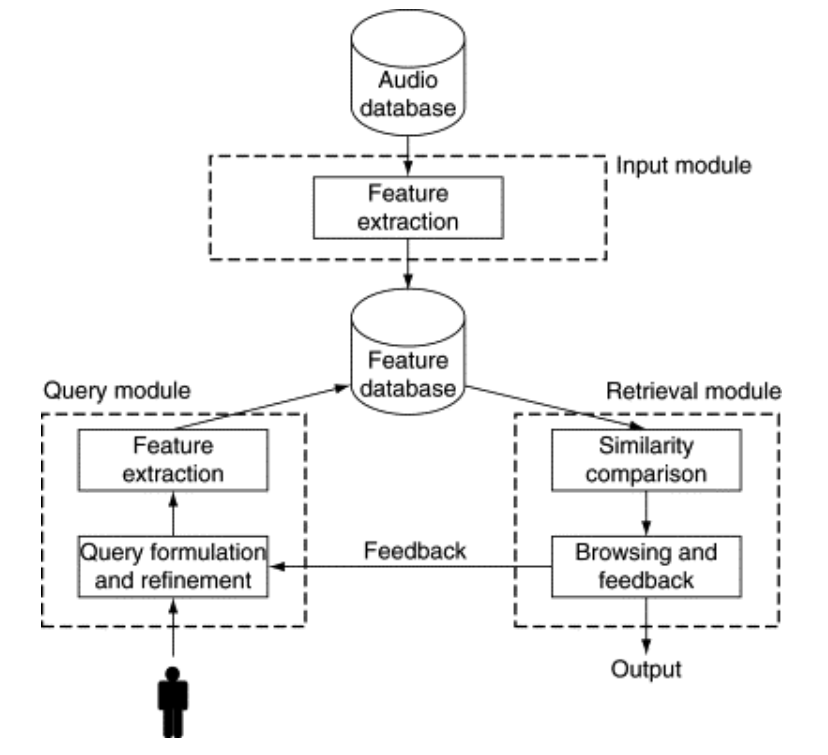
# Conclusion

This project covers the use of deep neural networks in three different facets, with performance accuracies ranging over 54-64%. Over the years, there have been a plethora of domains data scientists have pursued and explored in order to approach audio classification and wrangling audio datasets. Research in respective areas aims to perform consistent and systematic monitoring of noise pollution scaled city-wise, describing said acoustic environments in context of source composition and aiding the participation of inhabitants in noise mitigation/reporting. Deep, high capacity models in conjunction with augmented training sets tend to outperform conventional neural netwtorks [9].

Class conditional data augmentation is an infamous approach that researchers use to prove that model performance relies solely on an individual influence on each sound class in a dataset. These classes can form contextual audio augmentation sets such as Time Stretching, Pitch Shifting, Dynamic Range Compression, and Background Noise. They can then be subjected to a “shallow” dictionary learning model. However, as is in other typical deep learning networks- to enhance results the projects can consist of varying types of regularizations (Dropout, L1, L2), picking from a wide array of activation functions, modifying number of classes, and testing optimization algorithms.

TensorFlow’s Estimator API- DNN Classifier is also another technique adapted to build a neural network that gives efficient results [3]. Feature columns constitute data that is fed into the estimator. Next, an input function is defined in order to mechanize fetching data from the dataset. Following that, a classifier is initialized and a training loop established in order to monitor metrics (loss curves and confusion matrix) whilst training said classifier.

When it comes to audio classification, there are three prime domains; automatic speech recognition (syntactic level analysis), music information retrieval (extracting pitch, duration, signal source), acoustic surveillance and consequently environmental sound retrieval. In this project, I chose to focus on environmental sound retrieval as it focuses of audio that is rendered as neither music or speech, and since the domain size is so arbitrary, techniques for classification and feature extraction are rather straightforward. While it is true that only humans can bridge semantic gaps in audio with prior knowledge and context, researchers have tried to narrow this semantic gap in machine learning by using the following methodology [10]:



In conclusion, undertaking this project shed light on how a monotonous, trivial task can be enhanced and undertaken by deep learning via persuing schemes of taxonomies involving available features (indexed, annotated data) and various other learning methodologies available to us. I chose audio because I wanted to learn about a novel facet differing from images or text. I hope you enjoy your experience with this project as much as I did.

# References

|  |  |
| --- | --- |
| [1] | Franky, "Deep Learning, Audio Classification, and Beyond," 24 07 2018. [Online]. Available: https://medium.com/@franky07724\_57962/deep-learning-audio-classification-and-beyond-4d2533080a33. [Accessed 06 12 2020]. |
| [2] | S. Mourya, "Urban Sound Classification," 04 05 2018. [Online]. Available: https://github.com/satyen95/Urban-Sound-Classification. [Accessed 30 11 2020]. |
| [3] | D. Glavas, "Let's Build An Audio Classifier," 20 05 2019. [Online]. Available: https://davidglavas.me/lets-build-an-audio-classifier/. [Accessed 30 11 2020]. |
| [4] | A. P. J. R. L. L. Venkatesh Boddapati, "Classifying Environmental Sounds using Image Recognition Networks," *Procedia Computer Science,* vol. 112, pp. 2048-2056, 2017. |
| [5] | A. Saeed, "Urban Sound Classification," 03 09 2016. [Online]. Available: http://aqibsaeed.github.io/2016-09-03-urban-sound-classification-part-1/. [Accessed 30 11 2020]. |
| [6] | C. J. J. P. B. Justin Salamon, "A Dataset and Taxonomy for Urban Sound Research," Music and Audio Research Laboratory, New York University, New York City. |
| [7] | L. L. H.-J. Z. J.-H. T. a. L.-H. C. Dan-Ning Jiang, "Music Type Classification by Spectral Contrast Feature," in *IEEE International Conference on Multimedia and Expo*, Lausanne, Switzerland, 2002. |
| [8] | M. S. a. M. G. Christopher Harte, "Detecting Harmonic Change in Musical Audio," in *1st ACM Workshop on Audio and Music Computing Multimedia*, New York, NY, 2006. |
| [9] | J. P. B. Justin Salamon, "Deep Convolutional Neural Networks and Data," *IEEE Signal Processing Letters,* p. 5, 2016. |
| [10] | M. Z. C. B. Dalibor Mitrović, "Features for Content-Based Audio Retrieval," *Advances in Computers,* vol. 78, pp. 71-150, 2010. |