Urban Sound Classification with Neural Networks

MACHINE LEARNING MODULE

Rijin Baby (954007)

Manasa Shankara Murthy (978401)

Table of Contents

Declaration

We declare that this material, which we now submit for assessment, is entirely our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of our work. We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by us or any other person for assessment on this or any other course of study.

1 INTRODUCTION:

Sound is one of the most important phenomena of Physics. It's a kind of vibration that's supposed to propagate in waveform. Generally, classification of urban sound falls under a category of two major fields- Natural language processing (NLP) and Digital Signal Processing (DSP). The advancement in the fields of control system and digital signal processing has led to rise of application of Artificial Intelligence and Machine Learning to recognize sound and classify them. Thus, in short, Sound classification is the process of listening to and analyzing different audio recordings and classify them to different predefined urban sound taxonomical categories. This technology is at the heart of a variety of modern AI technology like- Virtual AI assistant, Automated Speech recognition, and text to speech applications.

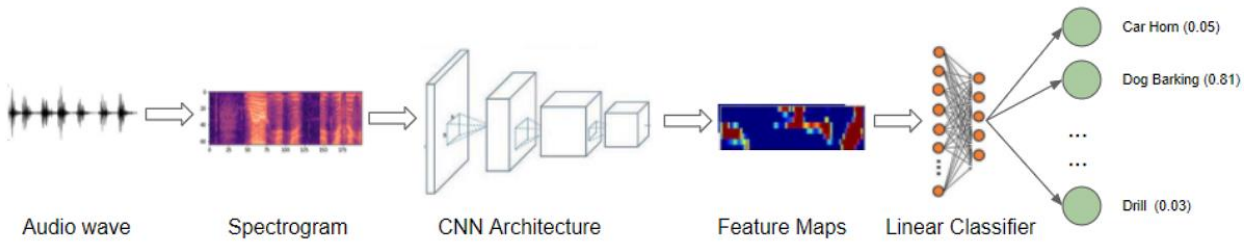
The most important task of this project is to understand the data and the type of data we are working with. The dataset is 'UrbanSound8K' dataset.

1.1 OBJECTIVE:

To develop a methodology and an efficient machine learning algorithm to be trained on pre-labelled dataset and efficiently recognize, predict and classify random sound signals into these predefined labels.

We employ CNN and DNN models to achieve this task.

The schematic process flow of the activity is given below:



2. UNDERSTANDING THE DATA & FEATURES:

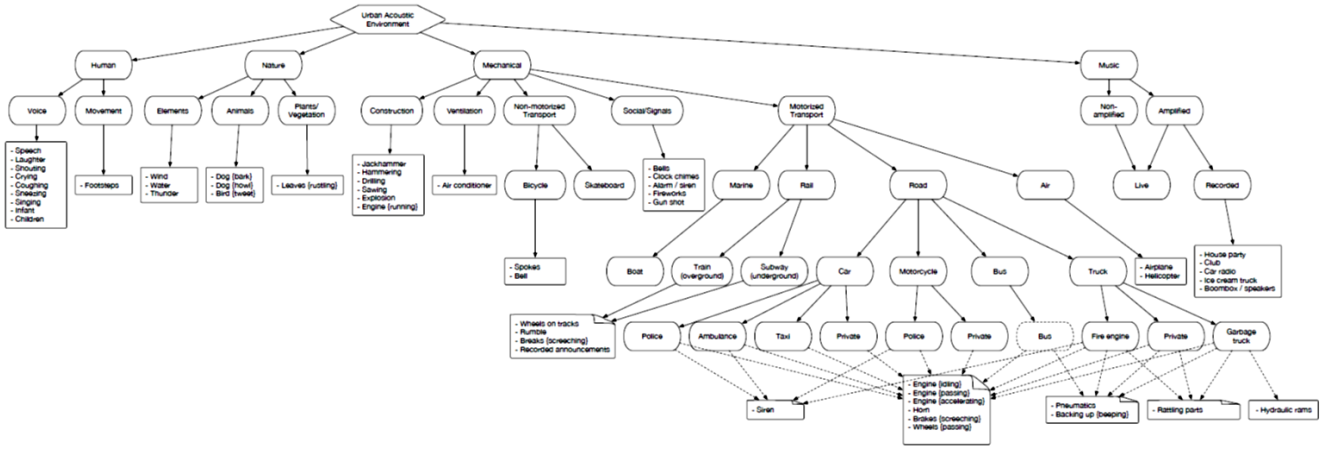
2.1 DATASET DESCRIPTION:

The dataset has files mainly in .WAV format and there are 8732 audio files of urban sounds hence the name 8K 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
- street_music

The classes are drawn from the urban sound taxonomy [Fig -1 given below].



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

After downloading the dataset, we see that it consists of two parts:

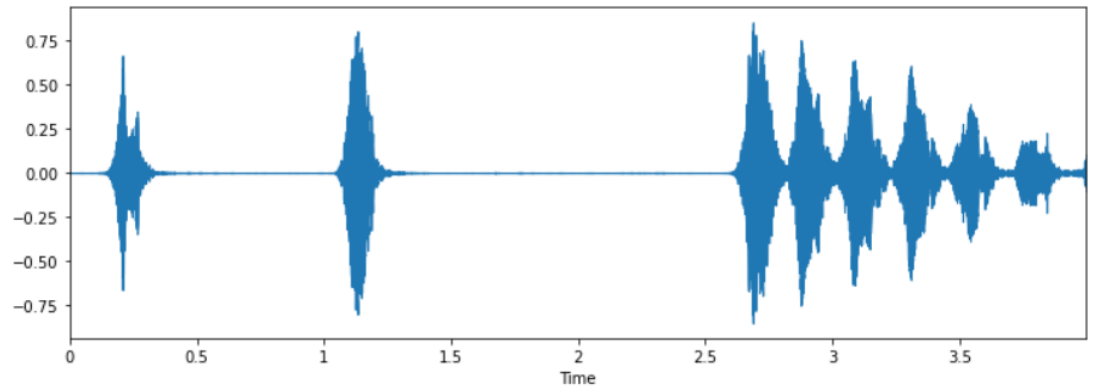
- **Audio files** in the '*audio*' folder: It has 10 sub-folders named '*fold1*' through '*fold10*'. Each sub-folder contains several '*.wav*' audio samples eg. '*fold1/103074-7-1-0.wav*' The files are pre-sorted into ten folds (folders named fold1-fold10) to help in the reproduction of and comparison with the automatic classification results.
- **Metadata** in the 'metadata' folder: It has a file '*UrbanSound8K.csv*' that contains information about each audio sample in the dataset such as its filename, its class label, the 'fold' sub-folder location, and so on. The class label is a numeric Class ID

from 0–9 for each of the 10 classes. e.g., the number 0 means air conditioner, 1 is a car horn, and so on.

