# Comparing Techniques for Audio Style Transfer For Converting One Voice to Another

*Jesse Galef uni: jbg2160*

Columbia University

**ABSTRACT**

This project aimed to explore audio style transfer in the domain of speech, comparing how well three different feature extraction methods represent a speaker's style: AutoRegressive modeling coefficients of the vocal tract, Speaker identification faux i-Vectors, and a wide side of random convolutional filters.

Using these three different feature sets to model a speaker's style profile, a content audio was modified through a series of convolutional and residual layers to more closely match the speaking style of the late Alan Rickman.

While the neural nets proved capable of converting a given vocal tract profile to match that of the target's, the modifications proved insufficient to sound as though the resynthesized audio came from a different speaker. By only adjusting the vocal tract information – leaving the vocal cord excitation signal's pitch and amplitude unchanged – the alterations left out many important subjective elements of Alan Rickman's voice. Future work could include both elements of the source-filter model in their style transfer and would likely produce higher quality results.

In a desire to learn more about signal processing, many components are coded from scratch rather than using pre-built functions. The TensorFlow Signal module in particular would have made many things possible, but would have masked important concepts.

The project was written and run in Python 3 in Google Colabaratory, which allowed the use of a GPU to train the neural nets and greatly reduce runtime. The notebook can be viewed or run by others, and a link is provided in section 6.

## 1. INTRODUCTION

Style transfer has been the focus of many papers and commercial applications in the last few years, sparked by the 2015 paper by Gatys et al. 'A Neural Algorithm of Artistic Style' [1]. In it, they demonstrated a way to separately encode the content and style of an image, enabling the them to create new images that match the content of one sample and the style of another.

Many techniques for image processing have been shown to work for audio processing by treating the audio spectrogram as a 2-dimensional image where one dimension is time. However, taking the style-transfer process and applying it to audio requires additional thought and work.

With images, the question of content vs style is generally agreed upon: content tends to refer to the edges and geometric/structural relations in the scene, while style tends to be the space-invariant relations within a patch and the overall color palette of the image. Making the same division for audio is less established and depends on context and purpose. The content of audio could refer to the phonemes in speech, the notes' pitch in a musical score, or the rhythms of a percussive track. However, pitch and rhythm might instead be considered style when in the context of speech, along with intonation, speed, emotional tenor, and other traits.

This project explored style transfer specifically for speech, aiming to generate a speech clip with the content of one audio sample in the style of a different speaker. It compared three types of features to judge which best convey speaker style for transfer.

## 2. PREVIOUS WORK

The style transfer architecture designed by Gatys et al. iteratively improved an initial image of random static to minimize a balance of two loss functions: The content feature difference from the initial input, and the stylistic difference from the style input.

Others including Johnson et al. [2] adapted this idea to be more general by using a series of content image as the initial samples, and using residual layers so the network was learning how to modify this input rather than generate a new image from static. A neural net trained this way could convert a new image to the target style with a single forward pass rather than requiring a new optimization, producing results orders of magnitude more quickly.

Grinstein et al. [3], who were exploring audio style transfer, found this technique of using residual layers to modify the content sample enabled them to remove the content loss term altogether. Because the generated sample was based on the content sample, the local optimum their neural net produced was close in content. They tested different feature extraction methods to represent audio style. They were unimpressed by the results of extracting features with the image-trained VGG net or even the audio-trained WaveNet.

However, they did find two techniques that they considered worked well for audio texture transfer. They report good results using hand-designed statistics based on McDermott and Simoncelli's analysis of the auditory system [4]. Following the results described by Ulyanov and Lebedev in a 2016 blog post [5], Grinstein et al. also tried using a wide, untrained layer of 4096 random convolutional filters as style features and found it surprisingly promising.

Perez et al. [6] adapted Gatys' original structure to style transfer in prosaic speech by replacing the pre-trained VGGNet with an autoencoder with several layers of dilated and transposed convolutions. They found that the generated clips had some of the target style in a universal sense, but the emotional content was not conveyed.

