International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 09 | September - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

Urban Sound Classification

Pravisha Desale, Yash Vaykar, Prof. Manisha Bharti

Pravisha Desale, Department of Technology, SPPU
Yash Vaykar, Department of Technology, SPPU
Prof. Manisha Bharti, Department of Technology, SPPU

Abstract - There are many sounds all around us and our brain can easily and clearly identify them. Furthermore, our brain processes the received sound signals continuously and provides us with relevant environmental knowledge. Although not up to the level of accuracy of the brain, there are some smart devices which can extract necessary information from an audio signal with the help of different algorithms. Over the years, various models like **Convolutional Neural Networks (CNNs), Artificial Neural Networks (ANNs), Region-Convolutional Neural Networks (R-CNNs)**, and numerous machine learning techniques have been employed for sound classification. These methods have shown impressive results in distinguishing spectra-temporal patterns and different sound categories.

The novelty of our research lies in showing that the long-short term memory (LSTM) shows a better result in classification accuracy compared to CNN for many features used. Additionally, we've evaluated model accuracy using different techniques such as data augmentation and feature stacking. With our RNN model, we achieved a remarkable accuracy of 87%, setting a new benchmark in performance on the UrbanSound8k dataset.

Our findings not only advance the field of sound classification but also underscore the potential of LSTM networks and the importance of innovative techniques such as data augmentation and feature stacking in improving the accuracy of sound recognition systems.

***Key Words*:** Sound Classification, Urbansound8k, Librosa, Spectrograms, deep learning, CNN, LSTM.

1. INTRODUCTION

Sound is an integral part of our daily lives, including conversations, music, and various environmental noises like rainfall or car sounds. Sound classification is a dynamic field of research with widespread applications. These applications include automatic speech recognition, security systems, text-to-speech applications, video indexing, content-based retrieval, and speaker and sound identification. It also has potential uses in security applications Convolutional Neural Networks (CNNs), known for their high accuracy in image classification, are being applied to the seemingly different field of audio classification, where sounds occur sequentially over time.

The primary aim of this project is to develop an audio classifier powered by Deep Learning techniques. The central challenge is to effectively handle audio data and create a model capable of categorizing different types of sounds. The project intends to leverage advances in Deep Learning, particularly in speaker identification and recognition, to improve accuracy and assist users effectively. This involves optimizing the model's performance through the selection of appropriate algorithms, feature representations, and hyper parameter tuning.

The project uses data from the Urban Sound Classification Challenge database, which contains short audio samples typically encountered in urban settings. These samples can include sounds like children playing, street music, or car horns. Prior to model training, the audio samples undergo pre-processing to extract Mel-frequency Cepstral Coefficients (MFCC) features. Furthermore, the MFCC features vector is used as an input for the CNN model for classification and to generate predictions. The neural network outputs a vector with the probabilities of the sample belonging to each of

registered class. This vector is used to generate a prediction of the class of the sound, guessing for the one with the highest probability.

Furthermore, the project involves a comparison of various techniques, including RNN, CRNN, MFCC, and ASR for sound classification. The CNN model, known for its efficiency in learning deep architectures, is employed. It's important to note that two processes of converting sound clips to images are explored: spectrograms and MFCC. In this project, the spectrogram conversion method is employed. Spectrograms use a direct dispersed frequency scale, while MFCC relies on a semi-logarithmic divided frequency scale, mirroring how the human auditory system processes sound. This differentiation is a significant aspect of the research.

2 Dataset and features

2.1.1Dataset:

The dataset, known as UrbanSound8K, consists of 8732 sound samples, each lasting around 4 seconds. These audio clips are categorized into ten different classes: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, and street music. The audio files are stored in .wav format. The dataset has already been randomized and divided into ten folds, making it suitable for conducting a ten-fold cross-validation. The dataset creators advise against re-shuffling the data during training because of the way sound classes are distributed within each fold.

