A Seminar Report
on

# SOUND CLASSIFICATION USING DEEP LEARNING


*By*

**ABHISHEK PRAVIN KAKADE**

**Roll No: 305A039**

TE (Computer Engineering)

*Under the guidance of*

**PROF. A. KOTALWAR**




**Department of Computer Engineering**
**Sinhgad College of Engineering,**

**Vadgaon (Bk.), Pune-411041**

**Accredited by NAAC**

Affiliated to Savitribai Phule Pune University, Pune


**2021-2022**

# CERTIFICATE

This is certified that the Seminar Report entitled

## SOUND CLASSIFICATION USING DEEP LEARNING

*Submitted by*

**Abhishek Pravin Kakade**

Has successfully completed his Seminar Report and Presentation under the supervision of Prof. A. Kotalwar for the partial fulfillment of Third Year of Bachelor of Engineering, Computer Engineering of Savitribai Phule Pune University. This work has not been submitted earlier elsewhere for any degree.

**Prof. A. Kotalwar**
**Seminar Guide**

**Prof. M. P. Wankhade**
**Head of Department**

**Dr. S. D. Lokhande,**
**Principal,**
**Sinhgad College of Engineering, Pune**

# Acknowledgement

With due respect and gratitude, I take the opportunity to thank those who have helped me directly and indirectly. I convey my sincere thanks to Prof. M. P. Wankhade HoD Computer Department and Prof. A. Kotalwar for their help in selecting the seminar topic and support.

I thank to my seminar guide Prof. A. Kotalwar for his guidance, timely help and valuable suggestions without which this seminar would not have been possible. Her direction has always been encouraging as well as inspiring for me. Attempts have been made to minimize the errors in the report.

I would also like to express my appreciation and thanks to my family who knowingly or unknowingly have assisted and encourage me throughout my hard work.

# Abstract

Sound Classification is beneficial and very demanding technology these days as it has many applications in surveillance systems, classifying music from a huge music database based on their genres and also in medical treatment. Sound classification using Convolutional Neural Network (CNN) is strong technique where first the audio feature is selected from various audio features such as spectrograms, Mel-spectrograms, Mel-frequency Cepstral Coefficients, log-mel-spectrograms, etc. From these audio features, one which bests suits the problem is selected and made compatible for the input for CNN algorithm. The CNN algorithm is a deep learning algorithm which is capable of finding it's own features from the input. This algorithm doesn't need to explicitly generate features for the algorithm to detect. This algorithm has proven to be powerful tool for image classification tasks. Various research has been going on to classify sounds and deploying them in an effective way. This paper discusses on classifying urban sounds using the most powerful algorithm for image classification, Convolutional Neural Network and MFCCs as the audio feature.

# Contents

# List of Figures

# Chapter 1
# Introduction

Convolutional Neural Network is the most popular algorithm for image classification tasks. It uses filters which convolves with the input, does the matrix multiplication and finds the output. To fit more complex data, it uses ReLU activation function and softmax activation function at the end.

Multi-class classification refers to the problem where we find to which class the given data belongs. In this paper, we are classifying 7 urban sounds namely sound of air conditioner, children playing, drilling, engine idling, jack-hammer, siren and street music. These sounds are converted to a audio feature which closely resembles to how humans' vocal cords works, they are MFCCs (Mel-frequency Cepstral Coefficients).

The goal of urban sound classification is to classify urban sounds which has it's applications in surveillance systems, industry, etc. using Convolutional Neural Network (CNN). The dataset used is UrbanSound8K.

## 1.1 Motivation

As sound classification finds it's applications in many birds conservation projects (Brandes, 2008), it also has many applications in audio-based automated surveillance methods (Crocco, Cristani, Trucco, & Murino, 2016). The monitoring of human activities has never been as ubiquitous and massive as today, with thousands of sensors deployed in almost every urban area, industrial facility, and critical environment, and increasing rapidly in terms of both amount and scope. As a consequence, studies on automated surveillance have grown at a fast pace, with hundreds of algorithms embedded in various commercial systems. Also, Detection algorithm and sound classification methods are applied to medical telemonitoring (Vacher, Istrate, Besacier, Serignat, & Castelli, 2004). This motivated me to implement sound classification which classifies various urban sounds.

# Chapter 2
# Literature Survey

## 2.1 Fundamentals

Different studies have been conducted to classify sound using Convolutional Neural Network (CNN), Deep Artificial Neural Network (ANN) and Recurrent Neural Network (RNN) (Garg et al., 2021). Some of the research papers focused on classifying birds sound which find it's applications in many conservation projects[link] or as an early warning of fire in a forest and some focused on classifying music based on their genre to efficiently handle massive music databases for music indexing.

## 2.2 Related Work

Permana et al., developed a deep learning model which classifies birds sound as an early warning method of forest fires (Permana et al., 2021). In this paper, they have implemented a binary classification model using CNN algorithm which classifies birds sound into two categories viz. (1) under normal circumstances or conditions and (2) under threaten or panic condition. For the data pre-processing step, they converted the sound to spectrogram thus the sound has been changed into visual form or image form. As the problem is reduced down to an image classification problem, they fed this generated spectrogram to CNN algorithm. With these methods they were able to achieve a high accuracy of 96.45%. Inspired from the same idea of sound classification using deep learning techniques which has many applications in conservations projects, I implemented urban sound classification model.

Mushtaq and Su (Mushtaq & Su, 2020) created a proficient AI based plan for Urban Sound Classification. In this paper they used sounds such as air_conditioner , drilling, engine_idling , jackhammer, car_horn, children_playing , dog_bark , boring, gun_shot, alarm and street_music . The prime objective of our project was to assess whether the CNN (Convolution Neural Network) can be effectively applied to Urban Sound Classification. They made a comparison of various machine learning algorithms such as CNN, Recurrent Neural Network(RNN), K-Nearest Neighbours(KNN) and DeepNN. In this paper they converted the sound to Mel-Frequency Cepstral Coefficients (MFCC). MFCCs are based on the known variation of the human ears critical bandwidths with frequency. Conducted experiments showed that CNN outperformed all other algorithms with accuracy of 86.44%. Thus CNN proved to be the better

algorithm for urban sound classification.

R. Thiruvengatanadhan (Thiruvengatanadhan, 2018) implemented music classification using Support Vector Machines (SVM) which helps in efficiently handling huge music databases by classifying music based on their genres and sub genres. They extracted acoustic features from sound namely Mel-Frequency Cepstral Coefficients (MFCC). The mel frequency cepstrum has proven to be highly effective in recognizing the structure of music signals and in modelling the subjective pitch and frequency content of audio signals.

# Chapter 3
# Methodology

## 3.1 Task Definition

The goal of the project is to classify urban sounds which we hear more frequently in urban cities. The sound clips are first converted into MFCCs which are then fed to CNN algorithm as gray scale images. The pipeline of this task is given as follows,

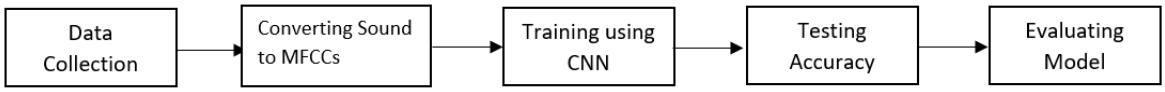


Figure 3.1: A general flowchart of urban sound classification based on CNN method.

### 3.1.1 Dataset

For the analysis of the problem statement, we have use UrbanSound8k dataset which comprehends 10 day to day sounds taken from urban territory, namely, air conditioner, siren, children playing, barking dogs, drilling, engine idling, gun shot, jack-hammer, alarm and music from the streets. It contains 8732 marked sound example passages of about 4 seconds stretch or less, 9.7 hours in total. All sound files are in .wav format.

