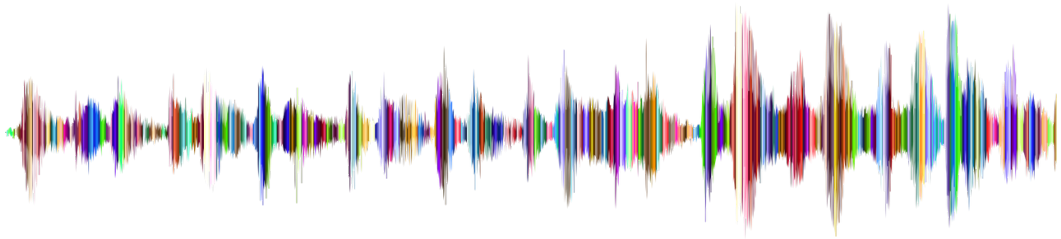
# CSML1020 - UrbanSound8K Classification

All code and source files available on our GitHub repository:

https://github.com/Powahowa/CSML1020-Group_5-Project-UrbanSound-8K

ALEX FUNG, York University - School of Continuing Studies
VISWESH KRISHNAMURTHY, York University - School of Continuing Studies
TONY LEE, York University - School of Continuing Studies
PATRICK OSBORNE*, York University - School of Continuing Studies



Classification of environmental data in a usable format has been critical to the "Big Data" revolution. Images and numerical data are extremely well represented in this space, but typically audio has been more obscure as the industry lacks a standard pipeline for audio classification. In this paper we examine various approaches to classifying environmental sound data. A purely numerical approach, as well as a translation of the problem to image classification, are attempted. Both traditional machine learning and deep learning through the use of convolution neural networks are attempted for classification. Cloud computing resources are utilized to accelerate performance but are not mandatory.

CCS Concepts: • **Computing methodologies** → **Neural networks**.

*All authors contributed equally to this research.

Authors' addresses: Alex Fung, York University - School of Continuing Studies; Viswesh Krishnamurthy, York University - School of Continuing Studies; Tony Lee, York University - School of Continuing Studies; Patrick Osborne, York University - School of Continuing Studies.


XXXX-XXXX/2020/7-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Additional Key Words and Phrases: audio classification, neural networks, UrbanSound8K, machine learning, Librosa, TensorFlow, Keras, sonicboom

**ACM Reference Format:**
Alex Fung, Viswesh Krishnamurthy, Tony Lee, and Patrick Osborne. 2020. CSML1020 - UrbanSound8K Classification: All code and source files available on our GitHub repository: https://github.com/Powahowa/CSML1020-Group_5-Project-UrbanSound-8K . 1, 1 (July 2020), 24 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 BUSINESS PROBLEM/ACCURATE PROBLEM DESCRIPTION AND HYPOTHESIS

We selected the *UrbanSound8K* dataset for use in our project. It consists of roughly 8 thousand common urban sounds across 10 different classes such as car horns, air conditioners, jackhammers, etc. The primary goal of the project is to accurately classify these sounds to their respective classes.

We targeted this goal for several reasons. Firstly, audio classification is a typically underrepresented area in machine learning. This is specifically true for sound classification as opposed to music classification. We hope to advance the state of the art in this field and provide useful information to others who attempt it.

Secondly, we believe the problem presents several different paths to a solution which will allow us to gain experience in underrepresented feature engineering techniques, as well as in a variety of techniques to apply machine learning models. At the onset of the project, we planned to attempt classification on both numerical features generated from the audio files and more traditional image recognition on one or more visual representations of the audio files. We then compare and contrast the two approaches.

Our expectation was that given a visual feature that was sufficiently different in its representation of each class, the overall metrics of image classification would be superior. This hypothesis is based mainly on the advanced state of image recognition in the field with many high-performing pretrained neural networks and helper packages available.

Conversely, we expected a challenge with the numerical features approach. Generating audio features was a new experience for our team and there was limited documentation on methods for doing so.

We planned to approach modelling from both a traditional angle (several different models) and a deep learning strategy (convolutional neural networks). We wrote code to be machine and operating system agnostic so that it could be deployed to the cloud easily.

Finally, we chose this dataset as we believe it has relevant real-world applicability. Classifying ambient sounds into specific labels has numerous uses. Possible applications include urban planning/municipal noise complain monitoring, machinery defect monitoring (i.e. identifying a specific noise associated with failure), and noise cancellation (identifying and removing specific noises) such as on Zoom.

## 2 DATA UNDERSTANDING/EXPLORATION

*UrbanSound8K* is a dataset which contains a collection of 8732 labelled sound excerpts (<=4s) of urban sounds from 10 classes: air conditioner operation sounds, car horns, children playing, dog barks, drilling sounds, engine idling noises, gunshots, jackhammer operation sounds, sirens, and street music.

Furthermore, the dataset is divided into 10 folds, which is the standard by which the creators intend to keep the comparison of the performance of machine learning models easy and fair.

Files are in WAV format. According to the creators, all excerpts are taken from field recordings uploaded to www.freesound.org. The crowd-sourced nature of the recordings in the dataset means that consideration must be given to various inconsistencies in the file format and specifications.

As with all machine learning projects, it is imperative that we first look in detail at the data we are dealing with. There is a myriad of reasons for this. Some of the more important ones have to do with bias, underfitting, or overfitting. For example, a dataset that has much more of one particular category compared to the others, otherwise called unbalanced data, may cause machine learning algorithms to start to memorize or favour that category over the others. The distribution of each feature in the data is also important to analyze in this context.

Another reason is simply to be able to understand what we are working with. It is always good to know what the minimum and maximum values are, what the mean and median is, and whether we are dealing with categorical or numerical data that we may have to encode into another format.

Taking into consideration the aforementioned reasons, and because *UrbanSound8K* contains sensory data that we humans understand innately, we have decided to look at: the distribution of the 10 categories of the audio files, the length of the audio files, and visualizing the raw sound waves and spectrograms in an effort to make sense of what we are working with.

### 2.1 Distribution of Categories

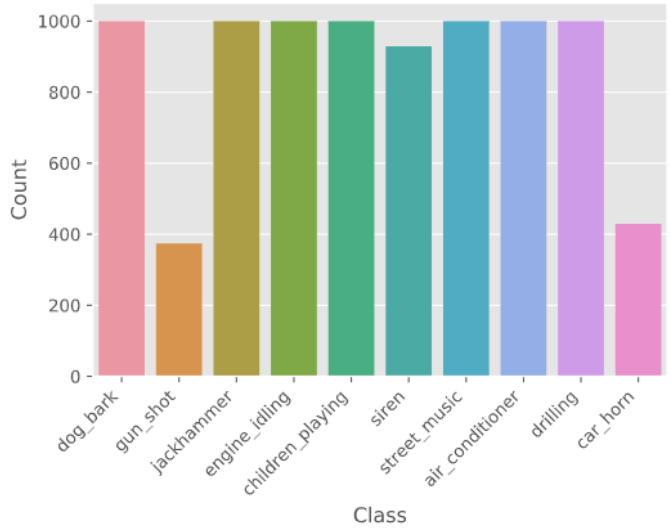



Fig. 1. Number of audio files by class

As seen in Figure 1, the 10 categories of sounds are mostly evenly distributed when viewed as a whole. Notably, audio samples of gunshots and car horns are about half as numerous as the other sounds. Since the distribution is not very unbalanced and because neural networks require a lot of training examples, we have decided that it is best not to limit the number of samples of the other 8 categories in the dataset.

Figure 2 shows that the creators have done a good job of sampling down the data for each fold while still maintaining the distribution of the counts of each category. We see this when break down the counts of the categories by fold.