2.1.2 Data Analysis:

Audio Data overview and Analysis All audio samples are in .wav format and are sampled at discrete periods of time at the standard sampling rate (44.1 kHz meaning 44,100 samples per second)

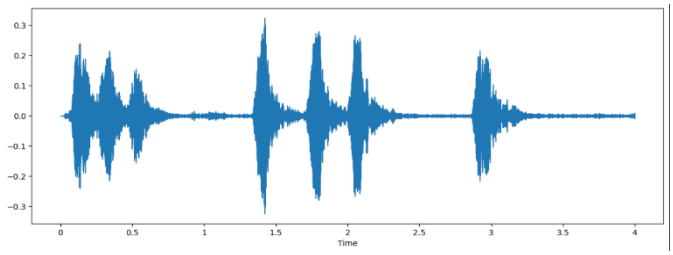


Fig -1: sample audio waveform

The bit depth determines how detailed the sample will be (typically 16-bit samples can range to about 65 thousand amplitude values) and every sample is the amplitude of the wave at a particular instance of time. Therefore, the data we will be using for every audio sample is basically a unidimensional vector of amplitude values.

2.1.3Visual Inspection of audio samples

We tried loading a sample from each class and visually inspecting the data for any similarities or patterns. We use librosa to load the sound files in an array and then use matplotlib. pyplot and librosa display to visualize the audio wave.

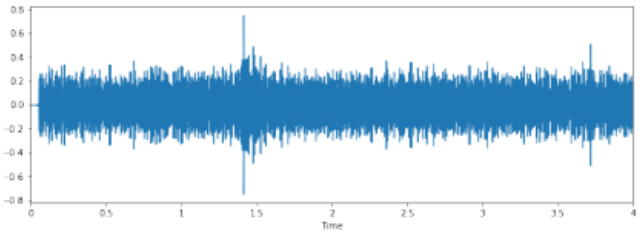


Fig -2: Air Conditioner

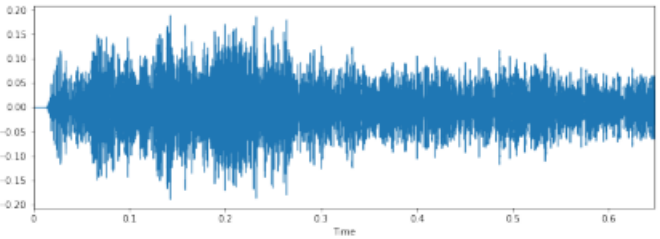


Fig -3: Car Horn

One cannot simply visualize the differences between some of the sound classes just by visual inspection. But a few observations can be made. Particularly, the waveforms which are repetitive such as jackhammer, drilling, air conditioner and engine idling have similar waveforms. Likewise the peak in the dog barking sample is similar in shape to the gun shot sample (albeit the samples differ in that there are two peaks for two gunshots compared to the one peak for one dog bark). Also, the car horn is similar too.

Fig -4: Children Playing

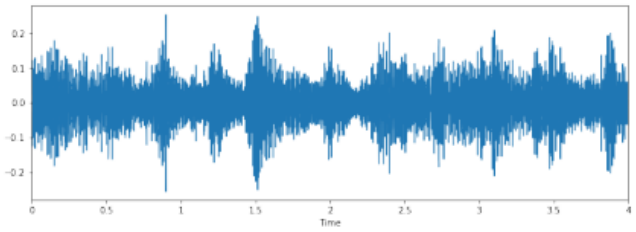


Fig -5: Dog Barking

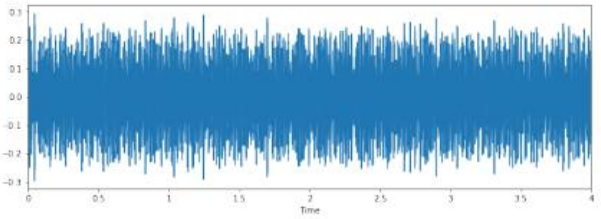
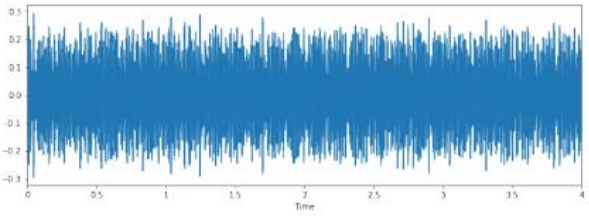


Fig -6: Drilling



Likewise the peak in the dog barking sample is similar in shape to the gun shot sample (albeit the samples differ in that there are two peaks for two gunshots compared to the one peak for one dog bark). Also, the car horn is similar too.

