# Machine Learning Engineer Nanodegree

## Capstone Project

Filipe Reis
May 21, 2017

## I. Definition

### Project Overview

Humans have the great capability of using their senses to interact with the world. One specific interaction is through sounds, as one is capable of easily determining a situation by just listening to the ambient sounds, in other words, we perform well the task of classifying sounds. As machines become increasingly more present and important on our lives, the need of interaction with the ambient grows more relevant, making Sound classifiers important. Traditionally, sound classifiers have been widely specially for automated call center and applications alike but there is still great potential to apply these machines to many other fields, such as hearing aid systems. In addition, current sound classifiers are heavily based on classical preprocessing of data [1] and rudimentary machine learning methods, which is good because made even hardware-level implementations possible but has a performance tradeoff. Also, there has been incipient application of newer technologies such data intensive machine learning and deep learning but some recent work such as using deep learning to improve hearing aid [2] and deep learning techniques for large audio datasets [3].

As mentioned earlier, there is enormous potential to apply sound classifiers to hearing aid systems as its users may need more than just amplification of sounds, as our brains usually performs some filtering to what is heard and this filtering may also be compromised, generating hearing selectivity problems. This problem motivates me as I have people in my family with selective hearing problems and even I have a mild level of hearing loss on my right ear, which makes me extra excited with the potential that this application has.

### Problem Statement

Sound classification is a task that traditionally relies heavily on dimensionality reduction strategies as its length is quickly converted into an

enormous number of samples due to the high sample rates necessary for acquisition without loss and aliasing. The traditional workflow for this classification is to calculate the Mel-frequency cepstral coefficients, which are coefficients obtained after transformations over the frequency response (resultant of a Fourier transform) and which approximate the human auditory systems' response [4] and use these coefficients as features on a machine learning classifier to generate labels corresponding to the sounds.

Sound classification imposes a quantifiable, measurable and replicable problem as it takes a standard audio file as input as returns a label to it. As the training dataset is labeled and contains 10 different classes, if either a new or an existing sound of the dataset is presented, the system will return a label, which is the corresponding sound class of it and this label can be compared to the actual sound contained on the file.

As for a typical classification problem, one can measure the accuracy of the results by comparing the predicted and original labels of the test files and even further tests could be carried out by introducing new samples from other sources.

### Metrics

To evaluate the performance of both the benchmark and the solution model, one approach is to consider the accuracy metric as its results for the benchmark are already presented on [5], allowing comparison between the two models. On the other hand, accuracy is not ideal as a performance metric for this unbalanced dataset as it could divert the results from the correct ones as it only accounts for correct answers over samples. This would allow the algorithm to, for instance, only choose the most recurrent class and achieve good precision even though it did not actually classify anything. To mitigate this effect, F1 score will be used to assess the overall performance of the model, as this method considers true positives, false negatives and false positives into account, resulting on a much more reliable result.

Additionally, a confusion matrix of the results will be evaluated to help analyzing the decisions the classifier made. This visualization allows quick and precise insights regarding misclassifications as it shows the correlation between inputs from each class and the classification given. The confusion matrix will also show if the classifier is opting only for a few classes or if it is working as expected.

## II. Analysis

### Data Exploration

The dataset considered for this project is the "Urban Sounds 8K", provided by NYU. This dataset was created from public samples available at freesound.org and is composed by 8732 wav (around 6.6GB) files divided into 10 **unbalanced** classes, as shown in Figure 3. This dataset excels others, such as AudioSet [4], by providing raw samples, which allow full experimentation on the data preprocessing stage.

As the data was collected throughout different contributions, it is not uniform, varying specially regarding duration and sample rate, which impose a special challenge as the first implies on number of features not constant and the latter may change important characteristics of the model as the time interval between each sample gets different. One way of dealing with the different signal duration would be to select only the samples, which have the most recurrent length, while other, and the chosen for this project, is to standardize the length during the preprocessing of the data.

The data is divided in 10 folds but will be reorganized into one group and then reordered into three sets, validation (70% of the total), train (20% of the total) and test (10%). Additionally, to provide better handling of data and allow for faster reprocessing, the three sets will be saved as pickle files as the time to load all sound files to memory is much higher than of loading three pickle blocks.

