Song Recognition via Feed-Forward Deep Neural Networks using Sonic Characteristics

Campbell Clarkson
Darla Moore School of Business, University of South Carolina, Columbia, South Carolina 29208,
campbell.clarkson@grad.moore.sc.edu

Drew Merrill
College of Engineering and Computing, University of South Carolina, Columbia, South Carolina 29208, merrila@email.sc.edu

Audio recognition is a popular, practical application of machine learning to help everyday consumers discover new songs or rediscover songs that they do not remember. We use a feed-forward deep neural network to identify songs from a selection based on the song's metadata. Our model differs from previous work, which uses the audio signal characteristics as opposed to the metadata, as is our case. Our specification, along with two robustness checks, show 94-97% Accuracy with several opportunities to work further on this project.

For a link to watch the presentation of this research, please visit `https://youtu.be/Qk8bSzFaNqs`. For Github files, please see footnote 3.

Key words: song identification, deep neural network, machine learning

1. Introduction

Machine learning is clearly ubiquitous throughout academic literature and in practice, from the smallest of start-up firms to large, established technology companies. One such area where computer science, and machine learning in particular, can benefit applications of ideas is dealing with searching through large data sets and solving heavily analytical problems. One such area, and the focus of this paper, is audio recognition. We examine the accuracy of a machine learning model at recognizing popular songs by name in this report.

Audio recognition has been deployed in many ways, but one particular area is in the music industry, where apps such as Shazam help users find particular songs, advertisements, and media without direct knowledge of the title, album, or artist. There is a clear practitioner benefit to applications like this because of an abundance of consumer demand. Further evidence of this benefit manifests itself in the recent news of Apple's purchase of

Shazam's platform[1] for a reported $400 million dollars, with the intention of making the application advertisement free.

Some factors which may inhibit traditional platforms of audio recognition include the presence of environmental noise which may obscure the song being presented, and the barrier to entry for music production being lowered with platforms like Spotify, which allow for unsigned and 'DIY' musical artists to produce and publish their own music. We attempt to address the first concern in this report and provide an avenue for mitigating the second. To understand the motivation for our particular application of machine learning to song recognition, we must understand what has been done in this particular area. As mentioned previously, Shazam is the largest platform which deploys audio recognition. To tackle some of the concerns about this type of model's robustness, research has examined implementing a model which tracks variation in qualities such as key, tempo, and ornamentation (Tao and Getachew 2020). These areas are where the robustness of the Shazam platform has fallen short (Xiao 2018). This research seeks to tackle the shortcomings of Shazam by examining factors not specific to the original song that the user seeks the name of. In contrast, we employ a set of characteristics very similar to Shazam.

The platform uses qualities which contribute to an overall fingerprint of the song being searched, such as the tempo, the energy, dance-ability, or even the happiness level of the song. While the particular qualities of songs that Shazam uses to create a 'fingerprint' of sorts[2]. We can circumvent the issue of non-accessible Shazam 'fingerprints' using publicly available metadata from Spotify to create our own fingerprint for the music upon which we test our model.

We use a curated data set compiled of metadata for 2000 popular songs from Spotify, and their accompanying characteristics, upon which we compute Mel Frequency Cepstral Coefficients for short snippets of each song. We then deploy a mid-sized feed forward network with just over 1.3 million trainable parameters, within which we find initial results of 94-97% accuracy, but offer challenges we encounter and possible solutions for the future.

[1] https://www.apple.com/newsroom/2018/09/apple-acquires-shazam-offering-more-ways-to-discover-and-enjoy-music/

[2] https://www.toptal.com/algorithms/shazam-it-music-processing-fingerprinting-and-recognition

2. Data and Methodology

In the following subsections, we describe our data and proposed methodology for this project. We have stored a repository for our code and data on Github[3].

2.1. Data

One of the most widely recognized aspects of machine learning is its ability to take large data sets and learn to classify examples it has seen via training. It is typically thought of as trivial for companies to obtain robust and large data sets, since they have the means and will to do so. For our project however we aim to take a much smaller data set and train models that will give both robust and accurate predictions. The data set we chose to train our models on is a data set of Spotify's 2000 most popular tracks (Singh 2020). This data set includes metadata such as title, artist, genre, year, BPM, "energy", etc. We compiled a list of 20 of these songs and downloaded them for our training examples. Some more traditional methods of song identification involve using metadata like this as a means to classify different songs (Cooper 2018). Our model takes three second segments of songs in order to classify the entire song. Section 2.2 provides more details into this segmentation. We also aim to provide a more robust model resistant to background noise provided in audio clips.