The dataset is distributed in 10 folders. It does not uniformly distributed in these folders. Also, each sound is situated in any of the 10 folders. To locate each and every audio, the dataset comes with the metadata file. The metadata file is a Comma Separated Values(csv) file.
Slice_file_name: It is the name of the sound file. It has the following arguments:
(fsID)-(classID)- (occurrenceID)(sliceID).wav, where:
1.1. fsID: The Freedsound ID of the profile from which this excerpt
1.2. classID: It is the numeric representation to all the class names
1.3. ccurrenceID: a numerical differentiator to recognize various events of the sound inside the first chronicle.
1.4. sliceID: a numeric identifier to recognize various excerpts taken from a similar event. start: It is the beginning time of the sound clip from the free stable chronicle. end: It signifies the

time where the sound clip ends in the first Free stable chronicle. salience: It is the abstract significant rating of the sound wherein 1 = frontal area and 2 = foundation.

fold: It is the fold number in which the corresponding sound clip has been placed.

classID: It is the numeric representation to all the class names which are as followed:

- 0: Air_Conditioner
- 1: Car_Horn
- 2: Children_Playing
- 3: Dog_Bark
- 4: Boring
- 5: Engine_Idling
- 6: Gun_Shot
- 7: Jack_Hammer
- 8: Alarm and
- 9: Street_Music

Class: it signifies the type of sound clip i.e., Children_playing, air_conditioner, dog_barking, siren,drilling, car_horn ,jack_hammer, gun_shot, street_music ,engine_idling.

### 3.1.2 Visualizing Dataset

The original dataset is not consistent meaning number of each type of sound is not the same. Also, the time duration of each type sound was different. The count of sounds for the class car_horn was only 203 and for the class gun_shot was only 16 sounds. This can cause bias problems if the less number of type of sound is used with huge number of other types. To address this issue, I've used 7 classes from original dataset. The classes used in the project are air_conditioner, children_playing, engine_idling, jackhammer, street_music, siren and drilling. The sampling rate of each sound clip varies. For the purpose of project and consistency of dataset, I've used only those clips which have 22.05K sampling rate and length of clip is 4 seconds.

To convert the sound into array of numbers, I've used Librosa library which is one of the python library for audio and music processing. After converting the sound into array each array of a sound clip will contain 88,200 integers. Here, I've used 5600 examples which is splitted

