Urban Sound Classification using CNN

Conference Paper · January 2021

DOI: 10.1109/ICICT50816.2021.9358621

3 authors:

Massoud Massoudi
Delhi Technological University
12 PUBLICATIONS **193** CITATIONS
SEE PROFILE

Siddhant Verma
Delhi Technological University
1 PUBLICATION **51** CITATIONS
SEE PROFILE

Riddhima Jain
Delhi Technological University
1 PUBLICATION **51** CITATIONS
SEE PROFILE

Proceedings of the Sixth International Conference on Inventive Computation Technologies [ICICT 2021]
IEEE Xplore Part Number: CFP21F70-ART; ISBN: 978-1-7281-8501-9

Urban Sound Classification using CNN

Massoud Massoudi
Software Engineering Deparment
Assistant Professor
Parwan University, Charikar, Afghanistan
PhD. Candidate at DTU, Delhi, India
massoud.massoudi@hotmail.com

Siddhant Verma
Department of Electronics and Comm.
Undergraduate Student
Delhi Technological Unversity
siddhantv10@outlook.com

Riddhima Jain
Department of Electrical Engineering
Undergraduate Student
Delhi Technological Unversity
jainriddhima00@gmail.com

2021 6th International Conference on Inventive Computation Technologies (ICICT) | 978-1-7281-8501-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICICT50816.2021.9358621

***Abstract*—This document illustrates a simple audio classification model based on deep learning. We address the problem of classifying the type of sound based on short audio signals and their generated spectrograms, from labeled sounds belonging to 10 different classes during model training. In order to meet this challenge, we use a model based on Convolutional Neural Network (CNN). The audio was processed with Mel-frequency Cepstral Coefficients (MFCC) into what's commonly known as Mel spectrograms, and hence, was transformed into an image. Our final CNN model achieved 91% accuracy on the testing dataset.**

***Index Terms*—MFCC, CNN, Spectrogram, Urban Noise, Classification, Deep Learning.**

I. Introduction

Sounds outline the context of our daily activities, ranging from the conversations we have when interacting with people, the music we listen to, and all the other environmental sounds that we hear on a daily basis such as a car driving past, the patter of rain, or any other kind of background noise.

Sound classification is a constantly developing area of research and is at the heart of a lot of advanced technologies including automatic speech recognition systems, security systems, and text-to-speech applications. There are numerous applications which are continuously improving like video indexing and content based retrieval, speaker and sound identification use cases and potential security applications. Moreover, we know convolutional neural networks (CNNs) are widely used in image classification and they achieve significantly high accuracy, so we try to use this technique in a seemingly different field of audio classification, where discrete sounds happen over time [1].

This project aims to build a deep learning powered audio classifier. The basic underlying problem to be able to manipulate audio data and build a model to classify sounds. The project aims to leverage progress achieved in the deep learning field for speaker identification and recognition problem in order to perform accurately and improve to effectively assist users.

The input of the algorithm used in this project is taken from the database of the Urban Sound Classification Challenge [2], which are short audio samples commonly found in an urban environment like children playing, street music, a car engine etc. The samples must first be pre-processed in order to extract the MFCC features of each audio signal. 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 probabilites of the sample belonging to each of the registered class. This vector is used to generate a prediction of the class of the sound, guessing for the one with the highest probability.

II. Related Work

CNN based architectures have been in use for sound classification for a while. The model proposed in [3] which works on extraction and use of STFTs along with MFCC make for a multi-feature fusion system. A similar architecture has been proposed in [4] which also uses a second order dense CNN and dual features. However, in our project, we are trying a simpler approach. Other methods such as unsupervised feature learning, K-means [5] and feature learning with deep scattering [6] have also been in use to address a similar classification problem.

III. Problem Statement

Given a short audio sample, we want to determine if it includes any one of the considered urban noises and if it does, to be able to recognise it.

The objective of this project is to use deep learning techniques and neural network architecture of CNN to classify urban sounds.

