Urban Sound Classification

by Vadim Spirkov, Murlidhar Loka

Abstract Cities are significantly noisier than rural areas. Almost a third of Europe's population lives in areas where noise levels exceed the 55 decibel sound limit recommended by the World Health Organisation.Until recently, urban noise was considered a mere annoyance, a by-product of living in close quarters on busy streets. Yet new research in Denmark funded by the European Research Council demonstrates that it causes stress, disturbed sleep, serious illness, and can even increase the risk of heart attack or stroke.(Ref: Echikson)

Background

Considering the harmful effect of the urban noise on people the classification of the urban sounds my be used to identify which particular kind of noise is the most disturbing. Knowing this the architects and builders may come-up with the better sound insulation materials and building designs to shield the dwellers from for the most distractive sounds. The machinery manufacturers could produce better soundproofed equipment...

Objective

Objective of this research is to build a robust urban sound classifier. We will be leveraging concepts from transfer learning and deep learning to build the classifier whereby, with any given audio sample belonging to one of our pre-determined categories, we should be able to correctly predict the source of this sound.

Data Analysis

This research employs the data set sourced from UrbanSounddataset. This dataset contains **8732** labeled sound excerpts (<=4s) of urban sounds from 10 classes: air_conditioner, car_horn, children_playing, dog_bark, drilling, enginge_idling, gun_shot, jackhammer, siren, and street_music. The classes are drawn from the urban sound taxonomy. The files are pre-sorted into ten folds (folders named fold1-fold10) to help in the reproduction of and comparison with the automatic classification results. We will follow a standard workflow of analyzing, visualizing, modeling, and evaluating our models on our audio data.

Data Dictionary

Audio Files

8732 audio files of urban sounds (see description above) in WAV format. The sampling rate, bit depth, and number of channels are the same as those of the original file uploaded to Freesound (and hence may vary from file to file).

Meta-data Files

This file contains meta-data information about every audio file in the dataset.This includes:

Slice_file_name

The name of the audio file. The name takes the following format: *fsID-classID-[occurrenceID]-sliceID.wav*, where:

sliceID

A numeric identifier to distinguish different slices taken from the same occurrence

fsID

The Freesound ID of the recording from which this excerpt (slice) is taken

Start

The start time of the slice in the original Freesound recording

end

The end time of slice in the original Freesound recording

salience

A (subjective) salience rating of the sound. 1 = foreground, 2 = background.

fold

The fold number (1-10) to which this file has been allocated.

classID

A numeric identifier of the sound class:

0 = air_conditioner 1 = car_horn 2 = children_playing 3 = dog_bark 4 = drilling 5 = engine_idling 6 = gun_shot 7 = jackhammer 8 = siren 9 = street_music

Data Exploration

Let's begin our data exploration exercise checking the content of the meta-data file supplied with the audio samples:

	FileName	ClassID	Class
0	100032-3-0-0.wav	3	dog_bark
1	100263-2-0-117.wav	2	children_playing
2	100263-2-0-121.wav	2	children_playing
3	100263-2-0-126.wav	2	children_playing
4	100263-2-0-137.wav	2	children_playing
5	100263-2-0-143.wav	2	children_playing
6	100263-2-0-161.wav	2	children_playing
7	100263-2-0-3.wav	2	children_playing
8	100263-2-0-36.wav	2	children_playing
9	100648-1-0-0.wav	1	car_horn
10	100648-1-1-0.wav	1	car_horn
11	100648-1-2-0.wav	1	car_horn
12	100648-1-3-0.wav	1	car_horn
13	100648-1-4-0.wav	1	car_horn
14	100652-3-0-0.wav	3	dog_bark
15	100652-3-0-1.wav	3	dog_bark
16	100652-3-0-2.wav	3	dog_bark
17	100652-3-0-3.wav	3	dog_bark
18	100795-3-0-0.wav	3	dog_bark
19	100795-3-1-0.wav	3	dog_bark

Figure 1: The Meta-data File Content Sample

Employing the meta-data file content and the audio samples we will try to gain better understanding of the data we are dealing with. This knowledge will help us to prepare data for model training.

We begin with counting the number of observations per category.