## 3.0: DATA SOURCE

Content audio was from a selected voice reading audiobooks, gathered from the LibriSpeech database. Style audio was drawn from 150 seconds of Alan Rickman reading The Return of the Native by Thomas Hardy obtained from Audible.com.

A link to these audio files is provided in section 6.

## 3.1 METHOD: SIGNAL PREPARATION

Audio from a speaker in the LibriSpeech dataset acted as the content audio, with 1,000 random 5-second segments taken from the 138 .flac files. A 150-second recording from Alan Rickman reading The Return of the Native by Thomas Hardy -- obtained from Audible.com -- served as the style audio. In training, random 5-second clips from this recording were used in batches.

Each signal went through a pre-emphasis filter which prevented the system from focusing too much on the higher frequencies, and was broken into overlapping frames of 512 sample points – roughly .03 seconds. Each frame was then windowed with a Hanning window to avoid artifacts which could be caused by sharp cutoffs.

There were many different potential representations of the audio signal to work with, including Mel Frequency Cepstral Coefficients (MFCCs), Spectrograms, and raw signal. There were drawbacks to each of these, however: MFCCs and Spectrograms discard phase information of a signal to operate in the frequency domain, rendering future reconstruction of an audio signal difficult. Working with the signal directly would make modification and reconstruction simple, but it would be difficult to extract desired features from a modified signal as part of the neural network architecture.

This project chose to represent the signal with an AutoRegressive model, calculating 12 coefficients that predict the next point in a signal from a linear combination of the previous points. These coefficients, known as Linear Predictive Coefficients (LPCs), were be obtained by applying the recursive Levinson-Durbin algorithm on the auto-correlation coefficients within the signal.

AutoRegressive representations of audio like this separate the source and filter of the signal. It models the signal as a series of unvoiced white noise or voiced noise generated from vibrations in the vocal cords, passed through the throat and vocal tract, which filters the sound. The 12 LPCs create an all-pole filter to model the vocal tract. Passing the signal back through its inverse (as an all-zero filter) provides the excitation, or the source sound which would generate the initial signal through this filter. This excitation signal is stored to be used as part of the resynthesizing process later.

This project focused solely on modifying the **filter** element of the source-filter model, trying to match a speaker's style by identifying patterns in their vocal tract movements. Leaving the source excitation largely untouched meant the number of features involved was dramatically reduced and simplified – but the decision clearly affected the final results.

One fortunate byproduct of using the Levinson-Durbin recursion to calculate LPCs was the Partial Correlation Coefficients, or Reflection Coefficients. These convey the non-redundant information and are considered to be more stable than LPCs. Reflection coefficients have a one-to-one correspondence with LPCs, meaning that we can convert back after passing through the neural nets. Counting the first and second derivatives (deltas and delta-deltas) of these 12 reflection coefficients, each frame contained 36 features. These were the input passed to each of the three neural networks.

## 3.2 METHOD: NEURAL NETWORK ARCHITECTURE

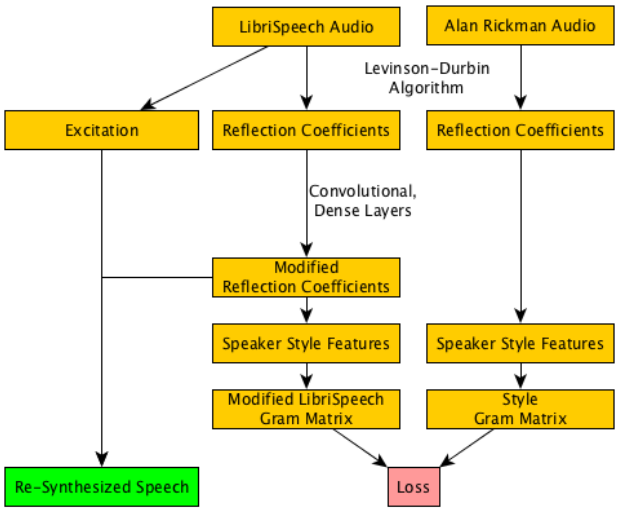


**Figure 1: Data Pipeline to Transfer Style. Three different techniques for extracting 'Speaker Style Features' were tested and compared.**

Each of the three neural networks tested followed a similar structure built in a combination of Keras and Tensorflow (Figure 1):

1. Take a batch of LibriSpeech reflection coefficients as input. Each 5-second clip contained 310 overlapping frames with 12 reflection coefficients, so a batch of 32 signals was of shape [32 x 310 x 12].