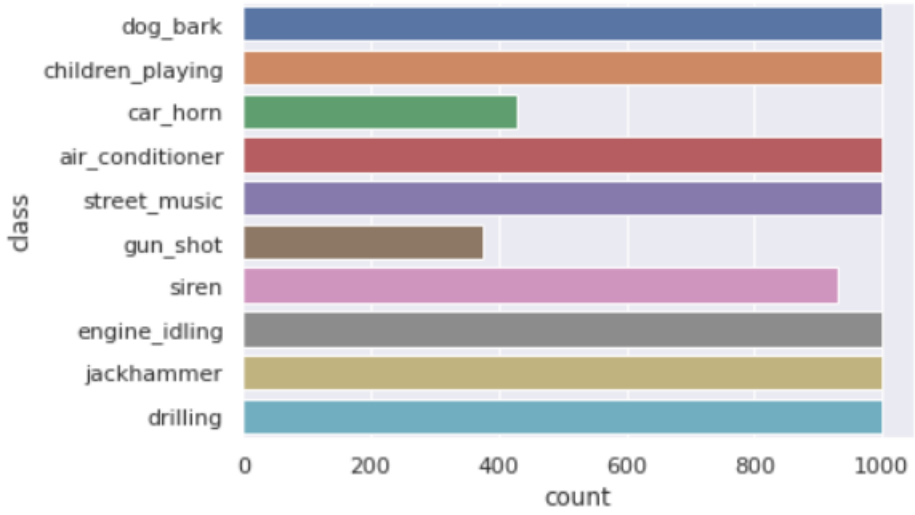
The samples are around 4 seconds in length. Here's what one sample looks like:

```
[8]  plt.figure(figsize=(12,4))
     librosa.display.waveplot(dog_bark, sr=sampling_rate)
```

```
<matplotlib.collections.PolyCollection at 0x7fcfd3e87310>
```

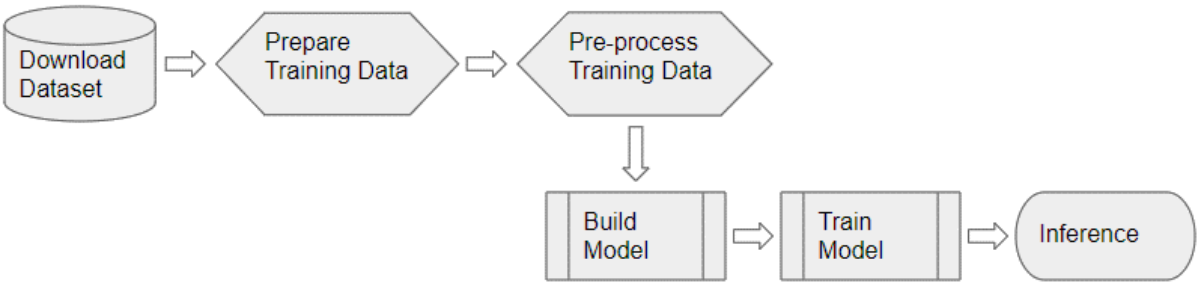


DATA DISTRIBUTION



2.2 PREPARING THE DATA

We will follow the following process flow for preparing the data for training:



The training data that has been extracted into a training metadata file which contains all the information simply represented as:

- The features (X) are the audio file paths
- The target labels (y) are the class names

We can prepare the feature and label data from the metadata.

2.3 PRE-PROCESSING OF THE AUDIO: DEFINING FEATURE EXTRACTION

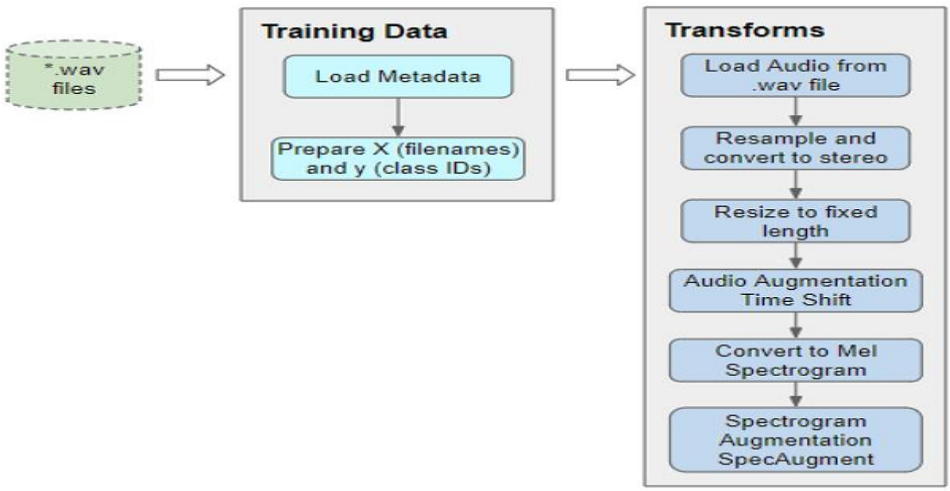
This training data with audio file paths cannot be input directly into the model. We have to load the audio data from the file and process it to a format that the model expects.

This audio pre-processing will all be done dynamically at runtime when we will read and load the audio files. This approach is similar to what we would do with image files as well. Since audio data, like image data, can be fairly large and memory-intensive, we don't want to read the entire dataset into memory all at once, ahead of time. So, we keep only the audio file names in our training data.

Then, at runtime, as we train the model one batch at a time, we will load the audio data for that batch and process it by applying a series of transforms to the audio. That way we keep audio data for only one batch in memory at a time.

The processing for audio data is very similar. Right now, we're only defining the functions, they will be run a little later when we feed data to the model during training.

All the audio files are preprocessed using Python librosa package. The audio files are individually loaded as a floating-point time series with sample rate of 22,050 hertz from sampling rate of 44,100 hertz.



In order to effectively classify the audio data, the features must be extracted from the audio sample. Three widely used techniques for audio classification research are Mel-Frequency Cepstral Coefficient (MFCC), Zero-Crossing Rate (ZCR) and Linear Predictive Coding (LPC). MFCCs have been used for feature extraction to improve speaker recognition.

Here we have looked into mainly 11 features has been identified to be worked upon, and they are : **Mel-Frequency Cepstral Coefficient** (MFCC), **Chromagram feature- Chroma Short-Time Fourier Transform** (chroma_stft), **Chroma Energy Normalized (CENS)**-(chroma_cens), **Root Mean Square Error** for each frame(rmse), **Zero-Crossing Rate** (zcr), **MEL Spectrogram**(mel), **Spectral Centroid** (cent), **Spectral Roll-Off** (rolloff), **Spectral Bandwidth** (spec_bw), **Spectral Contrast** (contrast), **Spectral Flatness** (flatness).

First, we need to understand the theory behind all of the audio features and pre-processing parameters

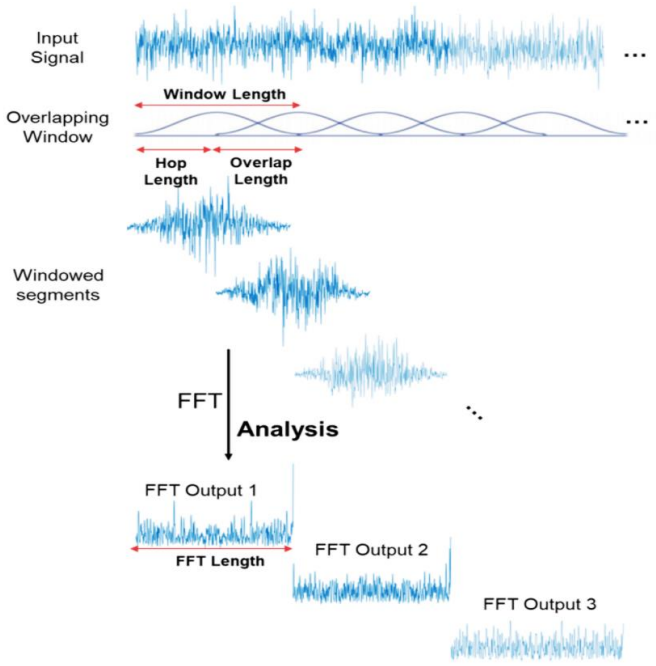
Fast Fourier Transform (FFT)

One way to compute Fourier Transforms is by using a technique called DFT (Discrete Fourier Transform). The DFT is very expensive to compute, so in practice, the FFT (Fast Fourier Transform) algorithm is used, which is an efficient way to implement the DFT.

However, the FFT will give you the overall frequency components for the entire time series of the audio signal as a whole. It won't tell you how those frequency components change over time within the audio signal. You will not be able to see, for example, that the first part of the audio had high frequencies while the second part had low frequencies, and so on.

Short-time Fourier Transform (STFT)

To get that more granular view and see the frequency variations over time, we use the STFT algorithm (Short-Time Fourier Transform). The STFT is another variant of the Fourier Transform that breaks up the audio signal into smaller sections by using a sliding time window. It takes the FFT on each section and then combines them. It is thus able to capture the variations of the frequency with time.