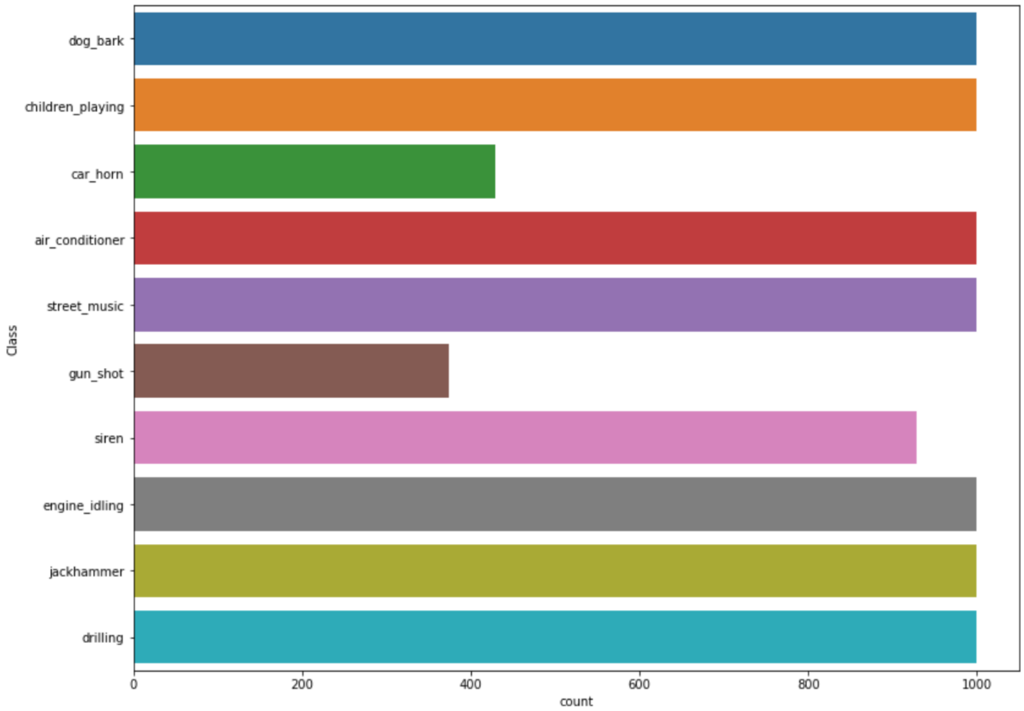

Figure 2: Number of Observations Per Category

As per the plot submitted above we can see that the **Gun Shot** and **Car Horn** categories are underpopulated; the data set is not balanced. To balance the dataset we could:

1. Upsample these categories
2. Downsample the populous categories
3. Add more labelled observations to the smaller categories
4. Leave the data set as is hoping that the categories with the smaller population will be large enough to train the network

We decided to go with option **1** since it gives us the best quality. But now we face a question how to upsample the audio. The answer is sound augmentation. We can take the existing audio samples and alter them adding noise, changing the pitch and time shift. This technique proved to be very effective to improve the accuracy of the models.

Now it is time to take a look at the audio sample rate distribution. As shown in figure 3 the sample rate of the sound files varies. We would have to re-sample the original data to bring it to the same standard.

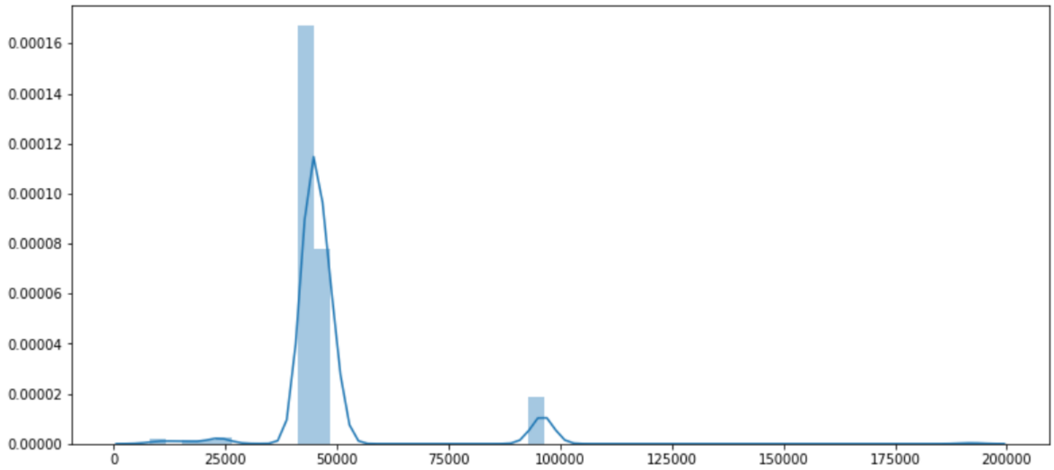

Figure 3: Audio Sample Rate Distribution

Let's calculate the length of the sound samples.

