# Sound Classification

**Jake Garrison**
12/6/2016
Department of Electrical Engineering
University of Washington
omonoid@uw.edu

## Abstract

Classifying sound can be a difficult task even for a human. Environmental sound is even more challenging given the uncontrolled conditions and background noise. In this paper, several machine learning techniques are explored for deriving sound features and using them to train a sound classifier. This paper is primarily focused on techniques that characterize sound using the short time Fourier transform (STFT) representation which enables sound to be treated as a spectrogram image rather than time series data. Classifier models are trained using deep neural net, as well as more traditional machine learning approaches.

## 1 Introduction

### 1.1 Inspiration

Deep neural networks (DNNs) have enabled many classification applications in the last few years. Convolutional Neural Networks (CNNs) coupled with high-performance GPU's have been crucial for creating modern image classifiers that when trained on large datasets such as ImageNet, are capable of classifying hundreds of different objects with acceptable precision. Audio classification has also benefited from modern advancements, but not nearly to the level of image classification. Part of this is due to a lack of labeled audio data, and additionally, there seem to be less meaningful applications to audio classifiers outside of speech. I am personally interested in training and deploying audio classifiers for use in medical screening applications, music composition, and supplementing real time image classification tasks.

### 1.2 Use Cases

Many respiratory illnesses can be screened using sound via smartphone or other portable low powered hardware. Coughs, for example, can be classified into 'healthy' or 'unhealthy' based on the sound; with 'unhealthy' coughs potentially further classified into infections such as pneumonia or tuberculosis (TB). Cough classifier hardware could then be deployed to endemic areas to help track the origin and spread of contagious infections like TB. Additionally, expensive medical hardware such as spirometers (used for measuring lung functionality) could be replaced with smartphones and enable daily screening for people with conditions such as asthma, or to help diagnose such conditions. Such technology could also be used to aid sleep studies for people suffering from disorders such as sleep apnea.

Classifying sound in order to distinguish between different instruments, rhythms and genres could also be revolutionary for music composition, discovery, and sorting. The record industry could use classification technology for copyright claims and musicians could use it as a tool for composition and sound generation.

Systems that traditionally rely on computer vision such as home surveillance and autonomous cars could be supplemented or replaced with audio classifiers which require much less processing power and storage space, and also can be used to easily triangulate the source of a sound event (with multiple microphones).

### 1.2 Related Work

Audio classification research has mostly fallen under the automatic speech recognition category, enabling systems such as Apple's Siri or Amazon's Alexa to respond to requests via speech. Most of this research is focused on segmenting speech into phonemes and then classifying words from there. Some speech research results such as Cepstral processing or the Mel-frequency scale can be used for broader audio classification problems and are used in this paper. There are examples of other classification problems, such as bird calls or musical instruments, but many of them are very specific to a particular domain.

The Urban Sound Dataset [1] used in this paper, was created in 2014 and select manuscripts have since been published on the topic of urban sound classification. The authors of the dataset published a paper [2] on the topic of using DNN's for classification, which inspired and set the baseline for the experiments in my paper.

## 2 Dataset

The Urban Sound dataset contains 8732 labeled sound excerpts less than 4 seconds each from 10 classes: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gunshot, jackhammer, siren, and street music. Note there were not many gunshot examples, so it was later removed. Since these recordings are collected from crowd-sourced data, the quality, volume and background noise varies significantly between samples, making this classification problem more difficult and realistic. The classes are drawn from the urban sound taxonomy described in the accompanying dataset article [1], which also includes a detailed description of the dataset and how it was compiled. The audio files are in WAV format, and 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).

Example plots for each class are shown below (Figures 1 & 2). The wave plots show the signal's amplitude versus time, and the spectrograms show how the magnitude of frequencies (y-axis) change over time (x-axis) where the magnitude scales from blue (low) to red (high). The spectrograms tend to provide more information about the sound since it includes frequency (pitch) content.