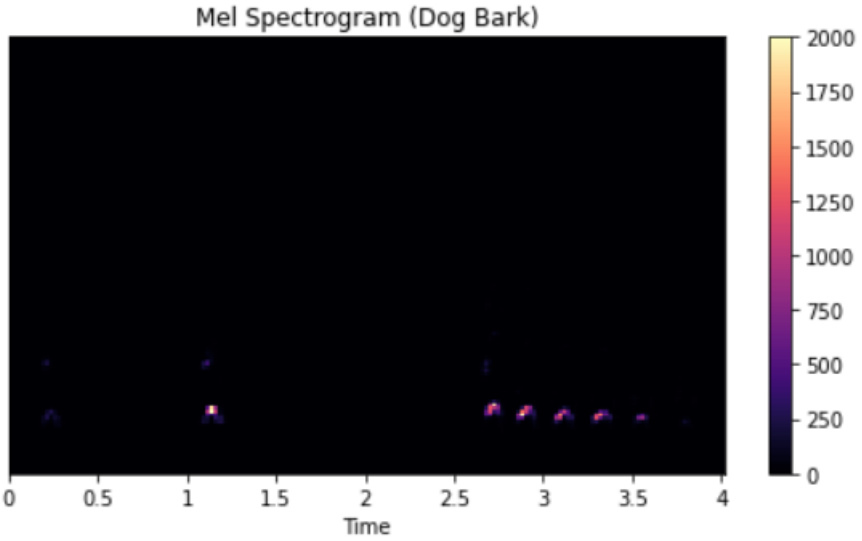
The signals whose frequency varies with time are known as **non- periodic** signals. We can compute several spectrums by performing FFT on several windowed segments of the signal, and it is called the **short-time Fourier transform.** The FFT is computed on overlapping windowed segments of the signal, and we get what is called the **spectrogram.**

The Mel Spectrogram:

A Mel Spectrogram makes two important changes relative to a regular Spectrogram that plots Frequency vs Time.

- It uses the Mel Scale instead of Frequency on the y-axis.
- It uses the Decibel Scale instead of Amplitude to indicate colors.

For deep learning models, we usually use this rather than a simple Spectrogram. The mel spectrogram demonstrated by only a couple lines of code.



Mel Spectrogram Hyperparameters

This gives us the hyperparameters for tuning our Mel Spectrogram. We'll use the parameter names that Librosa uses. (Other libraries will have equivalent parameters)

Frequency Bands:

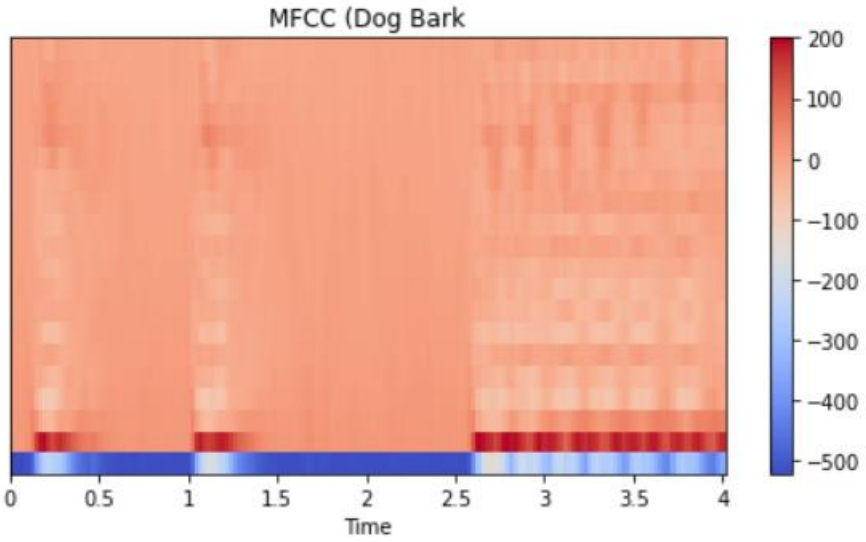
- fmin – minimum frequency
- fmax – maximum frequency to display
- n_mels – number of frequency bands (ie. Mel bins). This is the height of the Spectrogram

Time Sections:

- n_fft – window length for each time section
- hop_length – number of samples by which to slide the window at each step. Hence, the width of the Spectrogram is = Total number of samples / hops_length

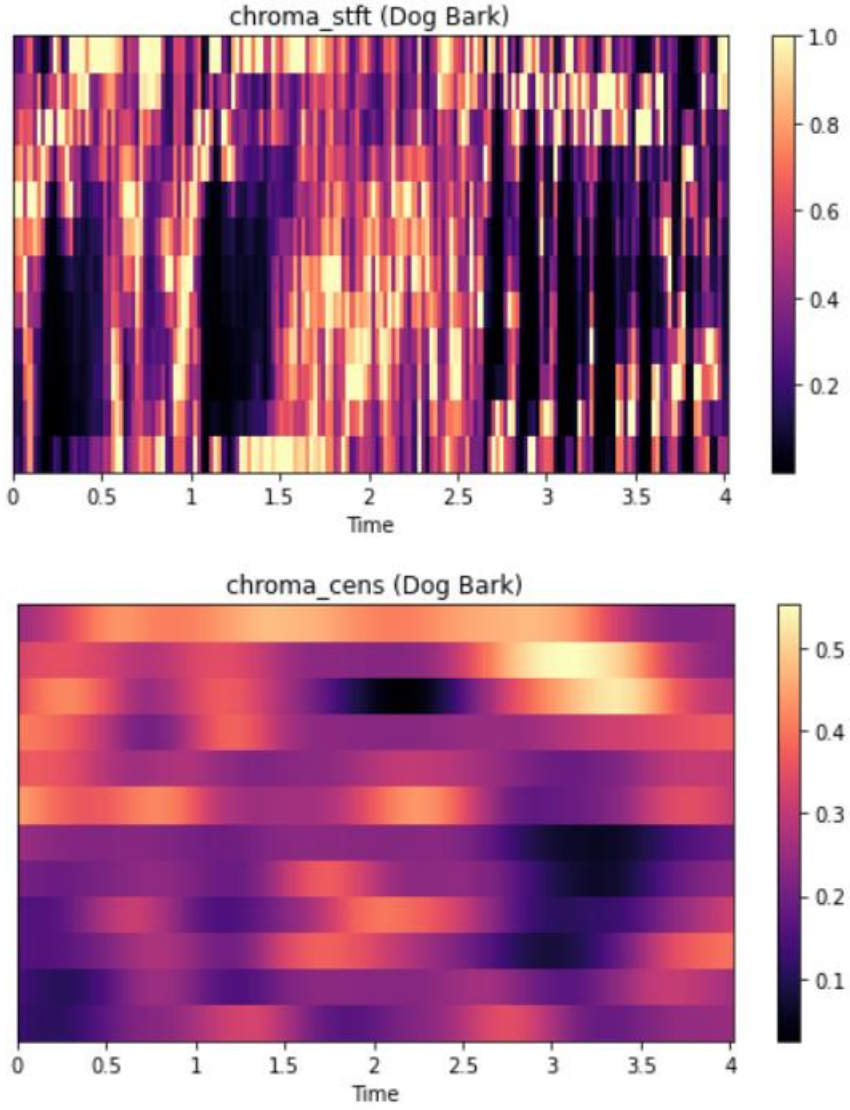
MFCC (model for Human Speech sound):

Mel Spectrograms work well for most audio deep learning applications. However, for problems dealing with human speech, like Automatic Speech Recognition, you might find that MFCC (Mel Frequency Cepstral Coefficients) sometimes work better. These essentially take Mel Spectrograms and apply a couple of further processing steps. This selects a compressed representation of the frequency bands from the Mel Spectrogram that correspond to the most common frequencies at which humans speak.