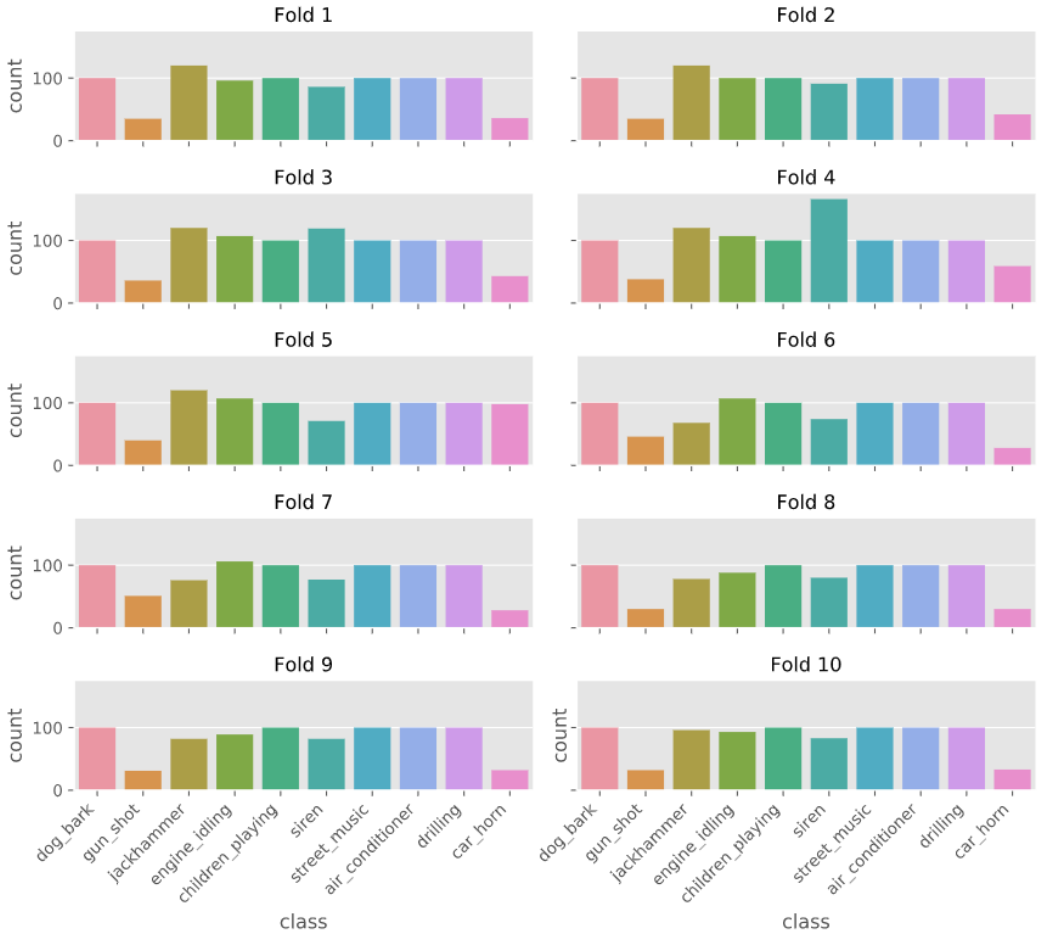



Fig. 2. Number of audio files by class, grouped by fold

## 2.2 Distribution of Audio Length/Duration

At first glance, it seems that the vast majority of the audio samples are 4 seconds long, with only a handful of audio samples that have various lengths ranging anywhere from 10s of milliseconds up to 4 seconds as shown in Figure 3.

Notably, the "Dog Bark", "Gun Shot" and "Car Horn" *UrbanSound8K* have a significant number of files with different lengths, as seen in Figure 4.

It is at this point where we decided that it may be wise to either pad, stretch or loop the sounds such that they all have a length of 4 seconds.

## 2.3 Distribution of Sample Rates

Referring to Figure 5, the vast majority of sample rates are at 44100 Hz, which is CD-quality. There are a lot also at 48000 Hz and 96000 Hz. There is a small selection of files that are equal to or lower than 24000 Hz. Since the sample rates are very important to feature extraction, especially in regards

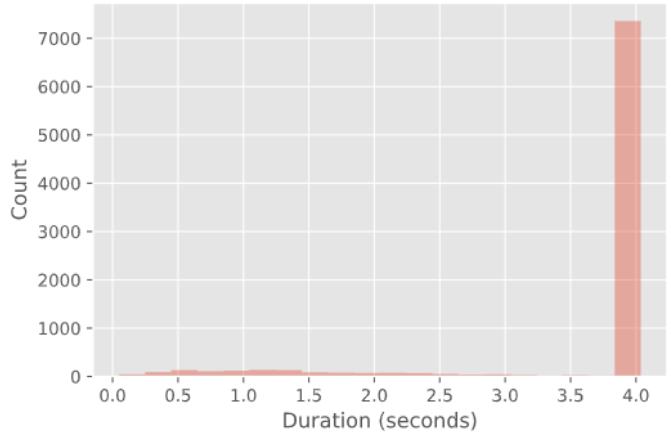



Fig. 3. Duration of audio clip lengths by class

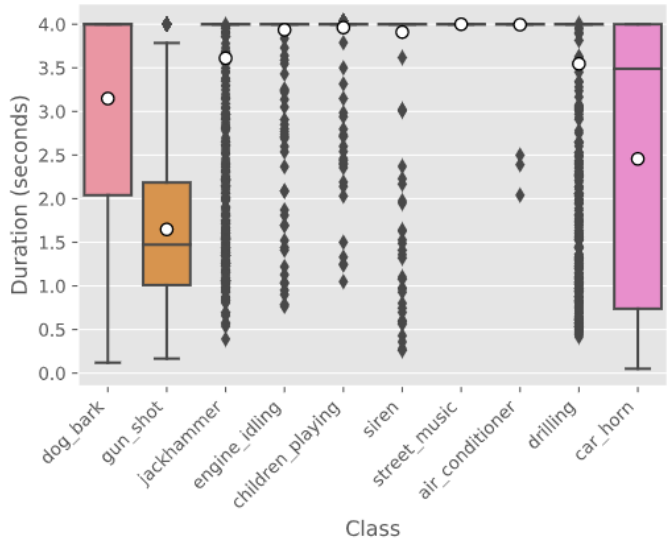



Fig. 4. Duration of audio clip lengths by class

to the shape of the feature, we decided to standardize the sample rate to 22050 Hz, the default for Librosa's load function.

### 2.4 Waves and Spectrograms

Arguably, the most common form of visualization of audio data seen by the average person is the wave plot. The x-axis is time, the y-axis is amplitude, or more plainly, volume. As seen in Figure 6, each sound is actually reasonably distinct in these plots. However, the amplitude is not necessarily a very good differentiating factor since, for example, air conditioner, engine idling, and jackhammer sounds can be confused for one another depending on the loudness of the sound. This is evident when comparing the plots.

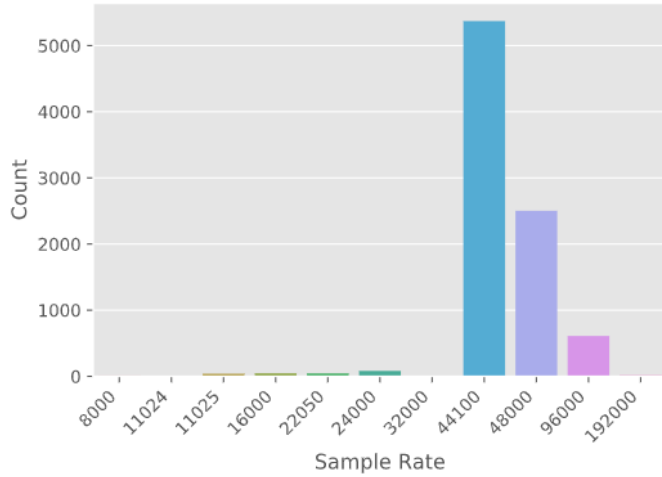



Fig. 5. Distribution of sample rates

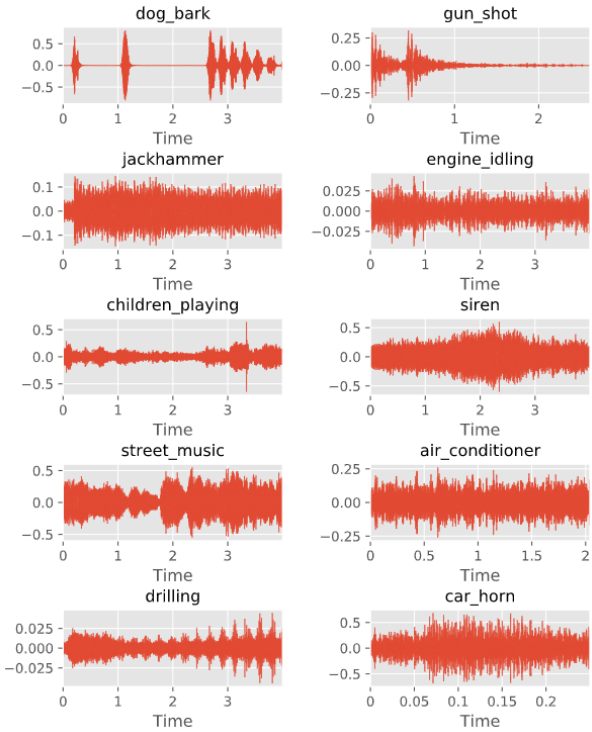



Fig. 6. Waveplot by Class

A spectrogram of each class is plotted in Figure 7. A spectrogram is a visual representation of the spectrum of frequencies and their amplitudes in a signal as they varies with time. Like the wave