IV. Evaluation Metric

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

$$Accuracy = \frac{Correct\ Classifications}{Number\ of\ Classifications} \quad (1)$$

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

Other metrics such as Precision, Recall were ruled out as they are more applicable to classification challenges that contain a relatively tiny target class in an unbalanced data set.

V. Data Exploration and Analysis

A. Dataset

We will be using a dataset from the Urban Sound Classification Challenge. The data contains 5435 labelled sounds from 10 different classes, which are:

- siren
- street music
- drilling
- engine idling
- air conditioner
- car horn
- dog bark
- jackhammer
- drilling
- gun shot

B. Class Distributions

Most classes are equally present in the dataset but there are two that have low representation. Most represent 11% of the data but one only represents 5% and another only 4%.

C. 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 (44.1kHz meaning 44,100 samples per second).

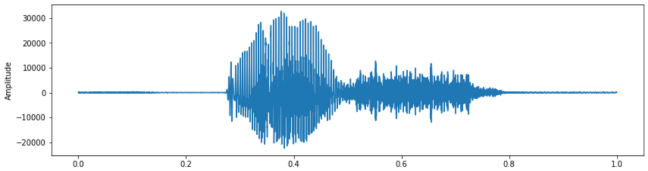


Fig. 1. sample audio waveform

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.

D. Visual Inspection of audio samples

We tried loading a sample from each class and visually inspect 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 visualise the audio wave.

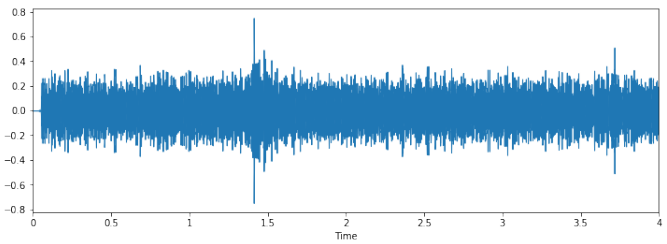


Fig. 2. air conditioner

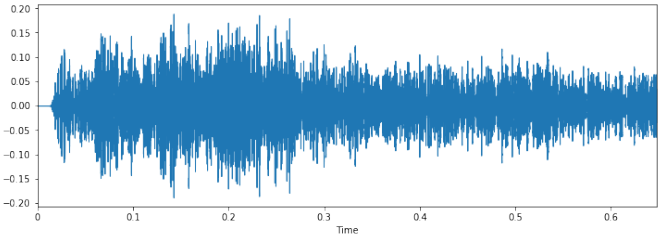


Fig. 3. car horn

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.

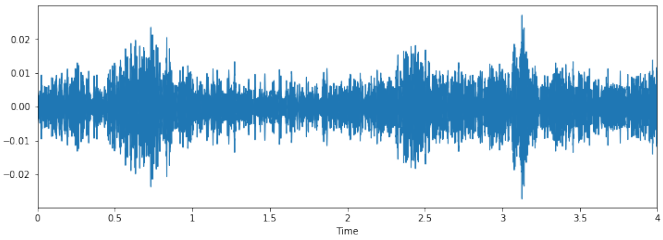


Fig. 4. children playing

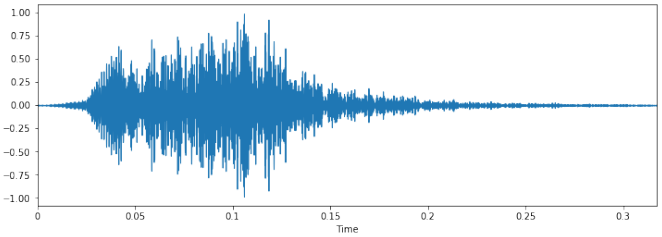


Fig. 5. dog barking

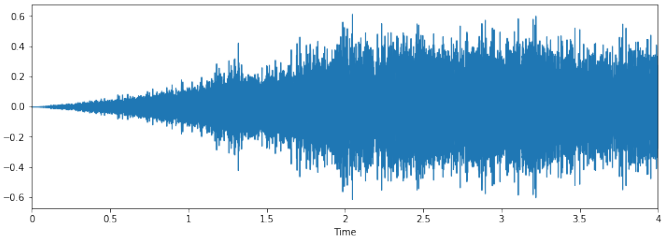


Fig. 6. drilling

Likewise the peak in the dog barking sample is similar in shape to the gun shot sample (albeit the samples differ in that