## Exploratory Visualization

The UrbanSounds8K dataset is composed by audio samples available at a public sound repository, selected only by their content class. This source variation creates a disadvantage of having files with different parameters such as sample rate, signal duration and file type, as shown on Figure 1.

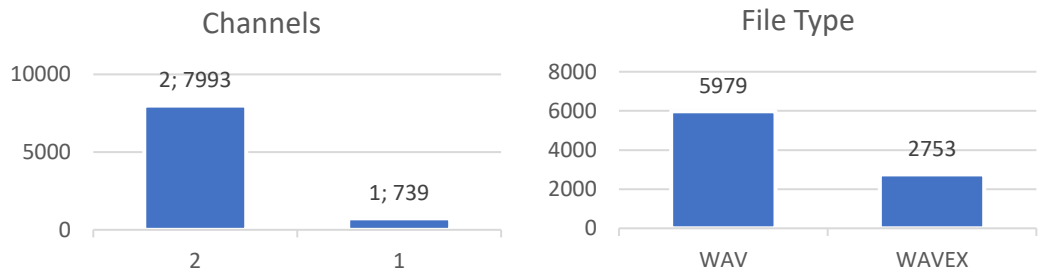


*Figure 1 - Dataset distribution regarding number of channels and File Type*

Most samples have two channels, which means that some strategy will have to be carried out to transform into only one channel. As for file type, most data is single WAV but a significant amount is WAVEX, which is much similar to the first with the exception of having higher bit depth, which can be troublesome when using specific libraries to open the file. In addition, duration and sample rate distribution for the dataset.

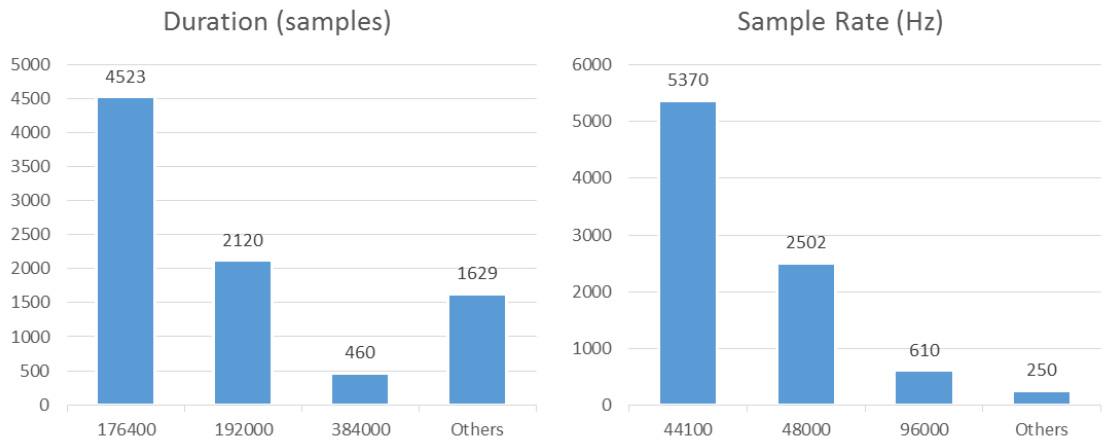


*Figure 2 - Duration and sample rate distribution for UrbanSounds8K dataset*

The duration variation among the dataset samples can be troublesome, as the input for the processing algorithms must have a fixed size, so a strategy will be required to deal with this size variation before proper use. In addition, the sample rate also varies considerably along the dataset, which creates a need to create some sort of compensation as the time representation for points of samples with different sample rates will be different.

Finally, Figure 3 shows the class distribution along the dataset. Clearly, the dataset sample distribution is not uniform, as classes car horn and gunshot have much less samples, while Siren has almost as much the seven others.