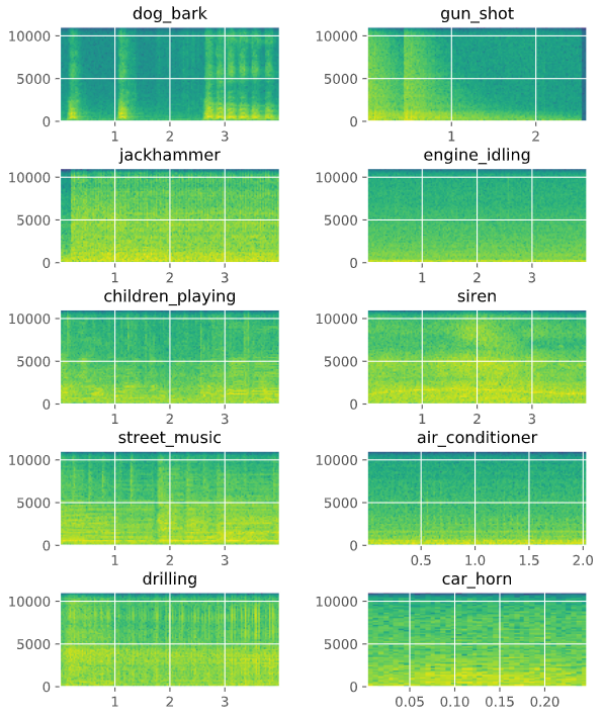



Fig. 7. Spectrogram by Class

plot, each class is seemingly quite unique. However, once again, the air conditioner is similar to the engine idle. Interestingly, the class "Jackhammer" is now seemingly similar to the class "Drilling".

## 3 PROBLEM APPROACH

The sound classification problem was approached from different angles, including different features and different ways to generate the features. One approach was to follow the classical sound classification methods, which is to import a sound file and engineer features like MFCCs, Librosa Delta, Mel-Scaled spectrogram, Short time Fourier transform, Chromagram, spectral contrast and Tonnetz. Another approach was to convert this into an image classification problem by taking the fast Fourier transform of the audio files, saving them as images and then classifying the image data set.

We'll review each of the generated features below.

### 3.1 Mel Scaled Spectrogram

A spectrogram is a visual plot of frequencies in an audio sample as they vary over time. A Mel Scaled Spectrogram has had its frequencies converted to the Mel scale. Figures 10 and 11 show two Mel Scaled Spectrograms.

The Mel Scale is a method of scaling frequencies such that they correspond more closely to what humans perceive. E.g. a very high frequency and a lower but still high frequency may be perceived as the same by a human. The Mel Scale replicates this. Since we are interested in identifying sound as heard by humans and discarding any noise that is not human interpretable, this makes Mel scaling desirable for our project.

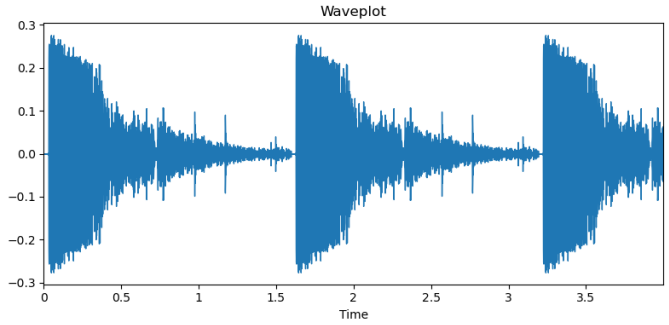



Fig. 8. Gun Shot: Waveplot

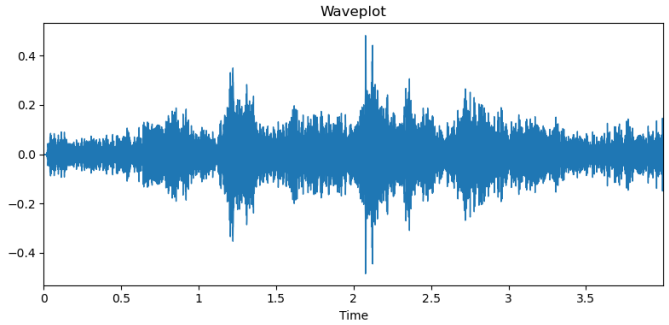



Fig. 9. Dog Bark: Waveplot

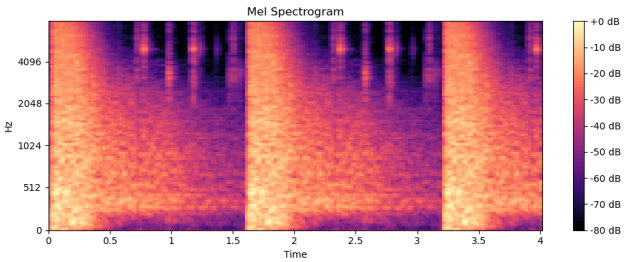



Fig. 10. Gun Shot: Mel Scaled Spectrogram

### 3.2 MFCC

MFCC stands for Mel Frequency Cepstrum Co-efficients. The term 'cepstrum' is 'spectrum' in reverse. 'Cepstrum' is defined as the rate of change in spectral bands. It is obtained by converting a digital signal from time domain to frequency domain (Fourier transform), applying a filter bank to place the frequencies on the Mel scale (as mentioned before), then taking the log of this Fourier spectrum and again converting it to the frequency spectrum. Since this signal is transformed from the frequency domain, it is neither in the time domain or the frequency domain and hence, Bogert et al, termed it as the quefrency (reverse for 'freque'ncy) spectrum. The MFCCs can be summarized

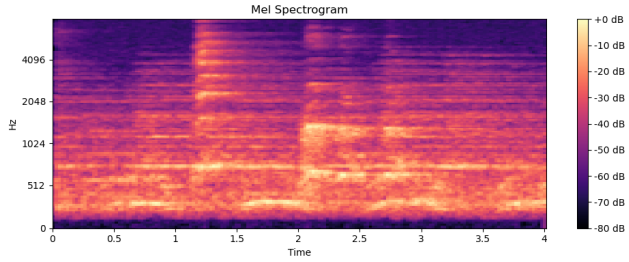



Fig. 11. Dog Bark: Mel Scaled Spectrogram

as a set of cepstral coefficients that represent specific, unique sounds in the audio signal. MFCCs are commonly used in speech recognition and music genre classification. MFCC is considered to be *the* state-of-the-art feature, even though it was created in 1976. Figures 12 and 13 show two MFCCs.

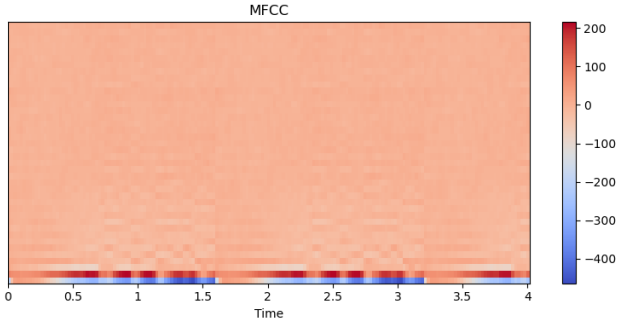



Fig. 12. Gun Shot: MFCC

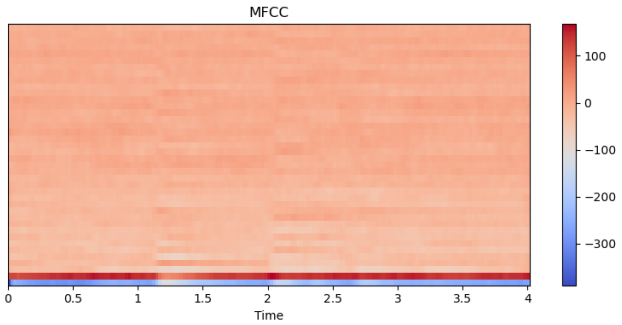



Fig. 13. Dog Bark: MFCC

### 3.3 Delta

Librosa's Delta feature computation calculates the local estimate of the derivative of the input data along the selected axis. Delta features are computed Savitsky-Golay filtering. The Delta feature we chose was the first derivative of MFCCs with 9 surrounding frames for differentiation.