Chroma Feature and related features like chroma_stft, chroma_sens:

The chroma feature is a descriptor, which represents the tonal content of a musical audio signal in a condensed form. Therefore, chroma features can be considered as important prerequisite for high-level semantic analysis, like chord recognition or harmonic similarity estimation. A better quality of the extracted chroma feature enables much better results in these high-level tasks. Short Time Fourier Transforms and Constant Q Transforms are used for chroma feature extraction.



In music, the term chroma feature or chromagram closely relates to the twelve different pitch classes. Chroma-based features, which are also referred to as "pitch class profiles", are a powerful tool for analysing music whose pitches can be meaningfully categorized (often into twelve categories) and whose tuning approximates to the equal-tempered scale. One main property of chroma features is that they capture harmonic and melodic characteristics of music, while being robust to changes in timbre and instrumentation.

Chroma features aim at representing the harmonic content (eg: keys, chords) of a short-time window of audio. The feature vector is extracted from the magnitude spectrum by using a short time fourier transform (STFT), Constant-Q transform (CQT), Chroma Energy Normalized (CENS), etc

Zero-Crossing Rate

The Zero-Crossing Rate (ZCR) of an audio frame is the rate of sign-changes of the signal during the frame. In other words, it is the number of times the signal changes value, from positive to negative and vice versa, divided by the length of the frame. ZCR can be interpreted as a measure of the noisiness of a signal. For example, it usually exhibits higher

values in the case of noisy signals. It is also known to reflect, in a rather coarse manner, the spectral characteristics of a signal. Such properties of the ZCR, along with the fact that it is easy to compute, have led to its adoption by numerous applications, including speech-music discrimination, speech detection, and music genre classification, to name but a few.

$$zcr = \frac{1}{T-1}\sum_{t=1}^{T-1} 1_{\mathbb{R}_{<0}}(s_t s_{t-1})$$

Spectral Centroid

As the name suggests, a spectral centroid is the location of the centre of mass of the spectrum. Since the audio files are digital signals and the spectral centroid is a measure that can be useful in the characterization of the spectrum of the audio file signal.

In some places, it can be considered as the median of the spectrum but there is a difference between the measurement of the spectral centroid and median of the spectrum. The spectral centroid is like a weighted median and the median of the spectrum is like the mean. Both measure the central tendency of the signal. In some cases, both show similar results.

Mathematically it can be determined by the Fourier transform of the signals with the weights.

$$\text{Centroid} = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1} x(n)}$$

were,

- x(n) is the weight frequency value
- n is the bin number.
- f(n) is the centre frequency of the bin.

Note- we consider the magnitude of the signal as the weight. Using the spectral centroid, we can predict the brightness in an audio file. It is widely used in the measurement of the tone quality of any audio file.

Spectral Roll-Off

It can be defined as the action of a specific type of filter which is designed to roll off the frequencies outside to a specific range. The reason we call it roll-off is because it is a gradual procedure. There can be two kinds of filters: hi-pass and low pass and both can roll off the frequency from a signal going outside of their range.

We can say the spectral roll-off point is the fraction of bins in the power spectrum at which 85% of the power is at lower frequencies

This can be used for calculating the maximum and minimum by setting up the roll percent to a value close to 1 and 0.

Spectral Bandwidth

Bandwidth is the difference between the upper and lower frequencies in a continuous band of frequencies. As we know the signals oscillate about a point so if the point is the centroid of the signal, then the sum of maximum deviation of the signal on both sides of the point can be considered as the bandwidth of the signal at that time frame.
The spectral bandwidth can be computed by:
(sum_k S[k, t] * (freq[k, t] - centroid[t])**p)**(1/p)
Where p is order and t is time.

Spectral Contrast

In an audio signal, the spectral contrast is the measure of the energy of frequency at each timestamp. Since most of the audio files contain the frequency, whose energy is changing with time. It becomes difficult to measure the level of energy. Spectral contrast is a way to measure that energy variation.

Spectral flatness

Spectral flatness or tonality coefficient, also known as Wiener entropy, is a measure used in digital signal processing to characterize an audio spectrum. Spectral flatness is typically measured in decibels, and provides a way to quantify how tone-like a sound is, as opposed to being noise-like. The meaning of *tonal* in this context is in the sense of the amount of peaks or resonant structure in a power spectrum, as opposed to flat spectrum of a white noise. A high spectral flatness (approaching 1.0 for white noise) indicates that the spectrum has a similar amount of power in all spectral bands – this would sound similar to white noise, and the graph of the spectrum would appear relatively flat and smooth. A low spectral flatness (approaching 0.0 for a pure tone) indicates that the spectral power is concentrated in a relatively small number of bands – this would typically sound like a mixture of sine waves, and the spectrum would appear "spiky".

2.4 DATA AUGMENTATION:

A common technique to increase the diversity of our dataset, particularly when we don't have enough data, is to augment data artificially. We do this by modifying the existing data samples in small ways. Here using the following data augmentation technique, the dataset was augmented 4 times.

Noise

Adding white noise to data augmentation means that the network is less able to memorize training samples because they are changing all of the time, resulting in smaller network weights and a more robust network that has lower generalization error. Here in this case, we introduced white noise only to augment.

Shift

Adding Random phase shifting into the data.

Time Shift – shift audio to the left or the right by a random amount.

For sound such as traffic or sea waves which has no order, the audio could wrap around.

The idea of shifting time is very simple. It just shifts audio to left/right with a random second. If shifting audio to left (fast forward) with x seconds, first x seconds will mark as 0 (i.e. silence). If shifting audio to right (back forward) with x seconds, last x seconds will mark as 0 (i.e. silence).

Pitch Shift – randomly modify the frequency of parts of the sound.

Stretch

Time Stretch – randomly slow down or speed up the sound. Stretches the sound. Also expands the dataset slightly.

Pitch

Pitch is a perceptual property of sounds that allows their ordering on a frequency-related scale, or more commonly, pitch is the quality that makes it possible to judge sounds as "higher" and "lower" in the sense associated with musical melodies. Pitch can be determined only in sounds that have a frequency that is clear and stable enough to distinguish from noise. Pitch is a major auditory attribute of musical tones, along with duration, loudness, and timbre.

We also introduced a combination of **Speed and Pitch tuning**.

3. Model Building and Training

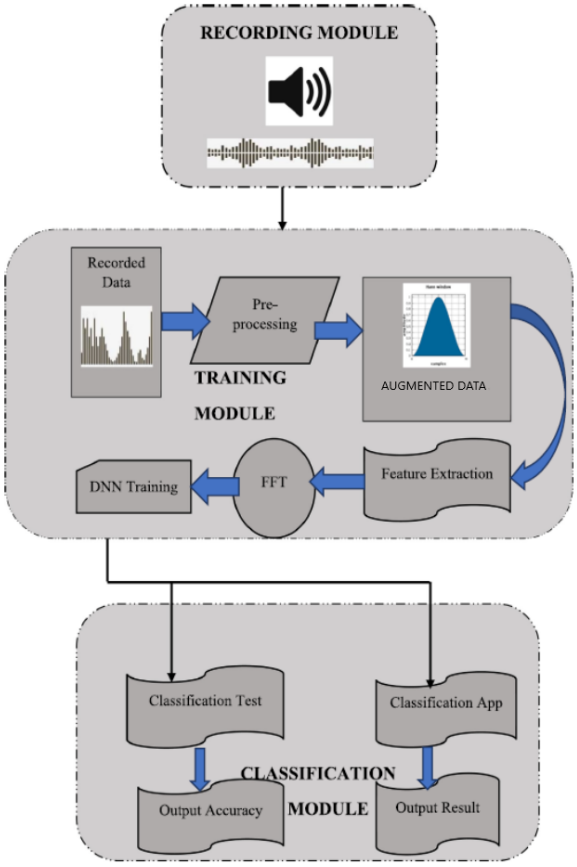


Figure: Showing the DNN process flow for Urban Sound Classification

Deep Neural Network (DNN)

The audio classification is used to distinguish audio samples by keywords, intonation and accent. Audio classification can lead to real time transcription and translation of audio. The majority of audio classification research focuses on a specific classification task to obtain high accuracy. However, due to the complexity of audio data, various techniques must be employed to analyse the data.