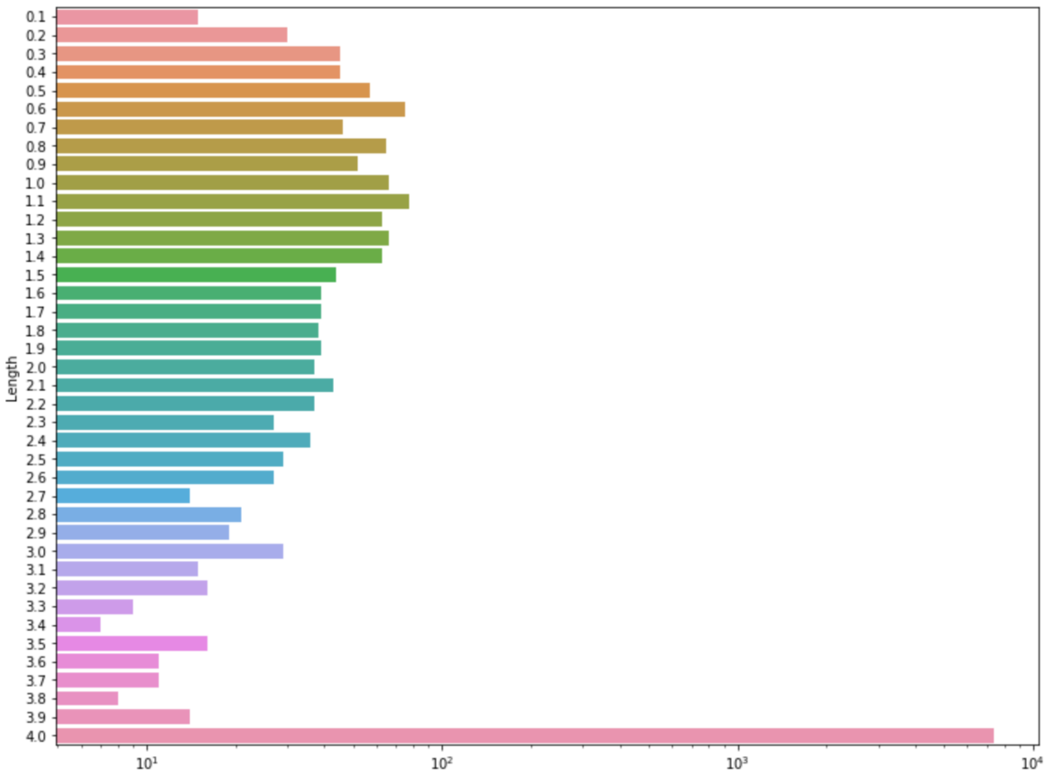

Figure 4: Audio Length Distribution

The majority of the sound files are **4** second long (fig: 4). But there are some file that are less than a second long. Designing the model we would have to make sure that the input layer is able to deal with the sound samples of various length and a sample rate. We might also want to filter out the samples that less than **0.5** second long, because most likely they do not carry much of valuable information.

Lastly we are going to verify how many channels the recorded audio file have (stereo vs. mono)

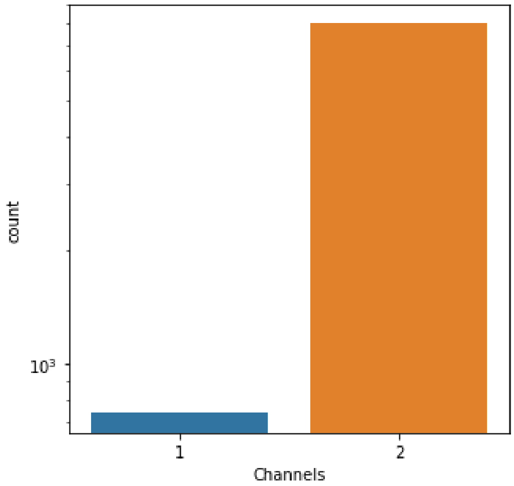

Figure 5: Mono vs. Stereo Sample Distribution

Just a few audio files were recorded in mono. Though each stereo channel carries slightly different information for the sake of simplicity we will be converting each audio sample to monophonic.

