A Comparison of Methodologies in Audio Analysis and Classification

Isayha Raposo
Computer Science Department
University of Northern British Columbia
Prince George, Canada
iraposo@unbc.ca

Tyler Martens
Computer Science Department
University of Northern British Columbia
Prince George, Canada
tmartens@unbc.ca

Alex Kitsul
Computer Science Department
University of Northern British Columbia
Prince George, Canada
kitsul@unbc.ca

***Abstract*—In this paper, we present and discuss a program capable of analyzing and ultimately classifying input .wav files of one of 10 distinct sound classes. The program utilizes two models, based off of the k Nearest Neighbour (KNN) and Logistic Regression algorithms respectively, allowing for a comparison of the two models and their prowess at solving such classification problems. The program also utilizes three pre-processing methodologies, allowing for further discussion regarding audio file analysis techniques for better audio file classification. Experiment results are based on a data set of approximately eight-thousand data files ranging from fractions of a second to four seconds in length, collected from *urbansounddataset.weebly.com* [1] [2].**

***Index Terms*—audio, analysis, accuracy, comparison, pre-processing, classification, model, KNN, Logistic Regression**

I. Introduction

The construction and subsequent training of a machine learning model capable of classifying or otherwise quantifying input media of a particular form is not a new concept in the realms of data science and artificial intelligence, and yet the practical applications of such a model seem to grow indefinitely over time and especially as general availability of computational power increases. Utilities such as speech-to-text converters and copyright identification systems are just a few examples in a nearly endless and diverse range of modern applications for such models. Other examples include those as recent and relevant as assistance in diagnosis and monitoring of SARS-CoV-2 and it's various symptoms by means of human speech analysis [3]. As demand for such models grows, especially where they are intended to be embedded in real-time systems (such as those responsible for operation of autonomous vehicles [4]), so too does the need for improvements in efficiency of underlying data pre-processing, training, and analysis techniques.

In this paper we present a proof-of-concept program featuring two distinct classification models capable of accurately classifying testing data via training data, both of which are provided: The program utilizes the UrbanSound8K Dataset [1] [2], a collection of over eight-thousand .wavs files of ten different, distinct classes (air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, and street music) divided into ten pre-defined "folds" for easy 10-fold cross-validation (note that each fold contains .wav files of varying classes, as opposed to each fold containing .wav files of only a single class). More specifically, the program contains a .wav file pre-processor with three pre-processing methodologies to choose from, and two multi-classification models based on the k Nearest Neighbour (kNN) and Logistic Regression algorithms respectively. In addition to presenting the program itself, we also compare and contrast results yielded via selection of all three pre-processing methodologies for both multi-classification models, in order to better understand their performances, strengths, and weaknesses. Lastly, we conclude with which of the pre-processing methodologies and which of the models reign supreme overall, and suggest improvements for both, as well as modifications that may allow for improved (e.g. more precise) future comparisons.

II. Background/Related Work

As mentioned prior, classification of audio by use of artificial intelligence and data mining algorithms is nothing new in terms of research. Another study very similar to our own involved the development and delivery of a KNN-based (described in detail in later sections) model used to control a robotic arm via voice commands [5]. The Arduino-based "arm" was controlled via a program built in Python that extracted critical information from input audio and ran said information through a trained KNN model capable of accurately classifying the input as one of two command classes: *"Ambil"* and *"Simpan"* (*"Pick"* and *"Place"* in Indonesian, respectively). The methodology employed in order to extract such critical information involved a combination of Mel-frequency Cepstral Coefficients (MFCCs - also described in detail in later sections) and Linear Predictive Coding (LPC). The experiment yielded accuracy rates of eighty to eighty-five percent, and the robot arm responded to commands as expected where classification of said commands was also accurate [5].

Another similar but much more complex study involved the use of a Convolutional Neural Network (CNN) and a Tensor Deep Stacking Network (TDSN) to classify input audio (environmental sounds) [6]. Interestingly, unlike our own project and the project described prior [5], *this* project first converts input audio into spectrograms, visual representations of audio signals capable of describing attributes such as frequency and power, and compresses said spectrograms as much as possible while still preserving the ability to extract

critical information from them. Experimentation lead to results including but not limited to demonstratively high accuracy (i.e. seventy-three percent for the CNN) relative to other common spectrogram-based classification systems, such as combined usage of MFCCs and Support Vector Machines (SVMs) [6]. In summation, the paper demonstrates a clever and effective means of sound classification via *image* classification.

