Machine Learning: Dog Bark Identification

Bryan Khuu, Pawel Juda. Spring 2020

This research paper introduces and executes approaches to identifying whether a dog has barked in an audio file using machine learning statistical models.

The project repository is located on GitHub at paweljuda/MLDogBarkIdentification

Part A: Introduction

Background and Rationale

Speech recognition technology has become increasingly advanced within the passing decade. The increase in computing power of our personal devices has allowed for larger clusters of data to be obtained and managed. This is particularly significant as voice recognition is a computationally heavy process. Audio signals contain numerous features were can be extracted for data. Human voice for example, can be analyzed for unique frequencies, bandwidth, and other features which make it distinguishable from other audio or people.

In the following sections, audio samples will be processed and analyzed to determine if similar voice recognition technology can be implemented to recognize dog speech.

Data Sourcing

To improve reliability and prevent overfitting, the models are trained on multiple breeds. A reliable dataset which was found and used comes from kaggle, provided by the user mmoreaux. The link to the dataset is https://www.kaggle.com/mmoreaux/audio-cats-and-dogs and contains 113 dog audio files and 167 cat audio files. Although cat audio files do not fall within the scope of the project's goal, they contribute to a part of the negative train/test set. The second dataset was provided by a handy dandy bark maker, Leo, a personal dog assistant for this project. This dataset provides 14 longer audio files, containing numerous barks. Lastly, a third dataset was created from a mix of audio files from various YouTube videos.

Part B: Preprocessing and Feature Extraction

Data Preprocessing

Before any feature can be extracted, the audio files must go through pre-processing. This process sets up and modifies audio samples to make it easier and more useful for analysis.

The first step of pre-processing is noise reduction. This helps to eliminate unwanted background audio. This process is visualized in the spectrograms shown in figure 1 and 2. Spectrograms plot frequency and signal strength with respect to time. As can be seen, this process only preserves

significant events, where a possible dog bark might have occurred.

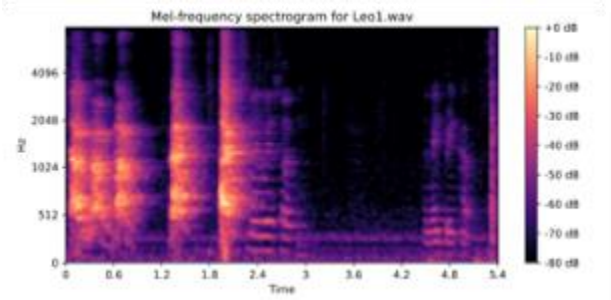


Figure 1: Original mel-spectrogram of a sample audio input.

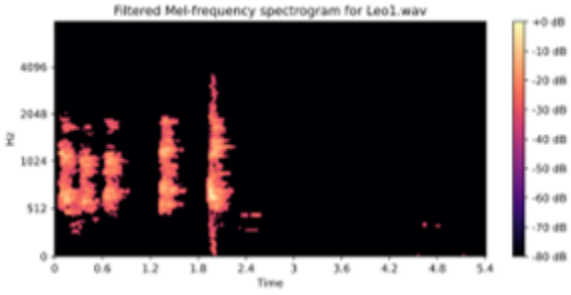


Figure 2: Amplitude filtered mel-spectrogram.

The new spectrogram, seen in figure 2, displays a sharper edge of where a significant audio event happens. These edges are used to help splice long audio files into multiple smaller files which only contain the desired audio. One spliced sample is shown in figure 3.

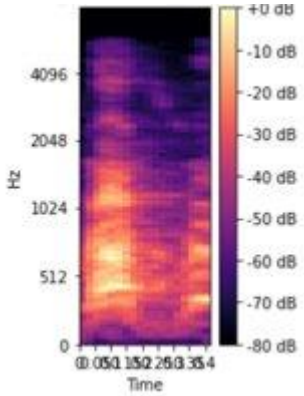


Figure 3: Sample of a spliced region from mel-spectrogram determining a significant event.

The next step in this process is to classify the newly spliced audio files. This is done by individually examining each audio split and labelling it as 1 if it contains a dog audio, or 0, if it contains no dog audio. This is known as machine supervised learning, a process where input and corresponding outputs are manually written into the system. These data points are used to train the model.