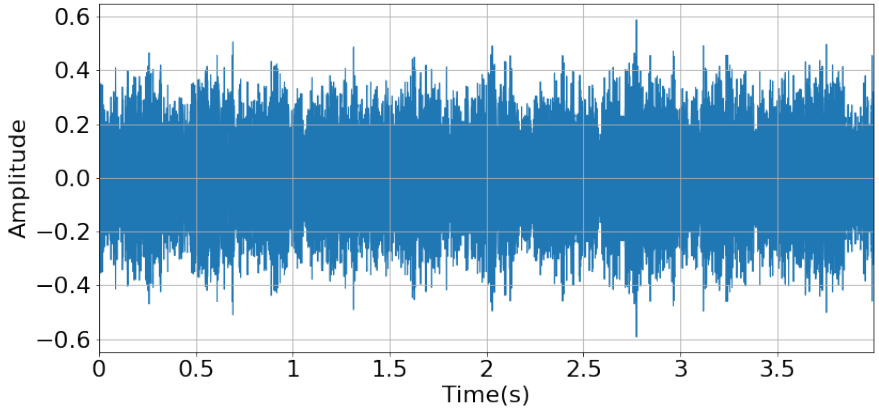


Figure 3.2: Air Conditioner sound clip

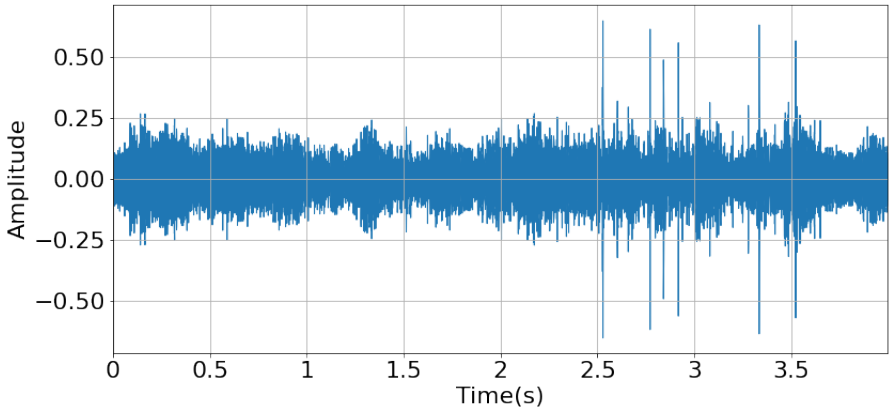


Figure 3.3: Children Playing sound clip

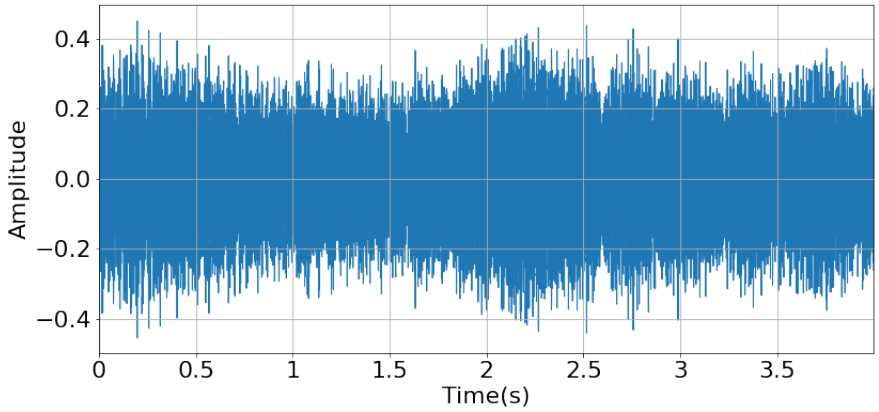


Figure 3.4: Drilling sound clip

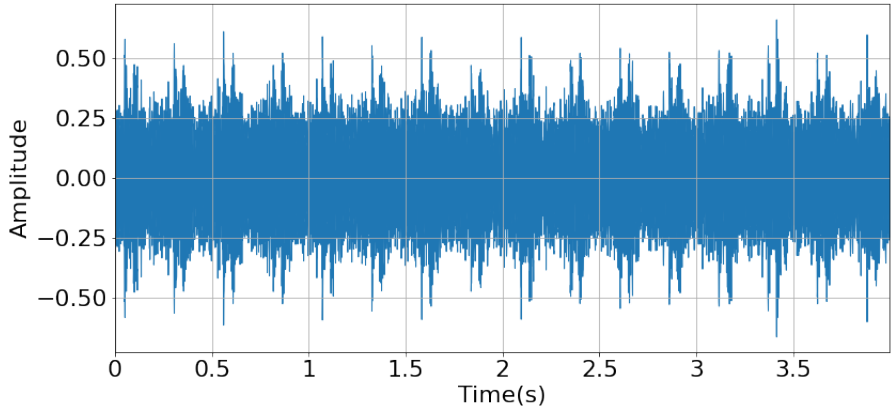


Figure 3.5: Engine Idling sound clip

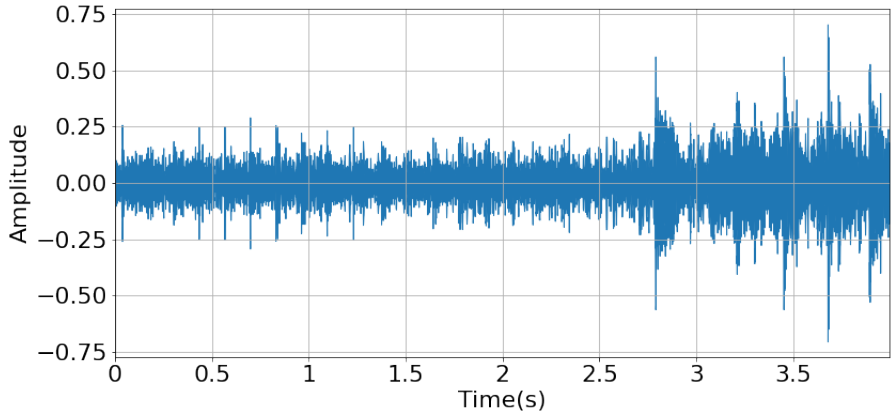


Figure 3.6: Jackhammer sound clip

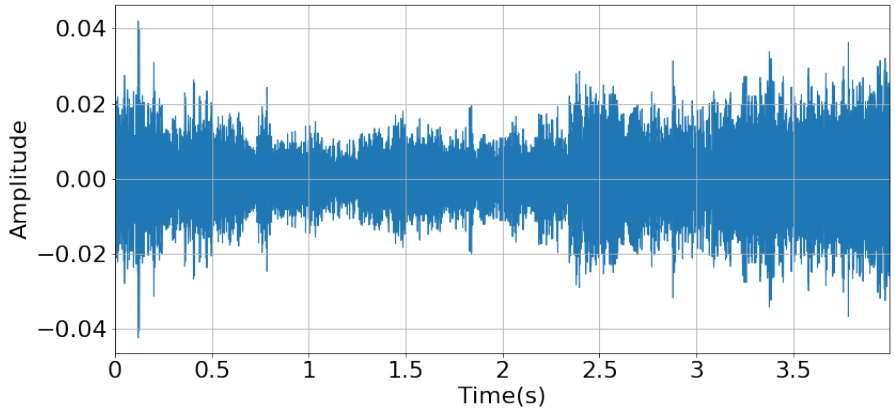


Figure 3.7: Siren sound clip

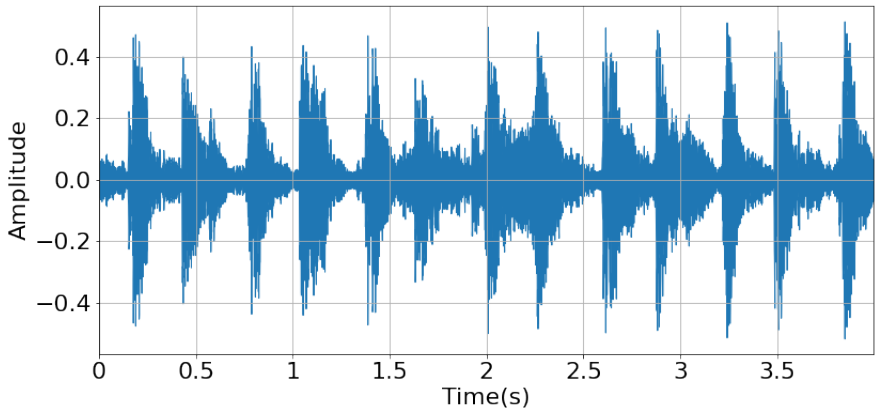


Figure 3.8: Street Music sound clip

into 3 categories namely training data, validation data and evaluation data. Total 4000 examples were used out of 5600 to train the data and 800 examples for each validating and evaluation purpose.

### 3.1.3 Converting Audio Data to Mel-Frequency Cepstral Coefficients (MFCCs)

Mel Frequency Cepstral Coefficients (MFCCs) are short-term spectral based and dominant features and are widely used in the area of audio and speech processing. The mel frequency cepstrum has proven to be highly effective in recognizing the structure of music signals and in modeling the subjective pitch and frequency content of audio signals.

MFCCs are based on the known variation of the human ears critical bandwidths with frequency. The filters are spaced linearly at low frequencies and logarithmically at high frequencies to capture the phonetically important characteristics of speech and audio. Magnitude spectrum is computed for each of these frames using Fast Fourier Transform (FFT) and converted into a set of mel scale filter bank outputs.

The human ear resolves frequencies non-linearly across the audio spectrum and empirical evidence suggests that designing a front-end to operate in a similar non-linear manner improves the performance. Therefore MFCCs are found to be better feature of audio for classification purpose. As Convolutional Neural Network(CNN) needs image as input data, all sound clips are converted to MFCCs. The shape of MFCCs are (number of MFCC coefficients, number of time frames). The time frames can be calculated as

$$number\_of\_time\_frames = \frac{time * sampling\_rate}{hop\_length}$$

For the purpose of project, only first 13 MFCCs coefficients were taken into consideration because the further coefficients doesn't contain much important information about the signal therefore, they can be omitted. Mel frequency to implement this filter bank, the window of audio data is transformed using a Fourier transform and the magnitude is taken. The magnitude coefficients are then binned by correlating them with each triangular filter. Here, binning means that each FFT magnitude coefficient is multiplied by the corresponding filter gain and the results are accumulated.

Thus, each bin holds a weighted sum representing the spectral magnitude in that filter bank channel.Logarithm is then applied to the filter bank outputs. Discrete Cosine Transformation (DCT) is applied to obtain the MFCCs. Since the mel spectrum coefficients are real numbers, they are converted to the time domain using the DCT. The cepstral representation of the speech spectrum provides a good representation of the local spectral properties of the signal for the given frame analysis. Typically, the first 13 MFCCs are used as features.

The MFCCs for air conditioner, children playing, drilling, engine idling, jackhammer, siren and street music used in the project are displayed here.

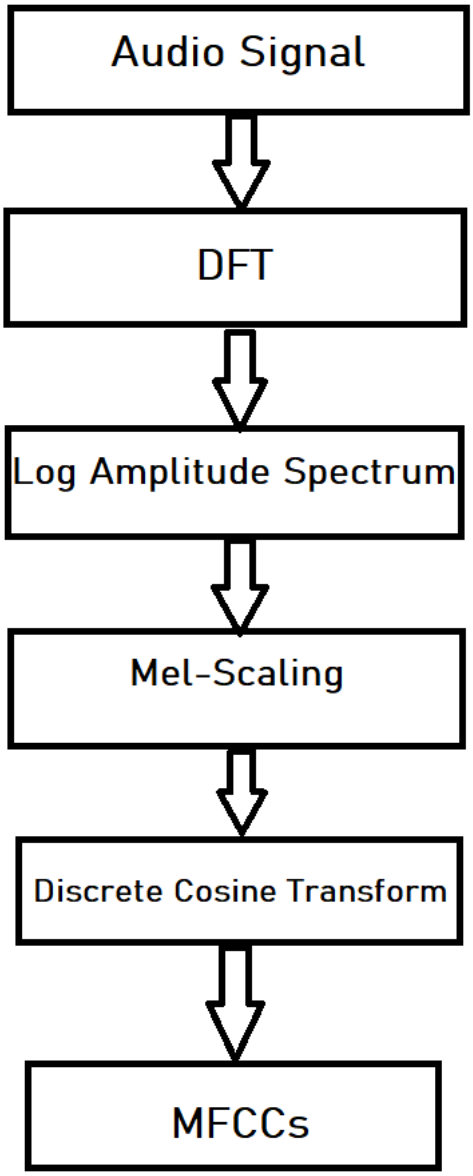


Figure 3.9: Procedure for extracting Mel-Frequency Cepstral Coefficients (MFCCs).