Sound Characteristics of Each Category

To successfully classify the urban sounds each sound category must have distinctive features. Applying audio feature extraction library **librosa** (Ref: Source) we are going to take a look at one sample from each category to witness the difference.

The most common and well understood audio chart is a `waveform amplitude plot`. The pattern of variations contained in the waveforms gives us information about the signal.

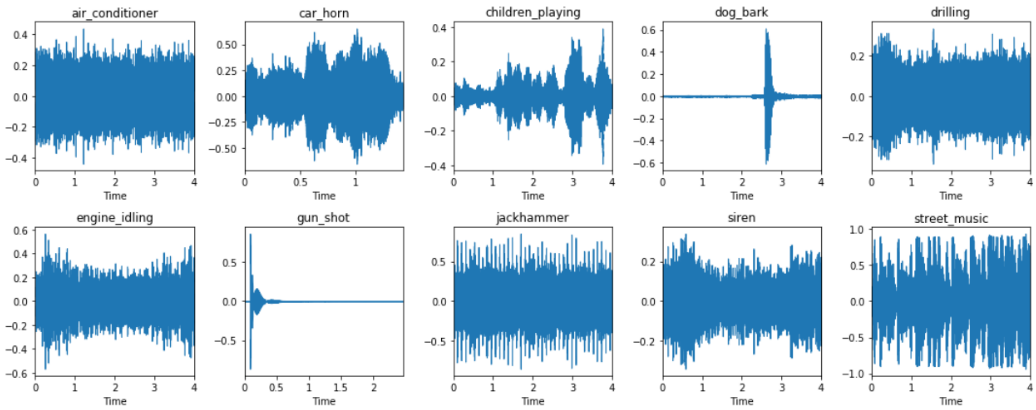

Figure 6: Waveforms of Sounds from All Ten Categories

Looking at the waveforms rendered in figure: 6 we can observe that some sounds have very distinctive shape. On the other hand the plot of the **air conditioner** and **jackhammer** are somewhat similar. Is there a better way to pick the distinctive sound features? In fact over time the sound engineers and scientist came up with many way to describe the unique characteristics of the sound. We have decided to employ **Mel spectrogram**. Mel frequency spectrogram is a lossless presentation of a signal that gives an acoustic time-frequency representation of a sound: the power spectral density. It is sampled into a number of points around equally spaced times and frequencies (on a Mel frequency scale). The Mel scale is a **perceptual** scale of pitches judged by listeners to be equal in distance from one another.

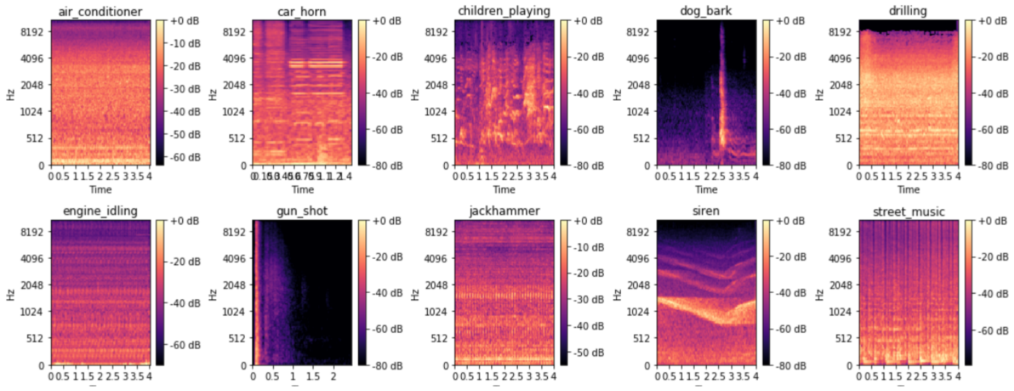

Figure 7: Mel Spectrograms of Sounds from All Ten Categories

As we can see the Mel spectrograms have more features than the amplitude/time waveform presentation. Now the distinction between the the **air conditioner** and **jackhammerr** is much clearer. Since the spectrograms are lossless they could be used to train the Neural Networks!

Data Preparation

Taking into consideration the finding discovered during the data exploration phase we are ready to design a dataset preparation and feature extraction approach.