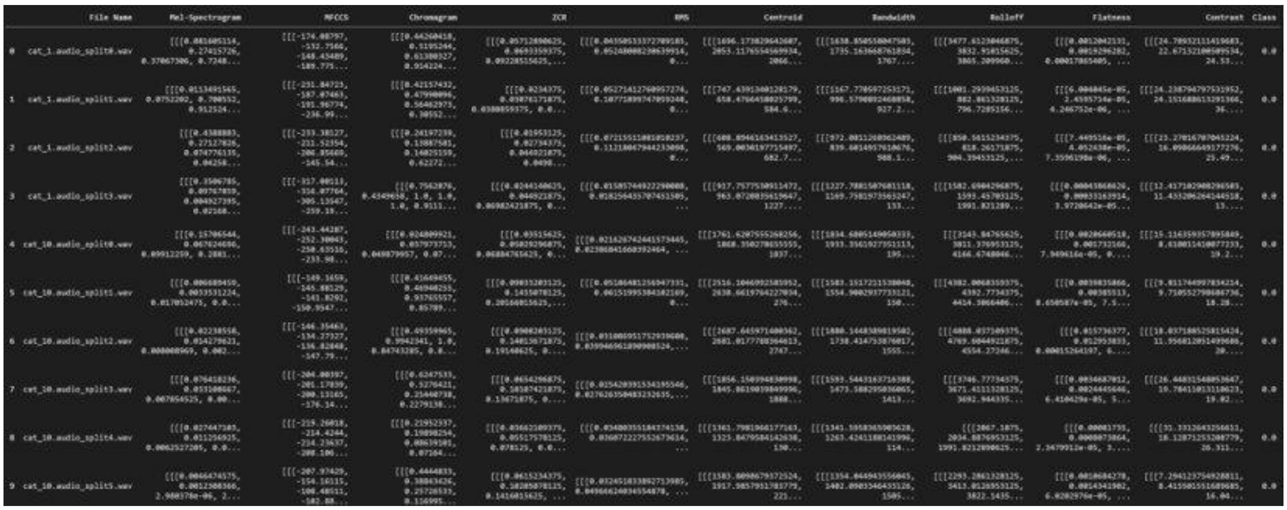
1111	Leo12-13.wav	1
1112	Leo12-14.wav	0
1113	Leo13-0.wav	0
1114	Leo13-1.wav	1
1115	Leo13-2.wav	1
1116	Leo13-3.wav	1
1117	Leo13-4.wav	1
1118	Leo13-5.wav	1
1119	Leo13-6.wav	0
1120	Leo13-7.wav	1
1121	Leo13-8.wav	1
1122	Leo13-9.wav	1
1123	Leo13-10.wav	1
1124	Leo13-11.wav	1
1125	Leo14-0.wav	0
1126	Leo14-1.wav	1

2086	monkey-43.wav	0
2087	monkey-44.wav	0
2088	monkey-45.wav	0
2089	monkey-46.wav	0
2090	monkey-47.wav	0
2091	monkey-5.wav	0
2092	monkey-6.wav	0
2093	monkey-7.wav	0
2094	monkey-8.wav	0
2095	monkey-9.wav	0
2096	pig noise-0.wav	0
2097	pig noise-1.wav	0
2098	pig noise-2.wav	0
2099	pig noise-3.wav	0
2100	Recording_1-0.wav	0
2101	Recording_1-1.wav	0
2102	Recording_1-10.wa	0

Figure 4: Samples of classified spliced audio file data.

Feature Extraction

Feature extraction is performed on all the samples and stored in the data frame using a schema as shown in figure 5. The features used are mel-spectogram, MFCC, chromagram, zero-crossing rate (ZCR), root-mean square (RMS), spectral centroid, spectral bandwidth, spectral roll-off frequency, spectral flatness, and spectral contrast. The library, LibROSA, was used for extracting these features.



	File Name	Mel-Spectrogram	MFCCS	Chromagram	ZCR	RMS	Centroid	Bandwidth	Rolloff	Flatness	Contrast	Class
0	cat_1.audio_split0.wav	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.0
1	cat_1.audio_split1.wav	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.0
2	cat_1.audio_split2.wav	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.0
3	cat_1.audio_split3.wav	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.0
4	cat_10.audio_split0.wav	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.0
5	cat_10.audio_split1.wav	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.0
6	cat_10.audio_split2.wav	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.0
7	cat_10.audio_split3.wav	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.0
8	cat_10.audio_split4.wav	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.0
9	cat_10.audio_split5.wav	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.0

Figure 5: First 10 rows of dataset dataframe holding extracted features and file classification

Part C: Model Setup and Results

Baseline Method: Logistic Regression

Logistic regression is a basic way of separating and classifying samples. This method can be visualized using the 3D and 4D models generated below. There are areas where similarly classified data points are clumped together. Using logistic regression, these data points can be separated using boundaries like hyperplanes. Data points which land on same side of the boundary get the same determined classification. This logic also holds true for higher dimensional models as more features are included, thought it becomes impossible to visualize. The logistic regression model utilized 7 of the features extracted: RMS, ZCR, spectral centroid, bandwidth, roll-off, flatness and contrast. These features were used to calculate corresponding β values and an intercept.

