# Urban Sounds

**CLASSIFICATION**

Author: Yashaswi Aryan

Reg. No.: 200968186

Date: 30th October, 2022

Serial no.: 1

Paper type: Project Report

## Goal

Given a short audio we need to implement a model that can determine  if it contains any one of the urban sounds and if contains the model needs to recognize it.

## Dataset

This dataset contains 8732 labelled sound excerpts (<=4s) of urban sounds from 10 classes: `air_conditioner`, `car_horn`, `children_playing`, `dog_bark`, `drilling`, `engine_idling`, `gun_shot`, `jackhammer`, `siren`, and `street_music`. The classes are drawn from the urban sound taxonomy.

In addition to the sound excerpts, a CSV file containing metadata about each excerpt is also provided

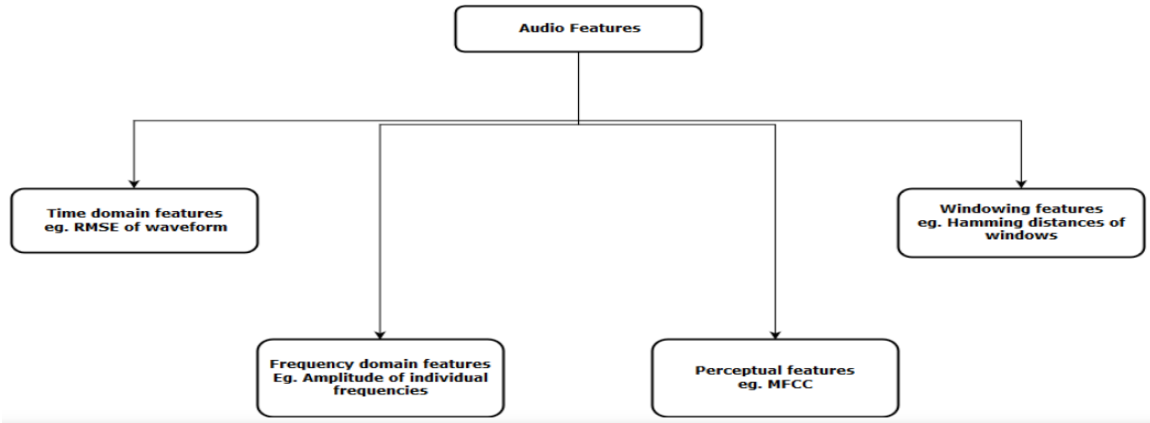
### Metadata

UrbanSound8k.csv has following columns:

- slice*file*name:
  The name of the audio file. The name takes the following format: [fsID]-[classID]-[occurrenceID]-[sliceID].wav, where:
  [fsID] = the Freesound ID of the recording from which this excerpt (slice) is taken
  [classID] = a numeric identifier of the sound class (see description of classID below for further details)
  [occurrenceID] = a numeric identifier to distinguish different occurrences of the sound within the original recording
  [sliceID] = a numeric identifier to distinguish different slices taken from the same occurrence
- fsID:
  The Freesound ID of the recording from which this excerpt (slice) is taken
- start
  The start time of the slice in the original Freesound recording
- end:
  The end time of slice in the original Freesound recording
- salience:
  A (subjective) salience rating of the sound. 1 = foreground, 2 = background.
- fold:
  The fold number (1-10) to which this file has been allocated.
- classID:
  A numeric identifier of the sound class:
  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
- class:
  The class name: air*conditioner, car*horn, children*playing, dog*bark, drilling, engine*idling, gun*shot, jackhammer,
  siren, street_music.