- Firstly we shall balance the data set employing sound augmentation techniques as previously discussed.
- We re-sample all audio using a sample rate of **22050 Hz**
- All sound samples will be converted to **mono**
- As it was noted the length of the audio samples varies from 0.1 to 4 seconds. To preserve as much data as possible we will apply a sub-sampling technique known as a *sliding window approach*.
  - We select a window size of **0.5** seconds, which we use against each audio sample to slice it into 0.5-long audio segments.
  - The window will be moving from the beginning of the audio file to the end with overlap of 0.35 seconds.
  - Most of the times the sliding window would not cut the audio file precisely from the beginning to the end. Thus the last slice will be taken from the and f the audio sample.
  - This approach will result in getting multiple overlapping samples from each audio file.

Feature Extraction

Now let's ponder about the features we would have to extract to design a robust classifier. Long before the neural networks gained their popularity the scientist had already come up with pretty accurate methods to classify the sound. The star feature of the time was **Mel-frequency cepstral coefficients (MFCCs)** (Ref: Halfdan Rump). With the expansion of the Neural Networks multiple research works demonstrate that **log-scaled Mel spectrogram** is superior to the other features extraction techniques in the context of deep learning models (Ref: Keunwoo Choi).

Our feature extraction approach is based on the latest scientific findings and largely inspired by the "Hands-On Transfer Learning with Python" book (Dipanjan Sarkar (2018)). Below we outline our feature extraction sequence.

To capture more data we are going to use **96-band** Mel spectrogram over **96** frames for each audio sample. When the spectrogram is extracted we can use it as an input for a Convolutional Neural Network. In the end the spectrograms could be rendered as an image. The CNN would learns the features of the spectrograms and then classify them. But there is a better approach... There are literally dozens of pre-trained models. Thousand hours have been spent on the model training and design. Can we take advantage of this work? Absolutely!

We are going to employ **VGG-16** convolutional neural network trained on millions of images from "imagenet" to learn the features of the Mel spectrograms, which we described just a moment ago. We would have to modify the VGG-16 architecture, namely:

- We shape the input layer as follows: $96 * 96 * 3$
- We drop the softmax output layer
- We flatten the last layer of the network

Here is the the modified VGG-16 architecture:

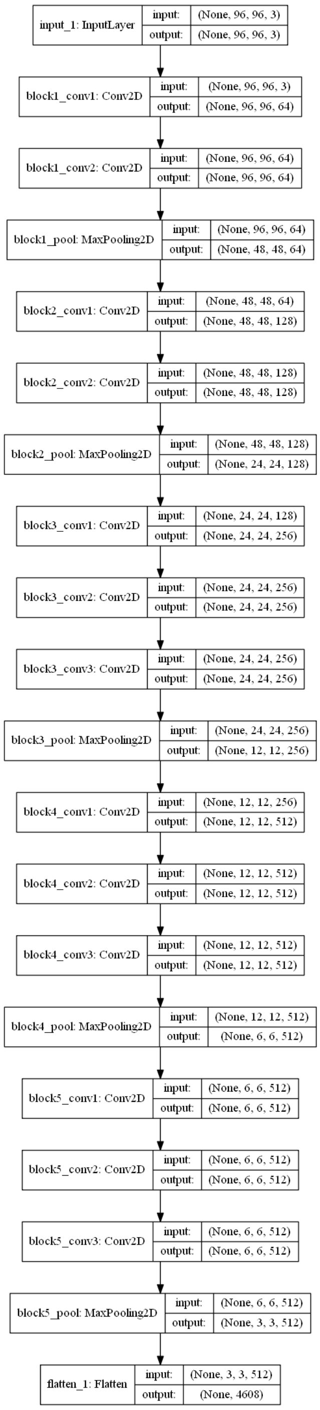

Figure 8: VGG-16 Architecture

The output of the model is a one-dimensional feature vector of size **4608**. There is one small issue though. VGG-16 is trained on the three-channel (RGB) images. In our case we have only one $96 * 96$ matrix. We can consider a few solutions... The first one is to extract two more features. The best candidates would probably be *Harmonic* and *Percussion* sequences. But earlier in the paper we have noted that the Mel spectrograms are lossless. Addition of new feature would undoubtedly make negative performance impact, but would it add more data? We do not believe so. Thus we decide to replicate the same matrix three times.