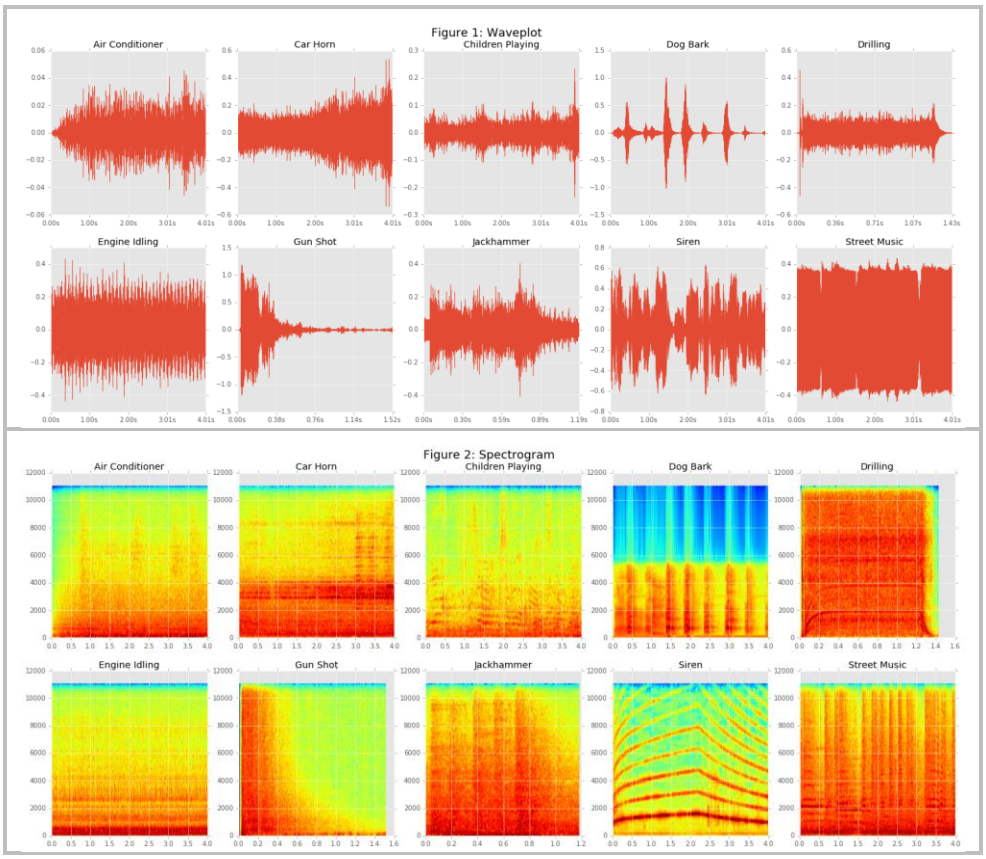



Figure 1 and 2: Show the waveform and spectrograms for the 10 classes

The plots above help to visualize the different attributes of the classes in the dataset, but further feature extraction is necessary in order to reduce the tens of thousands of data points for each audio clip into a smaller more reasonable set.

## 3 Feature Extraction

To reduce the points into different feature sets, various methods are utilized. The main idea is to compress and filter the data in some way such that the redundant information is removed, leaving only the most pertinent information necessary for classifying the sound. Following the feature extraction, each sound file can be represented with the same set of features, making them much easier and efficient to compare and classify.

The Librosa library was used for feature extracting as it comes with several useful methods for representing audio in different ways. The methods used in this paper are outlined in the table below with links to more details. These methods are all related to the frequency content of the audio. Other methods not explored in this paper could be used to further derive features from the natural rhythm or repetition of sounds. This is less useful for this application since most of the classes are impulses or aperiodic by nature.

| | |
|---|---|
| Mel-frequency cepstral coefficients (MFCC) | Coefficients usually used for speech recognition |
| Chromagram of STFT | Projects bins representing the 12 semitones (chroma) of the musical octave |
| Mel-scaled power spectrogram | Uses Mel scale to provide greater resolution for the more informative (lower) frequencies (based on human ear). |
| Octave-based spectral contrast | Focuses on musical octave patterns |
| Tonnetz | Estimates tonal centroids as coordinates in a six-dimensional interval space |

Table 1: Feature Extraction Methods

Two separate feature sets were derived from the dataset. The first representation uses each of five methods outlined above and then concatenates, averages and flattens them to give a consistent feature vector of 193 values for every processed audio clip. These features can be thought of as an ensemble of all methods, averaged so the information can be represented with fewer data points. This feature set is referenced as the 'mean' set.

The second set is a 128 x 128 representation of the Mel-scaled power spectrum that can be visualized as an image as shown below. This feature set is referenced as the 'specs'. Mel-scaled power spectrograms were used in place of regular spectrograms because they tend to be more sparse and represent the same information with less data.

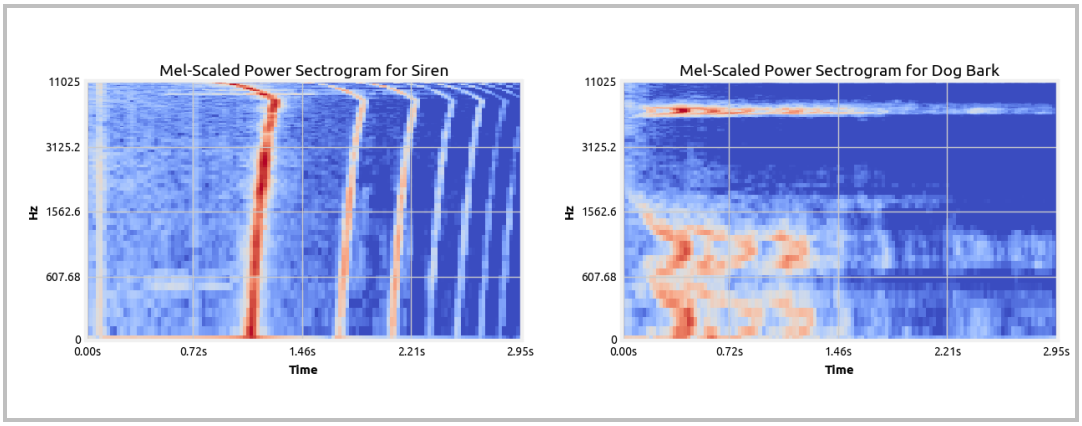



Figure 3: Mel-scaled power spectrogram

Deriving both of these feature sets takes over an hour on a high-performance workstation, but only needs to be done once and then stored as a file for later access.