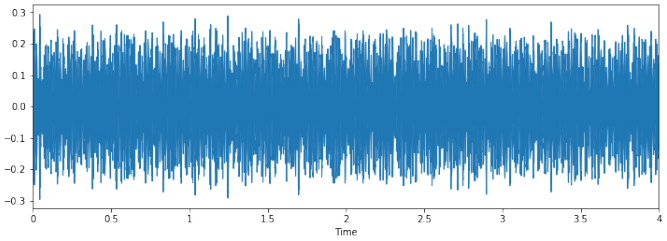


Fig. 7. idle engine

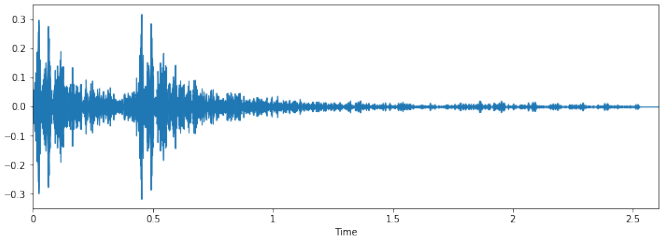


Fig. 8. gunshot

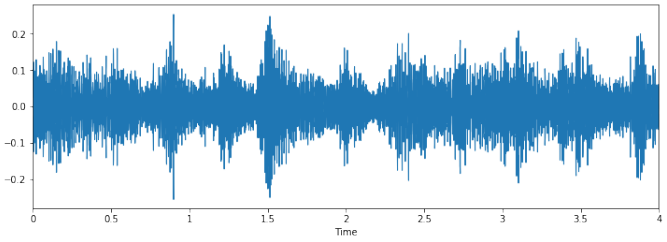


Fig. 9. street music

there are two peaks for two gunshots compared to the one peak for one dog bark). Also, the car horn is similar too.

VI. Algorithm

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 [7], [8] [9].

Initially we will extract Mel Frequency Cepstral Coefficients from the sound samples. It summarises the relation of the perceived frequencies of the audio sample to the actual measured values of frequency, so it is able to analyse both the time and frequency characteristics of the sample. These audio representations allow us to distinguish features required for classification.

Convolutional Neural Networks (CNNs) 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 model to perform in two essential steps. First, the feature extraction phase in which a filter window slides over the input and extracts a sum of the convolution at each location, the feature map stores these features from each window. The CNN layers include a pooling process in between them. Usually, each window's maximum value is taken, retaining the essential data while decreasing the feature map size. Pooling is crucial as it reduces the network's dimensionality, which reduces both the possibility of overfitting and the training time. Lastly, we have the classification phase. After this, the 3-D data within the interface is flattened and output into a 1-D vector. - [10], [11].

Because of the above discussed reasons, CNNs are better classifiers when it comes to image classification because of their feature extraction and classification capabilities.

Algorithm 1

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, MaxPooling 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=250
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**

VII. Data Preprocessing and Splitting

A. MFCC

Figure 10 shows a log scaled mel spectrogram generated from a sample audio signal from the dataset.