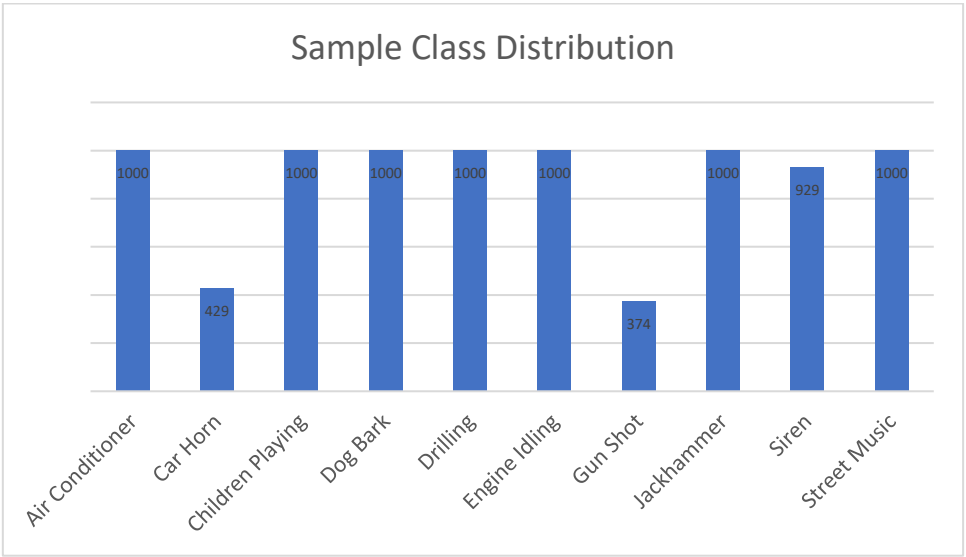


*Figure 3 - Sample Class Distribution for UrbanSounds8K dataset.*

## Algorithms and Techniques

In this project, the necessity for data preprocessing is clear, as there is a significant difficulty on classifying sounds by temporal series, due to their

distinction being most significant on frequency domain. For data preprocessing, the following techniques will be considered:

- Principal Component Analysis (PCA): feature selection technique based on orthogonal transformations, which generates coefficients capable which account for parts of the features variance [7]. The number of components to be considered will be defined by assessing the explained variance of each element and selecting those with expressive variance;
- Independent Component Analysis (ICA): much used with audio for source separation, where multiple independent sources are acquired. ICA is based on the assumption that the data is a mixture of multiple nongaussian and mutually independent variables [8];
- Mel-frequency cepstral coefficients (MFCC): this technique involves calculating coefficients obtained after transformations over the frequency response (resultant of a Fourier transform). These coefficients are calculated to be approximated to the human auditory systems' response [4]. These coefficients are the most used on classic sound classification devices because they are relatively simple to calculate and have impressive performance on representing significant attributes of the sound.

To classify the data, performance of the following algorithms will be evaluated, considering the default parameters as per scikit-learn library:

- Decision Tree Classifier: decision trees are classifiers with a tree-based structure, working in layman's terms as a series of yes and no questions, which allow to navigate through the tree and achieve a classification label at the end. This method is computationally good and extremely helpful on situations where the client wants to understand how the classifiers decides, as its steps are clear and easy to understand. The parameters used were the default ones, with gini function to measure the quality of the splits, no limitation for maximum number of splits and depth and equal class weights;
- Support Vector Machines (SVM): are classifiers based on linear separation of the data, either on the original data space or on others mapped with kernel functions, as the original may not be linearly separable. These classifiers are extremely versatile as the use of kernel function mapping allows the classification of the most varied datasets. We used the C-Support Vector Classification algorithm of the SVM class as, although it has high complexity for high number of features, it presented good results for the data size used. The kernel function considered was radial basis function (RBF) with auto gamma (1/n_features) and penalty parameter C of 1.

- Multi-Layer-Perceptron(MLP): this classification algorithm is based on presenting the samples to many mathematical functions (neurons) and analyzing whether the result of this function is higher or lower than one threshold. This output is then fed to another neuron, changing only that this output is multiplied by a specific number (weight). This process continues as there more layers of neurons. The functions are defined as parameters, while the weights are defined iteratively, starting random but small and being adjusted using a backpropagation method, which is deriving the error function to minimize the total error of the process. Initially, a small number of groups of functions (hidden layers) was considered, only 100, 75, 50 and 25. The initial activation function was relu activation function, which is defined by f(x) = ln(1+$e^x$) [9]. Additionally, the method used for the gradient descent is called adam solver. For penalizing the errors, we used a regularization term of 0.0001. Finally, the batch sizes considered were automatic, the learning rate constant and a maximum of 200 iterations on each run.