Production Pipeline

Considering everything said above this is a diagram of the production data pipeline

Classification Model

Heavily relying on the power of the transfer learning our classifier is going to be rather simple. It is a dense neural network that has input layer of size **4608** that matches the output of the VGG-16 NN. The classifier has a few dropout layers to fight the overfitting. The output layer of the classifier has softmax activation function with ten outputs - one for each sound category. A you may recall we started with 8736 audio files of various length. After data processing and feature extraction steps we ended-up with the dataset of **130899** rows and **4608** columns!

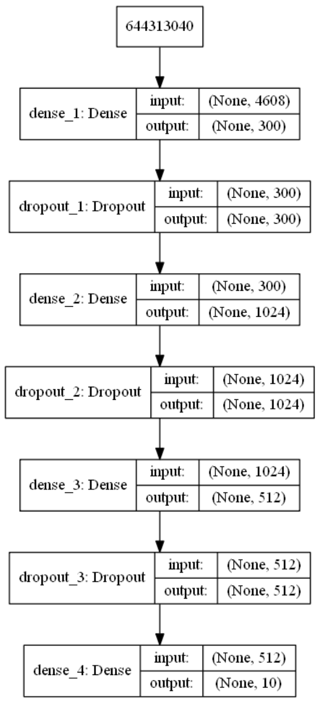

Figure 9: Classification Model

Model Deployment and Training on Google Cloud Platform

GCP offers many way to achieve the same goal. We frankly have been overwhelmed with the number of available articles and training materials, which in fact make the things more confusing than helpful. We have spent considerable amount of time studying various approaches. As it has been previously discussed our processing pipeline is quite complex. To implement it on the cloud we have designed the following workflow:

Training

The model training was done using 50 epochs, with the batch size of 256 and validation split of 25% of the training dataset.