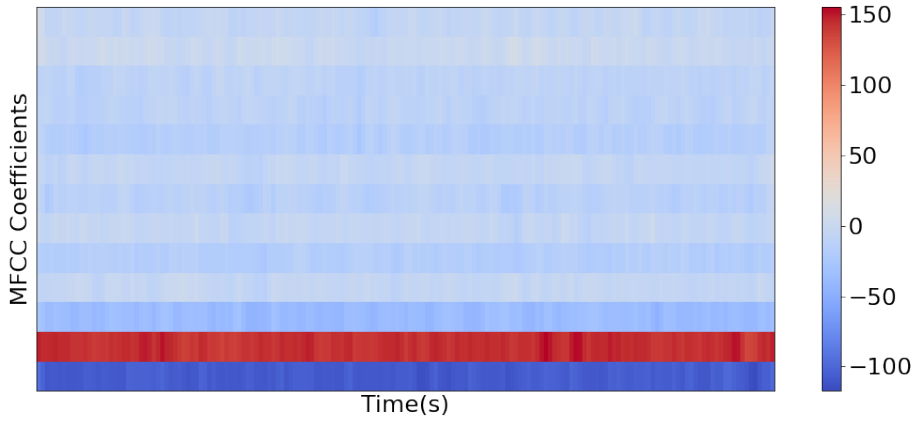


Figure 3.10: Air Conditioner MFCCs Plot

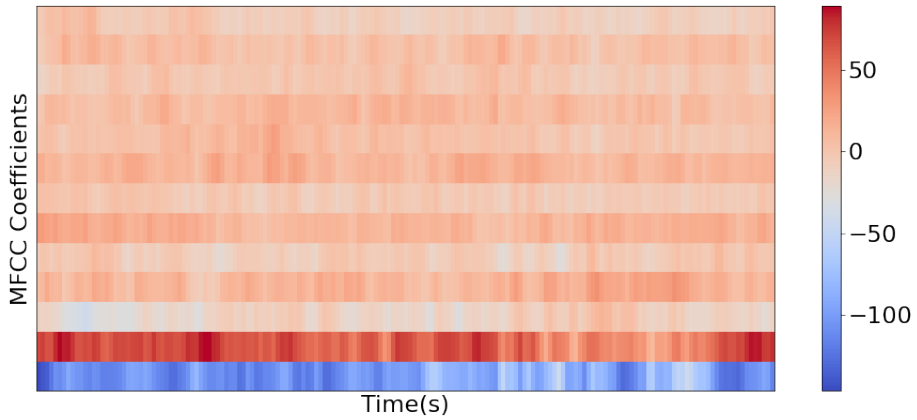


Figure 3.11: Children Playing MFCCs Plot

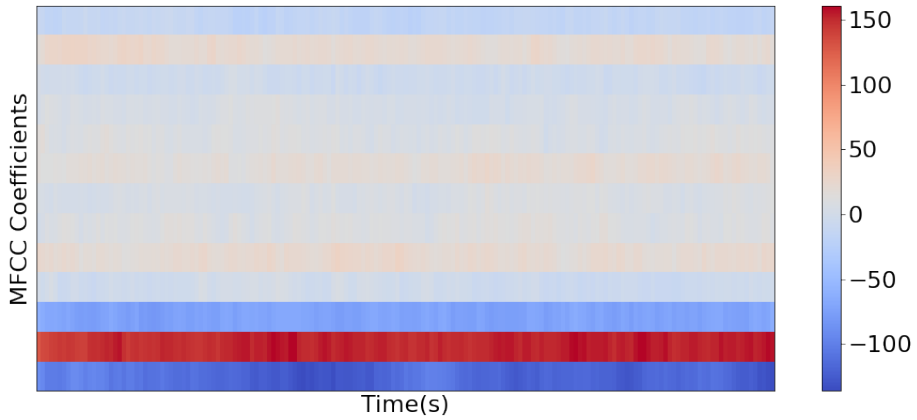


Figure 3.12: Drilling MFCCs Plot

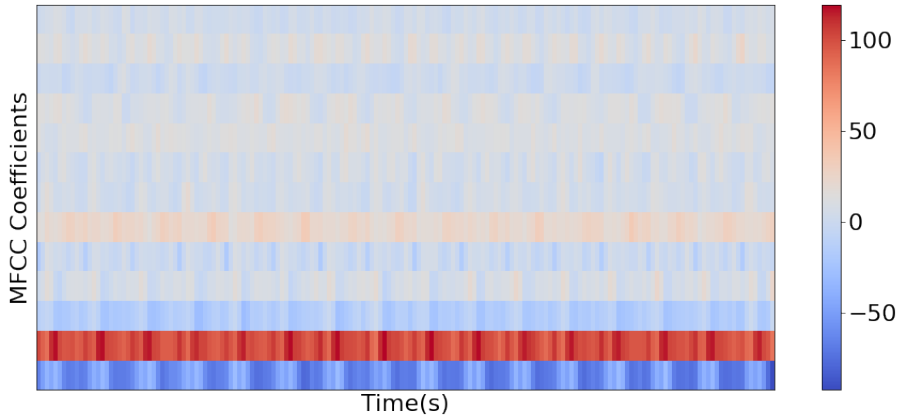


Figure 3.13: Engine Idling MFCCs Plot

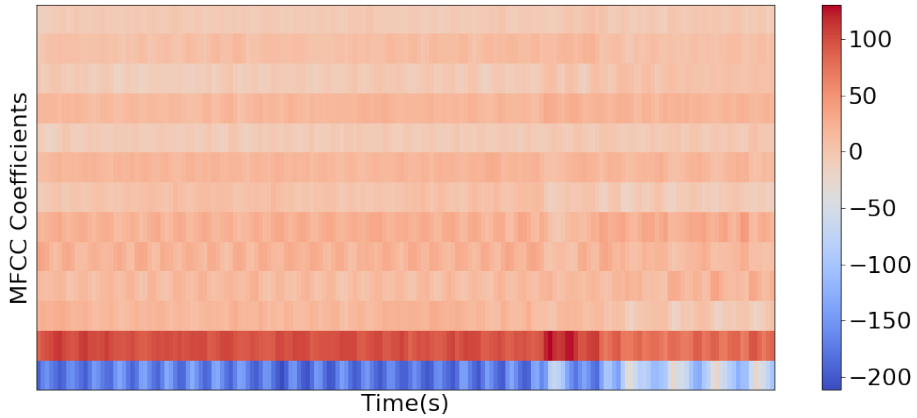


Figure 3.14: Jackhammer MFCCs Plot

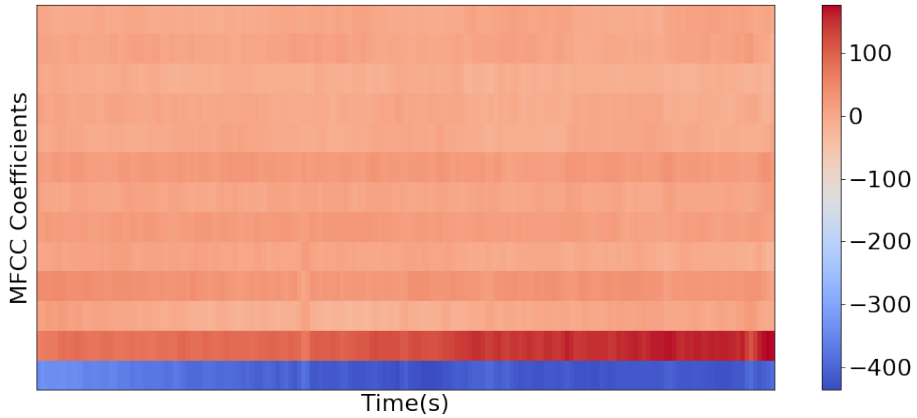


Figure 3.15: Siren sound clip MFCCs Plot

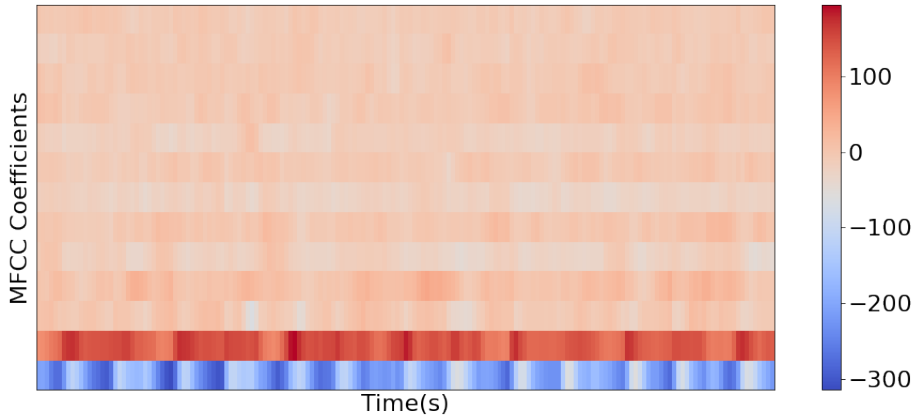


Figure 3.16: Street Music MFCCs Plot

## 3.2 Algorithm Definition

A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area. Conv2D is a layer from keras which is used to convolve images with RGB channels.

