# Deep Learning Project

Kenny Power, Matthew Brousseau, Miriã Rafante

August 14, 2018

# Contents

**1 Introduction** **3**

**2 Related Work** **3**

**3 Data** **5**
3.1 Sound properties . . . . . 6
3.2 Describing Data . . . . . 6
3.3 Comparisons . . . . . 9
3.4 Signal processing . . . . . 9

**4 Methods** **11**
4.1 Data processing . . . . . 11
4.2 Models . . . . . 12

**5 Experiments** **12**

**6 Future Work** **13**

**7 Conclusion** **13**

**8 Appendix** **14**

**Abstract**

We chose to tackle the problem of classifying noise in an environmental setting. This can be very difficult, because some sounds are not completely unique. For example, we can imagine that a chainsaw and a blender can sound quite similar in an audio recording. We look at different methods of extracting features from audio files, then building different networks. Some of these networks are based on our intuitions, and we have also tried models which have worked well for other tasks. Our best final result gave us 83% accuracy using top 3 prediction criteria (if the correction is in the top 3 predictions of our model), which was used in the competition. We discuss our methods and solutions to this problem here.

# 1 Introduction

The audio tagging problem is composed of various sub problems, each with varying degrees of the complexity and difficulty. For instance, tagging real life audio files presents many challenges. One of the main challenges concerns background noises. Often, when we want to classify an audio file, there are background noises that are irrelevant to our classification and can act as noise in the data. This is particularly dangerous when background noises happen at the same time as the sound which we would like to classify [1].

This work deals with a simplified version of the problem. The audio files that we used were recorded in isolation (there are no background noises). These files were then classified into one of forty one classes [1] based on what it was a recording of.

# 2 Related Work

Automatic tagging has been a desired solution as shown by the DCASE (Detection and Classification of Acoustic Scenes and Events)[1] challenges. These challenges have worked towards audio classification problems for a number of years, each year with a subtle change in focus[14][6]. This has contributed to the motivation of several other works in the area.

Hamel et al's [8] work compared mel-spectrum and Mel-frequency cepstral coefficients (MFCCs). Mel-spectrum and MFCC are two ways to represent audio which are used to build models for classification. They concluded that mel-spectrum performs better than MFCC at classification and the results can be further improved with the use of PCA. After establishing this, they compared pooling functions over time changing the size of the windows and combining the different pooling functions like mean, variance, maximum and minimum. They observed that the best outcome resulted from the combination of all four poolings and windows lengths around two to four seconds. Their final-best result presented 0.876-AUC when tested over a dataset called MagnaTagATune and it was obtained by adding a hidden layer before the pooling step.

Choi et al [5] also used mel-spectrum as input. In this case, the mel-spectrum was input to a CNN with no fully connected layers. Three to seven convolutions and pooling layers are used in order to have an output with dimensions 1x1, increasing the number of filters at each layer and a sigmoid as activation function for the output

layer. They reach a 0.894 on the AUC curve testing their approach on the same MagnaTagATune dataset.

Lee et al [10][5], accordingly, compared STFT (Power spectrogram), MFCC and mel-spectrum and observed the best model was the one using mel-spectrum. For their more outstanding approach, they used mel-spectrum to describe the data, with six convolutions by layer, each convolution followed by a max-pooling function. The mel-spectrum data were scaled using standard normalization. The filter length was fixed at 243 values with stride of 81. They also used batch normalization and ReLU activations for the hidden layers and sigmoid for the output layer. The cost function used was cross entropy. In the last convolution layer's output, they applied dropout of 0.5, stochastic gradient descent and a momentum of 0.9. The initial learning rate was 0.01 and decreased by a factor of 5 at each 3 epochs without improvement. The batch size was 23 for one database and 50 for the other. Their algorithm reached an AUC of 0.9059 for the same database used by Hamel et al [8] and Choi et al [5], MagnaTagATune dataset, out performing both of these.

In contrast, the recent work of Xu et al [13], used two CNN's combined with three RNN's. The descriptions chosen were spectrogram, raw data and MFB (Mel filter banks feature) as opposed to the MFCC (Mel- frequency cepstrum coefficient) used until now. All three used IMD (interaural magnitude differences), which is a linear measure to incorporate spatial features introduced in this work. Before proceeding with the learning step, each sample is segmented using a sliding window of 32ms and a hop of 16ms and reshaped into different dimensions according with the data description. Then, the CNN's have 128 filters, one size for each data description. The CNN was followed by 3 RNN's, that end up with 500 ReLU units connected with 7 sigmoid output units, one for each class. The metric used for evaluation is the EER and their best approach performed an EEU of 0.102 on average among the seven classes.

Aiming to compare Lee et al [10] work, with an AUC of 0.9059, against Xu et al. [13] work with an EEU of 0.102, [11] shows that ROC CURVE is the plot of true positive rate (TPR) in function of the false positive rate (FPR) and the results are measured in AUC (area under the curve), which represents how well the model correctly classified the positive class compared to the negative examples also classified as positives [11]. The EEU is the error, the point where the false positive rate (FPR) is equal to the false negative rate (FNR).