In order to effectively classify the audio data, the features must be extracted from the audio sample. Three widely used techniques for audio classification research is Mel-Frequency Cepstral Coefficient (MFCC), Zero-Crossing Rate (ZCR) and Linear Predictive Coding (LPC) [1, 2]. MFCCs have been used for feature extraction to improve speaker recognition. After this, Support Vector Machine (SVM) and Gaussian Mixture Model (GMM) are applied to do the classifications [3, 4]. Recently, Deep neural networks (DNNs) and more specifically Convolutional neural networks (CNNs) have been used to automatically learn feature representations from complex data [5]. This universal technique has been applied in many areas to replace ad hoc function designs and has shorten a decadelong development period to a few months. The related applications have been seen in the audio classification area. For example, DNNs in conjunction with transformed MFCCs have been used to improve the accuracy of speaker age classification [1]. Other researchers have used DNNs for cepstral feature extraction of audio samples [6]. CNNs can deal with complex nonlinear mappings and can share weights across the input, which allows for translation

invariance of the input. Most of the DNN-based algorithms need to convert the original audios into spectrograms before processing them. Spectrograms provide a visual representation of the frequencies with respect to time. Methods that use a time distributed approach [7, 8] split the spectrogram into frames to create a time-distributed spectrogram. The time-distributed spectrogram is used as the input into the CNN to train the model to distinguish local features at different time steps. A different approach [9] to audio classification splits the spectrogram along frequency to create a frequency-distributed spectrogram. Using this approach allows for the model to learn features based on various frequencies. Although the models based on spectrograms have achieved great successes, there are some intrinsic problems that are hard to eliminate. In particular, the function to generate spectrograms is independent from the later classification process. Practitioners must generate spectrograms from the audios before training the models. As a result, the spectrogram-generating function cannot be jointly optimized with the classification networks, which would considerably harm the performances of the algorithms. Besides, the spectrogram-generating process spans the originally one-dimensional audio data into three dimensions (one for time, one for frequency and one for three colour channels: red, green and red), which makes the representation sparse (thus hard to learn) and adds extra noises that could interfere the later classification process.

CONVOLUTIONAL NEURAL NETWORK(CNN)

Convolutional neural networks are applied for management data characterized by a grid topology. CNN focuses local reports on adjacency structures that appear in the data. These processes of knowledge extraction take place through adaptive learning of patterns that start from the bottom to reach the highest level. CNNs are widely used in machine vision for object recognition. Convolutional neural networks are deep neural networks created to operate on grid inputs characterized by strong spatial dependencies in local regions. An example of grid input is a two-dimensional image: it is nothing more than an array of values between 0 and 255. Adjacent pixels present in an image are interlaced with each other and together they define a pattern, a texture, a contour, etc. CNNs associate these characteristics with values called weights, which will be similar for local regions with similar patterns [35]. The quality that distinguishes them is also the operation that gives them their name, that is, convolution. It is nothing more than the scalar product between matrices, specifically between a grid structure of weights and a similar structure from the input. Convolutional networks are deep networks in which the convolutional layer appears at least once, although the majority use far more than one.
The input of a 2D CNN is an image, that is, an array of values of each single pixel occupying a precise position within the image. RGB (RED, GREEN, BLUE) images are described by a set of matrices of the intensities of the primary colours; the dimensions of an image are therefore not limited to height and width, but there is a third: depth, depth. The dimensions of the image, including the depth, will be given to the input layer, the very first layer of a CNN. The subsequent activation maps, that is, the inputs of the subsequent layers, also have a multidimensional structure and will be congruent in number with the independent properties relevant to the classification [36]. The convolutional neural networks operate on grid structures characterized by spatial relationships between pixels, inherited from one layer to the next through values that describe small local regions of the previous layer. The

set of matrices of the hidden layers, the result of convolution or other operations, is called a feature map or activation map, the trainable parameters are tensors called filters or kernels [37].
A CNN is composed of sequences of the following layers:

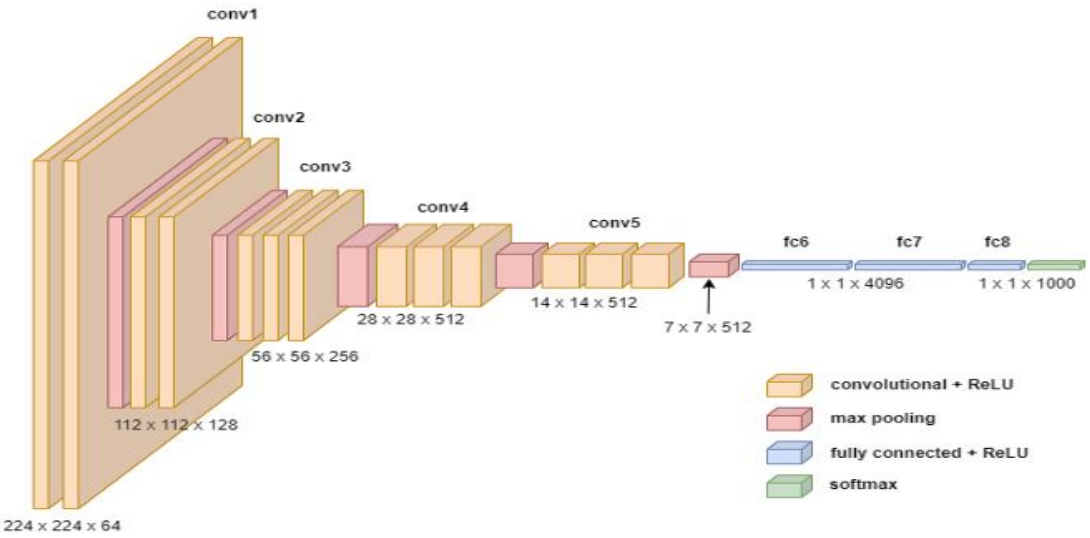
_ Convolutional layer
_ Activation layer
_ Pooling layer

From a mathematical point of view, a CNN can be regarded as a neural network densely connected with the substantial difference that the first layers carry out a convolution operation. We indicate with the following equation the relationship between input and output of an intermediate layer:

$$Y^{[j]} = K^{[j]} * I^{[j-1]} + b^{[j]}$$

Here:
_ K[j] is the kernel;
_ I[j-1] is the input passed to the convolutional layer;
_ b[j] is the bias.
Input passed to the current layer comes from a previous layer through the following equation:

$$I^{[j-1]} = g^{j-1}\left(Y^{[j-1]}\right)$$



Hyper parameter setting of CNN:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 1, 50, 64)         1664
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 1, 25, 64)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 1, 25, 128)        204928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 1, 13, 128)        0
_________________________________________________________________
dropout (Dropout)            (None, 1, 13, 128)        0
_________________________________________________________________
flatten (Flatten)            (None, 1664)              0
_________________________________________________________________
dense (Dense)                (None, 256)               426240
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
=================================================================
Total params: 769,546
Trainable params: 769,546
Non-trainable params: 0
_________________________________________________________________
```

Hyper parameter setting of DNN:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1024)              52224
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_3 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_5 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_6 (Dense)              (None, 10)                330
=================================================================
Total params: 751,914
Trainable params: 751,914
Non-trainable params: 0
_________________________________________________________________
```

4. Results

Data Splits:

- 5050: Training = fold 1, 2, 3, 4, 6

  Validation= fold 5

  Testing= fold 5, 7, 8, 9, 10

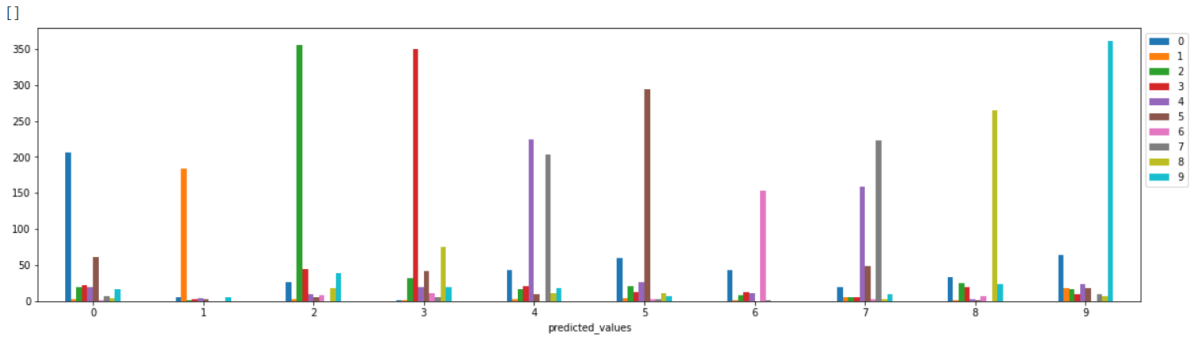
- 8020: Training = fold 1, 2, 3, 4, 5, 6, 7, 8

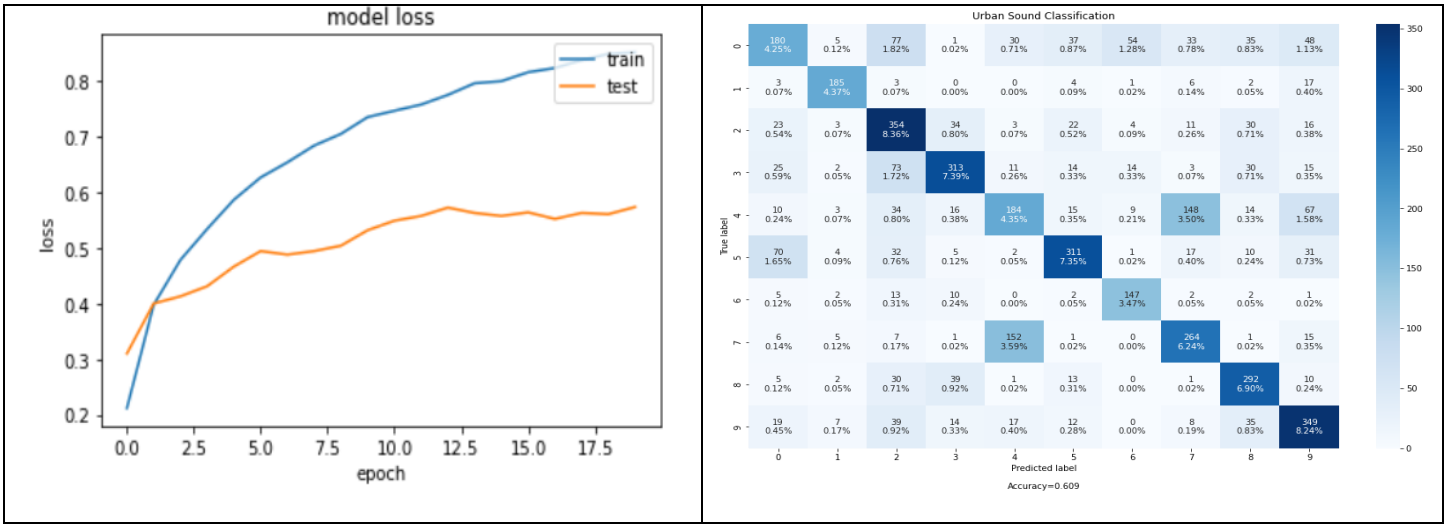
  Testing= fold 9, 10

As evident from the result in the present hyper-parameter setting, the maximum accuracy is achieved by Convolutional Neural Network model. The average model accuracy across 5 test folds and standard deviation obtained for each model is given below:

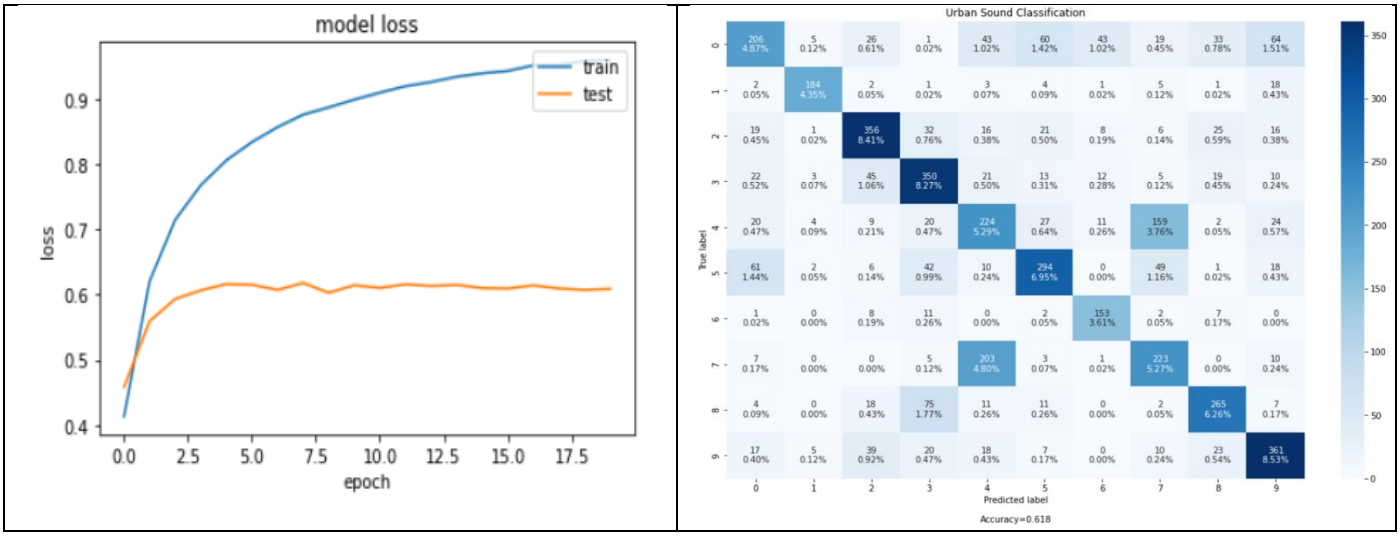
Split	NN	Features	Augmentation	Avg. Accuracy	Standard Deviation
5050	DNN	Only mfcc	No	50.77%	0.0295
			Yes	54.90%	0.0375
		all	No	56.34%	0.0262
			Yes	59.41%	0.0451
	CNN	Only mfcc	No	55.99%	0.0394
			Yes	58.09%	0.0527
		all	No	60.93%	0.0183
			Yes	63.38%	0.0478
8020	DNN	Only mfcc	No	56.14%	
			Yes	57.83%	
		all	No	62.98%	
			Yes	63.34%	
	CNN	Only mfcc	No	61.71%	
			Yes	65.34%	
		all	No	68.24%	
			Yes	68.91%	

Below is a snapshot of CNN model prediction, actual values and predicted values, label 4 corresponds to drilling and label 7 corresponds to jackhammer. These 2 are mostly misclassified in all the models.