### Benchmark

The creators of the dataset provide results for their own classifier trained on the data [5], which indicates a great comparison point for this project. The reference presents best results using Support Vector Machines using Radial Basis Function Kernel obtaining global accuracy 70%.

## III. Methodology

### Data Preprocessing

As the files had varied sizes, the most frequent one was considered (176400) and all the signals were read using this number of frames, either padding with zeros the smaller ones or discarding the end of bigger ones. Additionally, the dataset files were all normalized, which excluded the need to normalize them during this phase. For samples with two channels, the average between the two was considered, to account for possible differences between the two channels and consider information contained on the sample. On the other hand, different frequencies were handled as a parameter during the MFCC phase.

A feature reduction schema is vital to this project as the samples have a substantial number of features, which imposes a great difficulty for regular classification. To choose between PCA, ICA and MFCC, a SVM algorithm with default parameters was used to classify the samples and enquire which method

or configuration provided the best accuracy, yielding the results displayed on Figure 4.

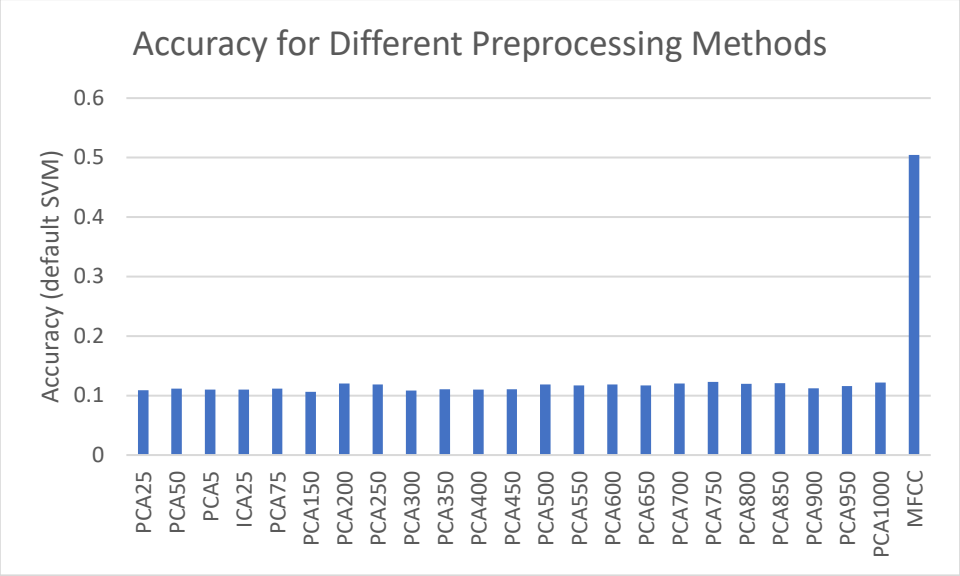


*Figure 4 - Accuracy for different preprocessing methods using a default SVM classifier.*

Therefore, MFCC preprocessing method was chosen and applied to the project as its performance excels greatly both ICA and PCA methods.

## Implementation

To separate the sets between training, testing and validation, the TrainTestValidationSeparation module from scikit-learn was used to group the metadata of the sound file into these three groups. These slices of the metadata were then used to load the correct audio files for each of the groups. As the files are quite big and loading them requires lots of RAM memory, the three groups were saved as a single pickle file, speeding the subsequent runs as it was more optimized to open one big file than thousands of smaller ones. For loading the audio files, the soundFile library [9] was used.

With the samples loaded, scikit-learn modules for PCA and ICA were applied. For the PCA method, the number of components was varied from 25 to 1000 to verify performance, as per refinement section of this report. For the MFCC preprocessing, pythonSpeechFeatures [10] libraries were used and the components for each element saved as a npy file, to avoid the need to recalculate after each run. Additionally, the parameters considered for the MFCC varied according to the input file.