III. Preamble

A great amount of pre-processing must be performed on the .wav files within the given data set in order to successfully extract a number of relevant features that should allow our two classification models to differentiate amongst the 10 different classes of sounds found therein. The vast majority of this pre-processing is performed with the aid of the Librosa 0.8.1 Python package. Librosa is specifically made for music and audio analysis applications, providing several helpful methods for retrieving and manipulating audio features [7]. Librosa was used in conjunction with other, more common Python packages, such as NumPy, which allows for a simplified means of writing and performing complicated mathematics. Both of these packages were integral to generating some of the most important features of this project, including Mel-frequency Cepstral Coefficients (MFCCs). MFCCs are known to help differentiate sounds exceptionally well (in short, they effectively quantify/describe timbre) and thus are commonly employed as a means of performing speech recognition [8].

Early on in development, we compiled a list of audio features that are relatively simple to extract from .wav files, whether via Librosa functions or otherwise. Members of the list were also considered to be of potential significance in regards to classifying such .wav files. This lead to our first pre-processing methodology. This methodology extracts and saves the mean spectral flatness and the mean spectral centroid of input .wav files, the former of which, according to Librosa documentation, effectively describes the *tonality* of audio (i.e. can be used to distinguish between tones and noise). Additionally, this methodology also extracts the mean and maximum amplitudes found within three pre-defined low, medium, and high frequency bands (definitions are given in **Results**). Some features (such as minimum and maximum overall frequency) were developed but ultimately excluded as they were found to be relatively insignificant.

Later, discovery of MFCCs lead to our second pre-processing methodology, which simply consisted of twenty mean MFCC values per .wav file. Lastly, our third and final pre-processing methodology was a combination of those which precede it, yielding a whopping twenty-eight attributes per .wav file.

It should be noted that, for *all* of the three pre-processing methodologies described above, each of the .wav files is bumped down from stereo to mono, trimmed of any silence, and, for both classification models developed, training and testing data points are standardized to accommodate for outliers in training data. Additionally, .wav files with given salience values indicating high levels of background noise were simply removed from consideration altogether (this is described with more detail in **KNN Methodology**.

Standardization is performed using the function StandardScaler, a part of the Sci-Kit Learn Python package that works by "removing the mean and scaling to unit variance" [11]. The standard score of a sample x is calculated as:

$z = (x - u)/s$

"... where u is the mean of the training samples or zero if with-mean=False, and s is the standard deviation of the training samples or one if with-std=False. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using transform" [11]. Standardization of data sets is common practice as models such as KNN and Logistic Regression may behave adversely when trained using data sets that do not follow at least a loose representation of normal distribution [12].

A. MFCC Introduction

In order to derive MFCCs from an audio file, one must first evaluate the Fourier Transform of the signal wave. A Fourier Transform is a mathematical equation for transformation that decomposes a function based on space or time into a new function based on space or time frequency of the wave. A Fourier Transform can be used to turn a musical chord (a singular wave when recorded) into the individual waves of the notes that produce the chord. Separating a sound into the sum of its parts may help in classifying the sound itself as this opens up the possibility of analyzing the compositional waves allowing for discovery of a larger range of/more specific features, which, in turn, should ideally increase a classifier's output accuracy. A Fourier Transform can be performed using the following formula [9]:

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \xi}, dx, \quad \forall \xi \in \mathbb{R}.$$

"The Fourier transform is denoted here by adding a circumflex to the symbol of the function. When the independent variable *x* represents time and, the transform variable ξ represents frequency (e.g. if time is measured in seconds, then frequency is in hertz)." [9]

After Fourier Transforms have been calculated, generation of an MFCC can continue: The next step is to map the powers of the spectrum to the mel scale. The time window of a single amplitude is mapped in a triangular pattern to the mel scale. The mel scale is related to the idea of hertz and can be converted to hertz using the following formula [8]:

$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$ (where m is the number of mels and f is the frequency in hertz)

We then calculate the *log* of the powers of the resulting mel frequencies. Lastly, we calculate the discrete cosine transform of the resulting list of mel log powers, which represents a finite

set of data points as a series of cosine functions. The MFCCs are the resulting amplitudes of the result spectrum [8]. As stated earlier, Librosa was made for audio feature extraction and manipulation, and includes a flexible MFCC generation function [7], which removes the onus of the laborious process of calculating these values via a manually written set of functions.