All data was processed with Mel-frequency cepstral coefficients (MFCC) which are based on processing audio with a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. In effect, this pre-processing of the audio files created a spectrogram, a feature representation from an audio signal to an image. This translation requires the use of parameters (i.e. Fourier-Transform, Hop-length, the mel coefficient, etc.), thus the conversion to an image for a richer set of features comes at a cost of reducing information about

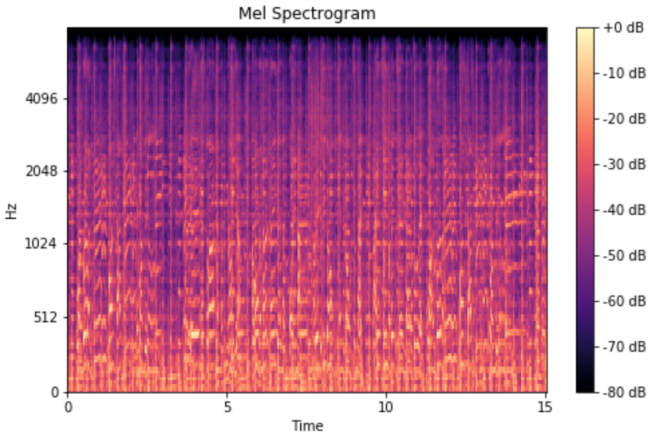


Fig. 10. Log Scaled Mel Spectrogram

the audio file. The log-frequency (y-axis) and time domain (x-axis) are affected by parameter choice. [7]

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.

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.

B. Feature 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.

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 [0,1] [12]. Hence, we use `ImageDataGenerator` function to rescale the parameters of the images in the dataset, thus scaling the array of original image pixel values to be between [0,1].

```
datagen=ImageDataGenerator(rescale=1./255.)
```

VIII. Model Implementation

A. 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. A convolution layer works by sliding a filter window over the input and performing matrix multiplication. It then stores the output in the form a feature map. This overall operation is known as convolution.

Our model consists 4 convolutional layers with increasing filter density. We found it optimal in order 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 maxpooling and dropout layers along with each conv-2D layer. After obtaining the pool feature map, we vectorize it into a single column using a flatten layer and feed it to the fully connected layer which is achieved using the dense function.

The size of the layers increase 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. [10], [13].

The first layer gets the image input in the shape (64, 64, 3) where in the 3 represents the 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. [7].

$$f_{ReLu}(x) = max(0, x); \tag{2}$$

Three convolutional layers have an associated pooling layer of *MaxPooling2D* type. By reducing computation requirements and the parameters in order 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 [10].

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 generalisation 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. *Softmax*, the activation function used, makes the output sum up to 1 allowing us to interpret the output as probabilities and also since the number of outcomes is more than two. The option with the highest