For the processing, SVM, MLP and Tree Classifier modules from scikit-learn library were used as they provide good performance and are easy to use. Additionally, accuracy and F1 score were calculated using the respective modules

from scikit learn and a confusion matrix was created using seaborn and matplotlib.

## Refinement

During the preprocessing method analysis, the number of components for PCA was changed between 25 to 1000, aiming higher accuracy considering the standard SVM classifier. The same should have been done with the ICA method but my computer ran out of RAM memory for higher number of components.

With the optimal preprocessing, each one of the considered classifier was applied to the dataset, obtaining the results shown on Figure 5

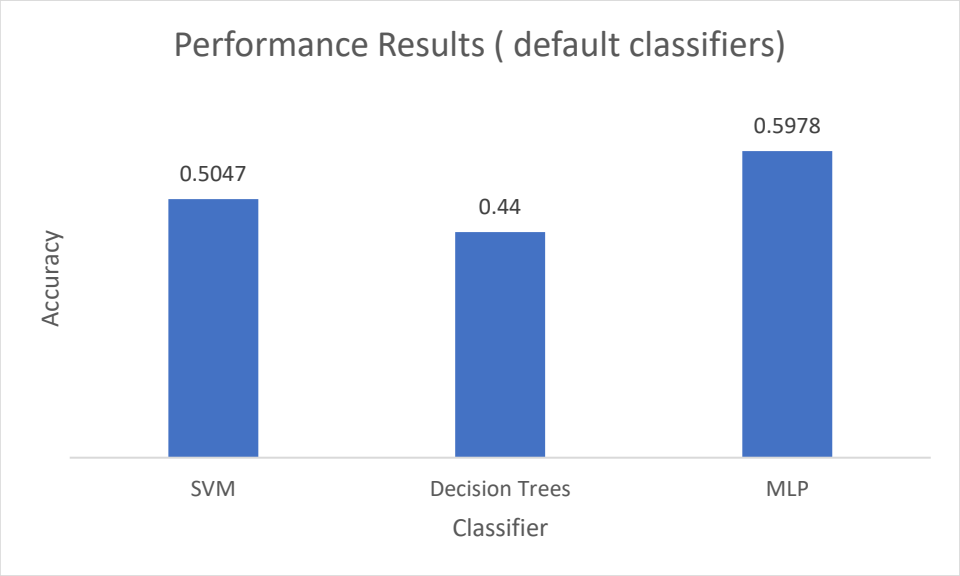


*Figure 5 - Performance Results for the three standard classifiers after MFCC preprocessing*

After the option for the MLP algorithm, extensive grid search was performed to find the best parameters. The parameters considered were:

- Hidden layer size: affect the quantity of neurons on each layer of the MLP. Although increasing these sizes slow the process, it also provides greater capacity to understand the data;
- Activation function: function that is applied to each node to verify if the neuron is active or not based on the inputs;
- Solver: solver used for weight optimization;
- Learning Rate: defines the value by which the weighs of each node is multiplied after each learning iteration.

Of these parameters, the most impactful ones were activation function and hidden layer size. Figure 6 shows accuracy scores for the test dataset considering different hidden layer sizes.