IV. KNN Model introduction

One of the two classification models employed in this the project is a weighted KNN, or K Nearest Neighbor, model. KNN classifies an entry based on a poll of the k (a number) nearest existing entries. The effect these points have on the final decision is based on their weight which is affected by distance from the newly added point in question. the formula for determining the weighting of neighbors is given below [10].

$weight = (1/((distances[neighbour])^2))$

Choosing a value for k that best represents the data can be a very difficult task. Larger values are best to reduce noise and outliers but can blur the distinction between classes. Values considered too small can cause entries close to the class boundaries to be classified incorrectly. k should never be an even number as this opens up the possibility of a tie in the poll when classifying. Tests of the data set have indicated that 5 is probably the optimal k value for this project and returns the best results. The KNN model was selected for this project because of how simple it is to implement in comparison to other models, all while retaining consistent and accurate results. However, one downside to KNN is that, if implemented naively, KNN may prove computationally expensive on large data sets [10] [13].

V. Logistic Regression Introduction

The other classification model employed in this project is a Logistic Regression model. In such a model, a mathematical function represents a binary dependant variable or question (e.g. "is this sound the sound of dogs barking?"). Natively, this model only supports binary classification, but by adding a few extra steps, Logistic Regression can be applied to a multi-classification problem such as that which is described within this project. The logistic function is applied on each possible class and the logarithm of the odds of each of these is some value between 0 and 1 (0 meaning the model is confident the input is not of the class in question; 1 meaning that the model is confident that it *is* of said class). Whichever class receives the highest probability is chosen as the classification for the entry. For a multi-class implementation of Logistic Regression, there are two options: One-vs-Rest, or One-vs-One. In One-vs-Rest, the multi-class problem is split into a binary classification problem per class, and results for each are weighed against one another. In One-vs-One, the multi-class problem is split into a binary classification problem for each *pair* of classes [12]. The One-vs-Rest solution was selected for this project for it's relative simplicity in understanding. However, if naively implemented, One-vs-Rest can be quite troublesome, as it requires a model be made for each individual class. A Python package known as Sci-Kit Learn includes a set of functions and classes for easy development and implementation. Use of this package was absolutely vital to the project as it simplifies development and model comparisons immensely (i.e. by avoiding simple developer error).

VI. KNN Methodology

As mentioned prior, the project employs the use of the UrbanSound8K data set. Of the 8733 .wav files, nine of the ten folders, or roughly 7900 .wav files (note that the folders are not uniform in size), are used as training data (with the remaining 800 or so files being instead used as test data) at any given time. A provided metadata file includes some key information regarding all of the .wav files, including but not limited to duration/length, class, and salience. Salience values refer to whether or not the sound is very loud and clear (i.e. as if it was recorded very close to a microphone), as opposed to being slightly muffled (i.e. as if it was recorded at a distance). .wav files with salience values implying the latter were, as mentioned prior, removed from consideration altogether because they were found to make (accurate) classification exceedingly difficult for a project of this scope. The downside to this removal is it notably reduces the size of the data set: The smaller data set size could, ironically, contribute to a lack of accuracy as well.

As mentioned prior, the .wav files are also trimmed of beginning and trailing silence, converted from stereo to mono where applicable, and sample rates are left unchanged. Using attribute sets generated via whichever of the three pre-processing methodologies (described in the **Preamble**) is chosen on a given run of the program, data points stemming from test data are compared to those stemming from training data by calculating and checking the euclidean distance from the 5 nearest neighbors for each test data point, in addition to calculating said neighbour's influence(s) via the weighting formula described prior. This process returns a classification and a certainty or confidence in that classification in the form of a percentage. The actual classifications of the test data points are, of course, already known, and so the model's output is then compared against the known values and an accuracy percentage is calculated.