## Dealing with Audio Data

*Audio data is represented by converting it into a different domain of data representation.*

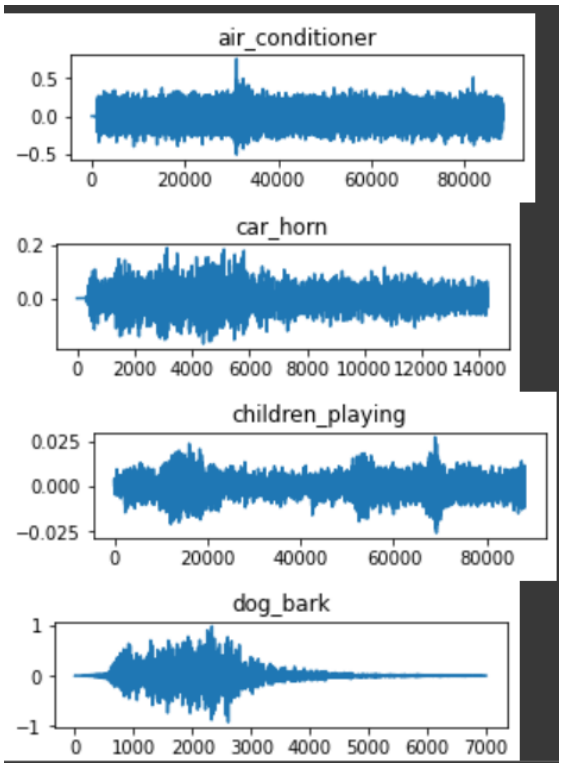


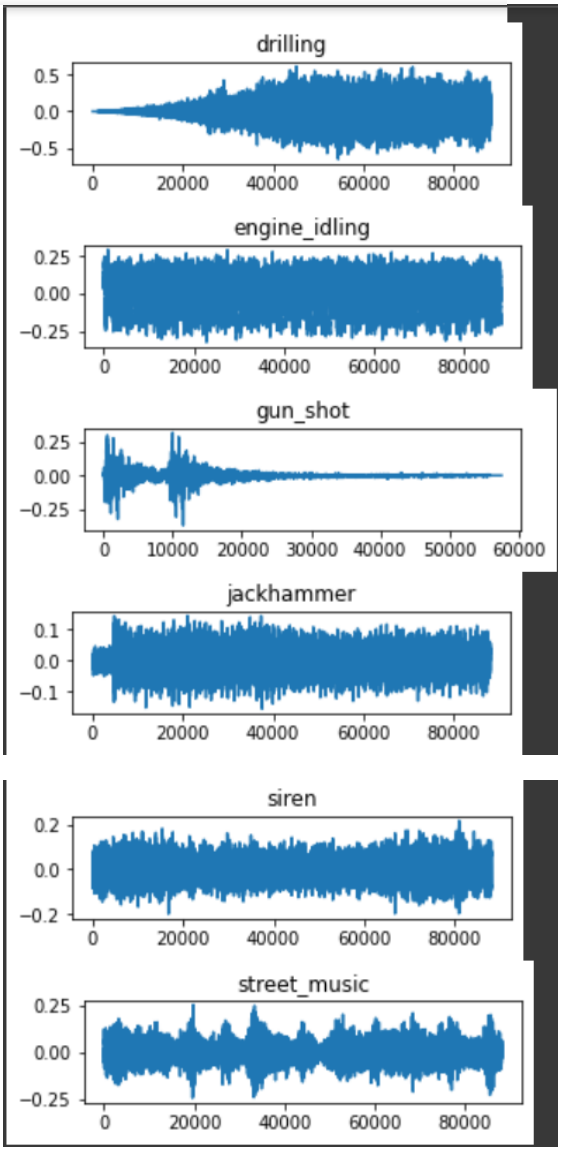

# Exploratory Data Analysis

## Visualizing Audio

We can plot the audio array using **librosa.display.waveplot**.