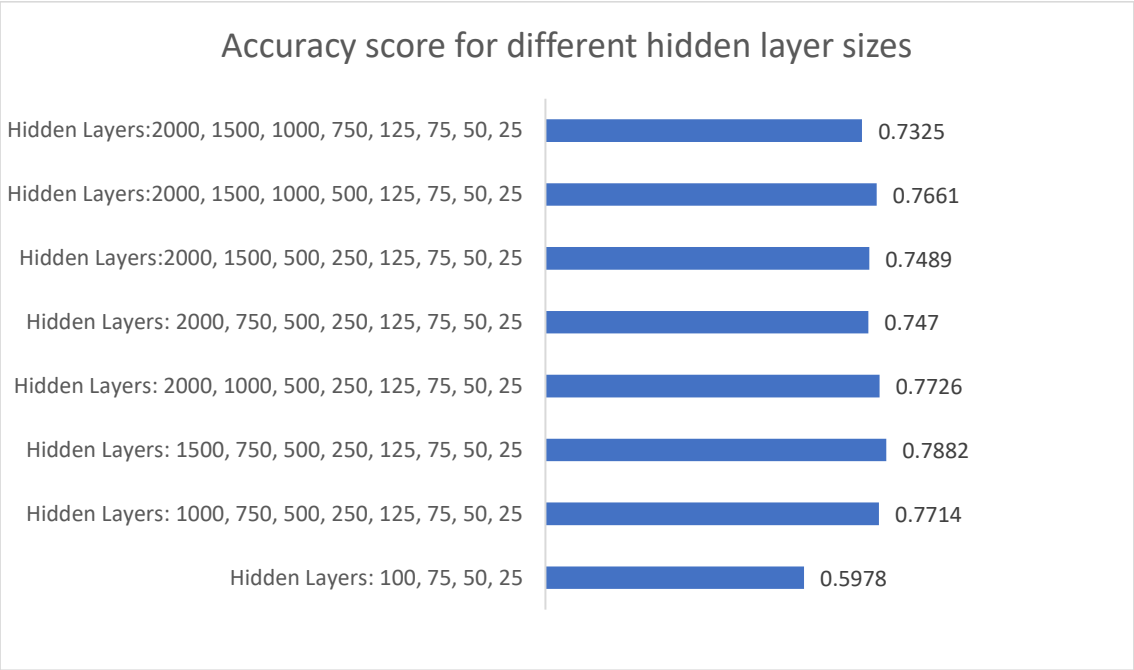


*Figure 6 - Accuracy score for test set using different hidden layer sizes.*

Along with performance improvement, increasing the hidden layer sizes also demand a bigger run time, as there are more nodes to be processed and the backpropagation process takes more resources. Although increasing the layer sizes increases performance, there is a threshold when it makes the classifier perform poorer, such is the case with 2000, 1000, 500, 250, 125, 75, 50, 25 and 1500, 750, 500, 250, 125, 75, 50, 25 as the first is around 1% less accurate than the smaller second. The configuration chosen was the one with best score, 1500, 750, 500, 250, 125, 75, 50, 25, which scored 0.7882 for the test set.

Finally, the extensive grid search resulted on an optimal model with constant learning rate, adam solver, tanh activation function and hidden layer size of 1500, 750, 500, 250, 125, 75, 50, 25.

# IV. Results

## Model Evaluation and Validation

The final model is reasonable considering the proposal as it can classify sounds from the different classes with good accuracy. Having the model saved makes it easier to apply it to new data and its performance with the validation set, which was only used after the final model was defined, was close to the results with the training set, 78.43%, which indicates the capacity to generalize well to unseen data. Additionally, the f1 score using weighted averages to account for

the different classes was close to the accuracy, 78.45%, confirming that the model is precise.

### Justification

The presented model excels the performance of the benchmark solution by 11%, achieving 78.43% versus original 70%, which is a great result considering that there are still ways to improve the method.

The final solution is able to solve the problem as it is capable of classifying sounds from all classes with good accuracy.

## V. Conclusion

### Free-Form Visualization

This project achieved satisfactory performance classifying sounds belonging to the different classes. To analyze which mistakes the model makes, Figure 7 shows a confusion matrix for the validation set of the final model.