### 3.4 Short Time Fourier Transform

A Short Time Fourier Transform helps one understand an audio signal in terms of its frequencies and how it changes over time. A standard Fourier transform transforms the entire signal to the frequency domain while STFT does this over multiple sections (or windows) of the same audio signal, preserving time information.

### 3.5 Chromagram

Chroma features are an interesting and powerful representation for music audio in which the entire spectrum is projected onto 12 bins representing the 12 distinct semitones (or chroma) of the musical octave. Since, in music, notes exactly one octave apart are perceived as particularly similar, knowing the distribution of chroma even without the absolute frequency (i.e. the original octave) can give useful musical information about the audio – and may even reveal perceived musical similarity that is not apparent in the original spectra. Figures 14 and 15 show two STFT Chromagrams.

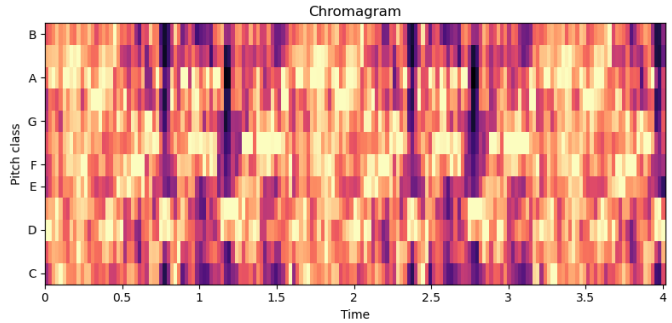



Fig. 14. Gun Shot: (STFT) Chromagram

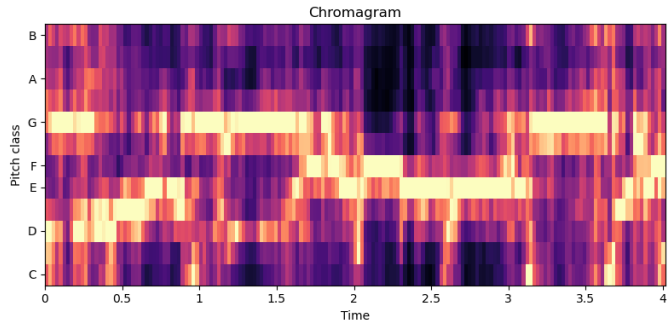



Fig. 15. Dog Bark: (STFT) Chromagram

### 3.6 Spectral Contrast

In features extraction, the music piece is first segmented into frames by 200ms analysis window with 100ms overlapping. For each frame, FFT is performed to get the spectral components and

then it is divided into six octave-based sub-bands. Finally, Spectral Contrast is estimated from each octave sub-band. The raw Spectral Contrast feature estimates the strength of spectral peaks, valleys and their difference in each sub-band. Figures 16 and 17 show two STFT Spectral Contrast representations.

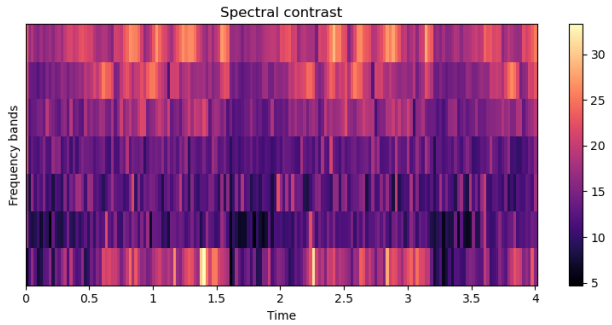



Fig. 16. Gun Shot: (STFT) Spectral Contrast

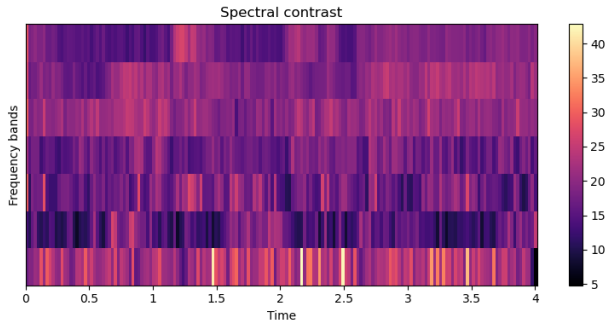



Fig. 17. Dog Bark: (STFT) Spectral Contrast

### 3.7 Tonnetz

The Harmonic Network or Tonnetz is a well known planar representation of pitch relations first attributed to Euler. For the purpose of audio classification, Tonnetz is used to compute tonal centroids as a feature. The six-dimensional tonal centroid vector, $\zeta$, for time frame $n$ is given by the multiplication of the chroma vector $c$ and a transformation matrix $\Phi$. Figures 18 and 19 show two representations of Tonnetz.

### 3.8 Fast Fourier Transform

Fourier transform converts an audio file from the time domain to frequency domain. The time-domain doesn't offer much for analysis. At a given point in time, an audio signal can be a combination of multiple frequencies which can be viewed and in turn further analysed only in the frequency domain. The Fast Fourier transform is just an efficient way of executing a Discrete Fourier Transform.

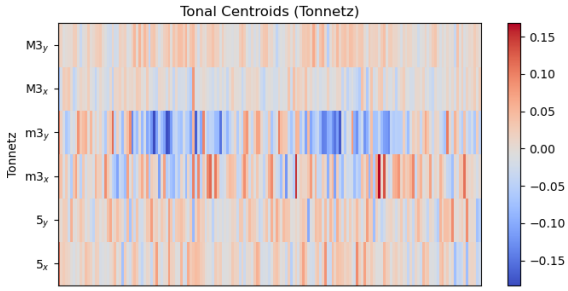



Fig. 18. Gun Shot: Tonnetz

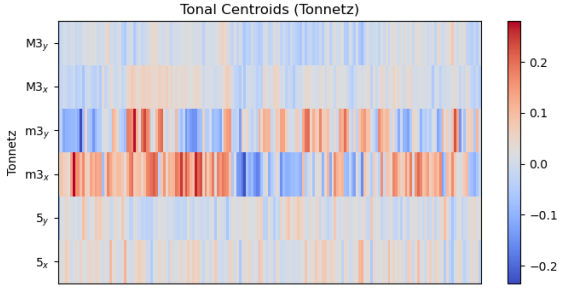



Fig. 19. Dog Bark: Tonnetz

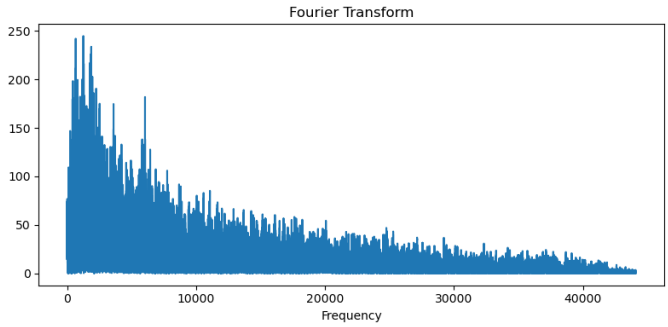



Fig. 20. Gun Shot: Fast Fourier Transform

## 4 DATA PREP

### 4.1 FFT

The raw audio files were recursively read in using Librosa library. All audio files were then put through Numpy's FFT and then plotted using Matplotlib. We know that any digital signal can be expressed as a combination of sines and cosines to convert it to the frequency domain. In turn, in the frequency domain, the negative frequencies don't give additional information about the signal

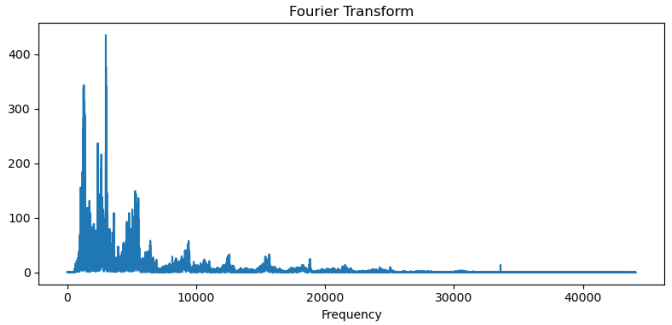



Fig. 21. Dog Bark: Fast Fourier Transform

and are hence filtered out. Audio signals thus converted were plotted and saved as PNG files. These image files were then used as input data for the classification algorithm. Figures 20 and 21 show two graphs of Fast Fourier Transform.