Following figure shows wave-plot of one particular audio from each class:

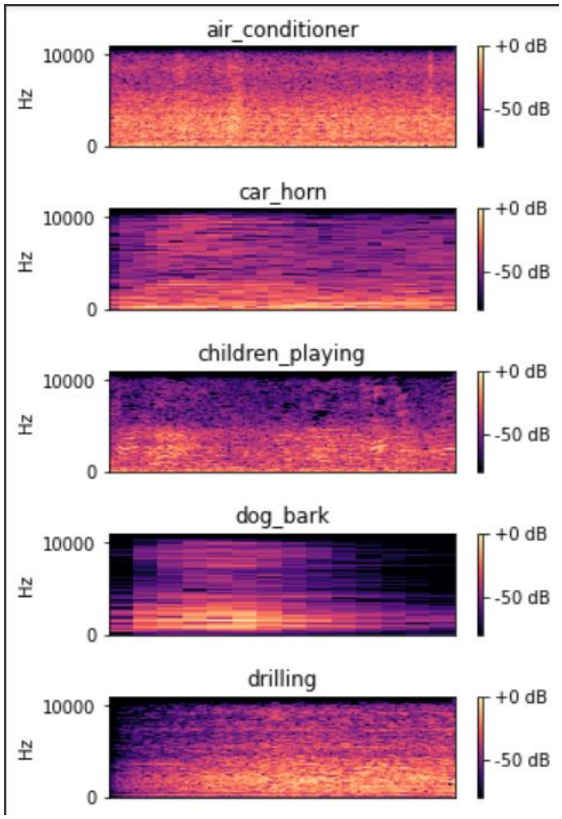


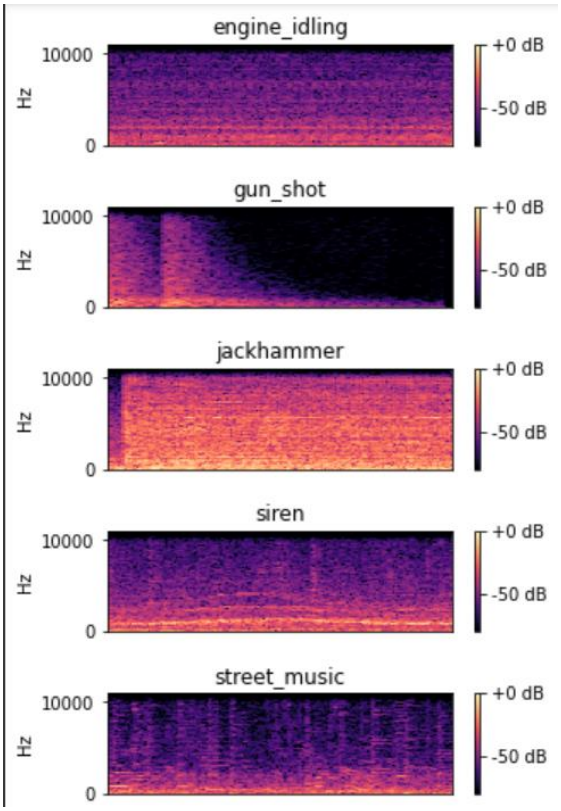



## Spectrogram

A spectrogram is a visual way of representing the signal strength, or “loudness”, of a signal over time at various frequencies present in a particular waveform. A spectrogram is usually depicted as a heat map, i.e., as an image with the intensity shown by varying the colour or brightness.

Following figure shows spectrogram of one particular audio from each class:



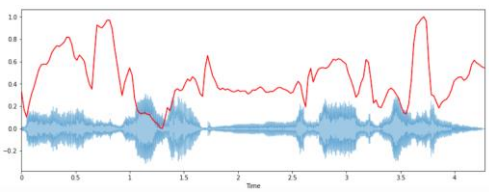




## Feature Extraction

The **spectral features** (frequency-based **features**), which are obtained by converting the time-based signal into the frequency domain using the Fourier Transform, like fundamental frequency, frequency components, **spectral** centroid, **spectral** flux, **spectral** density, **spectral** roll-off, etc.