# 4 Classifiers

## 4.1 Classifier Overview

There is a broad range of algorithms that fit under the category of multi-class classifiers. The Scikit-Learn library was used for a majority of them and the remaining DNN based models were implemented using Keras with a Tensorflow backend. The discussion of classifiers is delimited by the library for clarity.

### 4.2 Scikit-Learn

The Scikit-Learn library makes it convenient to try several different classifiers without writing much code. Ten different classifiers were evaluated including: Logistic regression, SVM, K nearest neighbors, Decision trees, Random forests, AdaBoost, Naïve Bayes, Gradient Boosting, Linear discriminant analysis, and Quadratic classifier. See the Scikit-Learn documentation for information on these classifiers.

Each of these classifiers was trained and evaluated on the 'mean' feature set and the 'spec' set (flattened into a single vector). Note that not all classifiers were evaluated with the 'spec' set due to training time constraints resulting from the high resolution of the 'spec' set.

### 4.3 Keras Deep Learning

Keras enables complex deep learning models to be implemented and evaluated with a few lines of code. As a result, the majority of the work was spent on tuning and optimizing the networks rather than creating them from scratch. That being said, obtaining results from the Keras models took significantly more effort than those made with Scikit-Learn.

Two model architectures were explored, each having two variants for a total of four DNNs. The simpler of the two architectures is the feed forward neural network (FNN), and the second is a convolutional neural network (CNN). In general, CNN's are better for image input as it convolves a filter over the input extracting patterns in order to train weights. Details of the four variants of the two architectures are presented in the table below. All models used the Adam optimizer and used some form of dropout after each layer. The models used Softmax on the last layer, ReLu activation on preceding layers and were evaluated with a cross entropy loss function.

| | |
|---|---|
| **FNN 1 (3 layer)** | **Input**: Spec features flattened to length 16384 (128*128)<br>**Hidden layers**: 3 each with 512 units<br>**Batch size:** 128<br>**Learning Rate:** 0.0005<br>**Training Epochs:** 30 |
| **FNN 2 (3 layer)** | **Input**: Mean features length 193<br>**Hidden layers**: 3 with 100, 200, 200 units respectively<br>**Batch size:** 100<br>**Learning Rate:** 0.0005<br>**Training Epochs:** 30 |
| **CNN 1 (5 layer)** | **Input**: Spec features 128x128<br>**Convolutional layers**: 3 with 24, 48, 48 filters respectively<br>**Filter Size:** 3<br>**Max Pooling:** 4x2<br>**Fully connected layers:** 2 size 64, then 10 for output layer<br>**Batch size:** 30<br>**Learning Rate:** 0.001<br>**Training Epochs:** 30 |
| **CNN 2 (8 layer)** | **Input**: Spec features 128x128<br>**Convolutional layers**: 6 each with 32 filters each<br>**Filter Size:** 5<br>**Max Pooling:** 4x2<br>**Fully connected layers:** 2 size 128, then 10 for output layer<br>**Batch size:** 30<br>**Learning Rate:** 0.0005<br>**Training Epochs:** 20 |

Table 2: Deep learning architectures

# 5 Training and Tuning

## 5.1 Training Split

The dataset is comprised of 10 ‘folds’ each containing around 800 samples. The data was split such that 80% of the data (8 folds) was for training and the rest for testing. This way after the models train, they can be evaluated on the remaining unseen data. The models were evaluated with three-fold cross validation to better represent the limited data and prevent overfitting. Unfortunately, there are less than 10 samples of gunshots, so the classifiers did not have adequate training examples to learn and classify a gunshot.

## 5.2 Hyperparameter Tuning

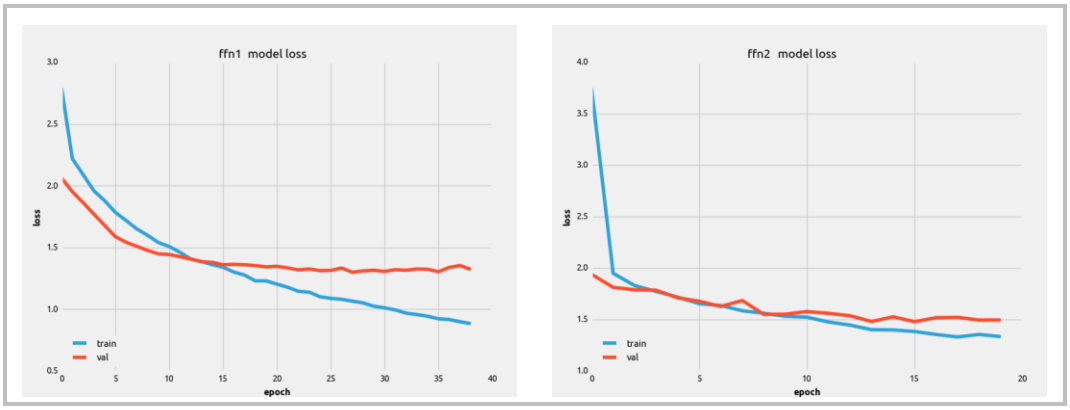
In order to get the most out of the classifiers, model tuning was necessary, especially for the DNN’s. For the first few experiments, parameters were manually tweaked in order to observe the effect on performance. Once the models were decent, further tuning was utilized.