If we let the AUC be the percentage of correctly classified samples and EER be the percentage of misclassification samples, and ignoring that different datasets were used, Lee et al [10] and Xu et al. [13] generated similar results with an error of 9.41% and 10.2%. Lee [10] presented a model with a much simpler structure.

In this work, besides exploring data descriptions, we also attempted to implement Lee's [10] algorithm and build models based on it.

The comparison below shows the differences and similarities between this approach and Lee's [10] work.

- Lee used mel-spectogram, but our approach uses MFCC. This choice was made empirically.
- Sigmoid was the activation function used in the output layer of Lee's work instead of a global average pooling used in this approach.
- Lee uses stochastic gradient descent but this approach uses Adam Optimizer.

- Lee does not use Recurrent Neural Networks (RNN) and fully connected hidden layers. These techniques, when added to our model, improved the results greatly.
- Lee's work also uses momentum of 0.9 and a learning rate decay that are not explored in this approach due to lack of time.
- Convolutions, batch normalization, max-pooling, ReLU, cross-entropy and dropout are used in both solutions, this last one being 0.5 in Lee's work and 0.3 in this approach.
- Each solution was applied to a different dataset making the results non-comparable.

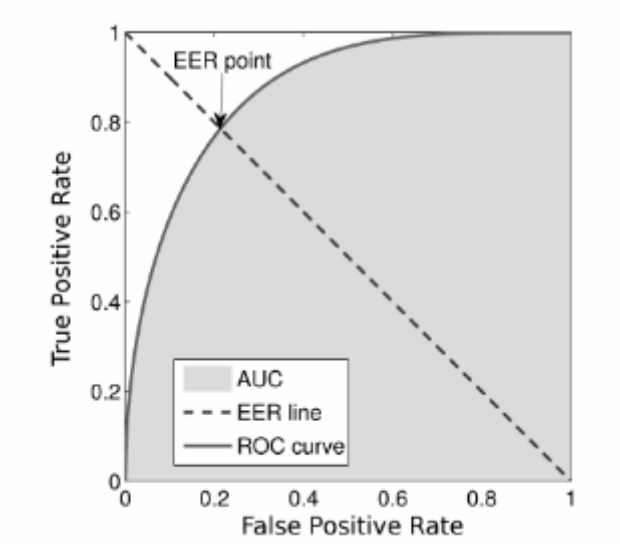



Figure 1: Comparison of ROC curve, AUC and EER measurements. Image source: https://www.researchgate.net/figure/An-example-of-a-ROC-curve-its-AUC-and-its-EER_fig1_225180361

# 3 Data

The dataset used is composed of 9473 samples, unequally distributed among 41 categories. According to DCASE [1], "the minimum number of audio samples per category in the train set is 94, and the maximum 300. The duration of the audio samples ranges from 300ms to 30s ...". Considering the rate is 44.1 kHz, 44100 values per second, and that the recording time is different for each sample, the number of points per sample variety drastically. The largest sample has 1323000 values.

Figures 2 and 3 show two sample's raw values (knock and an oboe) over time (ms). Some of the sample's label were manually verified and some were not. For treating this problem we are using only the verified samples. In other words, we are using only the 3710 samples that were verified manually.