### 4.2 Flattening and Normalization

Further preparation is necessary for the various numerical/vector features created above using Librosa. They are typically represented as 2-3 (or N) dimensional arrays. This poses a problem for some of our traditional models so it was necessary to write a function to flatten them.

The flattening function takes the mean of each array using Numpy, along axis 0. This has the effect of reducing an N-dimensional array to a 1-dimensional array, with each element being the mean of the various dimensions of that element.

We also wrote a function to iterate through each of our numerical features and normalize the data. This was necessary to ensure that all features are broadly comparable.

## 5 MODELLING

Two approaches in Modelling were used throughout the course of this project. In particular, two groups of Machine Learning algorithms were primarily used, namely Traditional Models, and Neural Networks. Firstly, more traditional algorithms and models were used to classify the *UrbanSound8K* dataset. The definition of "traditional" methods, or models, can be defined as Machine Learning algorithms that are not related to Neural Networks or Deep Learning. This would mean the latter approach used models that utilised Neural Networks or Deep Learning to some degree.

### 5.1 Traditional Models

Because of the multi-class classification problem, all of the traditional Machine Learning models listed below were wrapped in SkLearn's OneVsRestClassifier class to denote that the model was to be used in a One-vs-the-rest (OvR) multi-class strategy. In the OvR strategy, one classifier is calculated and fit per class, meaning technically there would be 10 classifiers per Traditional Model, as there are 10 classes in the UrbanSound8K Dataset.

As previously mentioned, the UrbanSound8k Dataset was curated into a number of engineered features, such as MFCCs, MelSpec, Chroma STFT and others. The Traditional Models used a number of the aforementioned features above, which were also flattened and normalized.

Lastly, all Machine Learning algorithms listed below were implemented with various SkLearn model API implementations. For most of the models listed below, the default parameters for the model API were used, unless stated otherwise.

*5.1.1 Logistic Regression.* Logistic Regression is a commonly used statistical parametric linear model based off the logit scale. Logistic Regression is primarily used for binary classification, although it can also be used for multi-label classification as well. The SkLearn implementation of the Logistic Regression algorithm is the LogisticRegression model API.

*5.1.2 K-Nearest Neighbours Classifier.* The K-Nearest Neighbours Classifier is another statistical model that is used fairly often. It is a non-parametric Machine Learning algorithm that attempts to classify an input object based on the vote of its neighbours. In SkLearn, the KNeighborsClassifier is the implementation of the K-Nearest Neighbours algorithm (k-NN). Aside from the number of neighbours parameter, which was defined to be 10, all other parameters in the model were default, as defined by SkLearn's KNeighborsClassifier API.

*5.1.3 Decision Tree.* The Decision Tree is one of the most commonly used, and most understood Machine Learning algorithms. The algorithm uses tree-like structures to decide what classification to make based on various inputs. SkLearn's DecisionTreeClassifier API was used to replicate the Decision Tree algorithm.

*5.1.4 Gaussian Naïve Bayes.* Gaussian Naïve Bayes is a statistical model based off the Naïve Bayes algorithm. The algorithm itself is based off the aptly named Naïve Bayes' Theorem, which makes a key assumption: that the probability of a hypothesis is dependent, and can be calculated based off of prior probabilities. GaussianNB, which is SkLearn's implementation of the Gaussian Naïve Bayes, was used.

*5.1.5 Linear SVM.* Support Vector Machine, or SVM, is a Machine Learning algorithm that attempts to find a hyperplane in N-dimensions that best classifies the data points. In this situation, the number of dimensions is equal to the number of features provided. To implement the Linear SVM algorithm, SkLearn's LinearSVC API was utilised.

*5.1.6 Bagging Classifier.* Bagging Classifier is a Machine Learning model that uses ensemble learning to fit a variety of base classifiers onto random subsets of the training dataset. Various other ensemble methods include boosting, forest of randomized trees, and others. Ensemble methods are typically used when not one particular model is encompassing and suitable to achieve acceptable classification metrics. As a result, multiple "weak" models are used to reduce the bias and achieve acceptable results. To implement the Bagging Classifier algorithm in SkLearn, a DecisionTreeClassifier of 2620 leaf nodes was used, with a BaggingClassifier of 100 estimators.

### 5.2 Deep Learning

Compared to Traditional Machine Learning Methods listed above, Deep Learning has recently gained more popularity as its effectiveness in various domains that other Traditional Machine Learning Methods normally struggle, such as classification tasks in Computer Vision, Natural Language Processing, and Audio Recognition. For example, the use of deep neural networks that utilize several layers of input, intermediate, and output layers has been used to perform image classification tasks.

In this project, the implementation of Deep Learning Neural Networks comprised of two different strategies: using Deep Learning Neural Networks to classify the UrbanSound8K dataset on both visual and audio domains. The Deep Neural Network that was chosen to classify the dataset on the visual domain was the Inception Neural Network based off of FFT (Fast Fourier Transform) images that are generated from the dataset, while the other Deep Neural Network that was chosen to classify the same dataset on the audio domain was a custom Convolutional Neural Network

based off of MFCC (Mel-frequency cepstral coefficients) and MFCC-Delta (the first-order derivative of the MFCC) features.

*5.2.1 Inception Neural Network.* For Image Classification, a pre-trained Deep Learning Neural Network model called Inception-v3 was used to classify the FFT (Fast Fourier Transform) images that are generated from the UrbanSound8k dataset.

The Inception-v3 is a pre-trained convolutional neural network typically used for Computer Visions tasks, such as Image Classification. The neural network model was created by Google, and is currently in its third version. It was trained with over 1 million training images on the ImageNet database, which comprises of over 1000 distinct classes.Inception-v3 contains approximately 7 million parameters, and is 42 layers deep with various convolution layers. The implementation of the Inception-v3 network is provided by Keras.

The Inception-v3 neural network was fitted to use the FFT (Fast Fourier Transform) images generated from the UrbanSound8k dataset. All 8732 audio files of the dataset were read, and then transformed through an FFT function provided by Numpy to generate FFT images. All the FFT images were saved as standard 256px x 256px images.

Upon the completion of the generation of FFT images, the FFT image dataset was then split into train, test validation sets, with 6286 samples for training, 1572 samples for test, and 874 samples for validation and blind test.

*5.2.2 Convolutional Neural Network (MFCC + Delta).* For Audio classification, a different convolutional neural network was built from scratch to classify sounds, based off the MFCC (Mel-frequency cepstral coefficients) and its Delta as its features The Delta was chosen as the first-order derivative of the MFCC (MFCC + Delta). Overall, the custom neural network model was implemented with Keras.

The custom convolutional neural network accepted the combined "MFCC + Delta" feature, as an input. The feature itself has an input shape of (40, 173, 2). The first dimension, 40, represents the number of MFCC coefficients that are kept. The second dimension, 173, represents the resultant data vector for each MFCC coefficient. Lastly, the third dimension, 2, represents the MFCC in the first channel, and the Delta in the second channel.

After each convolutional layer, an activation layer was used, with the activation function being RELU. After the RELU activation layer, a MaxPool layer was utilized. In the last layer, instead of a MaxPool layer, a regular Softmax layer was used to generate a probability of results. Lastly, the default Adam optimizer with a static 0.0001 learning rate was provide to compile the Keras model.

## 6 MODEL PERFORMANCE AND SELECTION

The following metrics were used to evaluate model performance and select the right model, whether the model is from a traditional Machine Learning algorithm, or based off from a Deep Learning Neutral Network.