Conv2D is a layer from keras which is used to convolve images with RGB channels. But as librosa returns a MFCCs of shape (173, 13), I've added a new axis using python numpy library which changes the shape to (173, 13, 1). By adding a new axis with values of 1, we assume that the MFCCs are grayscale images as grayscale images have only 1 channel. Convolutional Neural Networks contains a input layer which takes images as an input. Image is nothing but a multidimentional array of numbers. After the input layer, CNN contains one or more hidden layers which intuitively finds patterns in images. Then in the last layer we unroll the dimensions of convolutional layer into single dimensional array. This array is fully connected to 7 neurons which is our number of output classes. For the output, I used a softmax function which outputs probabilities of each of the class.

### 3.2.1 Input to CNN

As Librosa returns shape of MFCCs as $(number\_of\_time\_frames, MFCC\_coefficients)$ and for this project the shape is $(173, 13)$ it is not compatible for the CNN algorithm input as it needs an image with channels as input. Therefore to make this shape compatible I've added a new axis using numpy which changes the shape to $(173, 13, 1)$. By doing this we not only ensured that the input is compatible for the algorithm but also we're predicting that the input image is a gray scale image. Further, there are 4000 training examples and a batch size of 32. Therefore before training, we stack up the input with training examples of 32. The final shape of input becomes $(32, 173, 13, 1)$.

### 3.2.2 Convolutional Neural Network (CNN)

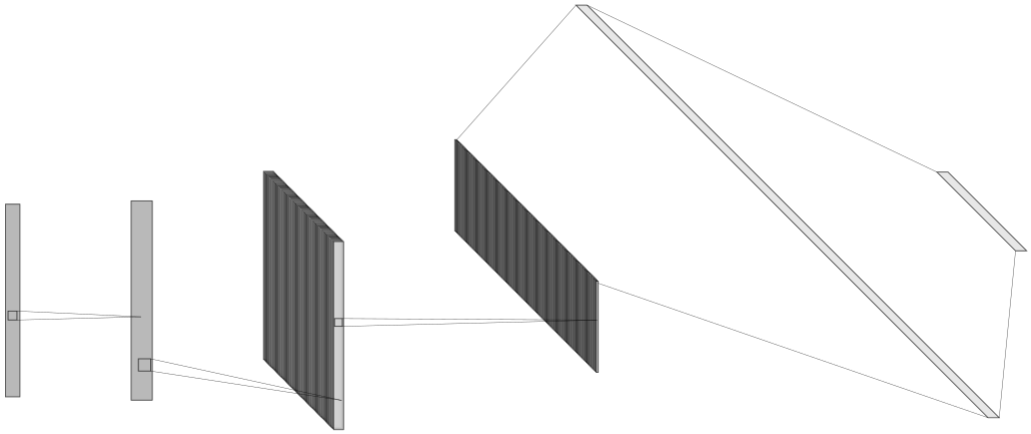

Figure 3.17: The CNN architecture for Training Data.

Keras is high-level API of Tensorflow which is open source library from Google for machine learning and artificial intelligence. In this project I've used keras' sequential API which stacks up network of layers one by one. The first layer is input layer which is of shape $(32, 173, 13, 1)$. The MFCC coefficients obtained from the sound clips are processed and are feed to this layer. The next layer is Zero Padding layer which pads the input shape with zeros with padding size of 3. This layer determines that the corners of the original shape should stay as it is while convolution as we can obtain much information from the corners as well.

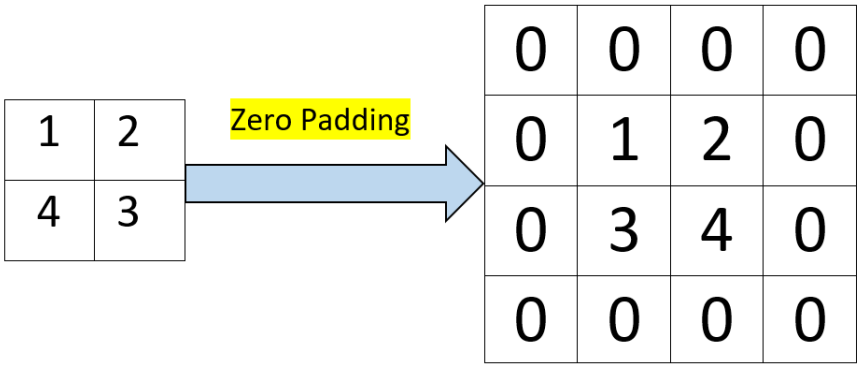


Figure 3.18: Zero Padding Illustration

After zero padding, the shape of input becomes $(m, 179, 19, 1)$,
where $m = number\_of\_training\_examples\_in\_a\_batch$.

In the convolutional layer, the filters/kernels slides through the input by number of strides specified within and every time it finds matrix multiplication operation between the kernel and the input matrix. Every time the kernel shifts from left to right in a row from top to bottom of the input matrix. The output of the matrix can be determined by the formula,

$$Output\_shape\_after\_convolution\_operation = \left(\frac{n+2p-f}{s}+1\right) \times \left(\frac{n+2p-f}{s}+1\right)$$