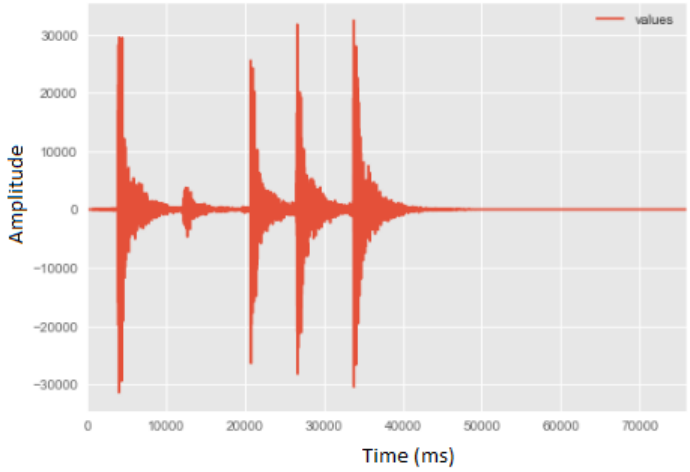



Figure 2: Plotting a knock sound

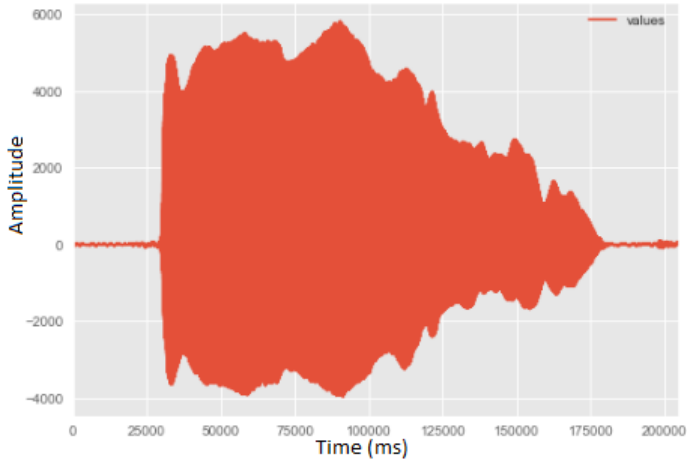



Figure 3: Plotting an oboe sound

## 3.1 Sound properties

Sounds are waves propagating over physical matter [3]. The height (or amplitude) of the wave and the number of waves flowing per second (or frequency) are examples of the properties of sound [4]. The higher the amplitude, the more energy the wave has. Intensity is the unit to measure the amount of energy a wave has in a given area [4]. Tones, overtones, harmonics, speed of sound, timbre and loudness are other properties of the sounds and they vary according to its source [4][3]. Therefore, in order to identify the source of a sound, we must study the change in the waves over time.

## 3.2 Describing Data

We started with using 4 functions from the Librosa package [2] which gave us different types of coefficients. This allowed us to perform feature extraction on the original data by using the following measures: tonnetz (computes the tonal

centroid features), spectral_centroid (computes a 6D description of chords), spectral_bandwidth and mfcc (computes the Mel-frequency cepstral coefficients (MFCCs)). These are plotted in Figures 4 and 5.

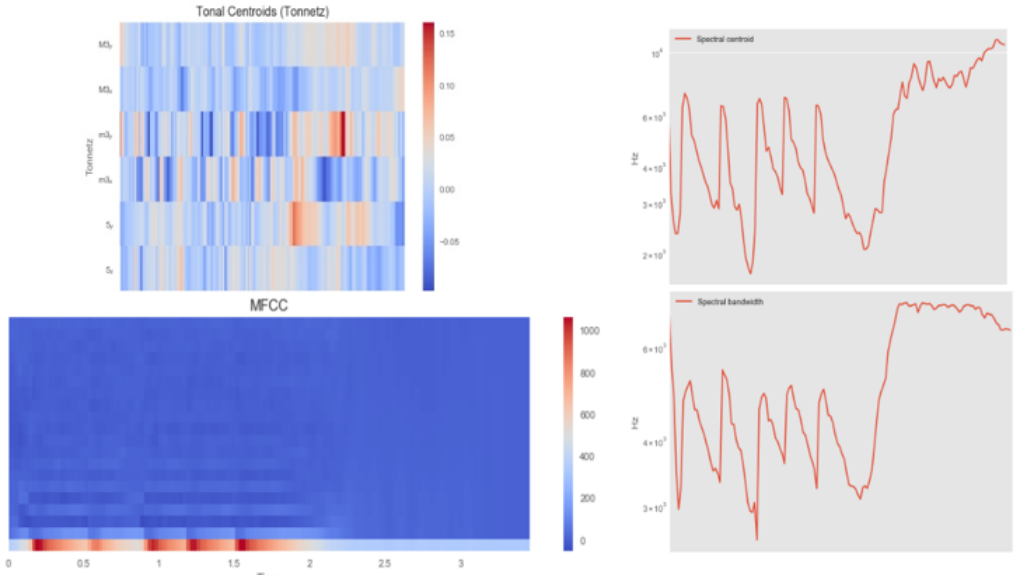



Figure 4: Visualization of the knock sample trough librosa's functions for feature extraction: (top-left) Tonnetz, (top-right) Spectral centroid, (bottom-left) MFCC and (bottom right) Spectral bandwidth. Source: https://librosa.github.io/librosa/index.html

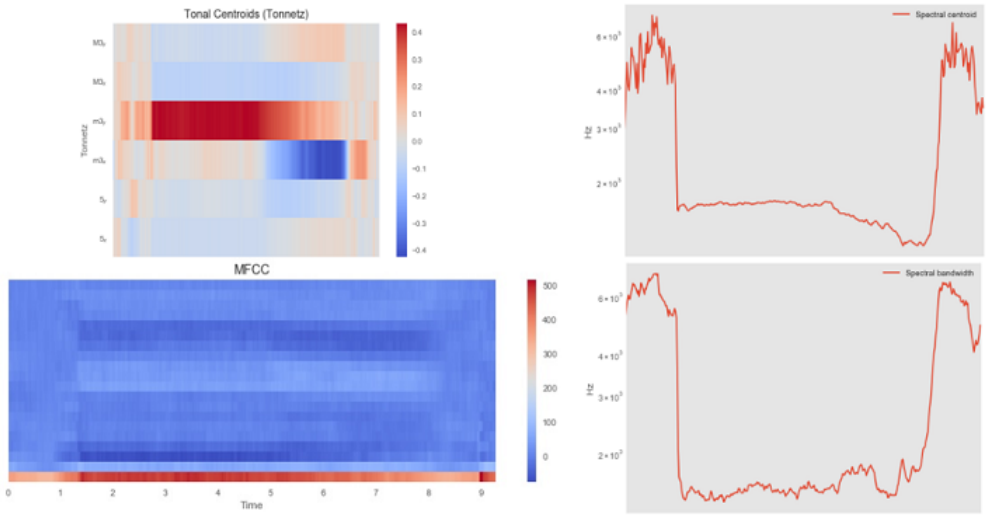



Figure 5: Visualization of the oboe sample trough some librosa's functions for feature extraction: (top-left) Tonnetz, (top-right) Spectral centroid, (bottom-left) MFCC and (bottom right) Spectral bandwidth. Source: https://librosa.github.io/librosa/index.html