- Confusion Matrix
- Accuracy
- Learning Curves
- ROC AUC Curve

### 6.1 Traditional Models Performance

See Figures 22-26. From the confusion matrices of the conventional models, their limitations become clear. The vast majority of the mis-classifications occurred when the models predicted the class "A/C" (Air Conditioner), even though the actual true classes were any other class but "A/C". Except

for Bagging ensemble model with Decision Tree base learner and K-Neighbours, all other models had less than "chance" accuracy.

Figures 27-31 show the learning rate plots for the different conventional models.

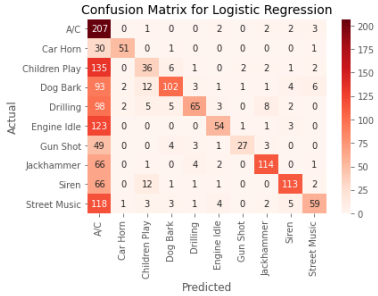



Fig. 22. Log Reg

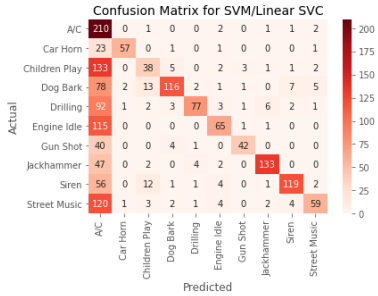



Fig. 23. SVM

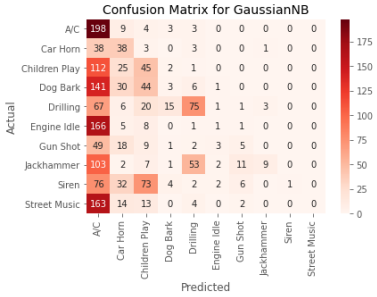



Fig. 24. Gaussian NB

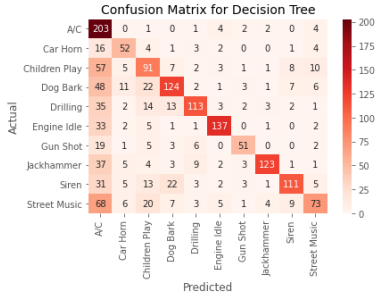



Fig. 25. Decision Tree

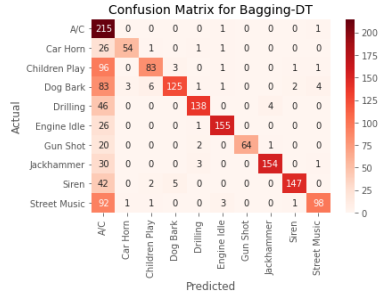



Fig. 26. Bagging DT

While ROC-AUC numbers are pretty good for the conventional models, the accuracy is greater than 0.5 for only two models. Therefore, it can be concluded that the conventional models don't perform very well, and are likely not suitable as capable models for classifying the multi-class dataset adequately.

## 6.2 Deep Learning Neural Networks Performance

Whereas the Traditional Methods and Models struggled to classify the multi-class dataset with an accuracy better than chance, the Inception Neural Network fitted to classify the dataset in FFT


Fig. 27. Log Reg

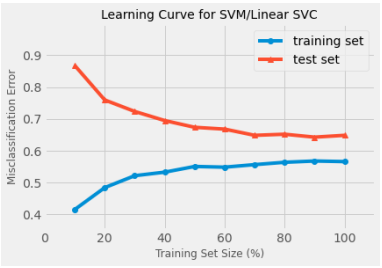



Fig. 28. SVM

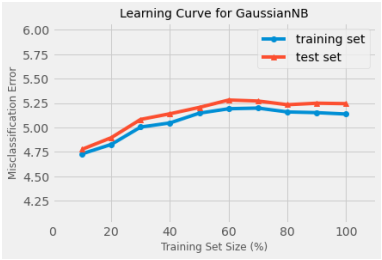



Fig. 29. Gaussian NB

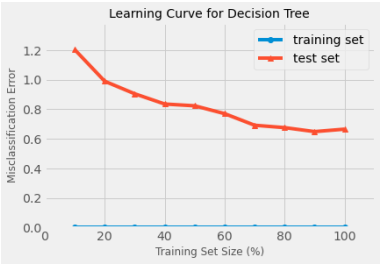



Fig. 30. Decision Tree

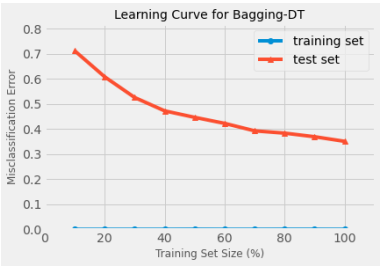



Fig. 31. Bagging DT

images, and the Convolutional Neural Network trained to classify the MFCC and Delta audio data, did not suffer from the same severe classification issues.

In terms of various metrics, such as ROC-AUC score, F1 Score, Precision, Recall, and Accuracy, the custom convolutional Neural Network (CNN) trained to classify the *UrbanSound8K* dataset using the MFCC and Delta audio data did significantly better than the Inception Neural Network, trained to classify the same dataset using FFT images.In terms of accuracy, the CNN fared better than the Inception Model by about 10 points in F1 Score, Precision, Recall, and Accuracy, and about 3 points in ROC-AUC. See Figures 32 and 33 for ROC-AUC and Accuracy. See Figures 34 and 35 for the Confusion Matrices.

Further analysis of the Accuracies by Class reveal that in the classification of most classes, the CNN Model fared significantly better than the Inception Model by some degree of points. In some classes, such as "Children Play" and "Street Music", the CNN model fared better than the Inception Model by approximately 20 to 30 points. The only class where the Inception Model classified more accurately than the CNN was the class "Car Horn", although this improvement was only by a relatively insignificant 2 points. See Figures 36 and 37.

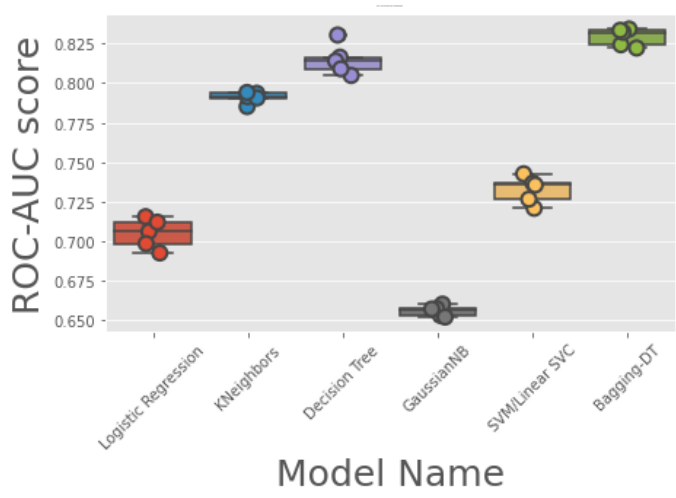



Fig. 32. ROC-AUC of various Traditional Models

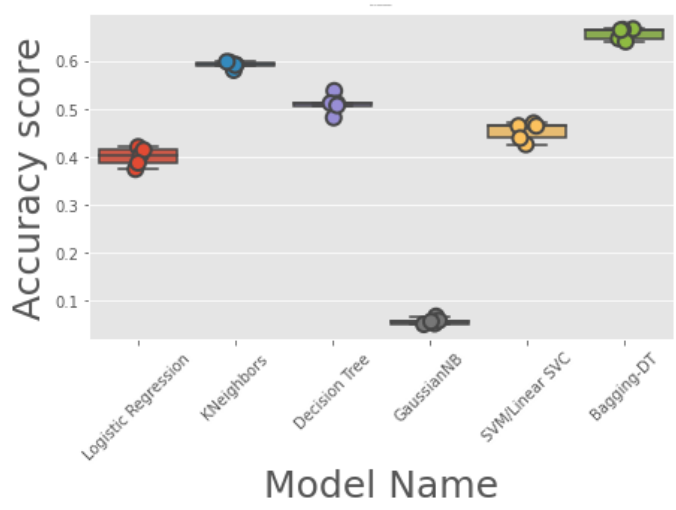



Fig. 33. Accuracy of various Traditional Models

The Learning Curve for Train and Test Accuracy of the Convolutional Neural Network as seen in Figure 38 revealed that the CNN model fit fairly well on both train and test datasets. The Learning Curve for Train and Test Loss of the CNN as seen in Figure 39 also revealed the same conclusion, as there was not much loss with the model itself across several epochs. However, there were some noticeable discrepancies related to the downward spikes of the of the test accuracy, and upward spikes of the test loss. The explanation of such spikes would require further analysis, but the current assumption is that on testing on certain data in specific folds, this caused such discrepancies. Another explanation could be the use of better hyper-parameter optimization, as well as the use of regularization so there is less over-fitting.