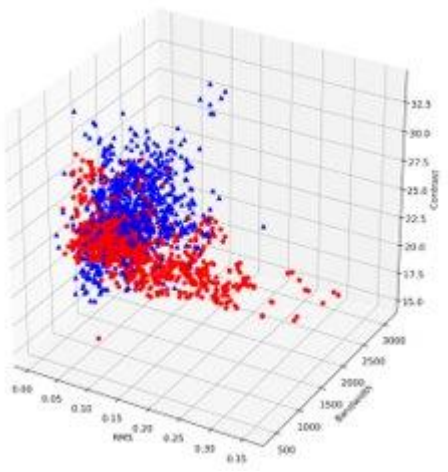


Figure 6: 3D Feature model, RMS, BW, Contrast

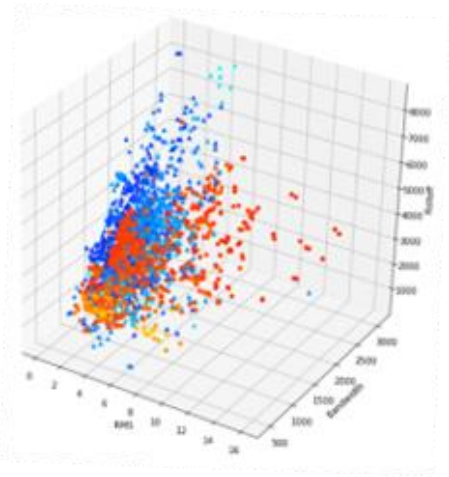


Figure 7: 4D Feature model, RMS, BW, Contrast, Roll-Off

The sklearn library provides a convenient function which calculates beta values with which a hyper plane can be constructed for the dataset. After separating the samples into a train and test set, these were the results: accuracy (76-81%), recall (~73%), precision (~77%). Utilizing an offset from –0.5 to 0 for the hyperplane realized even greater accuracy, thought it also runs small risks of decreasing accuracy. The following values were found using an offset of –0.5: accuracy (75-84%).

Convolutional Neural Network (CNN)

A second method of approach are convolutional neural networks (CNN). The results of the linear regression model were used as a baseline to compare against our second model choice, a CNN, whose architecture is shown in figure 8. The is composed of two layers of 2D convolutions and max pooling + batch normalization. The first 2D convolution utilizes a 5x5 kernel whilst the second one is a 3x3 kernel. After the two layers the result is flatten and passed through a two hidden layer neural network with a final activation layer utilizing a sigmoid function on the basis that the CNN is solving a binary classification problem.

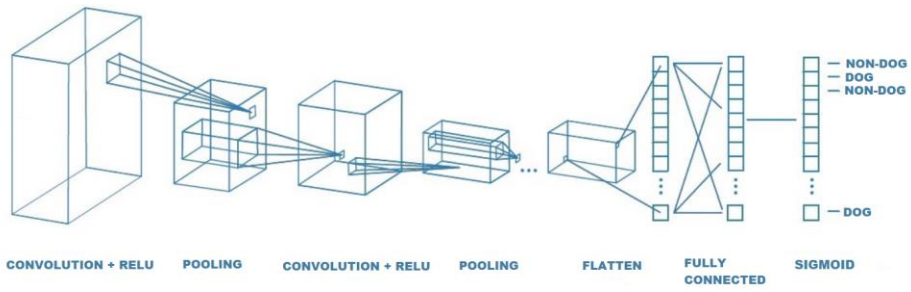


Figure 8: Convolutional Neural Network Architecture

One of the issues with using the mel-spectrogram as an image input for the CNN was that all image inputs need to be of the same size. This was fixed by padding spectrograms of all samples with the global minimum to a uniform maximum width (heights of spectrograms were the same). The padded spectrograms were then normalized to a value between 1 and 0 to improve performance of the CNN. A sample of the resulting padded and normalized spectrograms is shown in figure 9 below.

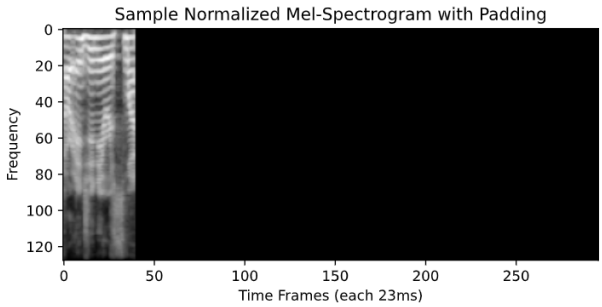


Figure 9: Example normalized and padded mel-spectrogram.