Many of the parameters listed in Table 2 were experimentally tuned using both grid search and random search. In grid search, an array of candidates for each parameter to be tuned is declared and each combination is evaluated using three-fold cross validation. With this method, tuning three hyperparameters with three candidates each will evaluate the model 3x3x3=27 times since for each combination, the model is evaluated three times due to cross-validation. This can be time-consuming, but can drastically improve the model’s accuracy. Random search was used for some parameters as well since it has been proven to converge to optimal parameters quicker than grid search [4]. For the CNN, the accuracy increased by 15% after these techniques were employed.

A select few of the Scikit-Learn classifiers were tuned beyond initial settings. The ones that showed most promise (gradient boosting, logistic regression LDA and SVM) were manually tweaked. For logistic and gradient boosting, grid search was used to parameterize the learning rate and the number of estimators. Note that random forests do not require tuning making it easy to employ.

## 5.3 Training and Validation Loss

The log entropy loss function was monitored while training the DNN’s and an early stop function was used to stop training once the validation loss started increasing. This helped tune the number of training epochs and also prevented overfitting. The noise in the loss plots is due to the batch size used in stochastic gradient descent.



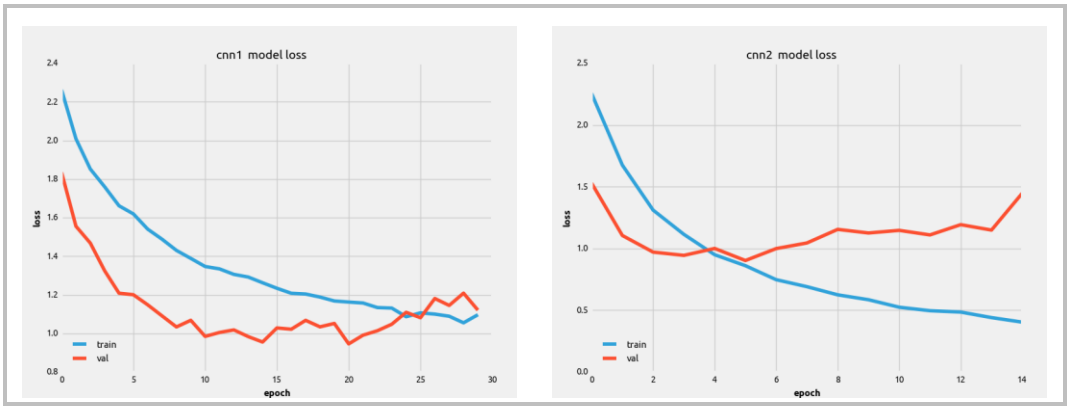


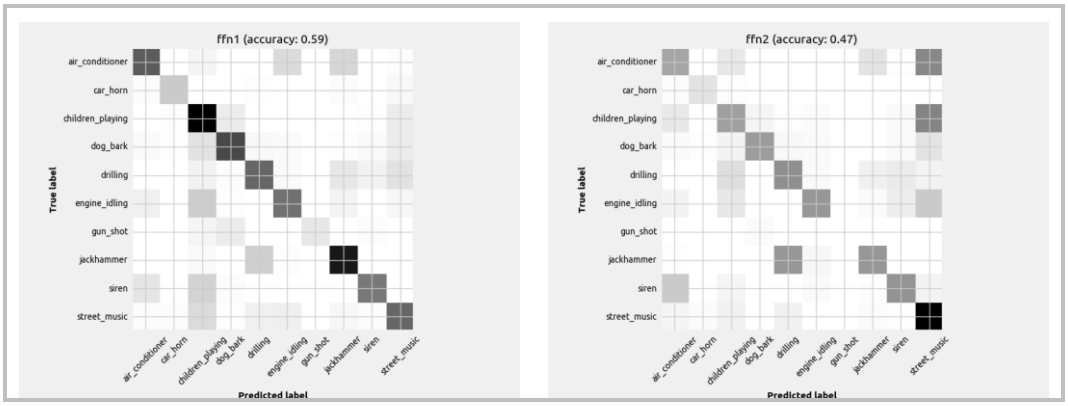

Figure 4: DNN Loss plots

Ideally, the validation line tracks well with the training line and they both increase as epochs increase. CNN2 looks like it may have an overfitting problem since the loss function doesn't track very well to the training line. While the FFN2 validation line seems to be well correlated with the training line, it stops after only 20 iterations, indicating it has stopped learning sooner than FFN1, which goes to nearly 40 epochs. This generally results in a worse model overall. This observation is also present when comparing CNN1 and CNN2.