The fifth description we tried was the spectrogram (Figure 6). The function is the STFT function from the Librosa package [2] and creates a 3-dimensional space combining the time series in milliseconds (ms) X 1025 frequencies (Hz) X the intensity in decibels (dB) at each time-frequency

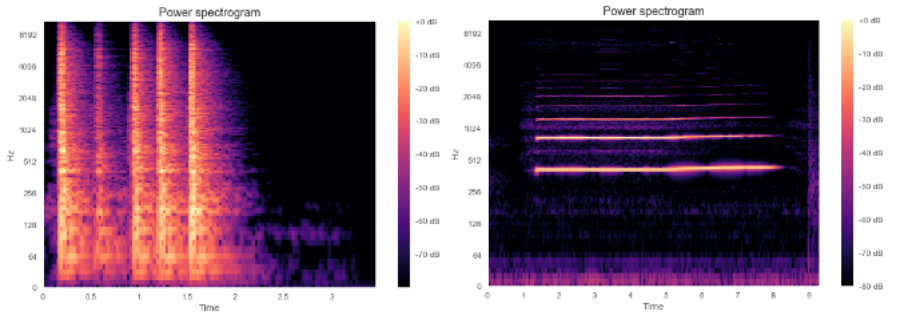



Figure 6: Power spectrograms of a Knock and an Oboe sound samples, respectively. Dimensions: 1025 frequencies x time x intensity at each time-frequency point.

Finally, considering Lee et al [10], we were also able to transform the data into mel-spectrograms. The plots of two sounds samples are shown in Figure 8. The mel-spectrogram calculates the spectrograms on the mel-frequency. The mel-frequency output dimensions for each sample are 128 groups of frequencies (Hz) X time in milliseconds (ms) X the intensity in decibels (dB) at each time-frequency. This spectrogram is much smaller than MFCC, presenting only 128 sets of frequencies versus 1025.

In order to get our initial results with mel-spectrogram and MFCC, we start with padding and trimming our samples to be the same length of time (Figure 7). We chose 1000 to be our sample length. 90% of our data is smaller than 1000, and so we pad this and we are only losing information on 10% of our data.

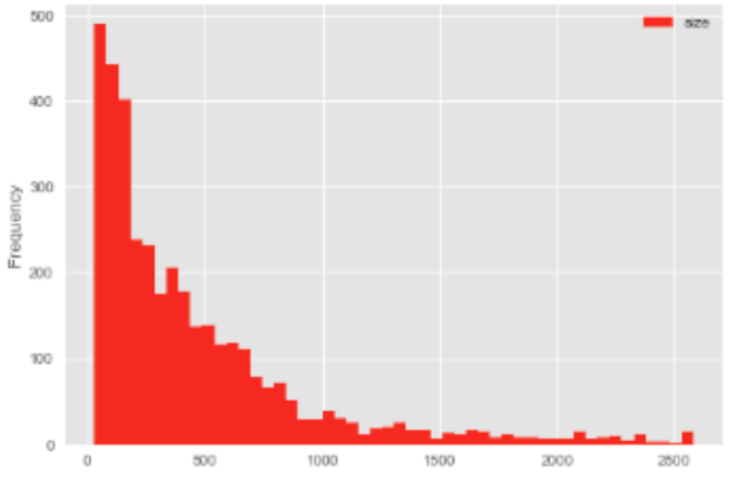



Figure 7: Number of samples of each size (size is proportional to the samples' length in ms)

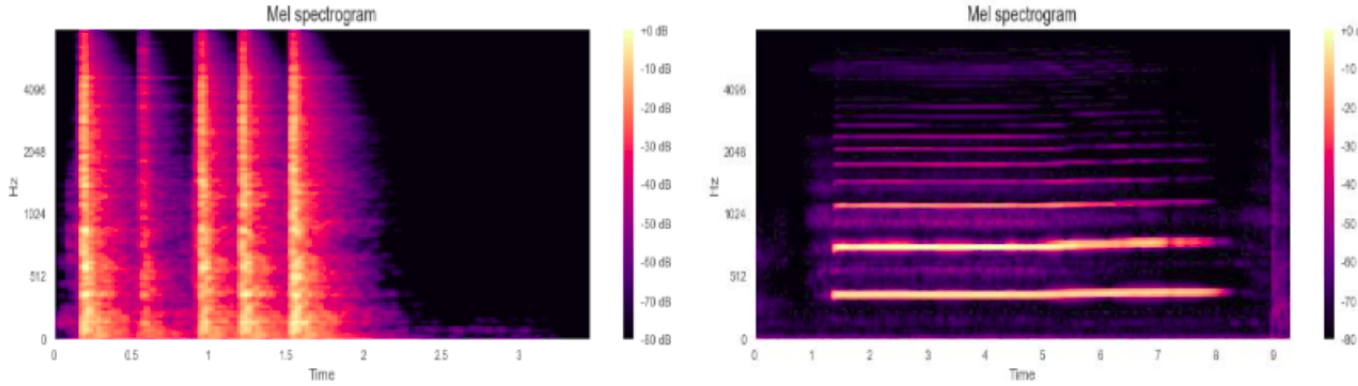



Figure 8: Mel-spectrograms of a Knock and an Oboe sound samples, respectively. Dimensions: 128 frequencies x time x intensity at each time-frequency point