where,

$n = input\ shape\ (\ height/width\ of\ input\ )$

$p = padding\ size$

$f = filter\ size$

$s = number\ of\ strides$

This can be visualized as follows,

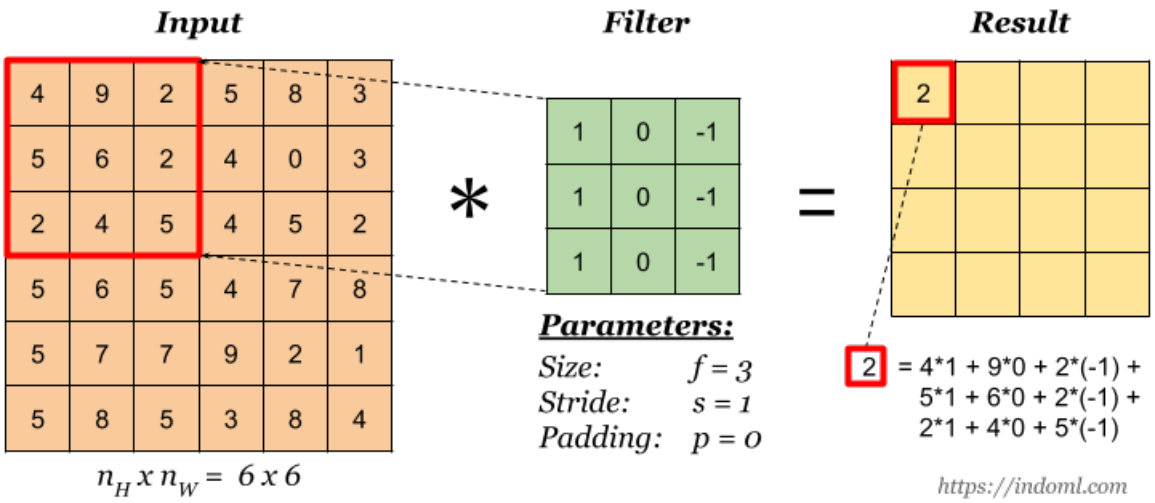


Figure 3.19: Single Convolution Operation

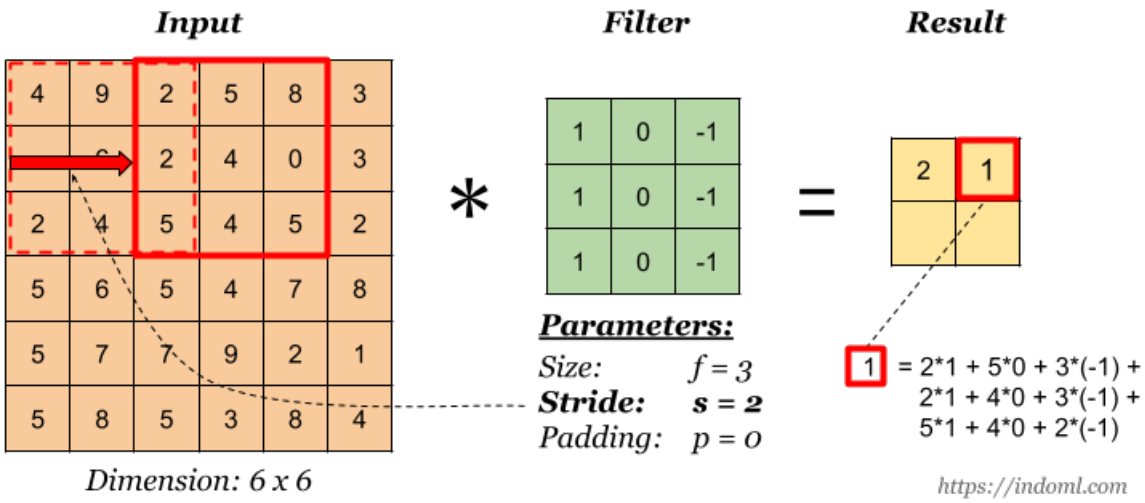


Figure 3.20: Stride

After convolution layer comes the activation layer. The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

$$R(x) = max(0, x)$$

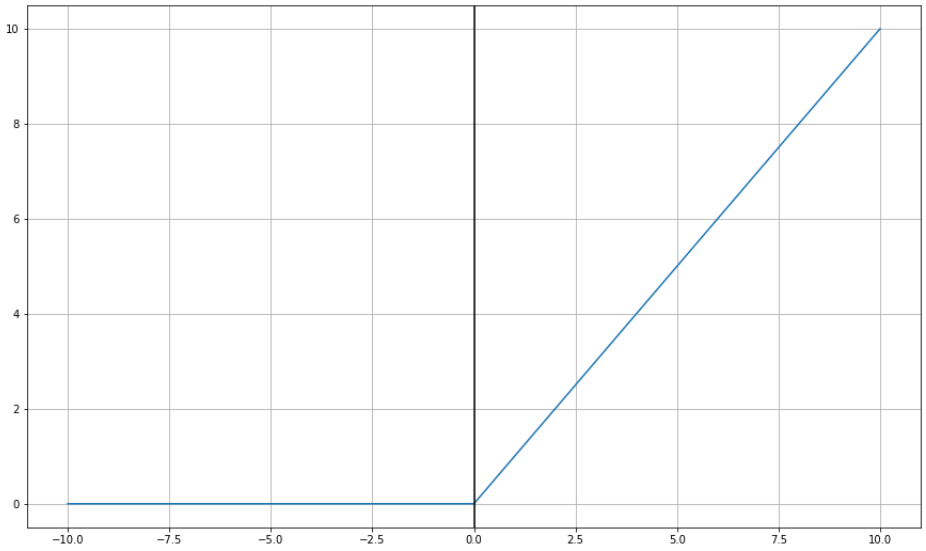

Figure 3.21: ReLU Activation Function

Pooling layers provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map. Convolutional layers in a convolutional neural network systematically apply learned filters to input images in order to create feature maps that summarize the presence of those features in the input. Convolutional layers prove very effective, and stacking convolutional layers in deep models allows layers close to the input to learn low-level features (e.g. lines) and layers deeper in the model to learn high-order or more abstract features, like shapes or specific objects. A limitation of the feature map output of convolutional layers is that they record the precise position of features in the input. This means that small movements in the position of the feature in the input image will result in a different feature map.

This can happen with re-cropping, rotation, shifting, and other minor changes to the input image. A common approach to addressing this problem from signal processing is called down sampling. This is where a lower resolution version of an input signal is created that still contains the large or important structural elements, without the fine detail that may not be as useful to the task. Maximum pooling, or max pooling, is a pooling operation that calculates the maximum, or largest, value in each patch of each feature map. The results are down sampled or pooled feature maps that highlight the most present feature in the patch, not the average presence of the feature in the case of average pooling. This has been found to work better in practive than average pooling for computer vision tasks like image classification. This has been found to work better in practice than average pooling for computer vision tasks like image classification.

Before the convolutional operation, the shape of the input was $(179, 19, 1)$. But after convolution layer with 64 filters, the shape becomes $(169, 9, 64)$. The ReLU function doesn't change the shape of input layer. According to the model, there is another convolution layer with 128 filters which further changes the output to shape of $(82, 2, 128)$. The max-pooling layer scales down the input from the $2^{nd}$ convolution layer. So the shape becomes $(41, 1, 128)$. After this layer, the input is flattened into a single dimentional array with 5248 values. This layer is densely connected to 7 output neurons of using a softmax activation function. The model summary can be seen below,

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 zero_padding2d (ZeroPadding  (None, 179, 19, 1)       0
 2D)

 conv2d (Conv2D)             (None, 169, 9, 64)        7808

 batch_normalization (BatchN  (None, 169, 9, 64)       256
 ormalization)

 re_lu (ReLU)                (None, 169, 9, 64)        0

 conv2d_1 (Conv2D)           (None, 82, 2, 128)        401536

 batch_normalization_1 (Batc  (None, 82, 2, 128)       512
 hNormalization)

 re_lu_1 (ReLU)              (None, 82, 2, 128)        0

 max_pooling2d (MaxPooling2D  (None, 41, 1, 128)       0
 )