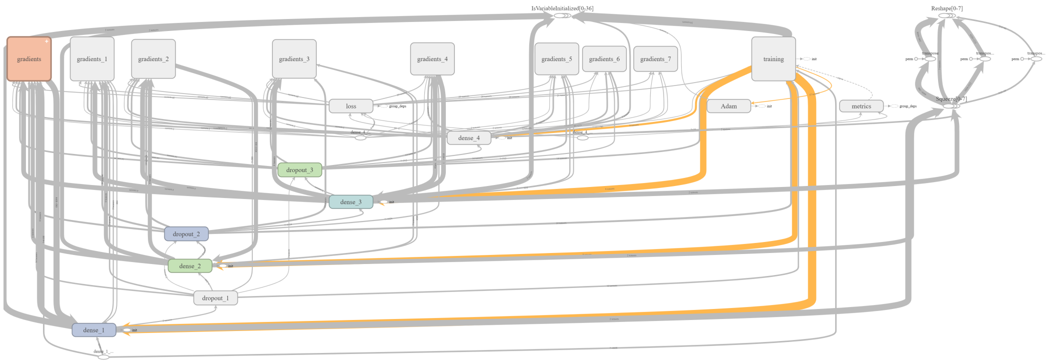

Figure 10: Model Training Graph

Let's review model training and validation curves

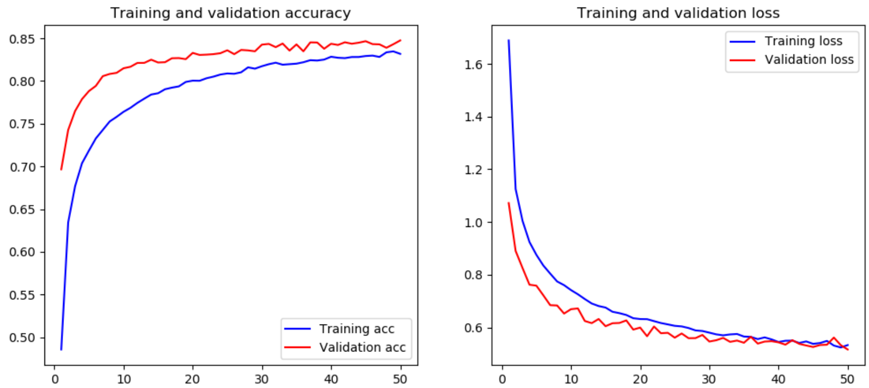

Figure 11: Model Training and Validation Curves

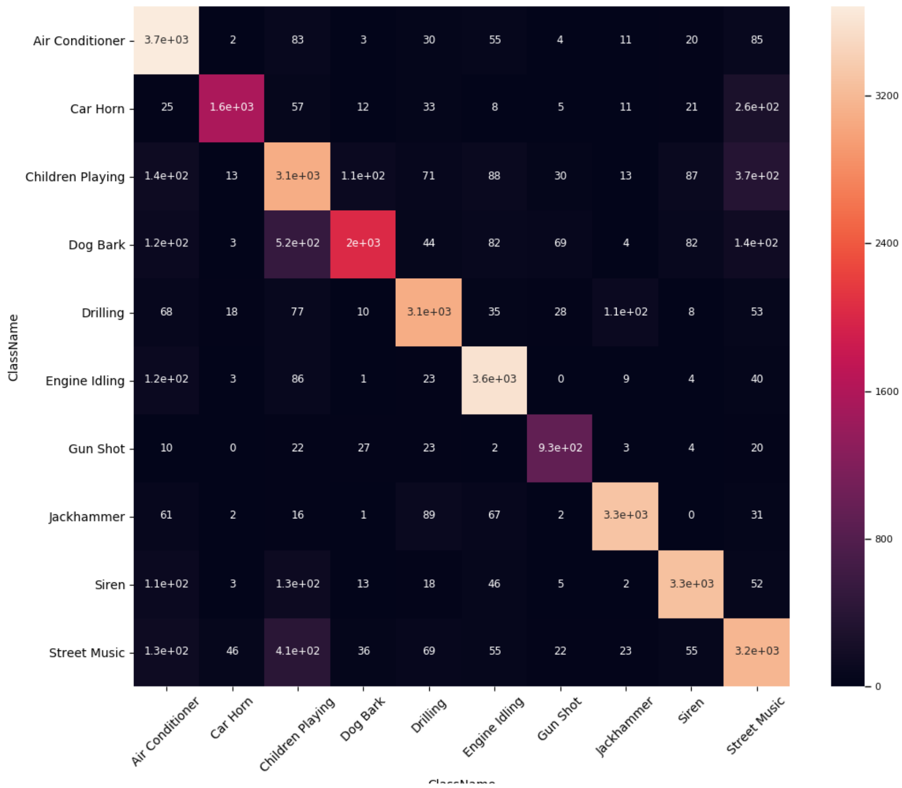

Figure 12: Classifier Confusion Matrix

	precision	recall	f1-score	support
Air Conditioner	0.83	0.93	0.87	3977
Car Horn	0.95	0.78	0.86	1998
Children Playing	0.69	0.77	0.73	4009
Dog Bark	0.91	0.65	0.76	3084
Drilling	0.89	0.88	0.88	3492
Engine Idling	0.89	0.93	0.91	3877
Gun Shot	0.85	0.89	0.87	1038
Jackhammer	0.95	0.92	0.94	3571
Siren	0.92	0.90	0.91	3656
Street Music	0.75	0.79	0.77	4023
micro avg	0.85	0.85	0.85	32725
macro avg	0.86	0.84	0.85	32725
weighted avg	0.85	0.85	0.85	32725

Conclusion

Bibliography

T. G. Dipanjan Sarkar, Raghav Bali. *Hands-On Transfer Learning with Python*. Packt Publishing Ltd., Livery Place 35 Livery Street Birmingham B3 2PB, UK, 2018. ISBN 978-1-78883-130-7. [p6]

W. Echikson. Combating urban noise. URL https://newcities.org/blog-combating-urban-noise-echikson/. [p1]

E. T. Halfdan Rump, Shigeki Miyabe. Autoregressive mfcc models for genre classificationimproved by harmonic- percussion separation. URL https://www.researchgate.net/publication/

220723615_Autoregressive_MFCC_Models_for_Genre_Classification_Improved_by_Harmonic-percussion_Separation. [p6]

M. S. Keunwoo Choi, Gyorgy Fazekas. A comparison on audio signal preprocessing methods for deep neuralnetworks on music tagging. URL https://www.researchgate.net/publication/319534366_A_Comparison_on_Audio_Signal_Preprocessing_Methods_for_Deep_Neural_Networks_on_Music_Tagging. [p6]

L. O. Source. Librosa is a python package for music and audio analysis. URL https://librosa.github.io/librosa/. [p5]

Note from the Authors

This file was generated using *The R Journal* style article template, additional information on how to prepare articles for submission is here - Instructions for Authors. The article itself is an executable R Markdown file that could be downloaded from Github with all the necessary artifacts.

Vadim Spirkov
York University School of Continuing Studies

Murlidhar Loka
York University School of Continuing Studies