## 3.3 Comparisons

Before inputting the various feature types into our deep learning models, some tests were performed in order to compare how effective each set of features were for the classification problem.

In order to test our extracted features, we attempted to first use a number of classification models that we believed would run more efficiently than a deep neural network. We started with Random Forest and then used Scikit learn's MLP on each. The results are shown in Appendix: Table 1. MFCC consistently had the best performance.

Considering the feature extractions used in [10], [13], and other works in Related work, Spectrograms (STFT) and Mel-spectrum were also tested. We attempted to use STFT, however, we ran into memory issues on our local machines and had to discard it. Despite the mentioned works have had best results with Mel-spectrum, MFCC still performed better in our tests (Appendix - Table 2). Therefore, the models were built using MFCC.

## 3.4 Signal processing

Our own implementation of MFCC extraction was highly based on [9] and [7] to give us more flexibility.

Given a set of .wav files, we can read these in using libraries such as Librosa or Scipy. This gives us the audio in the format of (time, amplitude). We normalized these waves, and their output was plotted:

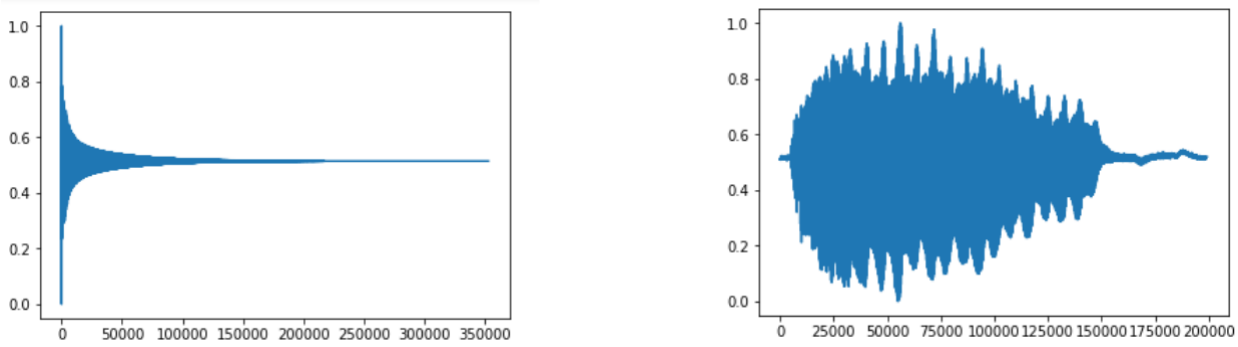



Figure 9: Normalized waves

These waves could be used as input, but it would not be very effective. We came across some great resources on how to handle audio data, and proceeded to transform these waves into MFCCs. We originally started using Librosa to extract MFCCs, but we found that the documentation didn't describe the process fully how they do it. We decided it would be a good idea to implement a hand coded MFCC extraction to have more flexibility of how the coefficients are extracted, in hopes of achieving better results. To do this, we first need to 'frame' the audio.

From [9]: "Because audio is a non stationary process, the FFT (fast Fourier transform) will produce distortions. To overcome this we can assume that the audio is a stationary process for a short periods of time. Because of that we divide the signal into short frames. Each audio frame will be the same size as the FFT. Also we want the frames to overlap. We do that so that the frames will have some correlation between them and because we lose the information on the edges of each frame after applying a window function."

We apply the hamming window to our data. The plot is shown here:

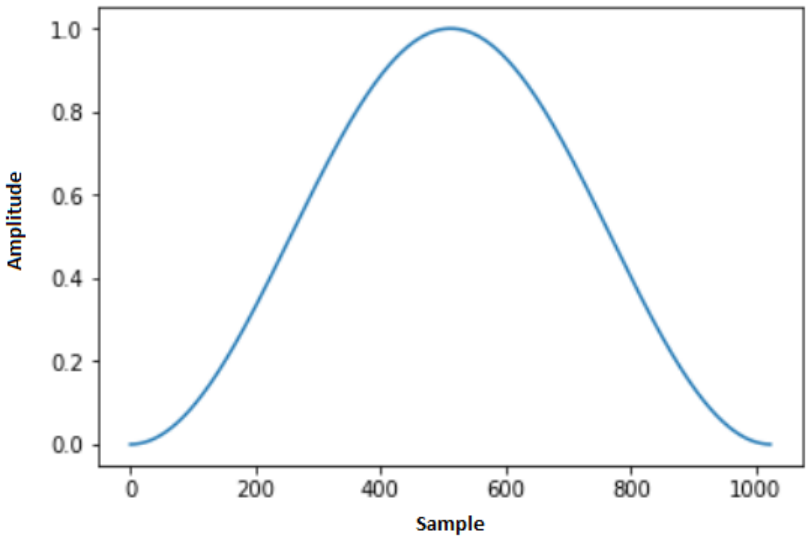



Figure 10: Hamming Window

Once the data is framed, we can see more interesting patterns shown below:

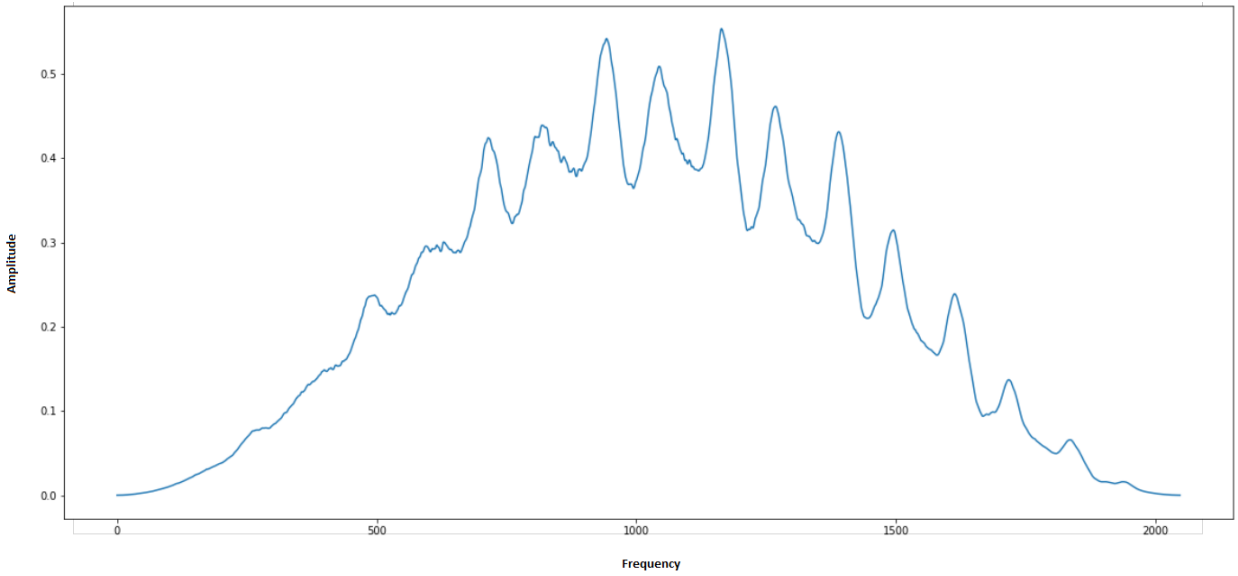



Figure 11: Audio frame

With the data framed, we need to apply filters on our frames. From [9]: "That (the filtered signal) will give us information about the power in each frequency band". We can see below how the filter banks look:

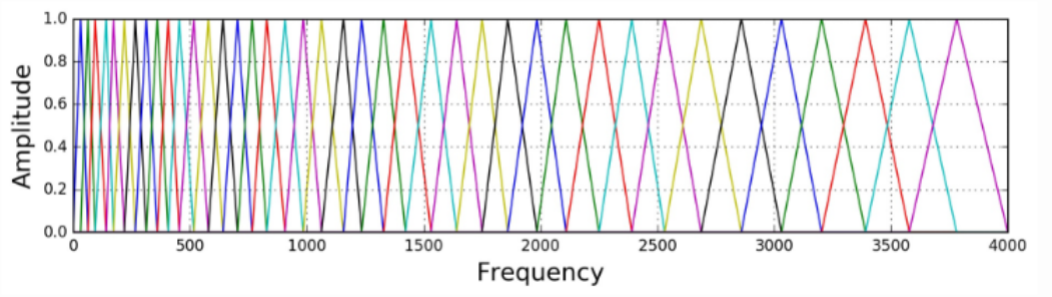



Figure 12: Filter banks, from [7]

Given the framed data and filters, we are then able to get our spectrogram. We apply a discrete cosine transform to decorrelate the filter bank coefficients. This gives us a compressed representation of the filter banks, and this allows to obtain the MFCCs.

# 4 Methods

## 4.1 Data processing

The first major task was to deal with the data using our methods described above. Our given data was in .wav format, which needs a lot of work to be put into a format usable for a neural network.

Since we found MFCC consistently performing the best, we attempted to extract MFCCs with our own program instead of using Librosa as it did not have very

detailed documentation of its steps and how to tune them. We believe there are many advantages to this approach. It gives us a powerful way of extracting information from the audio, which makes it easier to get good results from our classifiers. The downside however - is that it is tough to do a good job of extracting very useful MFCCs. There are things that influence our final output such as the window chosen to transform our data for example. It takes a lot of trial and error of tuning to get this right!