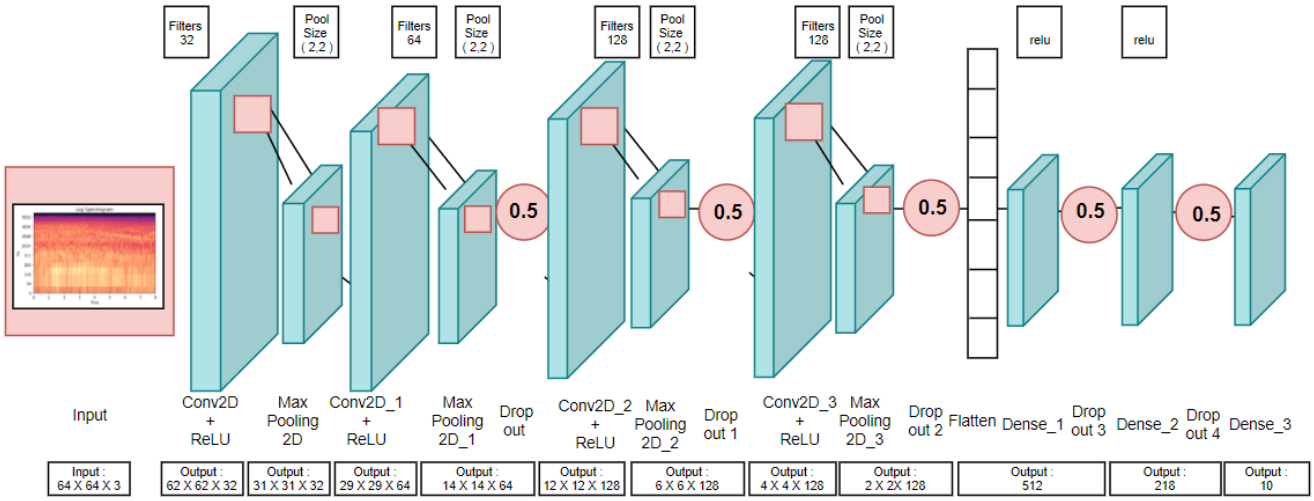


Fig. 11. Model Summary

```
MFCC feature extraction :  (40, 345)
```

probability is proposed as the prediction by the model - [1], [14].

B. Model Compilation

We use the following parameters for compiling our model:

- Loss Function : To compute loss, we use `categorical_crossentropy`, 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.

C. Training and Testing

We train about 250 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.

IX. Results

We were able to achieve a Classification Accuracy score of 91% on the testing dataset.Model accuracy and loss functions are shown in figures 13-14.

Justification

The model achieved an accuracy score of 91% on the testing data. The given model performs well when presented with a .wav file with a duration of a few seconds and returns a reliable classification. However, we do not know how the model would perform on Real-time audio and in a real world setting. We made no attempt to determine the effect of factors like noise, echos, volume, and salience level of the sample.

X. 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.

Conclusion and Future Improvements

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 CNN can be as effective of an algorithm for audio classification as it is for image classification.

Future improvements:

- Test the models performance with Real-time audio.
- Train the model for real world data by adding a variety of additional background noises, adjusting volume levels of the target sounds or adding echos.
- Experiment with other techniques for feature extraction such as other different forms of spectrograms.

References

[1] M. Smales, "Classifying Urban sounds using Deep Learning," 2018. [Online]. Available: https://github.com/mikesmales/Udacity-ML-Capstone/blob/master/Report/Report.pdf