# 6 Results

## 6.1 Classifier Performance

The results are published in figures below. The results of each model are displayed as a confusion matrix with the model name and accuracy in the title. The confusion matrix helps indicate what classes the model struggled to identify. Note there are different amounts of each class, so a darker box doesn't necessarily equate to better prediction. The missed predictions (elements not on diagonal) are the most insightful way to visualize the limitations of the model. As stated earlier, this dataset had very few gunshot examples and so it was removed as a class in the context of this paper.



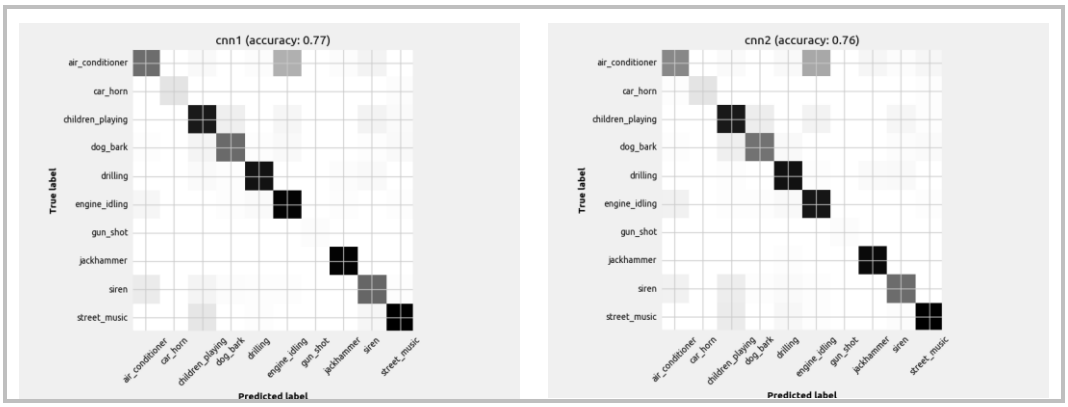
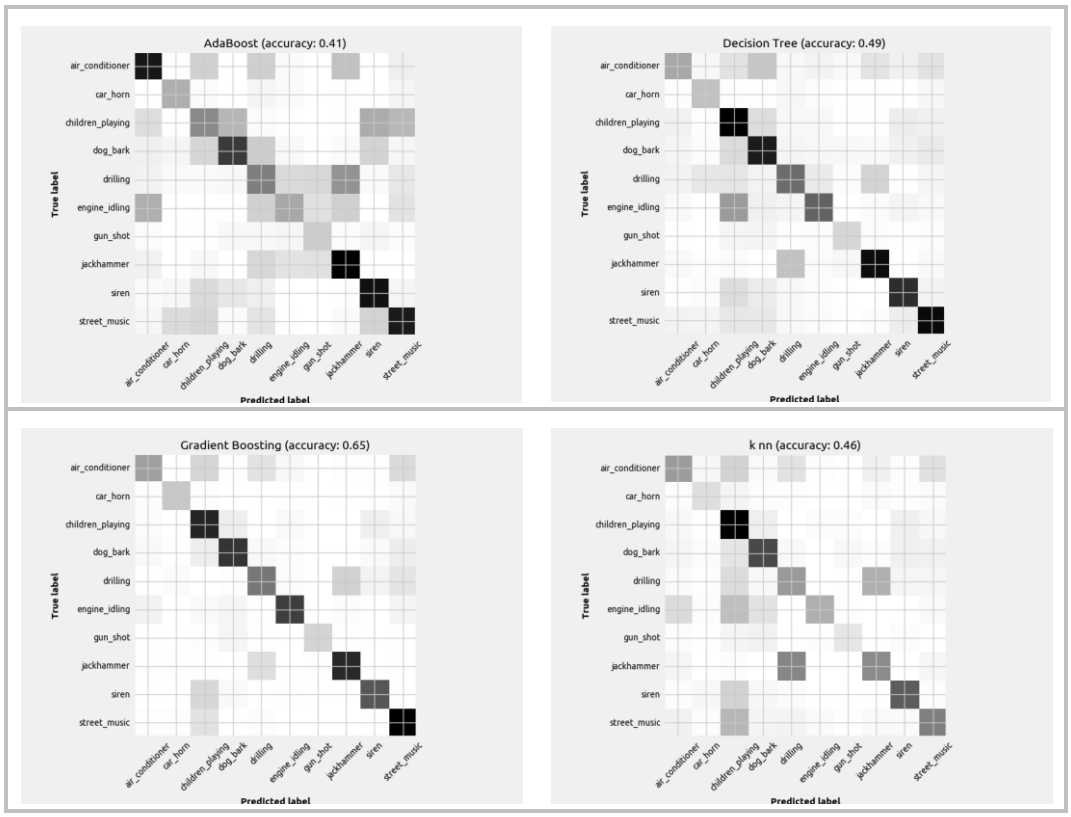



Figure 5: DNN results

As expected, the CNNs outperformed the FFNs. Additionally, the CNNs are nearly identical in accuracy and confusion matrix.