VII. KNN Evaluation

Before introducing weighting and data standardization into the KNN model, it would only classify sounds with an average of roughly 44.9 percent accuracy for a particular set of test data pre-processed using the MFCC-based methodology, which was far below our goal of 50 percent. After introducing distance weighting, the model was able to classify the very same set of test data with 51 percent accuracy, just slightly above our goal. With weighting and standardization added (using the StandardScalar function described prior), this average accuracy skyrocketed up to 69 percent. Later, averaging at 59.5 percent accuracy after cross-validation when using the best of our three pre-processing methodologies, the KNN model far exceeded

the goals set out for this project. The accuracy of the model could theoretically be improved with a larger valid data set, as well as data that is more normalized (meaning files all recorded in the same sample rate, same length and relative volume - not digitally amplified or recorded with extra gain, etc.). This way we would have data points that better relate to each other and better represent the class the belong to. When using a KNN model, it is highly recommended to reduce the number of dimensions if there exists more than 10 in order to avoid the curse of dimensionality [10]. When using the combined pre-processing method (that which yielded the best results for both KNN and Logistic Regression) there are 28 dimensions to consider. This is obviously far above the recommended 10, meaning the model may suffer from the curse of dimensionality. If this is true it may be beneficial to find a way to reduce the number of dimensions by either combining attributes or removing those of lesser relevance (e.g.: experimentally reducing the number of MFCCs used).

VIII. Logistic Regression Evaluation

As with the KNN model, the Logistic Regression model may have seen an increase in accuracy with better, more normalized data. Additionally, the Sci-Kit Learn package StandardScalar includes a lot of functions and options that were not fully explored in processing this data set; options that may have achieved better results (the very same goes for the KNN model). StandardScalar is just one of many standardization functions included, and a different means of scaling could very well better center the data. Another point of improvement to to consider is balanced weighting of the classes: Sklearn suggests that the 'class_weight' parameter of it's Logistic Regression function be set to balanced, in order to prepare the function for an imbalance in input per class. However, in the case of this project, switching from the default option to 'balanced' actually diminished the accuracy in most cases. Even so, there are still several other class weights and penalty/evaluation combinations possible that could potentially yield accuracy improvements. Similar to the KNN model, utilizing the Logistic Regression model with the first pre-processing methodology yielded an average of 43.9 percent accuracy, failing to meet our goal of 50. After switching to the second pre-processing methodology, the average accuracy rose to 52.1 percent, just above our aforementioned goal. The third, combinatorial pre-processing methodology yielded an average accuracy of 58.4 percent, again meeting project goals within a decent margin.

IX. Raw Results

A. Earlier/First pre-processing methodology

Selected by inputting 0 at the program's pre-processing methodology prompt.
Attributes (8 total) include:

- Mean Spectral Flatness
- Mean Spectral Centroid
- Low Frequency Band - 0hz (incl.) up to 200hz (exc.)
  - Mean Amplitude
  - Max Amplitude
- Mid Frequency Band - 200hz (incl.) up to 2000hz (exc.)
  - Mean Amplitude
  - Max Amplitude
- High Frequency Band - 2000hz (incl.) and up
  - Mean Amplitude
  - Max Amplitude

Test Fold	KNN Acc./Time	Log. Reg. Acc./Time
1	38%, 0s	32%, 2s
2	46%, 0s	45%, 2s
3	43%, 0s	40%, 2s
4	33%, 0s	38%, 2s
5	41%, 0s	43%, 2s
6	48%, 0s	46%, 2s
7	54%, 0s	43%, 2s
8	48%, 0s	53%, 2s
9	56%, 0s	42%, 2s
10	42%, 0s	57%, 2s
All Folds (Avg.)	44.9%, 0s	43.9%, 2s

B. Later/Second pre-processing methodology

Selected by inputting 1 at the program's pre-processing methodology prompt.
Attributes (20 total) include 20 averaged MFCCs.

Test Fold	KNN Acc./Time	Log. Reg. Acc./Time
1	56%, 0s	38%, 2s
2	54%, 0s	45%, 2s
3	49%, 0s	57%, 2s
4	43%, 0s	46%, 2s
5	58%, 0s	48%, 2s
6	55%, 0s	50%, 2s
7	55%, 0s	54%, 2s
8	59%, 0s	64%, 2s
9	63%, 0s	48%, 2s
10	69%, 0s	71%, 2s
All Folds (Avg.)	56.1%, 0s	52.1%, 2s

C. Combined/Final pre-processing methodology

Selected by inputting 2 at the program's pre-processing methodology prompt.
Attributes (28 total) include all those described by the two preceding processing methods.