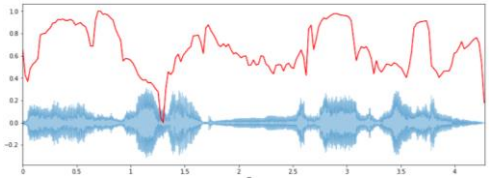
### Spectral Centroid

Indicates at which frequency the energy of a spectrum is cantered upon or in other words It indicates where the “centre of mass” for a sound is located.



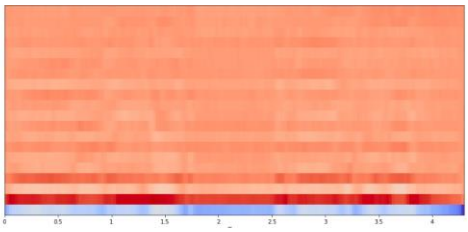
### Spectral Rolloff

It is a measure of the shape of the signal. It represents the frequency at which high frequencies decline to 0.



### Mel-frequency Cepstral Coefficients (MFCCs)

The Mel frequency cepstral coefficients (MFCCs) of a signal are a small set of features (usually about 10–20) which concisely describe the overall shape of a spectral envelope.



### Pre-processing

- First we need to extract features from the audio.
- To do so there are three ways:
    - Using the mfccs data of the audio files
    - Using a spectrogram image of the audio and then converting the same to data points (As is done for images). This is easily done using mel_spectrogram function of Librosa
    - Combining both features to build a better model. (Requires a lot of time to read and extract data).
- We will be using the first one

We will be using Librosa package for Python for EDA and Pre-processing of audio files

#### *Librosa*

It is a Python module to analyse audio signals in general but geared more towards music. It includes the nuts and bolts to build a MIR (Music information retrieval) system.

## Project Objectives

- We will define 4-5 models for classification of unknown audio to predefined class labels.
- The model reaches 80 ~ 85% accuracy, we will consider as successful model.

#### *Evaluation Metrics*

The evaluation metric for this project will be 'Classification Accuracy' which is defined as the percentage of correct predictions.

$$Accuracy = \frac{Correct\ Classifications}{Number\ of\ Classifications}$$

Classification Accuracy was deemed to be the optimal choice metric as it is presumed that the dataset will be relatively symmetrical (as we will explore in the next section) with this being a multi-class classifier whereby the target data classes will be generally uniform in size.

## Model

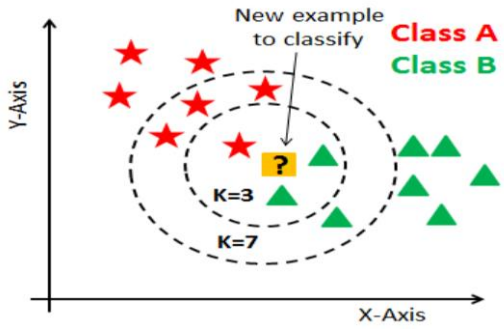
After going through several research papers, I have shortlisted following models to implement, compare and analyse for the project.

### K Nearest Neighbours Classifier

KNN is a supervised lazy learning, non-parametric algorithm. It uses data with

several classes to predict the classification of the new sample point. KNN is non-parametric since it doesn't make any assumptions on the data being studied, i.e., the model is distributed from the data.

The k-nearest neighbour algorithm stores all the available data and classifies a new data point based on the similarity measure (e.g., distance functions)




### Advantages of KNN

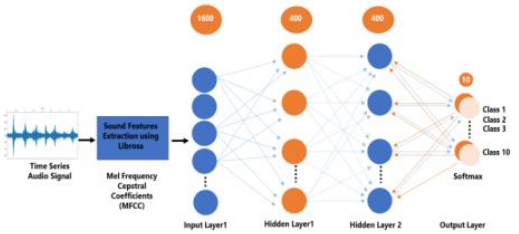
- Expect little to no explicit training phase which is fast
- Intuitive & simple
- No assumptions about data

### Disadvantages of KNN

- Sensitive to scale of data & irrelevant features
- Will require high memory
- Prediction can take a lot more time
- Imbalanced data can cause problems
- Not capable of dealing with missing values.

