Automated Detection of Converted Voices Generated by Sprocket

Vivek Sharath and Alex Liu
Applied Research Laboratories
The University of Texas at Austin

1 Abstract

Voice Conversion is the transformation of one speakers voice (the source) to sound like another speakers voice (the target) [1]. As voice conversion tools improve, generated and authentic voice clips become increasingly indiscernible to the human ear. This is where speaker recognition and machine learning come into the picture. This paper will explain the methodology I used to effectively distinguish between real and converted voice clips. Briefly put, this process includes using sprocket (an open source voice conversion tool) to create synthetic voice clips to go along with the original authentic audio clips. From there, the MFCC (mel-frequency cepstral coefficients), is extracted from each clip. Finally, these extracted features (which is in the form of a matrix) will be used to train a logistic regression model that aims to differentiate between generated voice clips and authentic voice clips. The models performance will serve as an indicator of MFCC's ability to distinguish authentic and converted voices.

2 Introduction

A group of astute engineers have created an online voice conversion application known as Lyrebird. This product takes in merely one minute of your own voice as input, and generates a synthetic version of your voice that can say anything as output. One of Lyrebirds most recent works includes teaming up with the ALS association to give a voice to those who are slowly losing the ability to speak. As impressive as these feats are, we must also use these breakthroughs and advances as markers for caution. If this type of technology falls in the wrong hands, the dissemination of spurious information becomes much more conducive. For example, converted voices that contain offensive material has deleterious ramifications for the original source speaker. In any case, seemingly genuine audio has the potential to wreak havoc for anyone involved. This paper serves as a rudimentary approach to detecting artificial voices which can be easily adopted and implemented.

3 Literature Review

Current research efforts have identified various methods of speaker recognition which have been quite successful in extracting unique characteristics in voice. The Automatic Speaker Verification Spoofing (ASV) and Countermeasures Challenge of 2015 and 2017 have taken great strides in distinguishing between an authentic voice and a spoofed voice. The competition in 2017 featured replay as the spoofed voices and does not pertain to the topics discussed in this paper [2]. The competition in 2015 had various types of spoofed voices, including voice conversion [3]. Results from the 2015 ASV challenge have shown us which detection methods work in certain situations. For example, spoofed voices that were synthesized using the HMM (Hidden Markov Model) were detected most efficiently by MCEP (Mel Cepstral Envelope) [4]. Another detection method that a few of the competitors used was MFCC. MFCC has been used in the past for speaker recognition and gave adequate results for team CRIM in the competition [5]. As mentioned before, the 2015 ASV challenge used various types of spoofed voices, including voice conversion. However, the algorithms use to generate voice converted clips were discrete in nature [3]. For example, some spoofed clips were purely made via GMM, some were generated from HMM, etc. Furthermore, the voice spoofing methods used in the competition were high level and not easily accessible to the general public [3]. Most spoofed voices that have circulated in the media will come from more accessible voice spoofing tools [3]. This paper aims to mend these gaps by using a voice conversion tool that is not only easily accessible to the general public, but also uses multiple algorithms at once which is something the 2015 ASV challenge did not feature.

4 Voice Conversion

In this section, I will go over the process used to create converted voices to analyze.

4.1 Sprocket

Sprocket is an open source voice conversion tool that allows for users to convert one speakers voice (the source) to another speakers voice (the target) [6]. The creator of Sprocket, Kazuhiro Kobayashi, wanted to develop a tool that allowed users to easily test voice conversion from the comfort of their own computer. Sprocket uses a vocoder-free framework based on differential GMM and F0 transformation [6]. This is an example of a voice conversion process that incorporates two different voice conversion algorithms which was non-existent in the ASV 2015 spoof challenge. Furthermore, Sprocket is one of the better voice conversion tools out there as it was used as the baseline system for the Voice Conversion Challenge of 2018 [6].

4.2 How Sprocket Works

Sprocket requires parallel data. In other words, the source utterance must be the same as the target utterance. For example, consider two folders of parallel data with 10 WAV files each. Out of the two folders, one is the source and the other is the target. Sprocket will take in the WAV files in both the source and target folders and output a new folder that contains 10 new WAV files which are the converted voices. To see in depth how this process works, sprocket has a github page that details the various steps that go into the voice conversion process. [7].