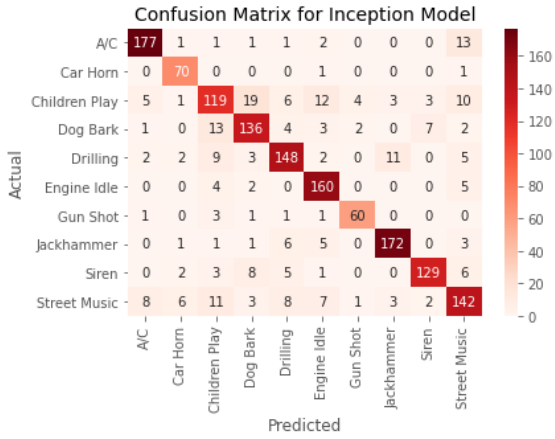



Fig. 34. Confusion Matrix for Inception Model

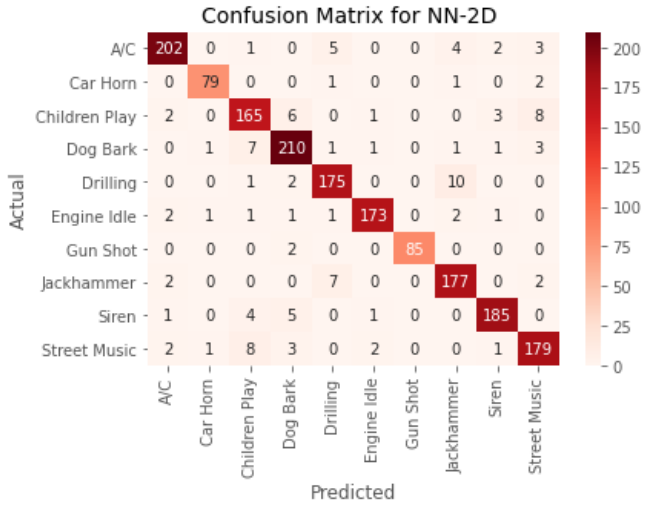



Fig. 35. Confusion Matrix for Convolutional Neural Network

On the other hand, the Learning Curve for Train and Test Accuracy of the Inception Model as seen in Figure 40, as well as the Learning Curve for Train and Test Loss of the Inception Model as seen in Figure 41, show that the Inception Model is over-fitting to a stronger degree than the CNN. This can be seen in the Train and Test Accuracy, where the Test Accuracy noticeably dips below the Train Accuracy towards a higher number of epochs. Similarly, this can also be observed in the Train and Test Loss, where the Test Loss increases very slightly and gradually at a higher number of epochs.

### 6.3 Final Model Selection

The Final Model selected depends entirely on a given business use case, as there are other factors of a model to consider, such as time and complexity to train and test, model interpretability. In

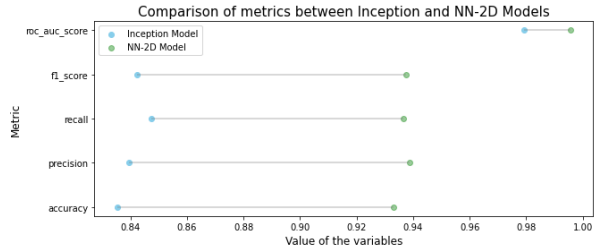



Fig. 36. Comparison of Metrics of the Deep Learning Neural Networks

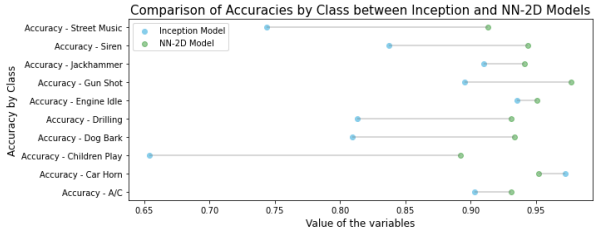



Fig. 37. Comparison of Class Accuracies of the Deep Learning Neural Networks

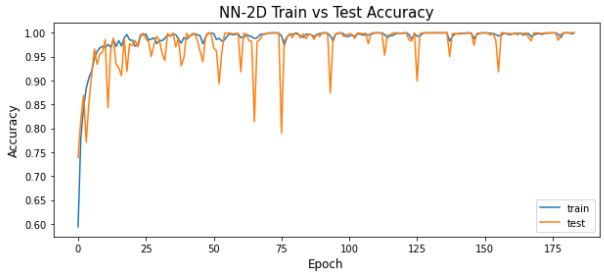



Fig. 38. Convolutional Neural Network Train vs Test Accuracy

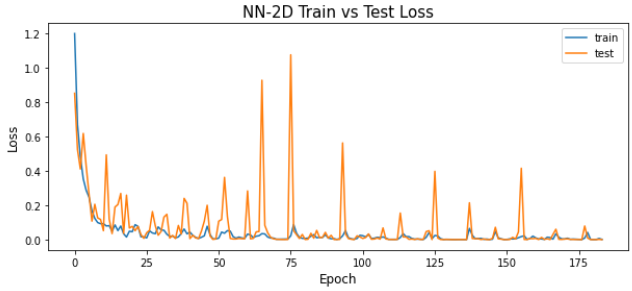



Fig. 39. Convolutional Neural Network Train vs Test Loss

an academic study, where the metrics of the model are normally given the most importance, the custom Convolutional Neural Network trained on MFCC and Delta would be chosen for the final

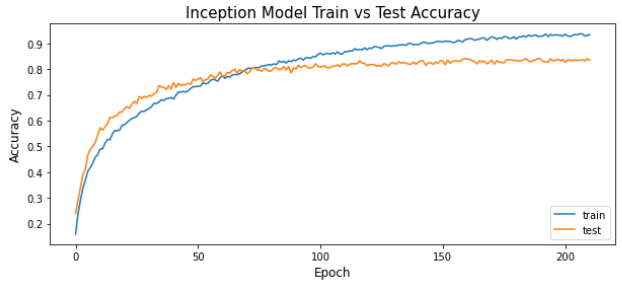



Fig. 40. Inception Model Train vs Test Accuracy

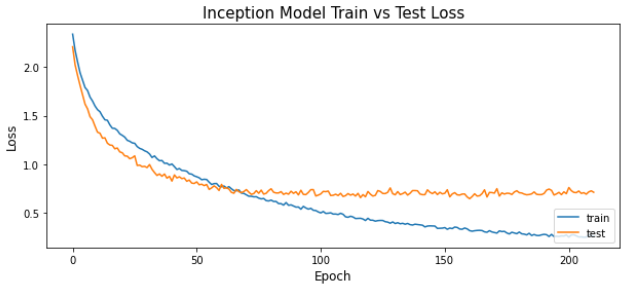



Fig. 41. Inception Model Train vs Test Loss

model, since it ranked highest in metric performance versus other models. In other situations where some loss in metric performance was acceptable given an improvement in model interpretability, then the Inception Model trained on FFT would be chosen for the final model, as even though it had worse metric performance by 10 points, it would be easier to explain how the Inception Model works, since it is a pre-trained model that has basis in numerous scientific journals, as well as online research articles.

## 7 CHALLENGES FACED

Overall, we faced numerous challenges in the execution of this project. As audio and sound classification are more unusual in the literature and online resources, when we encountered issues with code not running or working as expected we had to rely more on our own troubleshooting abilities. This also applied to to developing an overall approach. We do cite some papers that worked on similar problems but each approach was relatively unique and there was no consensus on the best way to tackle the problem. As such, we iterated on various options until we found what worked.

Generating numerical features posed another challenge. We were not initially sure how to approach this and looked at building some feature from scratch. Luckily, we found that the Librosa library does an excellent job of exposing cutting edge features and when used correctly can efficiently create many numerical features stored in arrays.