2.2. Methodology

The methodology for this project involves many common processes involved in typical machine learning. The majority of project infrastructure is used to preprocess and organize the data. Our team is too small to collect our own dataset so we opted to use the aforementioned Spotify dataset. We first select a subset of the full dataset, we developed tools to automate this process and others described here. Once a sample dataset of songs has been randomly selected we utilized Youtube-dl, a popular Python library to download the songs from YouTube.

Once the songs have been downloaded locally we use PyDub, which is a python wrapper for audio processing to segment the songs into variable lengths. We chose three second song snippets for this project. It is possible other lengths may yield superior results, something we would explore in the future. For our project there is no zero padding at the end of the final audio segment for a given song. If the length modulo three is not zero, the final

[3] https://github.com/RealColdFate/csce585

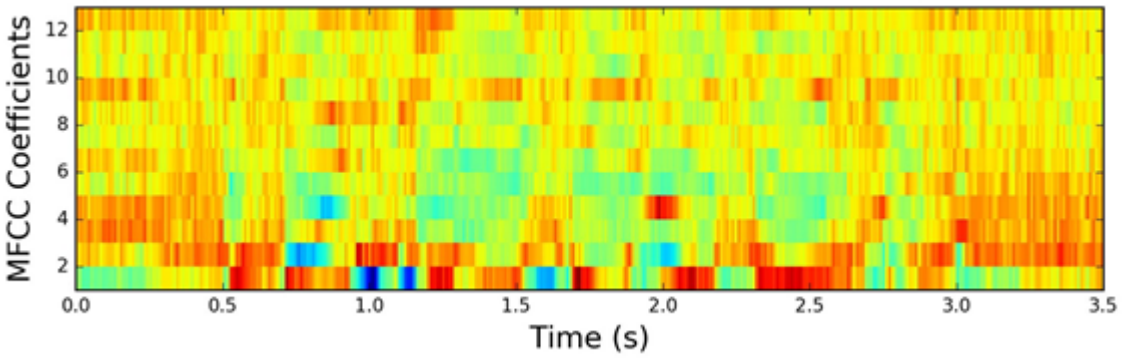

Figure 1 **Visual Representation of MFCCs over time**

segment will be discarded. The next phase in processing is generating additional training data. We have done this by overlaying outside sounds from the UrbanSound8k dataset (Salamon et al. 2014) to the song segments we already have. In some instances we add a small amount of random noise to the training examples.

The next portion of data processing is done primarily with the popular audio processing library Librosa (McFee et al. 2015). It is important to note there are some secondary dependencies not mentioned here such as ffmpeg that are crucial to our data processing. With librosa we compute Mel Frequency Cepstral Coefficients (MFCCs) for each of our audio segments. The MFCC is commonly used in sound recognition, particularly speech recognition, however this has been examined already for its application into music modeling by Logan (2000). This part of the data processing takes some time and for larger sets this portion of processing would likely be considered non-trivial compute time. Once the MFCCs are generated for a given segment they are normalized and the mean is taken over the y axis, for each time step represented by the x axis.

Finally the model is actually trained. The model training is straightforward; we use a mid-sized feed forward network with 1.3 million trainable parameters. The input is given as the mean values for the MFCCs over each time step. It is important to note that with larger song segments a new model would have to be trained. Our current model relies on the expectation that each segment is exactly 3000 milliseconds long. All model layers are densely connected and use relu as their activation function with softmax at the output layer. The model is trained for 250 epochs with a batch size of 128.