Fig-7: Idle Engine

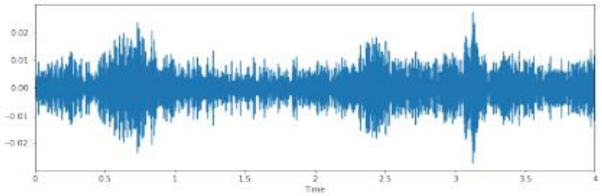
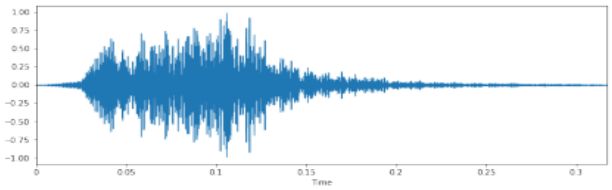


Fig -8: Gunshot



The solution we propose to this problem is to use Deep Learning techniques that have proved to be highly successful in the field of image classification. Initially we will extract Mel Frequency Cepstral Coefficients from the sound samples. It summarizes the relation of the perceived frequencies of the audio sample to the actual measured values of frequency, so it is able to analyze both the time and frequency characteristics of the sample. These audio representations allow us to distinguish features required for classification

2.1.4Feature Extraction and Data Labels:

As outlined previously, we will extract MFCCs from sound samples and create mel-spectrograms. We use the feature mfcc function of librosa library to extract the mfcc coefficients from the audio samples in the form a 2-D vector as shown below.

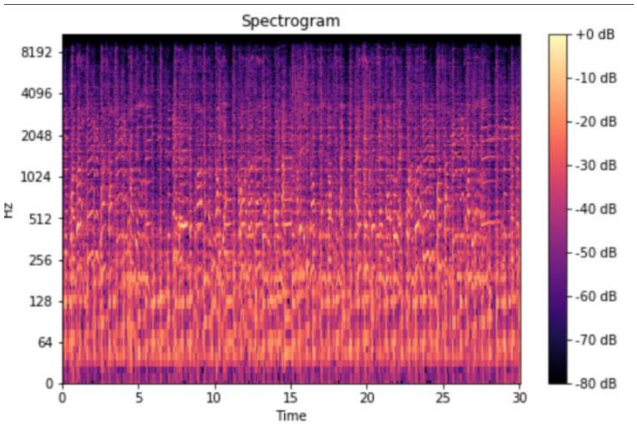


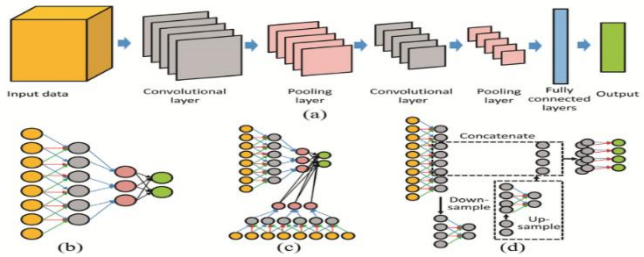
Fig -10: Log Scaled Mel Spectrogram

We modified the .csv file that came along the training dataset into a data frame with the titles of the audio files and their corresponding label. We extracted the features through a function that iterates through every row of the data frame accessing the file in the computer by reading the file's path. The

function make jpg helps label the names of all audio files and append them in the .csv file. Rescaling is a process in which we scale the data by a value before its processing in order to augment our data. Our dataset consists of images having RGB coefficients in the range 0-255. For a given learning rate, the values of the coefficients would be too high for our model to process, so we rescale the values between. Hence, we use Image Data Generator function to rescale the parameters of the images in the dataset, thus scaling the array of original image pixel values to be between datagen=Image Data Generator (rescale=1/255) probability is proposed as the prediction by the model.