The results of the CNN model predictions are shown in table 1 below where they are compared to the results of the baseline logistic regression.

Method	True Pos	True Neg	False Pos	False Neg	Precision	Recall	Accuracy
LogReg	33%	45%	10%	12%	77%	73%	78%
CNN	40%	57%	0.5%	3%	99%	93%	97%

Table 1: Model Performance Comparisons

Part D: Conclusions and Future Work

The results of logistics regression and convolutional neural network were strong. As a baseline, logistic regression got a fair 80%. The CNN provided a much stronger approach reaching a whopping 97% accuracy with a 99% precision. However, its validation accuracy was unstable, as seen in figure 10, which will require changes to the architecture, learning rate and decay. Additionally, the environment used (WSL Ubuntu 16.04) for training the CNN did not support CUDA and therefore an NVIDIA GPU was not recognized, therefore setting a limiter on the number of epochs used, which was 30 where it would have otherwise been set to 100. More epochs could stabilize the validation accuracy.

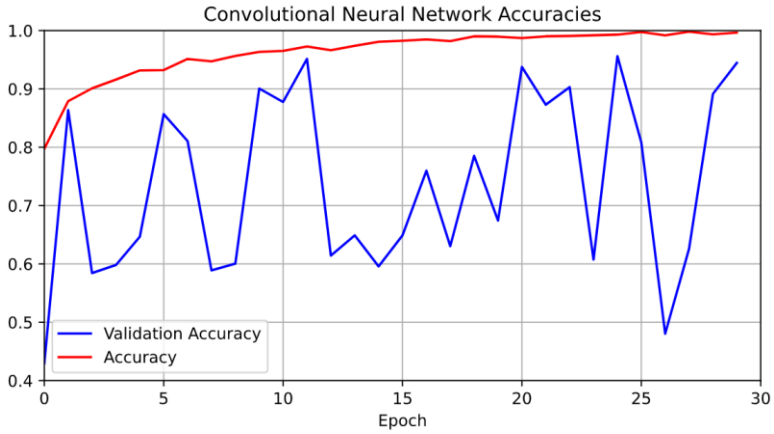


Figure 10: CNN Model Accuracies per Epoch

One improvement that may have improved results is a better splicing of the samples. There are cases where dog barks occur too closely together for the noise reduction method to finely separate the events. The dataset would produce more viable features if they were spliced apart into singular dog barks. This can be better achieved using an edge filtering convoluting kernel. As seen in figure.11 and figure.12 below, the filter makes each peak more distinct. The edge filtering convolutional kernel used was [-2,-1,1,4,1,-1,-2].

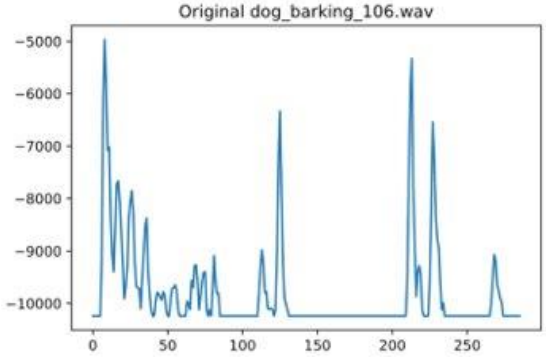


Figure 11: Original audio file's sum of mel-spectrogram columns

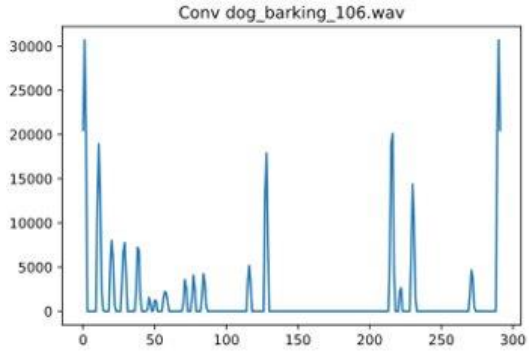


Figure 12: Convolution of audio file's sum of mel-spectrogram columns

For future work, there are many possible tasks. One thing that would be fun to implement is a machine learning model that can identify a specific breed or even a specific dog. Another task could include trying to trick the model into marking a fake bark as a real one. This could be made a person trying to imitate a dog. Lastly, an option for using our trained models on real life data would include using an open microphone to record audio and in real time predict if a bark occurred and take recordings of the number of barks as well as the timings.