2. Append the first and second derivatives – the deltas and delta-deltas – padding with zeros for the first frame.

3. Pass these features through a series of five 1-Dimensional Convolutional layers with non-linear activations and a growing number of filters between 32 and 128.

4. Apply a Time-Distributed Dense layer which outputs the amount to add or subtract from each of the 12 reflection coefficients per frame. This gives us the modified reflection coefficients we later use in re-synthesizing audio.

5. Append the deltas and delta-deltas for these new, modified reflection coefficients.

6. Extract features from these new, modified features in one of three ways being tested.

7. Calculate the Gram Matrix of these features, the dot products of each combination of features across all frames. A set of 310 frames each with 36 features would produce a 36 x 36 gram matrix indicating how much each pair of features is co-present in the signal.

8. As a second input, take the reflection coefficients from the style signal of Alan Rickman and append the deltas and delta-deltas just as with the LibriSpeech signal. Extract the same features as in step 6. Calculate the gram matrix of the style signal.

9. Fit the model weights to minimize the mean squared error between the modified coefficients gram matrix and the style gram matrix.

If the features extracted in step 6 accurately capture what we consider the "style" of a speaker, then the neural network should learn to modify the content audio to sound more like it was being spoken by Alan Rickman.

The first neural net calculated the gram matrix from the reflection coefficients and their delta/delta-deltas alone. Since the reflection coefficients are an attempt to form a numeric representation of the vocal tract, common patterns and correlations could describe a person's speaking style.

The second neural net took the reflection coefficients and their delta/delta-deltas and passed them through a static, randomly-initialized convolutional layer with 1024 filters. The idea, described in [5], is to bypass the process of learning content-specific features, and instead generate a large quantity of random features. Most of the thousand-plus features will be irrelevant, capturing nothing significant from the reflection coefficients. These filters will likely be uncorrelated, and the gram matrix will give them very little weight.

Some randomly-generated features, however, will track important characteristics of the data and will demonstrate patterns across time frames. A gram matrix will reflect this, and the neural net can learn to match these important features while ignoring the unimportant ones.

Although Ulyanov and Lebedev used 4096 features in [5], this project included only 1024 to speed up processing time. The filter weights are fixed and unchanging, so processing the gradients to update them was not an issue. However, with 1024 filters that span 5 time frames each with 36 features, across 310 frames for each of the 32 input signals, the number of calculations required per batch quickly surpassed a billion. Even with only 1024 filters, training this version of the modification net was significantly slower than the other two.

The third and final neural net was modeled after the original Gatys et al. style transfer project, which used mid-level filters from a pre-trained image classification model to populate its gram matrix. The idea was that an image

classification model was learning the space-invariant patterns which were most relevant in understanding what an object was.

The initial proposal for this project intended to use i-Vectors for this purpose. Designed to aid in speaker identification, i-Vectors are a series of values indicating how a particular utterance deviated from the 'average' audio signal which would be produced with those phonemes. It relies on pre-training a Universal Background Model (UBM) Gaussian Mixture Model, using audio aligned with transcript data to learn how the average man/woman sounds in producing particular phonemes. Since these i-Vectors indicate the elements that distinguish one speaker from another, they serve a similar purpose to the mid-level features used in image style transfer.