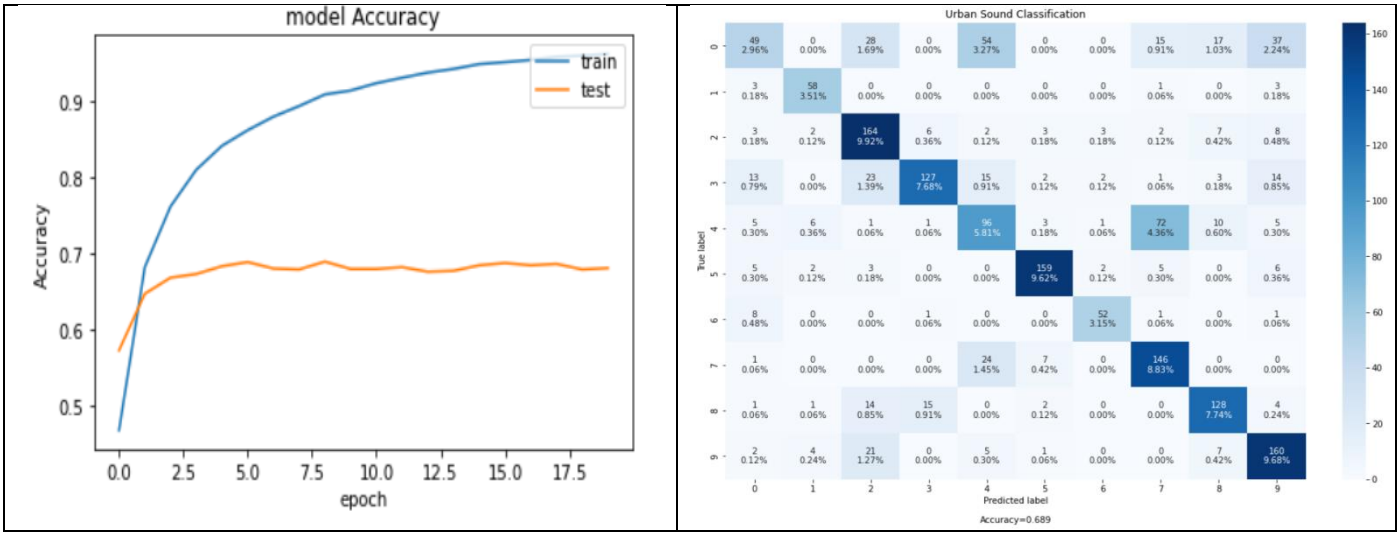




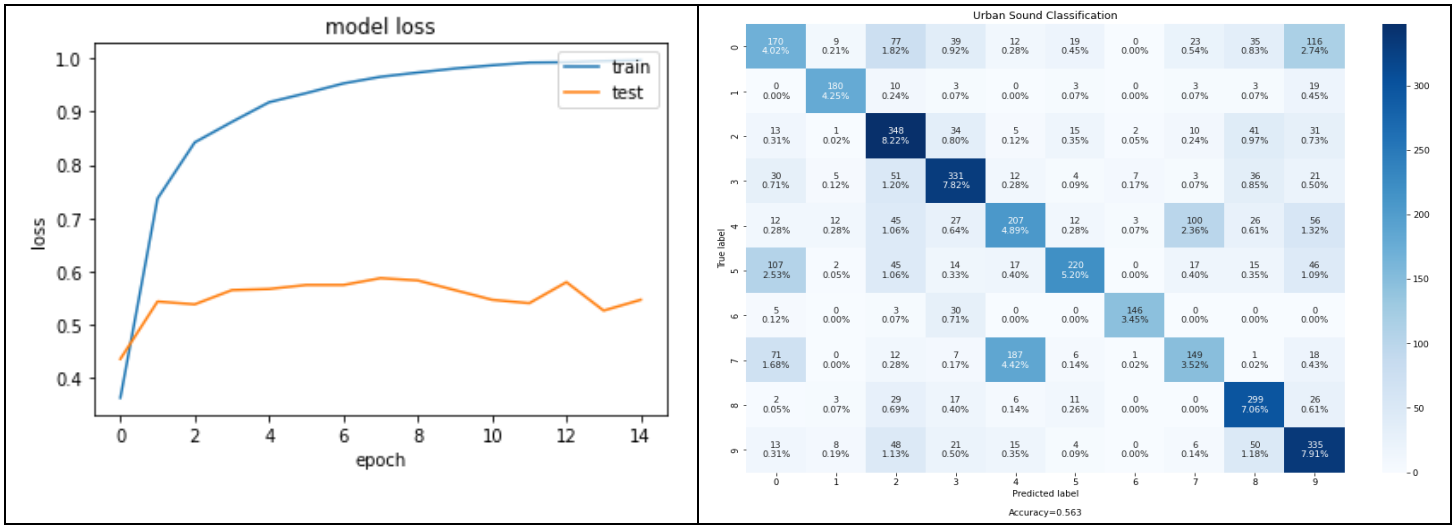
usc_5050_CNN_all model



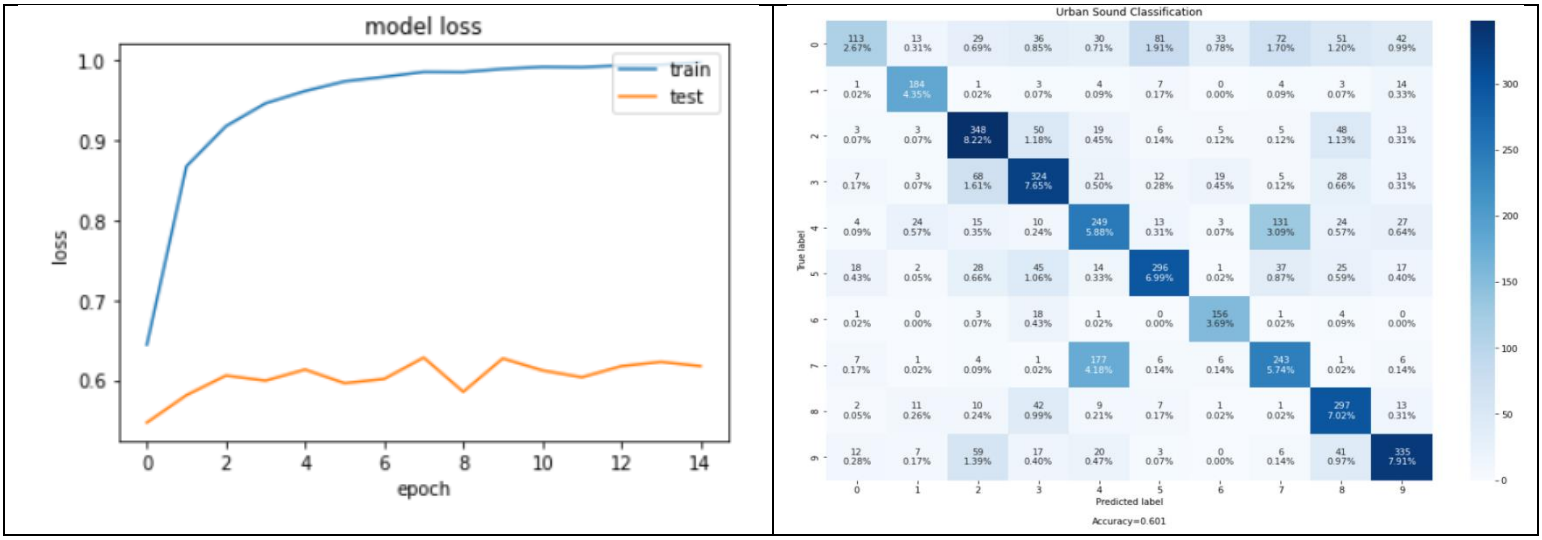
usc_5050_CNN_all_aug model



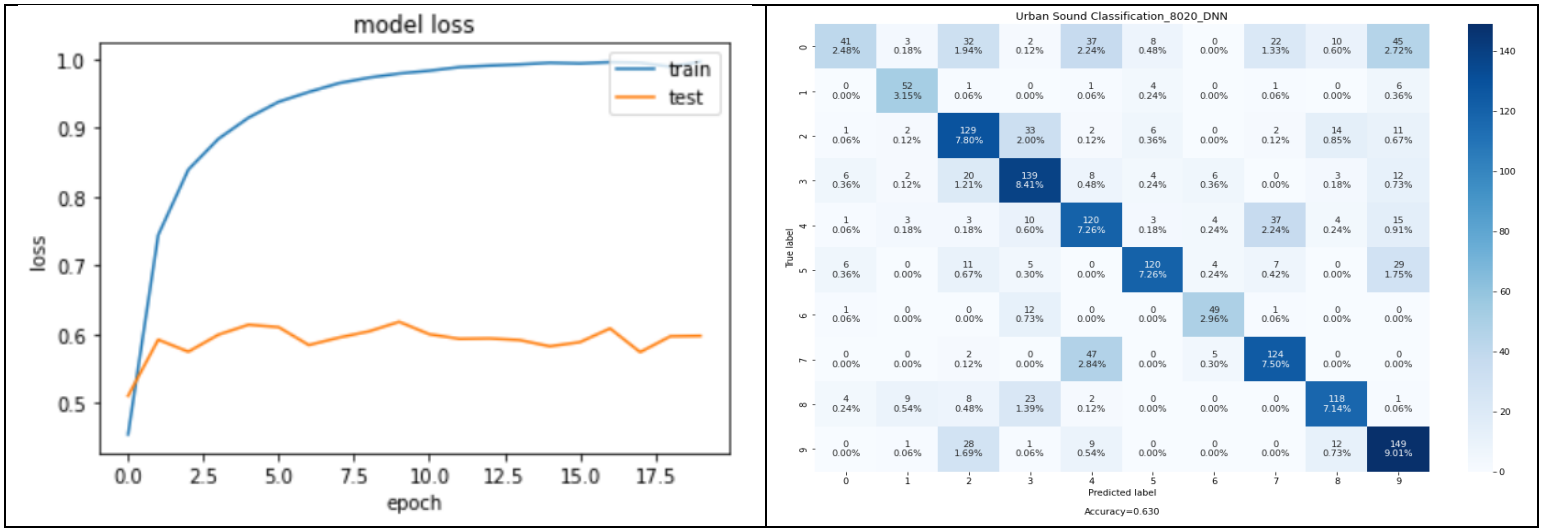
usc_8020_CNN_all_aug model



usc_5050_DNN_all model



usc_5050_DNN_all_aug model



usc_8020_DNN_all model

5. CONCLUSION & INFERENCE:

The experimentation proved that more features along with data augmentation gave the best results. More exploration on the feature selection needs to be done. We have implemented feature reduction technique PCA but it was ineffective and further decreased the performance so there is a room for improvement in feature engineering and feature selection process. With running more epochs, the data accuracy changed in just 2% for the convenience and time saving we have reduced epochs, but the results are quite good. There was a chance of multi-collinearity that may have reduced the overall accuracy of the models.

Finally, we showed that the improved performance stems from the combination of a deep, high-capacity model and an augmented training set: this combination outperformed both the proposed CNN without augmentation and the DNN model. Finally, we examined the influence of augmentation on the model's classification accuracy. We observed that the performance of the model for each sound class is influenced differently by each feature and the augmentation.

6.GitHub Link and File Description

https://github.com/rijinbaby/Urban-Sound-Classification

CNN Files https://github.com/rijinbaby/Urban-Sound-Classification/tree/CNN

DNN Files https://github.com/rijinbaby/Urban-Sound-Classification/tree/DNN

All the files follow this naming format:

"task_datasplit_NeuralNetwork_features_DataAugmentation"

for example: usc_5050_DNN_mfcc_aug = UrbanSoundClassification, using 5050 Data Split for training and testing (as suggested), Using Deep Neural Network, Only MFCC features (40), With data Augmentation to increase the size of dataset

usc_5050_DNN_mfcc = UrbanSoundClassification, using 5050 Data Split for training and testing, Using Deep Neural Network, Only MFCC features (40), No data Augmentation

usc_8020_CNN_all_aug = UrbanSoundClassification, using 8020 Data Split for training and testing, Using Convolutional Neural Network, MFCC and Other features, with data Augmentation to increase the size of dataset

REFERENCES:

[1] Z. Qawaqneh, A. A. Mallouh, B. D. Barkana, Deep neural network framework and transformed MFCCs for speaker's age and gender classification, Knowledge-Based Systems 115 (2017) 5{14.
[2] S. Poria, E. Cambria, A. Hussain, G.-B. Huang, Towards an intelligent framework for multimodal a_ective data analysis, Neural Networks 63 (2015) 104{116.
[3] C. Pedersen, J. Diederich, Accent classification using support vector machines, in: IEEE/ACIS International Conference on Computer and Information Science (ICIS), IEEE, 2007, pp. 444{449.
[4] P. Shegokar, P. Sircar, Continuous wavelet transform based speech emotion recognition, in: International Conference on Signal Processing and Communication Systems (ICSPCS), IEEE, 2016, pp. 1{8.
[5] Z. Yi, Foundations of implementing the competitive layer model by lotka {volterra recurrent neural networks, IEEE Transactions on Neural Networks 21 (3) (2010) 494{507.