2.1.5TABULAR REPRESENTATION OF THE DATA:

2.2Model Architecture:



2.2.1. Convolutional Neural Networks (CNNs) Model:

CNN are built on the architecture of Multi-Layer Perceptrops with several significant changes. Firstly, the three dimensions, height, width, and depth, are organized by layers. Moreover, the nodes in any layer are not connected to all the nodes in the following layers. The architecture allows the CNN mod the audio file. The log-frequency (y-axis) and time domain (x-axis) are affected by parameter choice. We used librosa to convert audio files into log spectrograms. We started with loading the .csv (comma separated) file containing all the titles of the audio files and their matching label. We defined a function for iterating through each row of the data frame to extract the features by reading the path of the file.

```
import pandas as pd

metadata=pd.read_csv("C:/Yash/B4B/Piyu Pro/UrbanSound8K/metadata/UrbanSound8K.csv")
metadata.head(10)
```

	slice_file_name	fsID	start	end	salience	fold	classID	class
0	100032-3-0-0.wav	100032	0.000000	0.317551	1	5	3	dog_bark
1	100263-2-0-117.wav	100263	58.500000	62.500000	1	5	2	children_playing
2	100263-2-0-121.wav	100263	60.500000	64.500000	1	5	2	children_playing
3	100263-2-0-126.wav	100263	63.000000	67.000000	1	5	2	children_playing
4	100263-2-0-137.wav	100263	68.500000	72.500000	1	5	2	children_playing
5	100263-2-0-143.wav	100263	71.500000	75.500000	1	5	2	children_playing
6	100263-2-0-161.wav	100263	80.500000	84.500000	1	5	2	children_playing
7	100263-2-0-3.wav	100263	1.500000	5.500000	1	5	2	children_playing
8	100263-2-0-36.wav	100263	18.000000	22.000000	1	5	2	children_playing
9	100648-1-0-0.wav	100648	4.823402	5.471927	2	10	1	car_horn

Fig -11: CNN Model

We obtained an array consisting of 193 features with their corresponding label which we used as X and Y for defining our training, validation and testing datasets. Further we scaled the datasets and selected 3435 samples for our training datasets, 1000 for our validation and testing datasets.el to perform in two.

Model Using Keras, we begin with building a Neural Network. We select a sequential model in order to construct the model layer wise. The model architecture is illustrated in figure 11 with all model parameters. Our model architecture comprises of 7 layers, a Conv2D and MaxPooling2D layer as the input and five hidden layers. Further, we can divide the hidden layers into three Conv2D layers with their corresponding MaxPooling2D layers, one Flatten layer and two Dense layers. Convolution layers are generally used for feature identification.

Algorithm

Data: Audio dataset converted into spectrogram images train generator - generates a spectrogram image data set from train file directory test generator - generates a spectrogram image data set from test file directory validation generator - generates a spectrogram image set from validation set

Result: Speech identification model using convolutional neural network

1: Initialize Sequential CNN Model:

Convolutional layer, Max Pooling layer, Dense layer, Flatten layer;

2: Train model

3: for all images in train generator & validation generator do

4: fit in model with validation steps=32 & epochs=50

5: calculate cross entropy loss

6: calculate accuracy for training data set & validation data set

7: end for

8: Test model

9: for all images in test generator do

10: predict & calculate accuracy

11: end for

Our model consists of 4 convolutional layers with increasing filter density. We found it optimal to extract the features of every image. Initially we constructed the model with 2 convolutional layers and gradually increased them in number to improve the performance of the model. To prevent overfitting and increase accuracy, we added max pooling and dropout layers along with each conv-2D layer. After obtaining the pool feature map, we vectorize it into a single column using a flattened layer and feed it to the fully connected layer which is achieved using the dense function. The size of the layers increases from 16, 32, 64 and till 128, with the filter parameter specifying the number of nodes in every layer. The kernel size parameter defines the size of the kernel window and has a value of 3 in our model which gives us a 3x3 filter matrix.