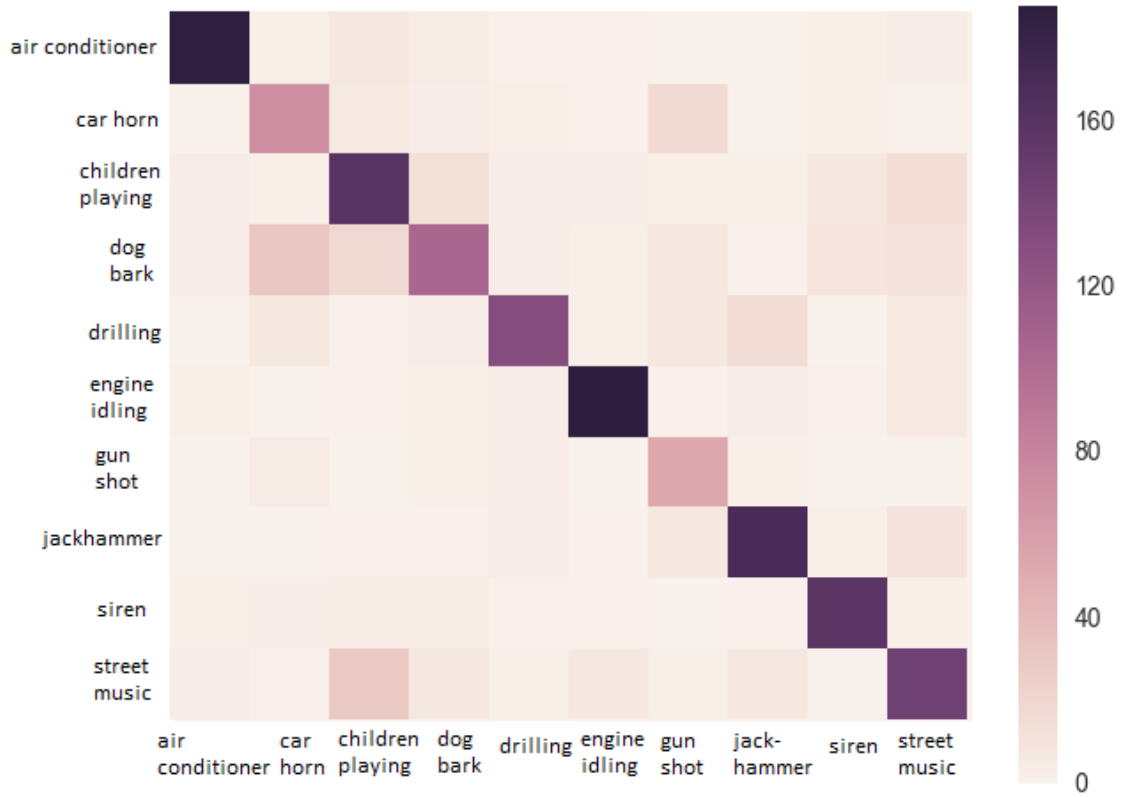


*Figure 7 - Confusion matrix for final model*

Figure 7 shows that the classifier struggles at differentiating dog barks from car horns and children playing, which confirms the fact that the MFC coefficients have a response close to human hearing, as these sounds could be mistaken by a human listener. Other significant source of error is between children playing and street music, which is also acceptable considering humans hearing abilities. Additionally, street music often is misclassified, which may be due to its diverse composition. Finally, car horn is commonly mistaken with gunshots, which is acceptable considering that both are short and high-pitched sounds. Additionally, gunshot and car horn are classes with significantly less samples than the others, which may contribute to a higher misclassification level.

## Reflection

This project was a wonderful experience of working with a real case and with its associated problems. As audio files are composed by a gigantic number of samples, feature reduction schemes are extremely necessary. Although PCA and ICA help to achieve modest improvements, using the right tool for the task generates much better results, as MFCC enormously outperformed these other techniques. The classification algorithms considered provided similar performance but neural networks showed their great capacity to generalize well even complicated datasets, especially with fine tuning of its parameters. The results obtained were good as they excelled the proposal from the creators of the dataset and ended up with errors close to what humans could normally mistake. The trickiest part of the project was handling with the size of the dataset and the amount of resources needed to process its data. Finally, the achieved model fit the expectations for the project, as it is capable of classifying sounds belonging to the 10 different classes with satisfactory performance.

## Improvement

This project leaves many possibilities for improvement, being the most relevant:

- **Possibility to process audio with only a few seconds of delay:** to improve the usability and range of application in which this project can be applied, online audio processing would be ideal. This way, the classification could be used in many instances such as subtitle creation.
- **Implementation of blind source separation:** in this version, ICA was applied only as feature reduction, but it could be also used to separate sources of the sounds. Although this would require a database with more sound channels and mixed sources, it would make it useful for hearing aid products as it could only amplify specified sound sources, greatly improving the systems for people with loss of selectivity;

- **Improvements on performance**: this code has lots of room to improvements with performance, both on running time and on used resources.
- **Implementation of deep neural networks**: as an alternative approach to preprocessing using MFCC and classifying with MLPs, deep neural networks could be applied with a possibility of yielding greater results, as described on the article about deep learning reinventing hearing aid [2]