4.3 Data

The data used in this research project comes from the Voice Conversion Challenge of 2016. The reason why I used this data is because it was freely available, professionally recorded, used in previous Voice Conversion Competitions, and consists of parallel data of both the source and target. There were a total of 10 speakers and thus, 10 folders holding the recordings for the respective speakers. Due to their parallel nature, each folder contains the same 216 utterances from the book The Call of the Wild by Jack London. There were a total of 5 source folders (3 female and 2 male) and 5 target folders (2 female and 3 male). These 2160 WAV files are the authentic recordings. When it came to creating the converted voices, I decided to do two of every gender to gender combination. In other words, there were two source male to target male conversions, two source male to target female conversion, two source female to target female conversions, and two source female to target male conversions. After running sprocket on these combinations, there were 8 new folders (216 WAV files each) that contained the converted voice recordings. In total, the dataset comprised of 18 folders that each had 216 WAV files of the same utterances. 10 of the folders contain authentic voice clips and 8 folders contain converted voice clips which yields 2160 authentic WAV files and 1728 generated WAV files.

5 Detection

In this section, I will go over the process used to differentiate between real and generated audio.

5.1 Initial Analysis

The first approach used to analyze the sound files of the real and converted voices was qualitative in nature. Using a matpllotlib and scipy [8] [9], I was able to visualize the signal of every WAV file. From this analysis, it seemed quite apparent that the lows for each of the signals differed ever so slightly. When it came to lows in the signals, authentic voice clips (both target and source) had a greater range of wave amplitudes than that of generated voice clips. This idea is illustrated in the figure below. This trend recurred throughout the corpora.

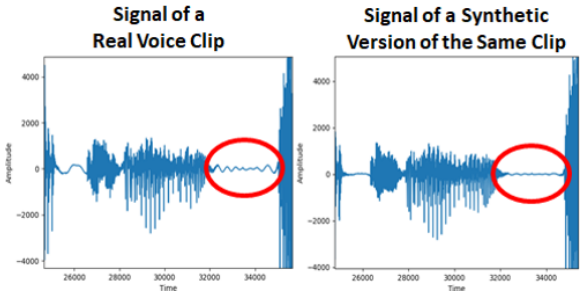

Figure 1: Signals of authentic and converted voice clip

5.2 What is MFCC

To better assess this discrepancy, I decided to extract the MFCC from each of the WAV files. As described from Wikipedia, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. [10]. In other words, extracting the MFCC from a signal gives us a numerical

representation of the sound. This type of representation would allow us to continue analysis and aid in detection of real and converted voices. MFCC extraction can be broken up into 5 stages. The first stage is to pass the signal through a Fourier Transformation (FT). FT breaks down a signal into several, smaller frequencies. Each of these frequency can be expressed quantitatively by its signal energy, or Hz. We then map each of these energies to the Mel filter bank. The filterbank ranges from 0 Hz to 8000 Hz and is divided into N bins. The typical amount of bins wavers around N = 40. (The algorithm used to extract MFCC in this project uses 52 bins). Once each of the signals are placed in its respective bin, each bin is summed based on the Hz of each signal. After that these N values are logged and then cosine transformed. The resulting N numbers are the MFCC of a signal.

5.3 Why did we pick MFCC

As mentioned earlier, I noticed a discrepancy in amplitude of the signals lows between real and converted voices. This discrepancy is taken into account by MFCC because MFCC analyzes the amplitudes of every frequency that makes up a signal. Furthermore, when a signal goes through a conversion, as in voice conversion, the new signal loses some of the phase information that was present in the original signal [5]. Although MFCC is a feature that is built on analyzing amplitudes, it still has the ability to perform just as well as feature extractions that take into account of the phase information loss [5]. Furthermore, extracting the MFCC has been a common approach among participants of the 2015 ASV Challenge and produced noteworthy results [5]. These are the three reasons why I used MFCC to analyze the difference between real and sprocket converted voices.