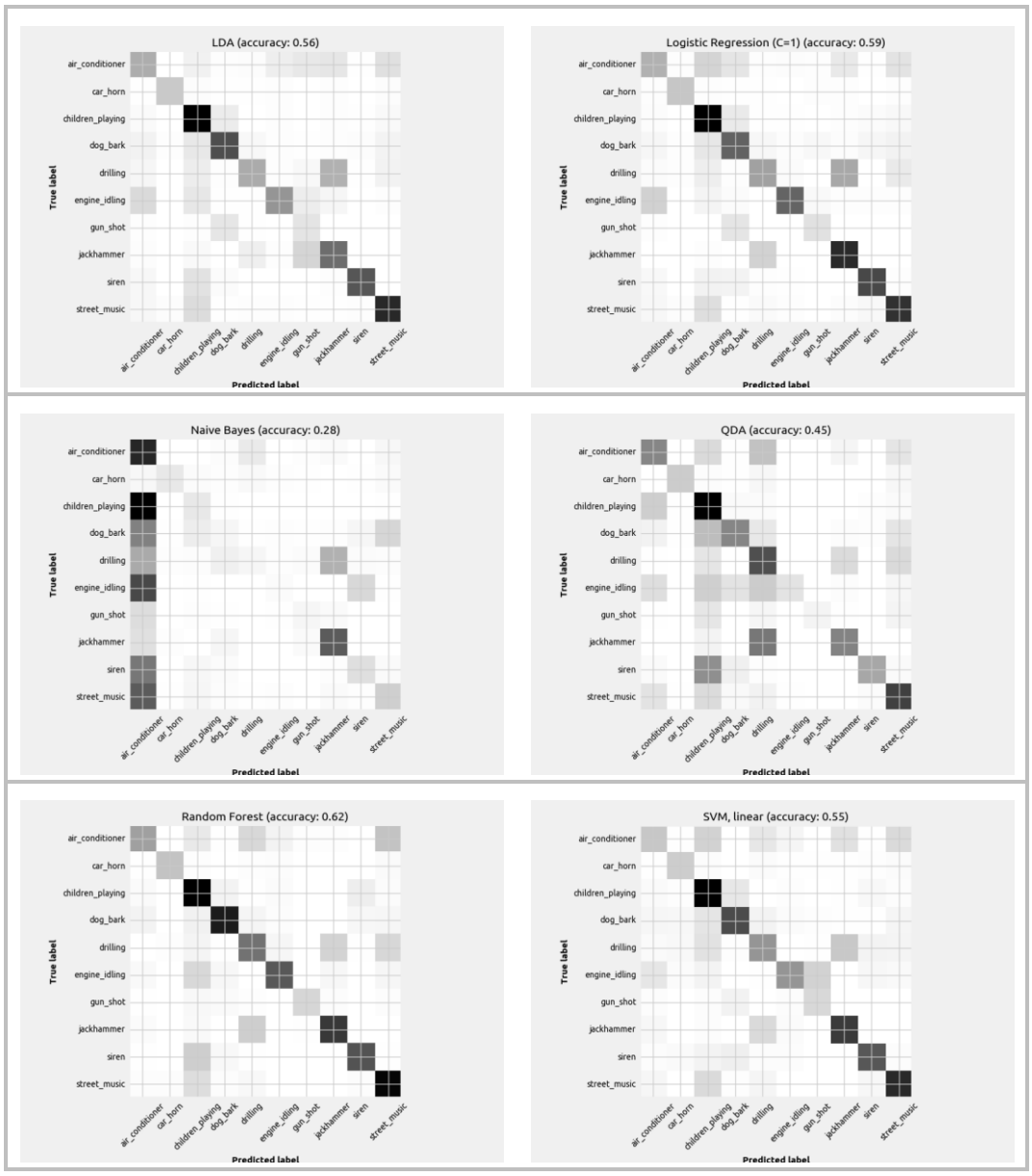
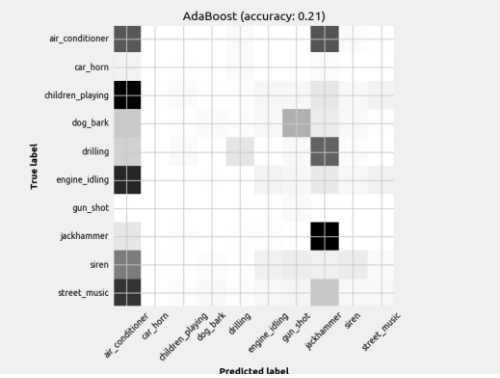
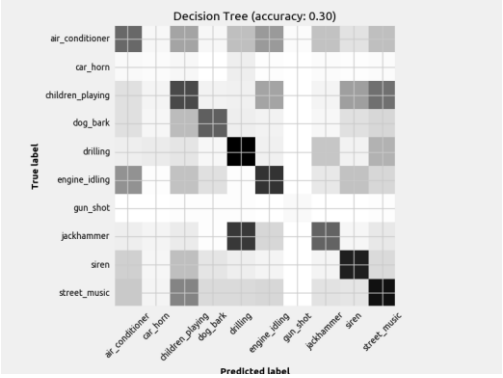
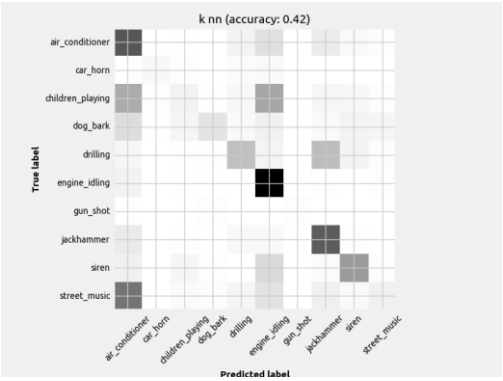