 flatten (Flatten)           (None, 5248)              0

 dense (Dense)               (None, 7)                 36743

=================================================================
Total params: 446,855
Trainable params: 446,471
Non-trainable params: 384
_________________________________________________________________
```

Figure 3.22: Model Summary

Formula for softmax activation function is given below,

$$softmax\ (\ x\ )_{\ i}\ =\ \frac{exp\ (\ x_i\ )}{\sum_j exp\ (\ x_j\ )}$$

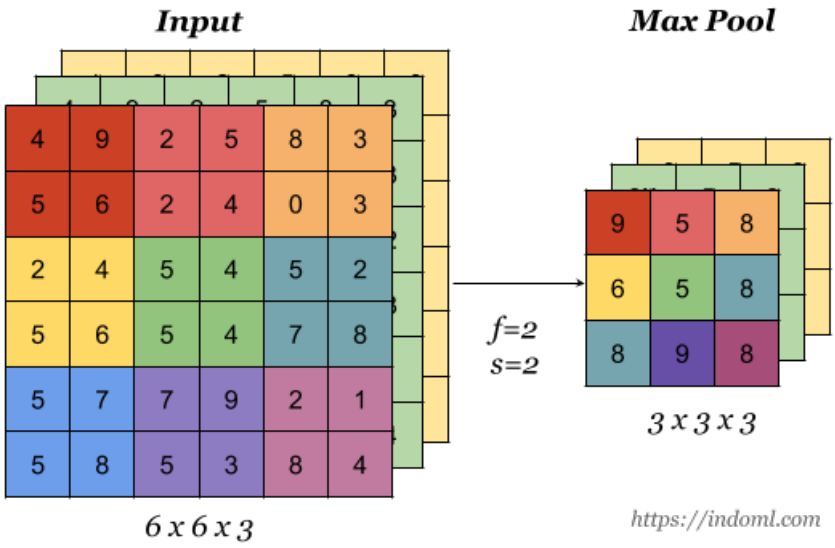


Figure 3.23: Max Pooling Illustration

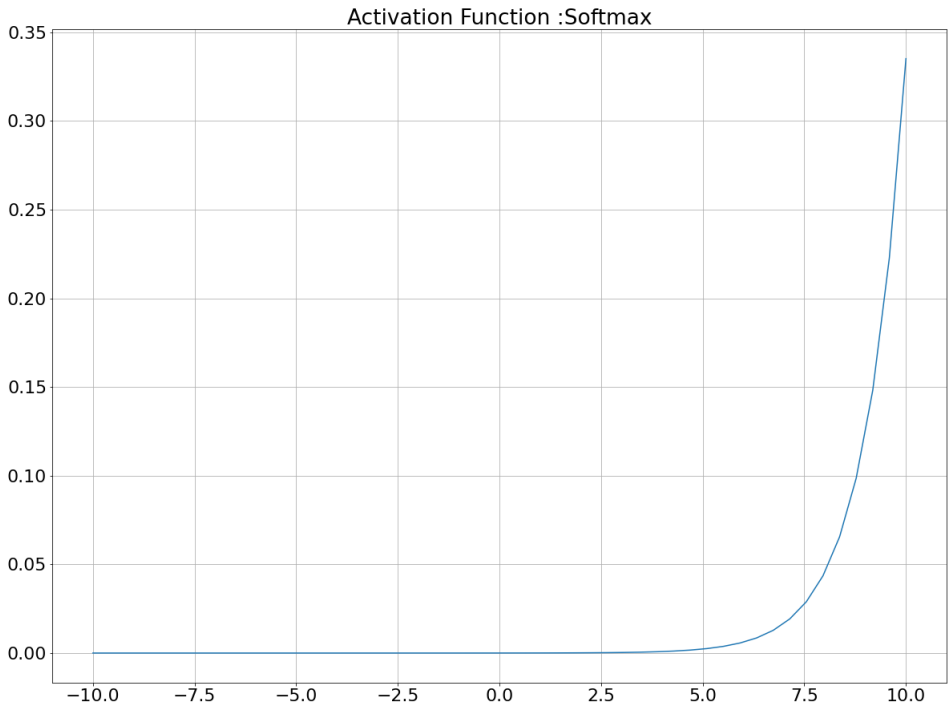


Figure 3.24: Softmax Activation Function

### 3.2.3 Cost Function

The loss function used is categorical crossentropy from tensorflow library. The formula for the calculating the loss is as follows,

$$L_i = -log\left(\hat{y}_{i,k}\right)$$

where,

$L_i = sample\ loss\ value$

$i = i^{th}\ example\ in\ set$

$k = target\ label\ index,\ index\ of\ correct\ class\ probability$

$\hat{y} = predicted\ values$

# Chapter 4
# Results

## 4.1 Evaluation Metric

The evaluation metric used is accuracy because as all the sound clips were mostly equally divided in train, validation and evaluation sets. The formula for accuracy is as follows,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where,

$TP = True\ Positives$

$TN = True\ Negatives$

$FP = False\ Positives$

$FN = False\ Negatives$

The validation accuracy of model was determined to be $87.37\%$ and training accuracy is $97.93\%$. This clearly shows that my model has regularization problem as it is over-fitting the training data but shows a high validation error of $10.56\%$. The model was evaluated on 800 validation examples. After training for 40 epochs with a batch size of 32 examples, the model gave a training loss of $0.0582$ and validation loss of $0.6482$.

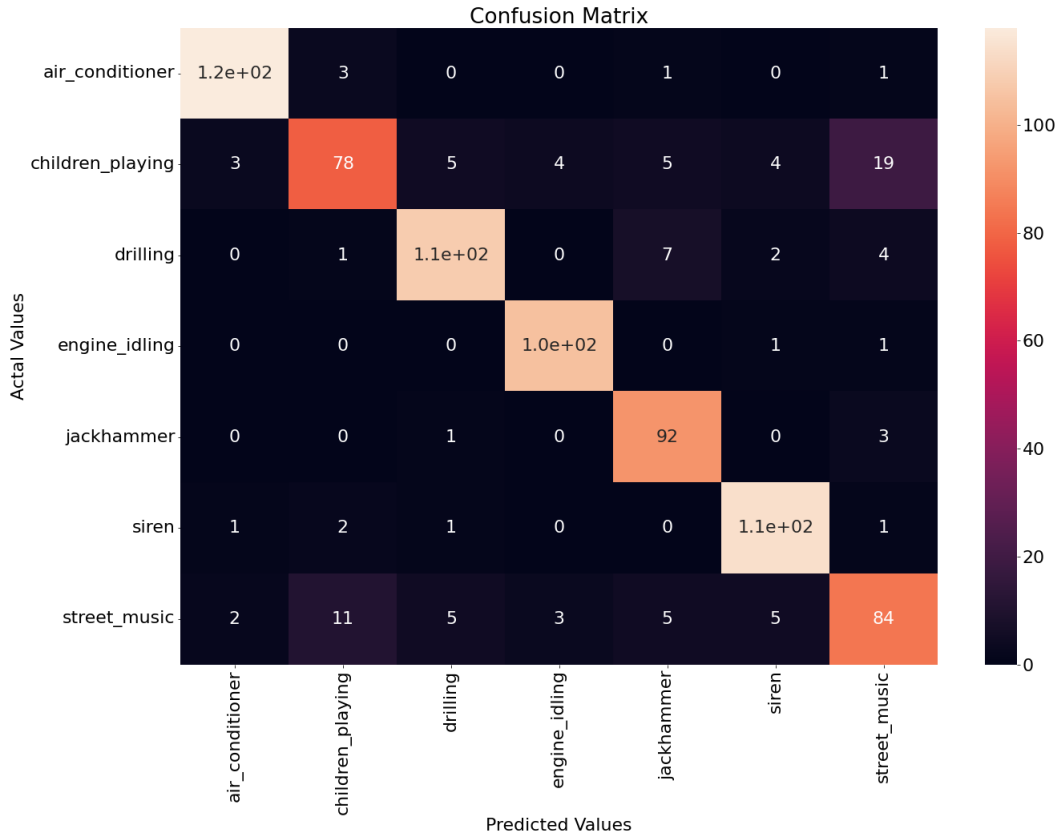


Figure 4.1: Confusion Matrix

In the graph of training accuracy, there is some noise which is because of batches. Since in TensorFlow, the data is fed in batches and for this project 32 examples were present in a single batch. While training, single step of gradient descent is calculated on this single batch size therefore, every time a new batch appears, the model founds it to be a completely new training example. This is the reason for noise in training graph.

The graph of loss and accuracy for training and validation data is shown below,

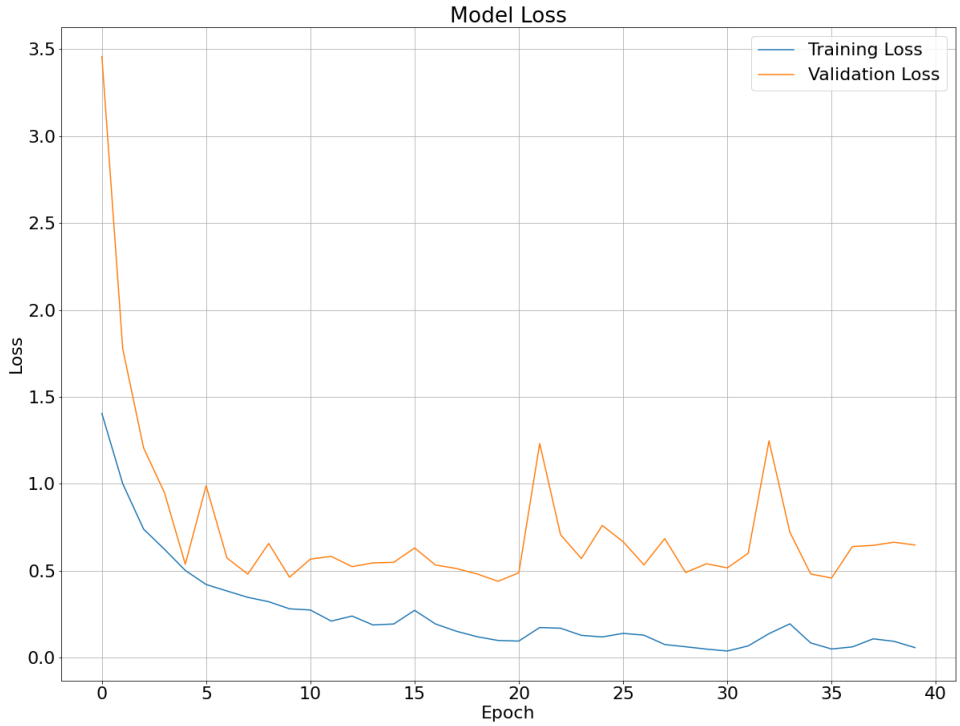


Figure 4.2: Model Loss

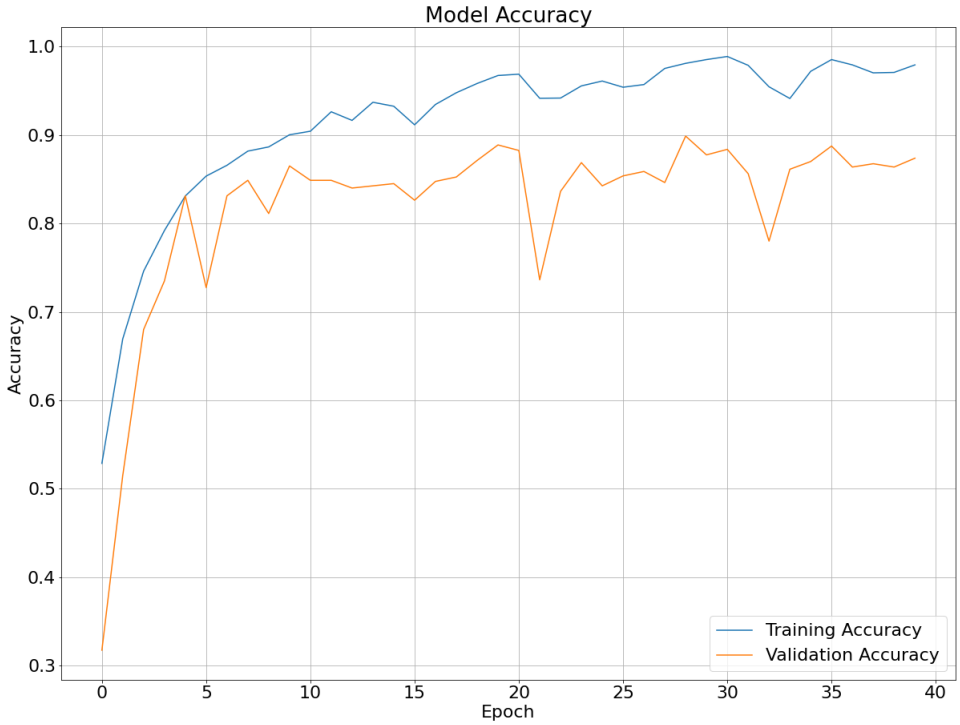


Figure 4.3: Model Accuracy

Classification Report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| air_conditioner | 0.95 | 0.96 | 0.96 | 123 |
| children_playing | 0.82 | 0.66 | 0.73 | 118 |
| drilling | 0.90 | 0.89 | 0.89 | 122 |
| engine_idling | 0.94 | 0.98 | 0.96 | 107 |
| jackhammer | 0.84 | 0.96 | 0.89 | 96 |
| siren | 0.90 | 0.96 | 0.93 | 119 |
| street_music | 0.74 | 0.73 | 0.74 | 115 |
| accuracy | | | 0.87 | 800 |
| macro avg | 0.87 | 0.88 | 0.87 | 800 |
| weighted avg | 0.87 | 0.87 | 0.87 | 800 |

Figure 4.4: Classification Report

# Chapter 5
# Discussion

From the results we can clearly see that urban sounds can be successfully be classified using Convolutional Neural Network (CNN) algorithm. The audio feature Mel-Frequency Cepstral Coefficients (MFCCs) has proven that for urban sounds it can be a better option over other audio features.