5.4 MFCC Extraction Process

There were a total of 18 folders (8 folders containing converted voices and 10 folders containing authentic voices). As mentioned before, there are 5 male genuine voice folders and 5 female genuine voice folders. The 8 folders of converted voices comprise of 2 male to male conversion, 2 female to female conversion, 2 female to male conversion, and 2 male to female conversion. Each of the 18 folders contain 216 WAV files of the same spoken utterances. This totals to 3888 WAV files (1728 Generated and 2160 Authentic). I extracted the MFCC for each of the 3888 WAV files by using the python library speech features [11]. The algorithm takes in a WAV file as input, and outputs 52 features (the MFCC). These 52 features go along with a class label of real or fake, based on which folder the WAV file came from. In total, each WAV file has 53 features: 52 MFCC features and 1 class label (real or fake). All 53 features for each of the 3888 WAV files are written into a CSV that will be used to train a machine learning model.

5.5 MFCC Visualization

With a total of 53 features, visualizing our data can be tricky. This is where we can use a dimensional reduction approach called t-SNE to get an idea what our data looks like [12]. Using t-SNE open-source code by Siraj Raval [13], I was able to get a sense of the data by reducing it to two dimensions. From this visualization, it was evident that there was some congregation of converted voices and authentic voices based on the given MFCC values. We will continue our investigation of MFCC in converted/authentic voices in the following paragraphs.

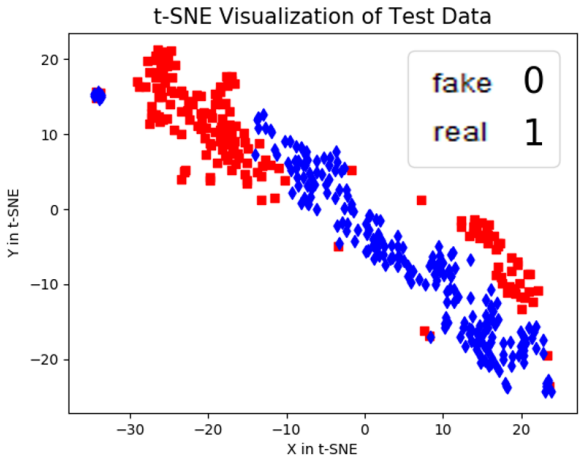

Figure 2: t-SNE of the created 53 feature data set

5.6 Logistic Regression

Our data has 53 features: 52 MFCC features and one class label (real or fake). We are curious to see how well the 52 MFCC features are in predicting the class label of real or fake. Due to the binary state of the dependent variable (class label), I decided to use logistic regression. The platform used to conduct the Logistic Regression was in R [14]. Since we had 1728 converted voices and 2160 authentic voices, I decided to keep this ratio of converted to authentic voices the same in both training and test data sets. I used 80% of the 3888 files as training data and the remaining 20% as test data.

5.7 Results

Once the model was created using the 53 features, I did some evaluation on the model performance. The

first model analysis I undertook was a density plot that seemed to separate fake and converted voices well on the training data. The next form of analysis were plotting the ROC curves for both the training data and test data. The results showed that both training and test data had about .98 area under the curve. The final form of analysis was assessing the models performance on just the test data. The model had an accuracy of 81.3% while precision was 99.5% and recall was 75%. These values indicate that the MFCC was indeed a powerful tool in differentiating between converted voices and authentic voices.