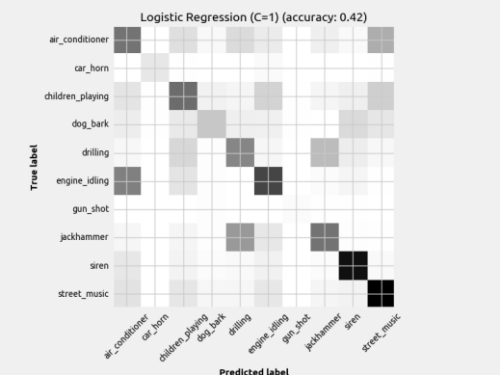

Figure 6: Scikit-Learn classifiers with mean feature set input

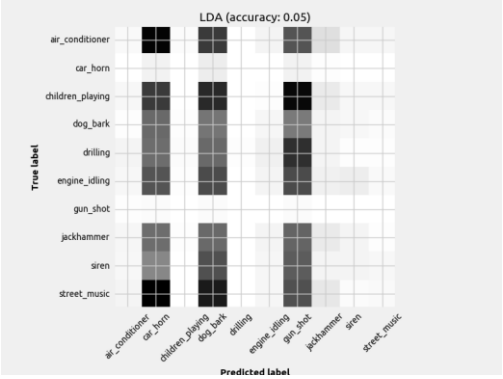
The distinguishable diagonal on most of the confusion matrices indicates the models (aside from Naïve Bayes) did indeed learn to classify the sounds to some degree. Also, notice how some models performed better at detecting certain classes when compared to another model with similar overall accuracy.

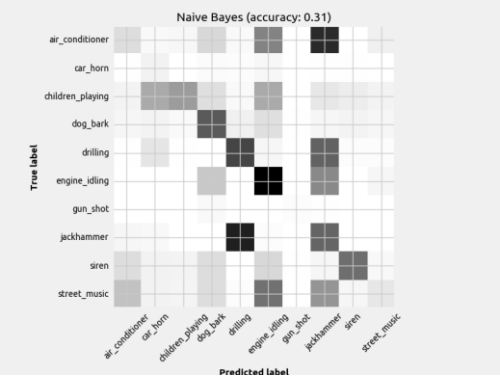

AdaBoost (accuracy: 0.21)
air_conditioner
car_horn
children_playing
dog_bark
drilling
engine_idling
gun_shot
jackhammer
siren
street_music
True label
Predicted label

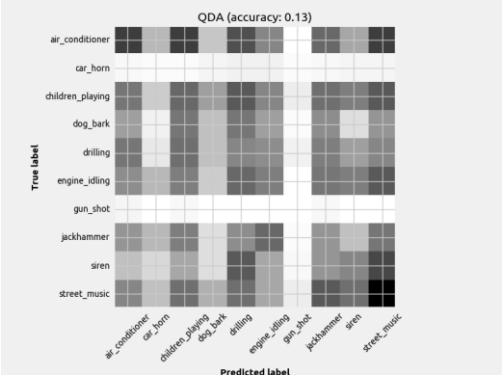

Decision Tree (accuracy: 0.30)
air_conditioner
car_horn
children_playing
dog_bark
drilling
engine_idling
gun_shot
jackhammer
siren
street_music
True label
Predicted label


k nn (accuracy: 0.42)
air_conditioner
car_horn
children_playing
dog_bark
drilling
engine_idling
gun_shot
jackhammer
siren
street_music
True label
Predicted label


Logistic Regression (C=1) (accuracy: 0.42)
air_conditioner
car_horn
children_playing
dog_bark
drilling
engine_idling
gun_shot
jackhammer
siren
street_music
True label
Predicted label


LDA (accuracy: 0.05)
air_conditioner
car_horn
children_playing
dog_bark
drilling
engine_idling
gun_shot
jackhammer
siren
street_music
True label
Predicted label


Naive Bayes (accuracy: 0.31)
air_conditioner
car_horn
children_playing
dog_bark
drilling
engine_idling
gun_shot
jackhammer
siren
street_music
True label
Predicted label


QDA (accuracy: 0.13)
air_conditioner
car_horn
children_playing
dog_bark
drilling
engine_idling
gun_shot
jackhammer
siren
street_music
True label
Predicted label