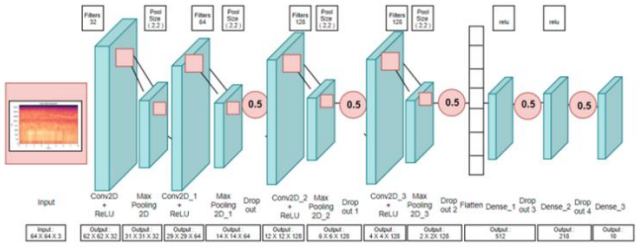


Fig. -12: CNN Model

The first layer gets the image input in the shape (64, 64, 3) where in the 3 represents the 3 RGB matrices and the 64 the number of frames. We used the Rectified Linear Activation or the Relu activation function for our 2 dense layers, which removes the negative part of the argument. F Relu(x) = max (0, x); Three convolutional layers have an associated pooling layer of MaxPooling2D type. By reducing computation requirements and the parameters to lessen the dimensionality of the model, a pooling layer reduces the training time and minimizes over-fitting.

The type of Max Pooling calculates the maximum size for every window to feed into our output layer. A dropout value of 50% is applied on all the hidden layers to regularize the neural network by randomly excluding nodes from every update cycle which results in a better generalization of the model and is to avoid over-fitting the data [10]. The output layer will have 10 nodes which matches the number of target classifications. Soft max, the activation function used, makes the output sum up to 1 allowing us to interpret the output as probabilities and since the number of outcomes is more than two. The option with the highest

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_4 (Conv1D)           (None, 38, 64)            256

 batch_normalization_5 (Batc  (None, 38, 64)           256
 hNormalization)

 max_pooling1d_4 (MaxPooling  (None, 19, 64)           0
 1D)

 conv1d_5 (Conv1D)           (None, 17, 128)           24704

 batch_normalization_6 (Batc  (None, 17, 128)          512
 hNormalization)

 max_pooling1d_5 (MaxPooling  (None, 8, 128)           0
 1D)

 global_average_pooling1d_2   (None, 128)              0
 (GlobalAveragePooling1D)