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 29, 29, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 6, 6, 128)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 4, 4, 128)         147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 2, 2, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 512)               262656
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               65664
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 570,442
Trainable params: 570,442
Non-trainable params: 0
_________________________________________________________________
```

Fig. 12. Model Summary

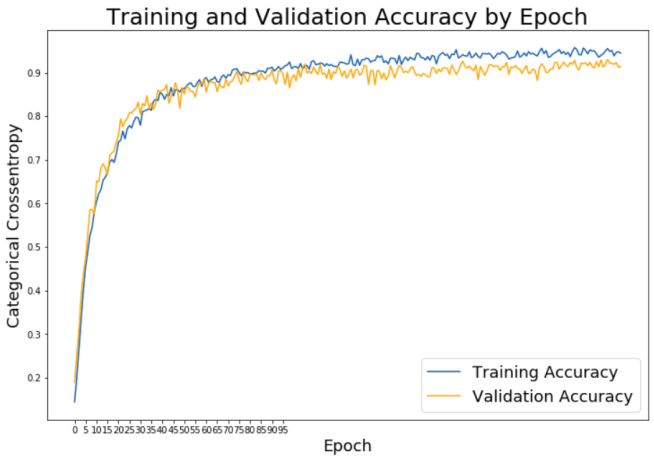


Fig. 13. Model Accuracy

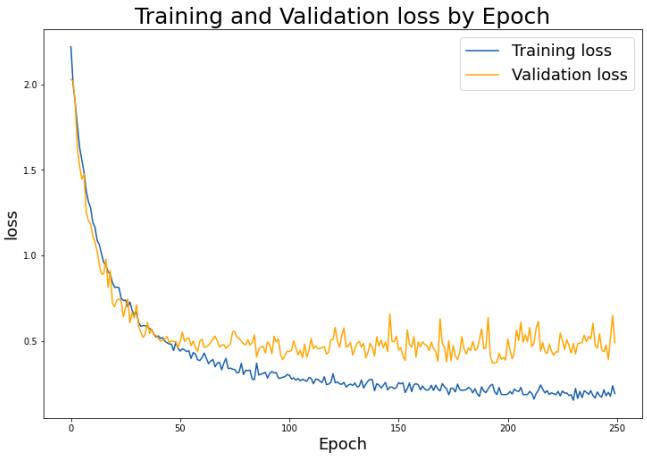
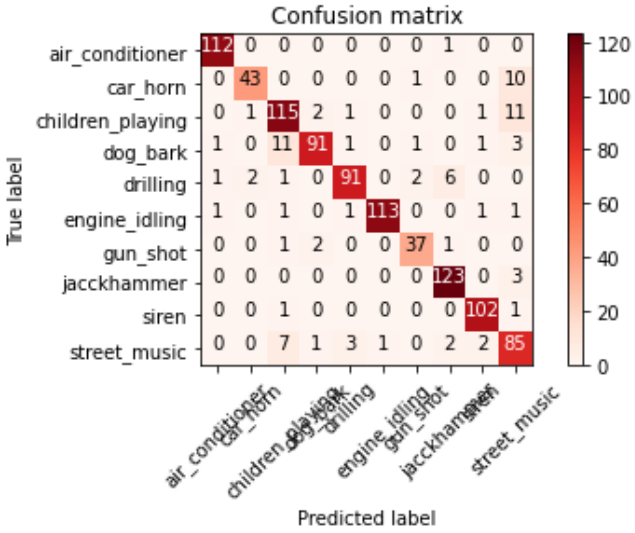


Fig. 14. Model Loss



Multi-Feature Fusion System," *Detection and Classification of Acoustic Scenes and Events 2019*.

[4] Z. Huang, C. Liu, H. Fei, W. Li, J. Yu, and Y. Cao, "Urban sound classification based on 2-order dense convolutional network using dual features," *Applied Acoustics*, vol. 164, p. 107243, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0003682X19312691

[5] J. Salamon and J. P. Bello, "Unsupervised feature learning for urban sound classification," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 171–175.

[6] ——, "Feature learning with deep scattering for urban sound analysis," in *2015 23rd European Signal Processing Conference (EUSIPCO)*, 2015, pp. 724–728.

[7] K. Jaiswal and D. Kalpeshbhai Patel, "Sound classification using convolutional neural networks," in *2018 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, 2018, pp. 81–84.

[8] J. Sang, S. Park, and J. Lee, "Convolutional recurrent neural networks for urban sound classification using raw waveforms," in *2018 26th European Signal Processing Conference (EUSIPCO)*, 2018, pp. 2444–2448.

[9] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[10] S. Ozarkar, R. Chetwani, S. Devare, S. Haryani, and N. Giri, "Ai for accessibility: Virtual assistant for hearing impaired," in *2020 11th In-*

[2] *Practice Problem: Urban Sound Classification*, 2017. [Online]. Available: https://datahack.analyticsvidhya.com/contest/practice-problem-urban-sound-classification/

[3] B. W. M. W. R. W. Jisheng Bai, Chen Chen, "Urban Sound Tagging With

ternational Conference on Computing, Communication and Networking Technologies (ICCCNT), 2020, pp. 1–7.

[11] Justin Salamon; Christopher Jacoby; Juan Pablo Bello, "Urban Sound Datasets," *MM '14 Proceedings of the 22nd ACM international conference on Multimedia*, no. 3, pp. 1041–1044, 2014. [Online]. Available: http://serv.cusp.nyu.edu/projects/urbansounddataset/

[12] F. Chollet, "Building powerful image classification models using very little data," 2016. [Online]. Available: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

[13] M. Sahidullah and G. Saha, "Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition," *Speech Communication*, vol. 54, no. 4, pp. 543–565, 2012.

[14] D. Cornelisse, "An intuitive guide to Convolutional Neural Networks," *free Code Camp*, pp. 1–19, 2018. [Online]. Available: https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/