## Deep Neural Network

A simple baseline model of neural network with an input layer passed from MFCC and an output layer consisting of 10 well-defined labels, and 2-3 hidden layers (which will be increased or decreased according to performance)




DNN Model Architecture

### Advantages of DNN

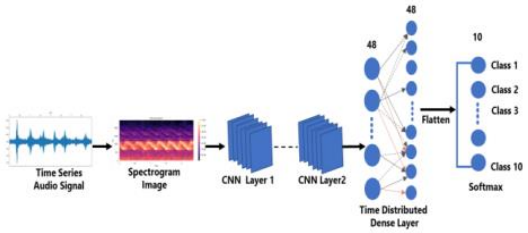
- Produces better results for higher amount amount of data as compared to other ml algorithms
- Can be used on any type of data. Eg sound, images, text etc
- Robust to outliers

### Disadvantages of DNN

- Computationally expensive
- How to understand how input maps to outputs via hidden layer (Blackbox)

## Convolutional Neural Network

This model will contain multiple convolution and pooling layers stacked together as block. This is connected to a fully connected(dense) layer which outputs values using SoftMax activation function.




CNN Model Architecture

### Advantages of CNN

- Better learning capabilities as compared to baseline models of neural network.
- Has a huge data capacity

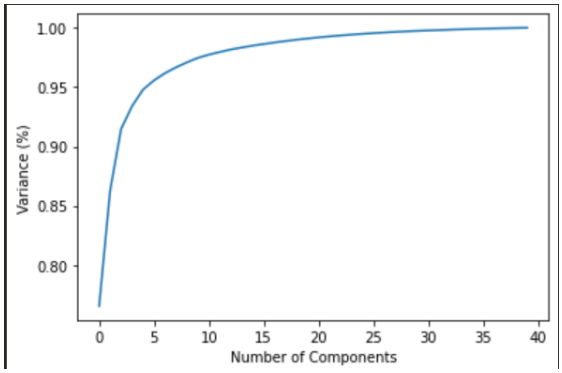
### Disadvantages of CNN

- Takes a lot of time in training model
- Exploding & vanishing gradient problems

# Methodology and Evaluation

An extracted feature dataframe was created by extracting features using mfcc(scaled down using mean) for KNN & ANN model

Before building model applying PCA for benefitting from little bit of noise reduction.




As we can see most of the variance is explained using all the features of the MFCC. This is expected since each feature gives information about the wave shape.

## K Nearest Neighbours Classifier

We first create a KNN classifier instance and then prepare a range of values of hyperparameter K as {3, 5, 7, 9, 11, 15} that will be used by GridSearchCV to find the best value of K.

Furthermore, we set our cross-validation batch sizes cv = 5 and set scoring metrics as accuracy as our preference.

Once the model is fit, we can find the optimal parameter of K and the best score obtained through GridSearchCV.

### Accuracy

Our model accuracy turned out to be ~0.89350 which was increased due to noise reduction we did with the help of PCA and it increased to ~0.9030267.

## Artificial Neural Network

Normal train/test split was used to split the dataset from folds to train, test and validation set from extracted feature dataframe.