 dense_10 (Dense)            (None, 256)               33024

 dropout_6 (Dropout)         (None, 256)               0

 dense_11 (Dense)            (None, 10)                2570

=================================================================
Total params: 61,322
Trainable params: 60,938
Non-trainable params: 384
```

Fig. -13: Model Summary

MODEL COMPILATION

We use the following parameters for compiling our model:

• **Loss Function:** To compute loss, we use categorical cross entropy, in which a lower score indicates that the model is performing better.

• **Metric:** We use the accuracy metric which enables us to calculate the accuracy score on the validation dataset while training the model.

Training and Testing

We train about 50 epochs with a epoch step size of 108 and 32 validation steps per batch. Our test data contains 1000 audio samples of random distribution of sounds.

RESULTS: We were able to achieve a Classification Accuracy score of 86% on the testing dataset. Model accuracy and loss functions.

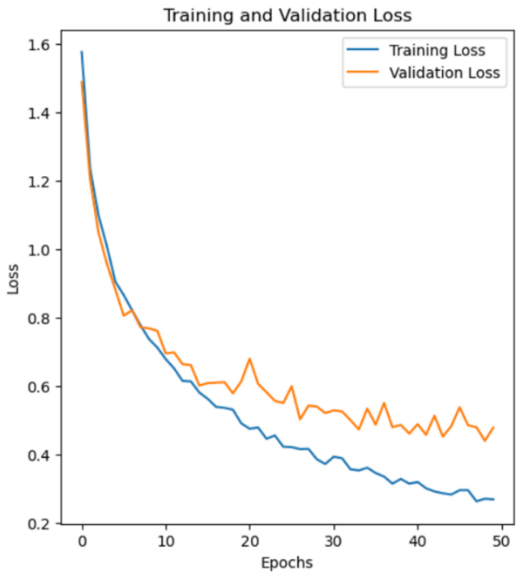


Fig -14: Training and validation loss

CONFUSION MATRIX

According to the confusion matrix, our model seems to struggle differentiating the following sub-groups:

• Street music, car horn and children playing

• Dog bark and children playing

• Drilling and jackhammer

This hints towards the problem being more complicated and nuanced than our assessment and helps us get an idea of the kind of features that the network is extracting to classify the sound samples. An example of it being street music, which is one of the commonly classified labels and has a lot of similarities with other classes, according to our model.

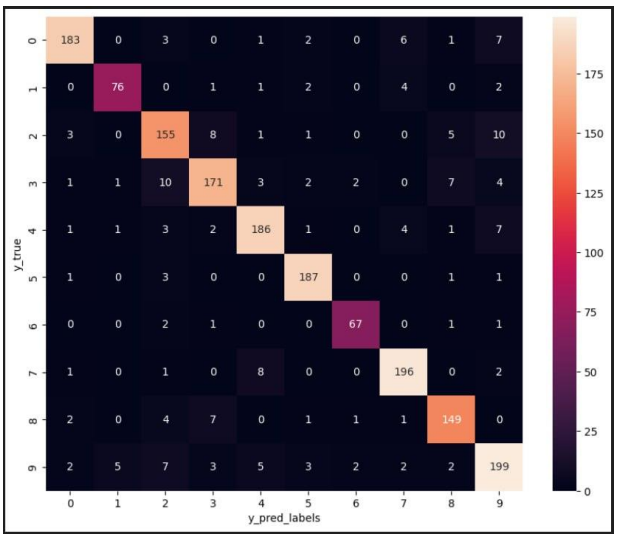


Fig-15: Confusion Matrix

2.2.2 LSTM MODEL:

In recent times, audio classification using Long Short-Term Memory (LSTM) networks has gained popularity due to its remarkable accuracy. To outline our approach in audio classification, we've employed a model architecture consisting of two LSTM layers, succeeded by two Time Distributed Layers, followed by a Flatten Layer, and finally, a Dense Layer, as illustrated in Figure.

Our audio classification model leverages the power of LSTM, a type of recurrent neural network (RNN), to effectively process sequential audio data and achieve impressive classification accuracy. The architecture of our model is designed to capture both short-term and long-term dependencies in the audio features. Below is a detailed breakdown of the model's layers.

Input Layer:

This is the initial layer that takes as input the audio features or spectrogram representations of the audio clips. The input shape is determined by the characteristics of our audio dataset.

LSTM Layer 1:

The first LSTM layer is responsible for capturing short-term temporal dependencies in the audio data. It processes the sequential data and outputs a hidden state vector.

LSTM Layer 2:

The second LSTM layer is stacked on top of the first one to capture deeper and more complex temporal patterns. This layer can capture longer-term dependencies in the audio features.

Time Distributed Layer:

After the LSTM layers, we apply a Time Distributed Layer to maintain temporal alignment with the output of the LSTM layers. This layer allows us to apply operations to each time step independently. Our model includes two Time Distributed Dense layers. In the first layer, the input size is 128, and the output layer comprises 256 nodes. As for the second Time Distributed layer, it takes an input size of 256 and outputs 512 nodes. In both of these layers, we have employed the Rectified Linear Unit (RELU) activation function.

Flatten Layer:

The Flatten Layer is used to convert the multi-dimensional output from the Time Distributed layers into a one-dimensional vector. This flattening step is necessary before feeding the data into the final classification layer.

Dense layer:

The Dense Layer serves as the output layer of our model. It typically contains a soft max activation function to produce probability distributions over the audio classes. The number of units in this layer corresponds to the number of classes in our audio classification task.

2.2.3. ANN MODEL:

Our audio classification model is built on a feedforward Artificial Neural Network (ANN) architecture. This architecture is designed to process audio features and make accurate predictions. Below, we provide a detailed description of the layers and components used in our model:

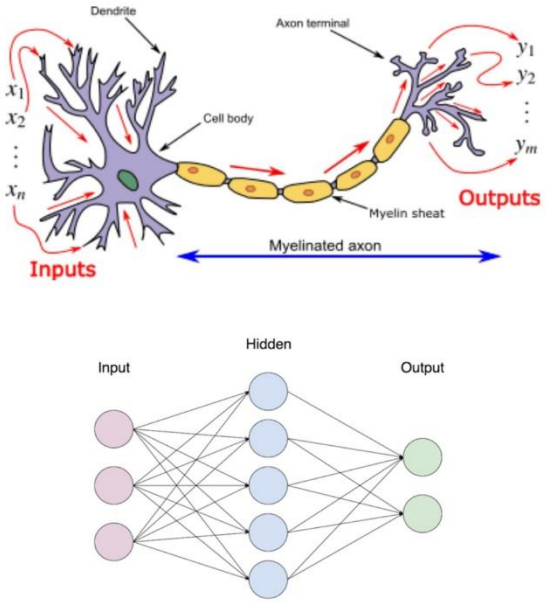


Fig -16: ANN Model

Input Layers:

The initial layer takes audio features or spectrogram representations as input. The input shape is determined by the characteristics of the audio dataset. Hidden Layers: These layers form the core of our ANN. The number of hidden layers and the number of neurons in each layer can be adjusted based on experimentation and the complexity of the task. Common activation functions include Rectified Linear Unit (RELU). Dropout Layers: Dropout layers can be added between hidden layers to prevent overfitting. The dropout rate can be adjusted to control the degree of regularization.

Output Layer:

The output layer produces predictions based on the audio input. The number of neurons in this layer is typically set to match the number of classes or categories in the audio classification task. The activation function used is often soft max for multiclass classification, sigmoid for binary classification, or linear for regression tasks.

Model Summary:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 100)               4100

 activation (Activation)     (None, 100)               0

 dropout (Dropout)           (None, 100)               0

 dense_1 (Dense)             (None, 200)               20200

 activation_1 (Activation)   (None, 200)               0

 dropout_1 (Dropout)         (None, 200)               0

 dense_2 (Dense)             (None, 100)               20100

 activation_2 (Activation)   (None, 100)               0

 dropout_2 (Dropout)         (None, 100)               0

 dense_3 (Dense)             (None, 10)                1010

 activation_3 (Activation)   (None, 10)                0

=================================================================
Total params: 45410 (177.38 KB)
Trainable params: 45410 (177.38 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig -17: Model Summary