[6] Z. Fu, G. Lu, K. M. Ting, D. Zhang, Optimizing cepstral features for audio classification, in: International Joint Conference on Artificial Intelligence, 2013.
[7] M. Espi, M. Fujimoto, K. Kinoshita, T. Nakatani, Exploiting Spectro temporal locality in deep learning based acoustic event detection, Journal on Audio, Speech, and Music Processing 2015 (1) (2015) 26.
[8] W. Lim, D. Jang, T. Lee, Speech emotion recognition using convolutional and recurrent neural networks, in: 2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), IEEE, 2016, pp. 1{4.
[9] Y. Leng, C. Sun, X. Xu, Q. Yuan, S. Xing, H. Wan, J. Wang, D. Li, Employing unlabelled data to improve the classification performance of SVM, and its application in audio event classification, Knowledge-Based Systems 98 (2016) 117{129.
[10] Y. Wu, H. Mao, Z. Yi, Audio classification using attention-augmented convolutional neural network, Knowledge-Based Systems 161 (2018) 90{100.
[11] J. H. Hansen, G. Liu, Unsupervised accent classification for deep data fusion of accent and language information, Speech Communication 78 (2016)19 { 33.
[12] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.
[13] J. Pons, X. Serra, randomly weighted cnns for (music) audio classification, in: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019.
[14] S. Dieleman, B. Schrauwen, End-to-end learning for music audio, in 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014, pp. 6964{6968.
[15] J. Pons, X. Serra, Designing efficient architectures for modeling temporal features with convolutional neural networks, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 2472{2476.
[16] J. Lee, J. Park, K. L. Kim, J. Nam, Samplecnn: End-to-end deep convolutional neural networks using very small _lters for music classification, Applied Sciences (Switzerland) 8 (1) (2018) 1{14.
[17] K. Choi, G. Fazekas, M. Sandler, Automatic tagging using deep convolutional neural networks, Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016 (2016) 805{811.
[18] R. Caruana, Multitask learning, Machine Learning 28 (1) (1997) 41{75.14

[19] J. Baxter, Learning internal representations, Proceedings of the 8th Annual Conference on Computational Learning Theory, COLT 1995 (1995) 311{320.
[20] B. Zhang, G. Essl, E. M. Provost, recognizing emotion from singing and speaking using shared models, in: International Conference on Active Computing and Intelligent Interaction (ACII), IEEE, 2015, pp. 139{145.
[21] H. Lee, P. Pham, Y. Largman, A. Y. Ng, Unsupervised feature learning for audio classification using convolutional deep belief networks, in: Advances in Neural Information Processing Systems, 2009, pp. 1096{1104.
[22] Y. Zeng, H. Mao, D. Peng, Z. Yi, Spectrogram based multi-task audio classification, Multimedia Tools and Applications 78 (3) (2019) 3705{3722.
[23] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: The International Conference on Learning Representations (ICLR), San Diego, USA, 2015.
[24] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097-1105.
[25] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770-778.
[26] J. Salamon and J. P. Bello.," Unsupervised Feature Learning for Urban Sound Classification", IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, Australia, April 2015.
[27] D. Stowell and M. D. Plumbley, "Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning," PeerJ, vol. 2, pp. e488, Jul. 2014.