### 4.2 Models

Many models were attempted. Some were put together based on our own knowledge, but we also tried using models that worked well in other research papers. [10] is one model which performed well for the authors. However, we were not able to replicate their results (or come close to them). Our final model however, was structured based on their architecture.

The model with which we had the most success started with a 1 Dimensional CNN. It had 256 filters, a kernel size of 20, a stride of 1 and a ReLU activation. The convolutions were then passed into a Batch Normalizer and then into a 3-Max Pooling. From there, the data was passed into another 1D CNN that was the same as the first CNN but with a stride of 3.

After the second CNN, we used a Bidirectional LSTM with 256 units and a Tanh activation, then a fully connected layer with 173 nodes and a ReLU activation, followed by another Batch Normalization. Finally, we used another 1D CNN with 128 filters, a kernel size of 20 and a stride of 3. That is passed into a Global Average Pooling and a Fully Connected Layer with 41 nodes and a softmax activation. An overview of this model can be found at Figure 13 in the Appendix.

## 5 Experiments

Using our model that was described in 4.2, we were able to run a number of experiments. First we tried to train the model using the MFCC data, cross-entropy loss, Adam optimizer, a batch size of 75 and 87 epochs. With this we were able to get a test accuracy of 71.4% which was our best result. We then tried to train this same model with a number of changes to the data itself.

We then tried to train the model using mel-spectrogram. The mel-spectrogram model, while initially showing some promise, did not train particularly well. It seemed to oscillate and never converge. Eventually, the training accuracy stayed at around .95 and the loss stopped changing. It provided an accuracy of only 45%.

We then attempted to change our normalization method from a traditional normalization to a tanh-estimator normalization. From our reading, it seemed that the tanh-estimator was not as susceptible to outliers as traditional normalization. It seemed like an interesting opportunity but we were not able to generate decent results with it. The normalization worked, however, no matter the model used, the loss would eventually go to NaN and the test accuracy never rose above 4%.

# 6 Future Work

Given more time to understand signals and signal processing, it would be great to tweak some parameters of the wave to MFCC process that we implemented to see if we can get more powerful coefficients extracted. We could also then look further into the other methods of extracting useful information from audio (CQT, STFT etc) and also play with these processes. The way that we present our data to our classifier is extremely important and is what will give us the ability to get a good classification.

Another approach we would like to explore is classify the images created when generated byb the MFCCs, STFT, etc. Perhaps this could give a better extraction of features. We tried to implement GoogLeNet [12], but ran into issues and time constraints. This network performed very well at image classification, and it could be very interesting to try it on our MFCC data or especially images/plots of it.

Based on suggestions (during our presentation), we attempted to implement a 2d convolutional variant of our best model, but we did not have enough time to get it right unfortunately. It certainly would be interesting to see how this improves performance of our best model.

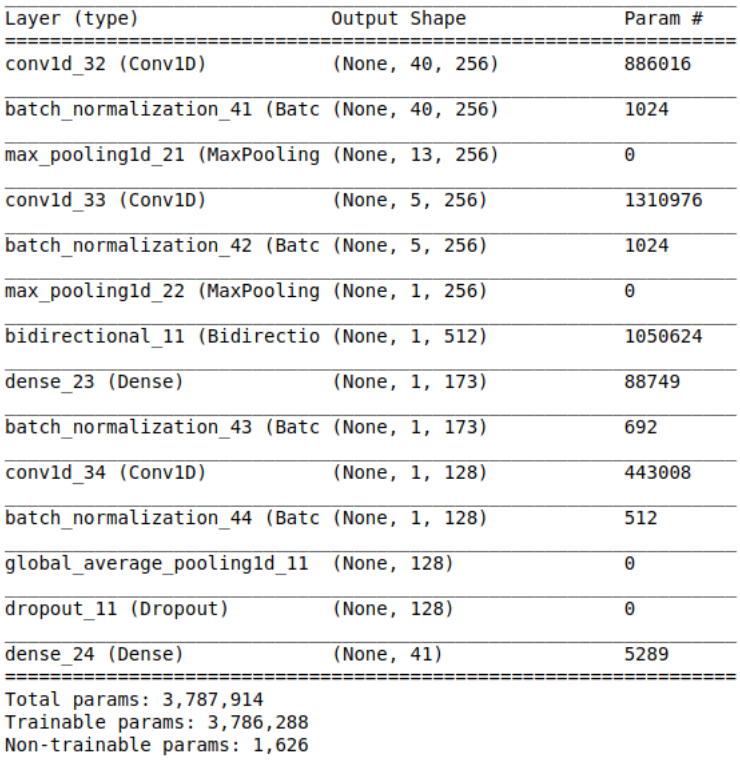
# 7 Conclusion

We began our project by researching various feature extraction functions and later testing them using numerous classification methods. Through this method, we were able to determine that MFCC provided the best features with which to build our larger models.

We later, using various online resources, attempted to implement our own MFCC function. While this greatly added to our understanding of MFCC's, it did not add to the accuracy of our models during our project. We believe there is great potential for this as mentioned in the further work however.