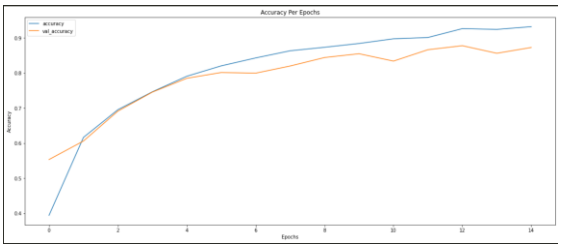
Model contains below mentioned layers:

- **Input layer:** The first layer is input layer which contains mfcc extracted features that is passed through model. The layer has input node of 1000, with an input shape of (40,) because during feature extraction, n mfcc was 40. Activation function is 'ReLu'.
- **Five hidden layers:** The nodes in five hidden layers decrease in order of 750, 500, 250, 100 & 50. 'ReLu' activation functions are used which will help in tackling the vanishing gradient problem. The negative part of the argument will be removed by using the 'ReLu' activation function.
    - f ReLU(x)=max(0, x)
- **Output layer:** The final layer is the output layer. The activation function used is 'SoftMax'. The output layer will be used to generate the output based on the number of labels, i.e., 10. SoftMax will give a probability of each class that will sum up equal to 1.

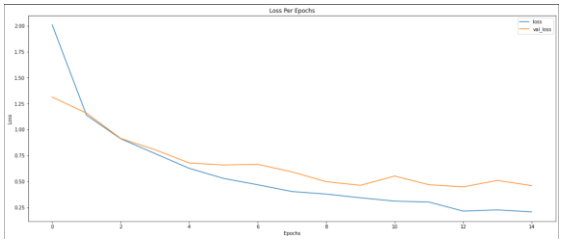
The optimizer used was 'Adam'. Adam optimizing algorithm is an extension of stochastic gradient descent, which is used to update the weights of the network in the training data.

### Accuracy

Following figure shows the plot which contain accuracy of train & validation sets:




Following figure shows the plot which contain loss of train & validation sets:



The classification report is as follows:

```
Classification Report :

              precision    recall  f1-score   support

           0       0.92      0.93      0.93       100
           1       0.97      0.86      0.91        43
           2       0.73      0.87      0.79       100
           3       0.83      0.80      0.82       100
...
    accuracy                           0.88       874
```

Modifications in architecture 2 used in notebook was done to reach better accuracy on train & validation.

## Convolutional Neural Network

This is a 2-D CNN model using the 'ReLu' activation function.(Shamsaldin et al. 2019) After feature extraction, data-frame has been created (the methodology used here is different than what used for ANN & KNN as we created spectrogram images to get a 2D format to feed into CNN model). This data-frame has been split into train and test set using sklearn's train test split. Test size of 20 % and train set of 80 % size have been created. Layers are added as follows:

- **Input Layer:** The first layer is an input layer, which has a filter = 16, the size of the kernel is 2, and the activation function is 'ReLu'. MaxPooling2D has been added with pool size of (2,2) and dropout has been set as 0.2.
- **Three Hidden Layer:** Hidden layers contain a total of 3 layers of filters 32, 64, and 128. All have the same activation function 'ReLu' with dropout as 0.2 and MaxPooling2D as (2,2). MaxPooling2D reduces the dimension of the image by retaining important information. GlobalAveragePooling2D is used as the next sub-layer, this is used to apply pooling on a spatial dimension, which is average pooling. A flatten sub-layer is used after the last hidden layer, which converts the image into the one-dimension image, which will be used as input for the next layer.
- **Output Layer:** The final layer is the output layer which is same as ANN discussed above.

The optimizer used is 'Adam' (discussed in ANN part)

Loss has been selected as 'categorical crossentropy' and metrics used is 'accuracy'. The batch size used is 256 and epochs is 100 at the time of fitting the model.

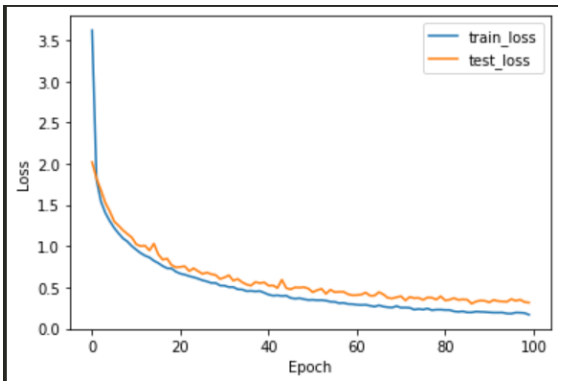
### Accuracy

Following are training & testing accuracy obtained:

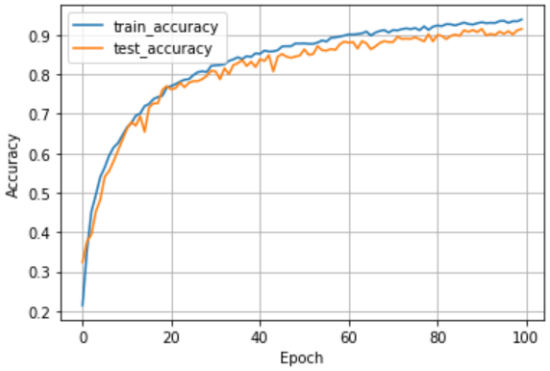
Training accuracy: ~0.97

Testing accuracy: ~0.91

Following figure shows training & test loss visualized over 100 epochs:




Following figure shows training & test accuracy visualized over 100 epochs:




The classification report is as follows:

Classification Report :

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.95 | 0.94 | 218 |
| 1 | 0.89 | 0.94 | 0.92 | 90 |
| 2 | 0.89 | 0.86 | 0.87 | 199 |
| 3 | 0.94 | 0.86 | 0.90 | 199 |
| 4 | 0.86 | 0.94 | 0.90 | 200 |
| 5 | 0.92 | 0.96 | 0.94 | 203 |
| 6 | 0.94 | 0.98 | 0.96 | 65 |
| 7 | 0.94 | 0.95 | 0.95 | 185 |
| 8 | 0.94 | 0.95 | 0.94 | 197 |
| 9 | 0.92 | 0.81 | 0.86 | 191 |
| accuracy | | | 0.92 | 1747 |

## Conclusion

Conclusively, deep learning models are not able to understand the audio files directly. This problem was solved by extracting features from all the audio files. Librosa library has been used to extract the features and a new data-frame was created with those extracted features. This data-frame was used by 2 deep learning models & 1 machine learning model. The train and test sets are created using 2 ways, ANN & KNN model is using a vector of extracted features for all audios. Despite being a very simplistic model K Nearest Neighbour Classifier was able to achieve 90% accuracy. ANN also achieved a good result on this data-set and was able to get 91 % accuracy after several modifications and optimization of parameters. Next, for CNN 2D model features were extracted in & saved as mel-spectrogram images. Overall, CNN 2D models perform best for the audio dataset with an accuracy of 97% with less error rate. In future, I'll be implementing LSTM & other handcrafted models.

## References

[1] M. Massoudi, S. Verma and R. Jain, "Urban Sound Classification using CNN," 2021 6th International Conference on Inventive Computation Technologies (ICICT), 2021, pp. 583-589, doi: 10.1109/ICICT50816.2021.9358621.

[2] M. Bubashait and N. Hewahi, "Urban Sound Classification Using DNN, CNN & LSTM a Comparative Approach," 2021 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), 2021, pp. 46-50, doi: 10.1109/3ICT53449.2021.9581339.

[3] I. Lezhenin, N. Bogach and E. Pyshkin, "Urban Sound Classification using Long Short-Term Memory Neural Network," 2019 Federated Conference on Computer Science and Information Systems (FedCSIS), 2019, pp. 57-60, doi: 10.15439/2019F185.

[4] Jederson S. Luz, Myllena C. Oliveira, Flávio H.D. Araújo, Deborah M.V. Magalhães, Ensemble of handcrafted and deep features for urban sound classification, Applied Acoustics, Volume 175, 2021, 107819, ISSN 0003-682X.

[5] https://www.kaggle.com/code/prabhavsingh/urbansound8k-classification/notebook

[6] Project report by Sanket Sonu on github:

https://github.com/SanketSonu/UrbanSound8K