Test Fold	KNN Acc./Time	Log. Reg. Acc./Time
1	57%, 1s	46%, 3s
2	59%, 1s	53%, 3s
3	56%, 1s	57%, 3s
4	51%, 1s	56%, 3s
5	60%, 1s	60%, 3s
6	63%, 1s	60%, 3s
7	55%, 1s	59%, 3s
8	61%, 1s	69%, 3s
9	68%, 1s	58%, 3s
10	65%, 1s	66%, 3s
All Folds (Avg.)	59.5%, 1s	58.4%, 3s

As recommended by the authors of the UrbanSound8K dataset [1] [2], we use 10-fold cross-validation in order to obtain a more refined sense of accuracy for each model. More specifically, the UrbanSound8k dataset comes divided into 10 pre-defined "folds" of .wav files. For each of our 3 aforementioned pre-processing methodologies, we classify (using both KNN and Logistic Regression) the contents of each of the 10 "folds" by using the other 9 as training data.

X. Conclusion

The results clearly demonstrate that both models average accuracies react very positively with each evolution in pre-processing, and to a similar degree. Additionally, the results demonstrate that MFCCs are a stronger attribute to employ when classifying audio than those used originally. Another increase in average accuracy was observed when the attribute groups were combined. This shows that the original attribute set must be helpful in classification of sounds in at least some capacity. The slight loss of accuracy in some cases also leads us to believe that the original attribute set may also cause the model to misidentify outlier points, despite average accuracy increases. Overall, the two models were very competitive when it came to accuracy, with the KNN model beating the Logistic Regression model by only 0.9 percent with the combined pre-processing methodology. The real discrepancy in ability between the two begins to show when comparing the time it took the models to classify a folder. The KNN model consistently took only one third the amount of time needed by the Logistic Regression model, though this could be due to the blackbox-esque nature of the Logistic Regression implementation. In conclusion, with a data set of this size and quality, a KNN model is likely to classify a sound with greater accuracy and speed than a Logistic Regression model, though the means through with the sound is processed and quantified are far more significant to the results than the model chosen.

References

[1] J. Salamon, C. Jacoby, and J. P. Bello, "UrbanSound8K," Urban Sound Datasets. [Online]. Available: https://urbansounddataset.weebly.com/urbansound8k.html. [Accessed: 06-Dec-2021].

[2] J. Salamon, C. Jacoby, and J. P. Bello, "A Dataset and Taxonomy for Urban Sound Research," Proceedings of the 22nd ACM international conference on Multimedia, pp. 1041–1044, Nov. 2014.

[3] G. Deshpande, A. Batliner, and B. W. Schuller, "AI-based human audio processing for COVID-19: A comprehensive overview," Pattern Recognition, vol. 122, p. 108289, Aug. 2022.

[4] H. Gao, B. Cheng, J. Wang, K. Li, J. Zhao and D. Li, "Object Classification Using CNN-Based Fusion of Vision and LIDAR in Autonomous Vehicle Environment," in IEEE Transactions on Industrial Informatics, vol. 14, no. 9, pp. 4224-4231, Sept. 2018, doi: 10.1109/TII.2018.2822828.

[5] D. Anggraeni, W. S. M. Sanjaya, M. Munawwaroh, M. Y. S. Nurasyidiek and I. P. Santika, "Control of robot arm based on speech recognition using Mel-Frequency Cepstrum Coefficients (MFCC) and K-Nearest Neighbors (KNN) method," 2017 International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA), 2017, pp. 217-222, doi: 10.1109/ICAMIMIA.2017.8387590.

[6] A. Khamparia, D. Gupta, N. G. Nguyen, A. Khanna, B. Pandey, and P. Tiwari, "Sound classification using convolutional neural network and tensor deep stacking network," IEEE Access, 2017.

[7] https://librosa.org/doc/latest/index.html

[8] T. Ganchev, N. Fakotakis, and G. Kokkinakis (2005), "Comparative evaluation of various MFCC implementations on the speaker verification task Archived 2011-07-17 at the Wayback Machine," in 10th International Conference on Speech and Computer (SPECOM 2005), Vol. 1, pp. 191–194

[9] Kaiser, Gerald (1994), "A Friendly Guide to Wavelets", Physics Today, 48 (7): 57–58, Bibcode:1995PhT....48g..57K, doi:10.1063/1.2808105, ISBN 978-0-8176-3711-8.

[10] Cover, Thomas, and, Hart, Peter. Nearest neighbor pattern classification[J]. Information Theory, IEEE Transactions on, 1967, 13(1): 21-27

[11] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[12] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[13] Fix, Evelyn; Hodges, Joseph L. (1951). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties (PDF) (Report). USAF School of Aviation Medicine, Randolph Field, Texas.