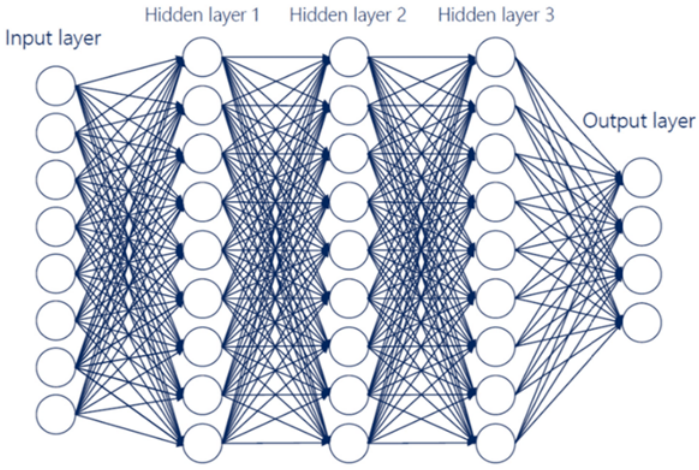

Figure 2 Model Concept

3. Results

Our results thus far are promising. We were able to attain 94-97% accuracy on our training set of 20 songs. This accuracy is fair, however we still have much to do with regard to testing the robustness of our model. With the additional training data we have generated via the methods mentioned above it is our hope that the model will prove to be more robust then other currently available solutions.

Our results show a high degree of variance of $\pm$ 3%. We believe this is due to the inherent nature of the difficulty of guessing a song with only a three second snippet. It is easy to imagine that the three seconds you were given just so happen to be a lull in the beat or an introduction that bears little resemblance to the rest of the song. This, however, does provide a unique challenge we are able to overcome this it will only serve to improve the robustness of our model as a whole. Auxiliary research shows that a time series model like an LSTM may improve robustness as well. We would like to look into this in the future. It is also important to note that our models so far only serve as a proof of concept since 20 songs is far from what a consumer might expect from a production model.

Our results of the primary model, shown in column (1) of Table 1, indicate that the average accuracy across 5 runs was 95.45%. Because a portion of our objective was to identify whether these audio recognition models would be robust, we conduct two robustness checks. First, we specify a model similar to the original, but with overlapped sound from the UrbanSound8k data set. These sounds can be things such as a dog barking, car horns, or construction sites. The reason this is an excellent model to compare to is its

Table 1 **Model Accuracy**

	Accuracy (%)		
Run No.	(1)	(2)	(3)
Run 1	96.59	96.87	96.02
Run 2	95.45	97.44	95.31
Run 3	94.89	96.58	95.45
Run 4	95.45	97.72	96.02
Run 5	94.89	95.73	94.89
Average Acc.	95.45	96.87	95.54
Original MFCC	Yes	Yes	No
Overlayed Noise	No	Yes	No
Normalized	No	No	Yes