N-Dimensional features were an issue with some of the models as mentioned earlier in the data preparation section, but we mostly overcame this by flattening and manipulating the features.

The next issue concerns the widely varying specifications of the audio files submitted by users on the Freesound.org website (which was used to build this data set by the original authors). Bit Depth,

Sample Rate, file length and salience (background/foreground) all varied widely. These caused various issues in both loading and analysing the data so we had to standardize. We re-sampled all audio files to 22050 Hz for consistency and repeated/looped any audio files there were shorter than 4 seconds to ensure consistency.

We struggled to find functions that did exactly what we needed in terms of loading and working the data as well as creating and manipulating features. We put a fairly extensive amount of work into creating our own "sonicboom.py" library which we coded to be platform agnostic (Linux or Windows). We ensure that our code was written in parallel using JobLib when possible as single-threaded code was very slow to execute.

Spinning up effective cloud resources quickly that could be used for collaboration posed another problem. We used Google's pre-built "Deep Learning VM" TensorFlow images to rapidly deploy a pre-compiled environment. We used JupyterLab to develop and collaborate in the cloud.

The final issue was with interpretability and built-in metrics on the Neural Networks. We struggled to double-check Keras' built-in validation function. There was some kind of error in that validation was showing as extremely high even when huge portions of the training data were removed. To correct this, we manually separate our own holdout set of validation data and validated it using a custom written function to call the predict function of the model and compare to the actual value.

## 8 ETHICAL CONSIDERATIONS

Various ethical and privacy concerns come into play when working on an audio recognition project. Since a large amount of audio/ambient sound is recorded, it's possible that sensitive audio such as private chats, personal info and unintended sounds could be captured accidentally. We must use care to filter and to treat the audio data appropriately.

Many of the features engineered in this project are extensively used in acoustic fingerprinting. Some commercially available products like Shazam and Lyrebird AI are built on a similar foundation. This could be used to infer personally identifiable information such as a user's location based on their sound profile.

The input dataset of urban sounds could also be replaced with the voices of different people and with minor edits, our models will classify people's voices. We're cognizant of the aforementioned capabilities and risks associated with our models and will exercise extreme caution in extending this solution for any other sound classification problem.

Due care will be taken to not (even inadvertently) use our models in such a way that can be, or perceived to be discriminatory and/or biased. It must be understood that the models built are specifically to solve the problem at hand only.

## 9 SUMMARY OF RELATED WORK

The vast majority of approaches found on the internet that attempt to classify the *UrbanSound8K* dataset apply a convolutional neural network to classify images of spectrograms, MFCC, etc. Their performance is usually good but not state-of-the-art. Again, this is a result of the lack of a pre-built framework or standard pipeline that the practitioners can rely on. Thankfully, research is ongoing. Only in the last year or two, has the accuracy breached the 90% threshold requiring very complicated deep learning models. One of the top approaches used what they call a Two-Stream CNN Based on Decision-Level Fusion (TSCNN-DS) model to achieve an accuracy of 97.2% in the year 2019. Their approach is detailed here.[19]

From our research, the best accuracy obtained is currently 97.52% on the *UrbanSound8K* dataset, which was achieved in 2020 using Multiple Feature Channels and Attention based Deep Convolutional Neural Network. The same model also achieves state-of-the-art performance on the ESC-10

and ESC-50 datasets of 95.7% and 88.5%, respectively. Apparently, for the ESC-10 and ESC-50 datasets, the accuracy achieved by the proposed model is beyond human accuracy of 95.7% and 81.3% respectively. Their approach is detailed in the journal here.[17]

## 10 CONCLUSION

It is clear from the above results that in both approaches, audio and images, the neural networks performed much better than the conventional models. If the solution were to be deployed and if the requirement is very high accuracy, then we would choose the 5 layer CNN which gave an accuracy of 93%. If the requirement is high accuracy and also high interpretability, then we would choose the image classifier inception V3 model as the pre-trained model has many existing work to leverage off of, including ways to plot and interpret the steps the NN takes the input through.

### 10.1 Future course of action

- Further feature extraction, which is a rabbit hole for simple classification tasks, but is an even deeper hole for audio classification because the relationship of time and frequency is so complicated
- Further Hyperparameter tuning of Models
- Experiment with model architecture similar to newer, more complicated, cutting edge models. Attention based DNN and Two-Stream CNN Based on Decision-Level Fusion (97% accuracy)
- Investigate discrepancies in Class Accuracies, particularly classes 'Children Play' and 'Street Music' (the performance on these two are a bit lower for CNN-2D and quite a bit lower with Inception)
- Investigate "Salience", the difference in classifying audio samples that are in the background vs the foreground
- Image Classification specifically: Ensemble methods + adding more features (multiple image features with ensemble voting)

## REFERENCES


[1] Seth" "Adams. 2020. DSP Background - Deep Learning for Audio Classification p.1. https://www.youtube.com/watch?v=Z7YM-HAz-IY&list=PLhA3b2k8R3t2Ng1WW_7MiXeh1pfQJQi_P.

[2] Avinash. 2019. Multiclass food classification using tensorflow. https://www.kaggle.com/theimgclist/multiclass-food-classification-using-tensorflow/.

[3] Jason Brownlee. 2020. Neural Networks Hyperparameter tuning in tensorflow 2.0. https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

[4] Karthik Chaudhary. 2020. Understanding Audio data, Fourier Transform, FFT and Spectrogram features for a Speech Recognition System. https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520/.

[5] Dan Ellis. 2007. Chroma Feature Analysis and Synthesis. https://labrosa.ee.columbia.edu/matlab/chroma-ansyn/.

[6] Enthought. 2015. Basic Sound Processing in Python | SciPy 2015 | Allen Downey. https://www.youtube.com/watch?v=0ALKGR0I5MA.

[7] Haytham Fayek. 2016. Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between. https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html

[8] Christopher Harte, Mark Sandler, and Martin Gasser. 2006. Detecting harmonic change in musical audio. *Proceedings of the ACM International Multimedia Conference and Exhibition.* https://doi.org/10.1145/1178723.1178727

[9] Dan-Ning Jiang, Lie Lu, HongJiang Zhang, Jianhua Tao, and Lianhong Cai. 2002. Music type classification by spectral contrast feature. *Proceedings. IEEE International Conference on Multimedia and Expo* 1 (2002), 113–116 vol.1.

[10] LibROSA. 2019. Feature Extraction - Librosa 0.7.2 documentation. https://librosa.org/librosa/feature.html.

[11] LibROSA. 2019. Image classification | TensorFlow Core. https://www.tensorflow.org/tutorials/images/classification.

[12] Derrick Mwiti. 2020. Build a deep learning model to classify images using Keras and TensorFlow 2.0. https://heartbeat.fritz.ai/build-a-deep-learning-model-to-classify-images-using-keras-and-tensorflow-2-0-379e99c0ba88.

[13] Pratheeksha Nair. 2018. The dummy's guide to MFCC. https://medium.com/prathena/the-dummys-guide-to-mfcc-aceab2450fd.
[14] Aaqib Saeed. 2020. Urban Sound Classification, Part 1. http://aqibsaeed.github.io/2016-09-03-urban-sound-classification-part-1/
[15] Aaqib Saeed. 2020. Urban Sound Classification, Part 2. http://aqibsaeed.github.io/2016-09-24-urban-sound-classification-part-2/
[16] SciPy.org. 2020. SciPy v1.5.1 Reference Guide. https://docs.scipy.org/doc/scipy/reference/generated/scipy.io.wavfile.read.html.
[17] Jivitesh Sharma, Ole-Christoffer Granmo, and Morten Goodwin. 2019. Environment Sound Classification using Multiple Feature Channels and Attention based Deep Convolutional Neural Network. arXiv:1908.11219 [cs.SD]
[18] SiDdhartha. 2019. Neural Networks Hyperparameter tuning in tensorflow 2.0. https://medium.com/ml-book/neural-networks-hyperparameter-tuning-in-tensorflow-2-0-a7b4e2b574a1
[19] Yu Su, Ke Zhang, Jingyu Wang, and Kurosh Madani. 2019. Environment Sound Classification Using a Two-Stream CNN Based on Decision-Level Fusion. *Sensors* 19 (04 2019), 1733. https://doi.org/10.3390/s19071733