Confusion Matrix:

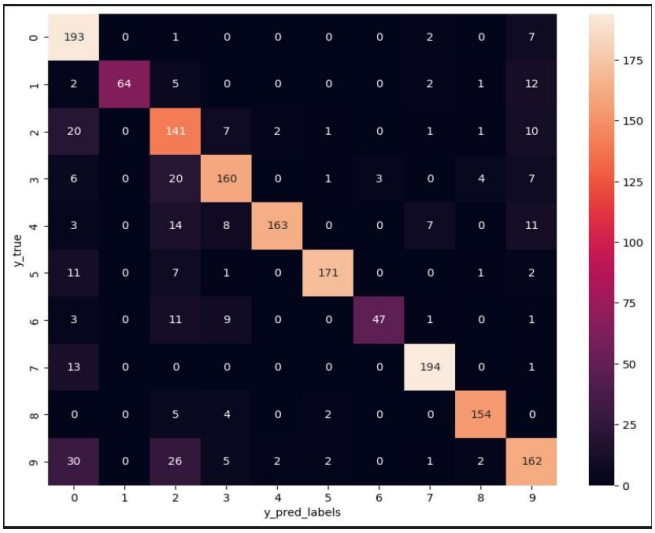


Fig -18: Confusion Matrix

Training and Validation Loss:

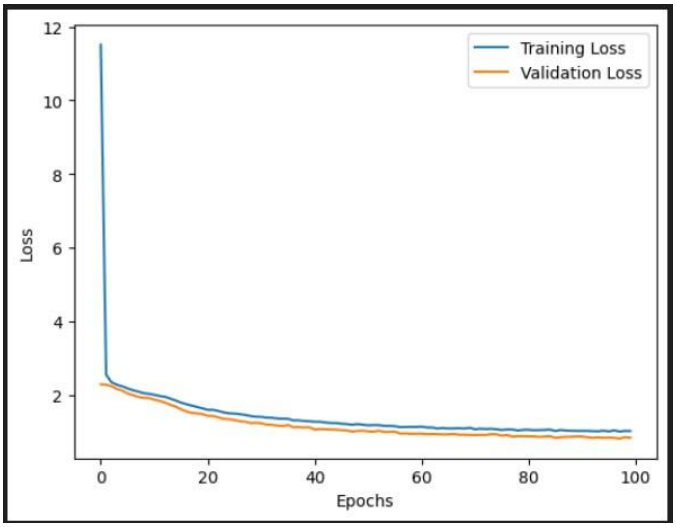


Fig -19: Training and validation loss

3. CONCLUSIONS

We successfully classified 10 different classes of audio samples from the given dataset by constructing a sequential 7-layer CNN model. We used MFCC to generate mel-spectrograms and used them as an input for the model. We can conclude that RNN can be as effective of an algorithm for audio classification as it is for image classification, the CNN was the second best and ANN came in last.

ACKNOWLEDGEMENT

We wish to express our deep sense of gratitude and honor towards our respected Prof. Manisha Bharti for her constant guidance that helped us greatly, to move forward with our research.
We wish to extend our sincere appreciation and gratitude to Professor Prof. Dr. Aditya Abhyankar, the Head of the Department of Technology (SPPU), for consistently motivating us and helping us in the successful completion of our project.
We are also thankful, to both the teaching and non-teaching staff at our institution, as well as our fellow classmates, who have directly or indirectly helped us in fostering our enthusiasm for the project work.
In closing, we want to emphasize that, much like a positive attitude has rewarded our diligent efforts in successfully completing this project, we believe it will continue to yield rewards. We aspire for this project to serve as a highly significant stepping stone in our careers, fulfilling our aspirations in every way.

REFERENCES

1) M. Smales, ˆaClassifying Urban sounds using Deep Learn-ing,ˆa 2018. [Online]. Available: https://github.com/mikesmales/Udacity-ML-Capstone/blob/master/Report/Report.pdf
2] Z. Huang, C. Liu, H. Fei, W. Li, J. Yu, and Y. Cao, ˆaUrbansound classidˇZcation based ´
On 2-order dense convolutional networkusing dual features,ˆa Applied Acoustics, vol. 164,
p. 107243, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0003682X19312691
J. Salamon and J. P. Bello, ˆaUnsupervised feature learning for urbansound classidˇZcation,ˆa ´ in 2015 IEEE International Conference on Acous-tics, Speech and Signal Processing
(ICASSP), 2015, pp. 171ˆa175.
ˆaˆa, ˆaFeature learning with deep scattering for urban sound analysis,ˆain 2015 23rd European

Signal Processing Conference (EUSIPCO),2015, pp. 724ˆa728.[7] K. Jaiswal and
D. Kalpeshbhai Patel, ˆaSound classidˇZcation using con-volutional neural networks,ˆa in
2018 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM),
2018, pp. 81ˆa84.
J. Sang, S. Park, and J. Lee, ˆa Convolutional recurrent neural networks forurban sound
classidˇZcation using raw waveforms ˆa in 2018 26th EuropeanSignal Processing Confer- ´
ence (EUSIPCO), 2018, pp. 2444ˆa2448.
J. Salamon and J. P. Bello, ˆa Deep convolutional neural networks anddata augmentation
For environmental sound classidˇ Zcation,ˆa IEEE SignalProcessing Letters, vol. 24, no. 3, ´
pp. 279ˆa283, 2017.
S. Ozarkar, R. Chetwani, S. Devare, S. Haryani, and N. Giri, ˆaAi foraccessibility: Virtual assistant for hearing impaired,ˆa in 2020 11th In