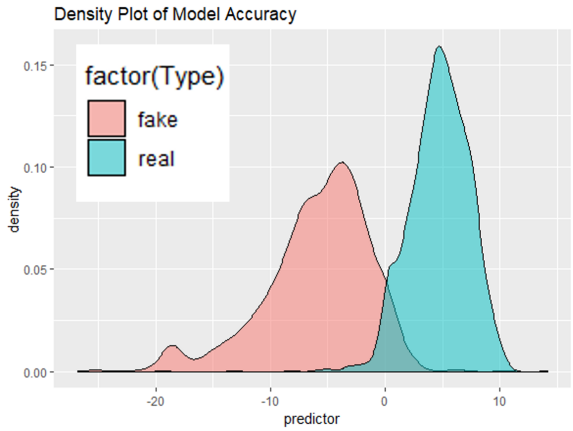

Figure 3: Density Plot of Class Seperation for Training Data

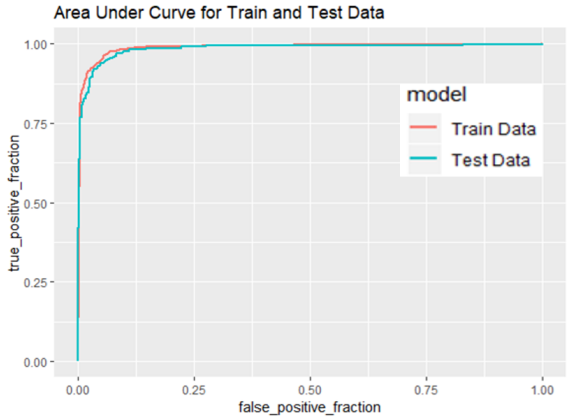

Figure 4: ROC Curves of Train and Test Data

Confusion Matrix

	fake <int>	real <int>
0	202	2
1	143	430

```
"Accuracy: 81.3%"
"Precision: 99.5%"
"Recall: 75%"
"f1: 85.6%"
```

Figure 5: Confusion Matrix of Training Data

6 Conclusion

The first form of analysis, as mentioned before, was qualitative in nature. The signals of real voice clips and converted voice clips differed in the lows as real voice clips seemed to have slightly higher amplitudes than converted voice clips, on average. Using MFCC, we were able to assess this discrepancy by decomposing each signal and representing it as a matrix of numbers. While this form of analysis gave us good results in differentiating between real and converted voice clips, there are a few things that need to be addressed. We had an AUC of .98 and precision of 99%. It seems that the model did extremely well in predicting clips that were authentic. On the other hand, with a recall of 75%, our model could have done better in predicting converted voice clips. In future experiments, I plan to add more features, along with MFCC to see if that will have an effect on aiding our model when faced with converted voice clip. It is important to note that all the converted voice data came from one tool: Sprocket. In the future, I will extract the signals of converted voices generated by other tools to assess if the lows in amplitudes are comparable to that of Sprocket converted voice signals. Future experiments will also use more easily accessible voice conversion tools similar to sprocket for a data set that is diverse in types of converted voices.

References

[1] Yannis Stylianou and Olivier Cappe. A system for voice conversion based on probabilistic classification and a harmonic plus noise model. In *Acoustics, Speech and Signal Processing, 1998.*

Proceedings of the 1998 IEEE International Conference on, volume 1, pages 281–284. IEEE, 1998.

[2] Tomi Kinnunen, Nicholas Evans, Junichi Yamagishi, Kong Aik Lee, Md Sahidullah, Massimiliano Todisco, and Héctor Delgado. Asvspoof 2017: Automatic speaker verification spoofing and countermeasures challenge evaluation plan. *Training*, 10(1508):1508, 2017.

[3] Zhizheng Wu, Tomi Kinnunen, Nicholas Evans, Junichi Yamagishi, Cemal Hanilçi, Md Sahidullah, and Aleksandr Sizov. Asvspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[4] Nanxin Chen, Yanmin Qian, Heinrich Dinkel, Bo Chen, and Kai Yu. Robust deep feature for spoofing detectionthe sjtu system for asvspoof 2015 challenge. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[5] Md Jahangir Alam, Patrick Kenny, Gautam Bhattacharya, and Themos Stafylakis. Development of crim system for the automatic speaker verification spoofing and countermeasures challenge 2015. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[6] Kazuhiro Kobayashi and Tomoki Toda. sprocket: Open-source voice conversion software. In *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, pages 203–210, 2018.

[7] Kazuhiro KOBAYASHI. *Sprocket*, 2018 (accessed June 20, 2018). https://github.com/k2kobayashi/sprocket.

[8] *Matplotlib*. Available at `https://matplotlib.org/`.

[9] *Scipy*. Available at `https://www.scipy.org/`.

[10] *MFCC*, (accessed July 10, 2018). `https://en.wikipedia.org/wiki/Mel-frequency_cepstrum/`.

[11] James Lyons. *Speech Features*, (accessed July 10, 2018). `https://github.com/jameslyons/python_speech_features/`.

[12] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[13] Siraj Raval. *How to Best Visualize a Dataset Easily*, 2016 (accessed June 20, 2018). `https://github.com/llSourcell/visualize_dataset_demo/`.

[14] *R*. https://www.r-project.org/.