Although we can see from the confusion matrix that the model is classifying little poorly on children playing audio and street music audio, this can further be improved using data augmentation techniques. The CNN algorithm is very useful for determining the features from the MFCCs plot which is also able to classify sounds which are nearly same for human ears to hear like children playing or street music.

# Chapter 6
# Future Work

From the results, we can clearly see the difference between training accuracy and validation accuracy is quite large. This means that the algorithm is fitting the training data very well but not on validation data. This problem can be reduced using regularization. Regularization will help to reduce the difference between training accuracy and testing accuracy.

Another way we can improvise the accuracy is by using Resnets. Resnets are very deep neural networks which also contains skip connections. They are efficient in finding accurate features from the data in very initial layers.

Also, if we want to improve accuracy and reduce the computational power required to find the output, we can use MobileNet architecture. This architecture was invented so that also mobile phones can run neural network computation.

Data augmentation can also be done so as to increase the number of training examples available. As deep learning algorithms often increase accuracy if more data is available this could increase accuracy.

# Chapter 7
# Conclusion

Sound classification using deep learning is very useful method and have various applications in fields such as surveillance systems, medical application and urban sound classification in industry. By improving these methods, sound classification has it's applications in bird sound classification where classifying birds sound can help forest officers identify forest fires where the forests are spread in thousands of square kilometres. Still there are many challenges in this field like other noise in the recordings and less data, but we still get a better accuracy without using data augmentation.

# References

Brandes, T. S. (2008). Automated sound recording and analysis techniques for bird surveys and conservation. *Bird Conservation International*, *18*(S1), S163–S173.

Crocco, M., Cristani, M., Trucco, A., & Murino, V. (2016). Audio surveillance: A systematic review. *ACM Computing Surveys (CSUR)*, *48*(4), 1–46.

Garg, S., Sehga, T., Jain, A., Garg, Y., Nagrath, P., & Jain, R. (2021). Urban sound classification using convolutional neural network model. In *Iop conference series: Materials science and engineering* (Vol. 1099, p. 012001).

Mushtaq, Z., & Su, S.-F. (2020). Environmental sound classification using a regularized deep convolutional neural network with data augmentation. *Applied Acoustics*, *167*, 107389.

Permana, S. D. H., Saputra, G., Arifitama, B., Caesarendra, W., Rahim, R., et al. (2021). Classification of bird sounds as an early warning method of forest fires using convolutional neural network (cnn) algorithm. *Journal of King Saud University-Computer and Information Sciences*.

Thiruvengatanadhan, R. (2018). Music classification using mfcc and svm. *International Research Journal of Engineering and Technology*, *5*, 922–924.

Vacher, M., Istrate, D., Besacier, L., Serignat, J.-F., & Castelli, E. (2004). Sound detection and classification for medical telesurvey. In *2nd conference on biomedical engineering* (pp. 395–398).