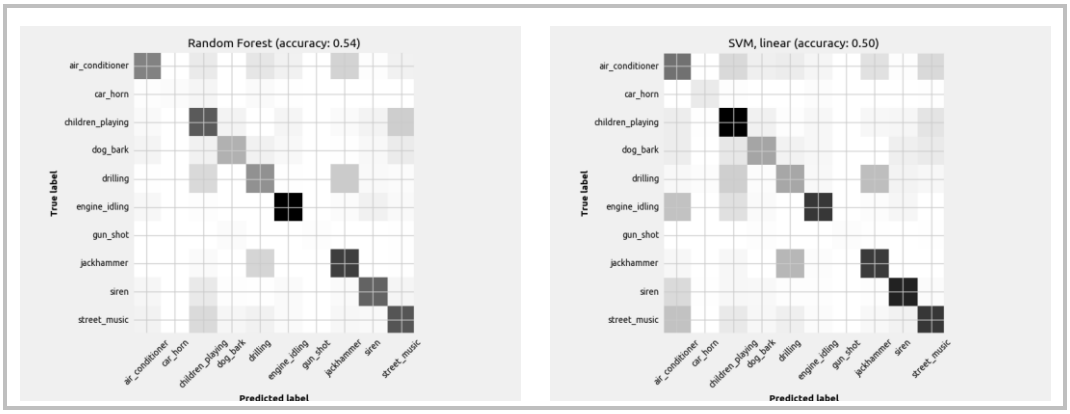



Figure 7: Scikit-Learn classifiers with specs feature set input

It should be clear that the specs features were less informative to the models and following training, the classifier performance suffered. Some classifiers such as QDA, were almost as bad as randomly guessing. Note that gradient boosting is missing. Since the spectrograms are thousands of points, gradient boosting takes a very long time to train and for this reason, was never fully evaluated.

| Position | Model | Features | Accuracy | Training Time (seconds) |
|---|---|---|---|---|
| 1 | CNN1 | specs | **0.77** | 111 |
| 2 | CNN2 | specs | **0.76** | 191 |
| 3 | Gradient Boosting | means | **0.65** | 82 |
| 4 | Random Forests | means | **0.62** | **5** |
| 5 | Logistic Regression | means | 0.59 | 511 |
| 6 | FFN1 | means | 0.58 | **12** |
| 7 | SVN | means | 0.55 | **14** |
| 8 | Random Forests | **specs** | 0.54 | 39 |

Table 3: Top classifiers

### 6.2 Discussion

As expected, the CNNs performed best by a large margin and both had roughly the same results. CNN1, however has a more ideal loss plot (see section 5.3) and is less prone to overfitting as a result and also trains faster, making it a superior model. Gradient boosting and random forests, generally regarded as the best decision tree based classifiers, were next in the list. Also, note that random forests trained extremely fast relative to other classifiers. The FFN1 model was further down in the list, but also trained very fast. Finally, even the random forests classifier using the 'specs' features (originally made for the CNNs) outperformed many other classifiers that use the superior 'means' features.

### 6.3 Conclusion

Given the noisiness in the data and the fact that there are ten classes to choose from, 0.77 accuracy is quite impressive. The paper on DNNs [2] published by the authors of the dataset achieved 0.72 accuracy, indicating the CNNs used in this paper are improvements. There is, however, some variability in the accuracy results of CNN1 and CNN2, and averaging several versions of the same CNN may bring down the accuracy from 0.77.

The CNN1 is the best classifier and it takes spectrograms which are relatively fast to compute on the fly for real-time classification. The downside is that it is slow to train and requires a high-end GPU compared to the FFN, gradient boosting and random forests which run fine on a CPU. The decision tree based methods, gradient boosting and random forests, also had

admiral performance. They rely on the ‘means’ features which take longer to compute than spectrograms since they involve several steps of feature extraction. Nevertheless, the decision trees train much faster than the CNN approach, partially because the ‘means’ features are sparser compared to the high resolution ‘specs’. Random forests also don’t require tuning making it a quick and dirty solution.

## 7 Future Work

The experiments performed drastically narrowed down the list of classifiers to a small set worth exploring further. This subset of classifier could benefit from more tuning as well as additional data to train on. An optimal classifier could be comprised of an ensemble of the top performing models. This ensemble could use several models to classify, then vote on the best choice based on the response of all classifiers in the ensemble.

State of the art time series audio analysis is done using Recurrent Neural Networks (RNN), specifically long short term memory networks (LSTM) [3]. These have the ability of remembering events earlier in time and are great at discovering and following patterns. An RNN could look at a siren sound as a repeating frequency sweep, or a gunshot as a large impulse followed by silence. The downside is RNN’s train very slowly and are very difficult to tune. An attempt was made to use an RNN in this experiment, but more tuning is necessary as it currently achieves an accuracy of around 0.6.

In the future, applying the top classifiers to new data, for example coughs, could be an insightful experience. In addition, generating more data by adding noise, pitch shifting and time shifting the original dataset, may yield better models less prone to overfitting.

### References

[1] Salamon, Justin, Christopher Jacoby, and Juan Pablo Bello. "A dataset and taxonomy for urban sound research." *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014.

[2] Salamon, Justin, and Juan Pablo Bello. "Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification." *arXiv preprint arXiv:1608.04363* (2016).

[3] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[4] Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." *Journal of Machine Learning Research* 13.Feb (2012): 281-305.