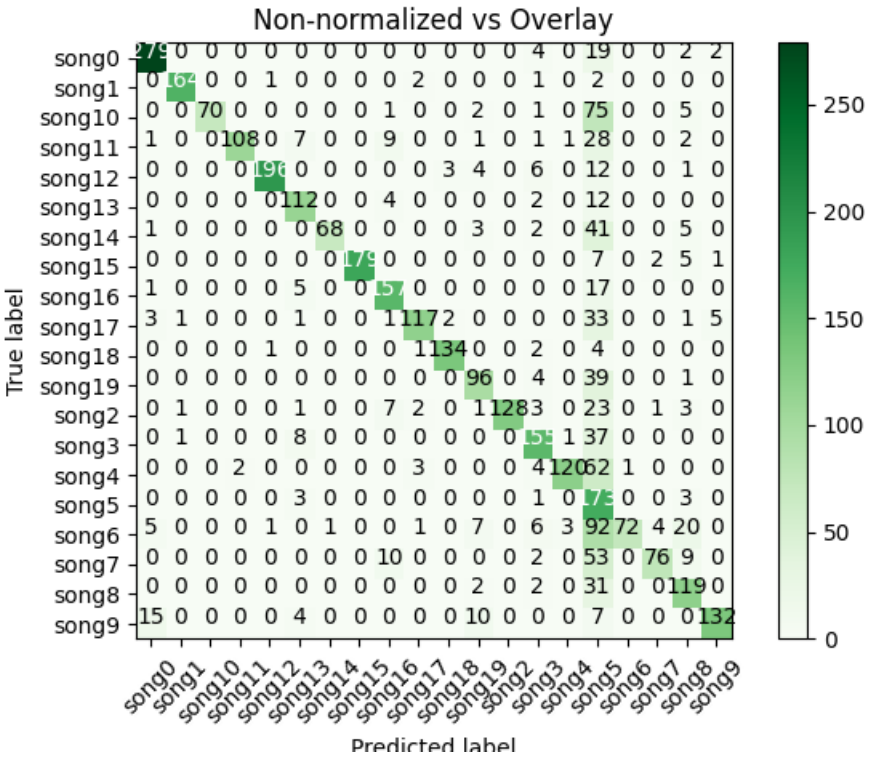

Figure 3 **Confusion matrix for non-normalized model fed overlay data**

usefulness in everyday life. When a user would be recording sound for recognition, it would be difficult to find music played in complete isolation. We find in column (2) that this had the highest accuracy and thus did not lose robustness. Second, we look at comparing the performance of the model with normalized MFCCs. We use Z-score normalization on the mean MFCC of the y-axis at each time period. With this model, we see in column (3) that it slightly outperforms the accuracy of a model defined without normalization, although the difference may prove to be negligible with further testing.

As an extension, we want to examine the model's ability to react to the other types of data we use in the other models. Of note in this experimentation is the most practical use case, where we would have to take data with environmental sounds and validate that against the model with non-normalized data. When we did this, our accuracy went down to 75.6%, meaning we should examine further why robustness struggles when the model receives data it has no way of seeing beforehand. The results of this are displayed in a confusion matrix in Figure 3.

4. Conclusion

In conclusion, we offer an approach to audio recognition that is very similar to deployed platforms like Shazam, which we imitate by compiling metadata for popular songs and their musical characteristics. While other work seeking to address the robustness in music recognition models looks at human variation in the input, like pitch and tempo, we examine the robustness of the characteristics that these platforms already deploy. We find that in the absence of any obscuring noise, we achieve 94-97% recognition accuracy with a mid-sized feed forward network. When we examine robustness with overlapping noises and a normalized MFCC model, we find similar levels of accuracy

We surmise that the initial results are promising, yet we have some issues we would like to address in future iterations of our model. Some problems associated with implementing noise are the lack of addressing audio recording. In practice, recording fidelity may play a huge role in the success of audio recognition platforms like Shazam or our model.

Despite these issues, the practical implications of a scaleable, financially feasible music recognition platform like Shazam (Wang et al. 2003) are highly beneficial to companies and end users. To discuss the possibilities of why these platforms and models are effective, we believe that the primary benefit is providing companies with a service to increase traffic to their platform. Should a company like Spotify develop a sophisticated copy of the Shazam service with access to its own databases of music, it would be able to keep up with the rapidly expanding creative content industry.

Acknowledgments

Thank you to Pooyan Jamshidi for giving guiding comments and value feedback during the 2020 Fall semester of CSCE 585: ML Systems.

We acknowledge that the results of this report are preliminary, and should not be cited as a reliable source in any academic publications without serious consideration.

References

Cooper T (2018) How shazam works. URL `https://medium.com/@treycoopermusic/how-shazam-works-d97135fb4582`.

Logan B (2000) Mel frequency cepstral coefficients for music modeling. *In International Symposium on Music Information Retrieval*.

McFee B, Raffel C, Liang D, Ellis DP, McVicar M, Battenberg E, Nieto O (2015) librosa: Audio and music signal analysis in python. *Proceedings of the 14th python in science conference*, volume 8, 18–25.

Salamon J, Jacoby C, Bello JP (2014) A dataset and taxonomy for urban sound research. *Proceedings of the 22nd ACM international conference on Multimedia*, 1041–1044.

Singh S (2020) Spotify - all time top 2000s mega dataset. URL `https://www.kaggle.com/iamsumat/spotify-top-2000s-mega-dataset/activity`.

Tao S, Getachew Y (2020) High fidelity song identification via audio decomposition and fingerprint reconstruction by cnn and lstm networks. *Stanford CS 230* .

Wang A, et al. (2003) An industrial strength audio search algorithm. *Ismir*, volume 2003, 7–13.

Xiao FF (2018) Experiments with the shazam music identification algorithm. *diss.* .