While implementing MFCC, we also attempted to build a number of published models that had success in the past at classifying sounds. While the implementations themselves did not work as well for us as they did for their original authors, they provided a basis for what became our best working model. This model, using a combination of ConvNets, LSTM's and fully connected layers was able to achieve a top 3 accuracy of 82.7%. Given some of the flaws we later learned about our model, we believe this to be a good score, one that was well above the original baseline. While there are many ideas that we have that could be used to improve our results (see Section 6), we are happy with our results and believe that our model shows potential for future works.

# 8 Appendix



| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv1d_32 (Conv1D) | (None, 40, 256) | 886016 |
| batch_normalization_41 (Batc | (None, 40, 256) | 1024 |
| max_pooling1d_21 (MaxPooling | (None, 13, 256) | 0 |
| conv1d_33 (Conv1D) | (None, 5, 256) | 1310976 |
| batch_normalization_42 (Batc | (None, 5, 256) | 1024 |
| max_pooling1d_22 (MaxPooling | (None, 1, 256) | 0 |
| bidirectional_11 (Bidirectio | (None, 1, 512) | 1050624 |
| dense_23 (Dense) | (None, 1, 173) | 88749 |
| batch_normalization_43 (Batc | (None, 1, 173) | 692 |
| conv1d_34 (Conv1D) | (None, 1, 128) | 443008 |
| batch_normalization_44 (Batc | (None, 1, 128) | 512 |
| global_average_pooling1d_11 | (None, 128) | 0 |
| dropout_11 (Dropout) | (None, 128) | 0 |
| dense_24 (Dense) | (None, 41) | 5289 |

Total params: 3,787,914
Trainable params: 3,786,288
Non-trainable params: 1,626

Figure 13: Model and Parameters

| Data description | RF Accuracies | MLP Accuracies |
|---|---|---|
| Tone | [0.297, 0.310, 0.297]<br>Mean: 0.3014<br>Std: 0.006 | [0.06869, 0.0672065, 0.069501]<br>Mean: 0.06847<br>Std: 0.00095 |
| Spectral Centroid | [0.4321, 0.4372, 0.4481]<br>Mean: 0.4391<br>Std: 0.0067 | [ 0.0687, 0.0688, 0.06787]<br>Mean: 0.0685<br>Std: 0.0004 |
| Spectral Bandwidth | [ 0.4489, 0.4308, 0.4325]<br>Mean: 0.4374<br>Std: 0.0082 | [ 0.06869, 0.06721, 0.0695]<br>Mean: 0.06847<br>Std: 0.00095 |
| MFCC | [0.6462, 0.6510, 0.6549]<br>Mean: 0.6507<br>Std: 0.0036 | -<br>-<br>- |

Table 1: Comparison of the effectiveness of each feature extracted in the classification problem using Random Forest (RF) and Multi-layer Perceptron (MLP).

| Data description | RF Accuracies |
|---|---|
| MFCC | [0.68, 0.66, 0.68, 0.64, 0.70, 0.67, 0.68, 0.66, 0.64, 0.70]<br>Mean: 0.67<br>Std: 0.02 |
| STFT | Not enough computer power |
| Mel-spectrogram | [0.66, 0.65, 0.67, 0.63, 0.67, 0.65, 0.64, 0.63, 0.61, 0.68]<br>Mean: 0.65<br>Std: 0.02 |

Table 2: Comparison of the effectiveness of each feature extracted in the classification problem using Random Forest (RF)

# References


[1] General-purpose Audio Tagging of Freesound Content with AudioSet Labels - DCASE.

[2] LibROSA - Librosa 0.6.0 Documentation.

[3] Sound. Wikipedia.

[4] The Components of Sound.

[5] Choi, K., Fazekas, G., and Sandler, M. Automatic tagging using deep convolutional neural networks.

[6] Eghbal-Zadeh, H., Lehner, B., ..., M. D. I. A. C., and undefined 2016. CP-JKU submissions for DCASE-2016: A hybrid approach using binaural i-vectors and deep convolutional neural networks. *researchgate.net*.

[7] Fayek, H. Speech processing for machine learning: Filter banks, mel-frequency cepstral coefficients (mfccs) and what's in-between, 2016.

[8] Hamel, P., Lemieux, S., Bengio, Y., and Eck, D. Temporal pooling and multiscale learning for automatic annotation and ranking of music audio.

[9] Ilyamich. Mfcc implementation and tutorial, 2018.

[10] Lee Jiyoung Park Keunhyoung Luke Kim Juhan Nam, J. Sample-level deep convolutional neural networks for music auto-taggings using raw waveforms.

[11] Schoonjans, F. ROC Curve Analysis with MedCalc.

[12] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 1–9.

[13] Xu, Y., Kong, Q., Huang, Q., Wang, W., and Plumbley, M. D. Convolutional gated recurrent neural network incorporating spatial features for audio tagging. In *2017 International Joint Conference on Neural Networks (IJCNN)* (may 2017), IEEE, pp. 3461–3466.

[14] Yun, S., Kim, S., Moon, S., Cho, J., and Kim, T. Discriminative training of gmm parameters for audio scene classification and audio tagging. Tech. rep., 2016.