Unfortunately, in choosing to do this project in Python and without the Kaldi toolkit, it became infeasible to train a UBM and extract i-Vectors from the training audio samples, let alone from the modified signals midway through the neural network. Instead, a similar purpose was served by a basic four-layer speaker recognition net trained to classify an utterance as either produced by Alan Rickman or the LibriSpeech speaker. This was less specifically-targeted at representing the principles which distinguish speakers, but served a similar purpose.

Once the intermediary model was trained, its layers were fixed and used in the third model to extract 64 features from sets of reflection coefficients and their delta/delta-deltas.

Although the neural networks technically took two signals as input and output the gram matrix difference – what the model was optimizing for – it was easy with Keras to use the trained layers to provide a single modified sample from a single set of reflection coefficients.

### 3.3 METHOD: RESYNTHESIS

Each neural network modified the given set of reflection coefficients to better match the style of reflection coefficients in samples of Alan Rickman audio. Reflection coefficients with a magnitude over 1 are capable of creating an unstable filter, so after modification each set of reflection coefficients was normalized to have magnitudes between -1 and 1. Once these newly generated sets of coefficients were normalized, they needed to be converted back to Linear Predictive Coefficients and re-combined with their excitation signals to generate the final audio.

Since this project focused on altering the filter component of the source-filter model, the source – or excitation signal – was unchanged. Passing it through the new set of filters frame-by-frame, we obtained the signal which would be generated by the same unvoiced and voiced vocal cord noise going through a new set of vocal tracts.

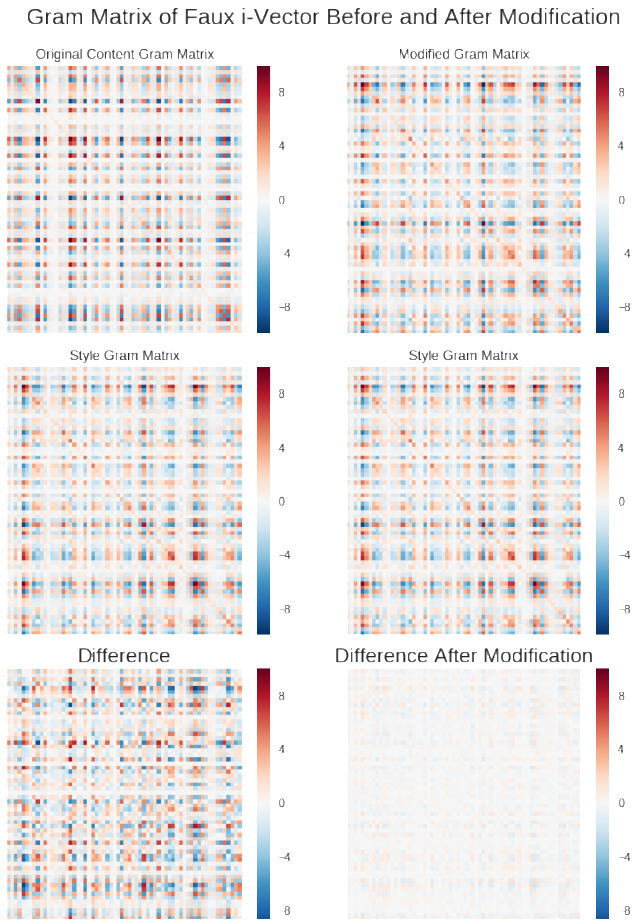


**Figure 2: Gram Matrices of faux i-Vectors before (left column) and after modifying (right column) the reflection coefficients.**

These resynthesized samples came from overlapping time frames, so in order to recombine them without sudden changes a Hanning window was applied again. After recombining each windowed, resynthesized frame into the full-length signal, it was run through the inverse pre-emphasis filter and the final product was obtained.

### 4.1 RESULTS: VOCAL TRACT

The neural networks proved capable of modifying the given LibriSpeech reflection coefficients to match the target style. The residual convolutional net architectures were able to match the chosen features of the vocal tract representation, as shown for an example signal in Figure 2. Before modification, the sample had a very different faux i-Vector profile compared to Alan Rickman. After modification, they were almost identical.

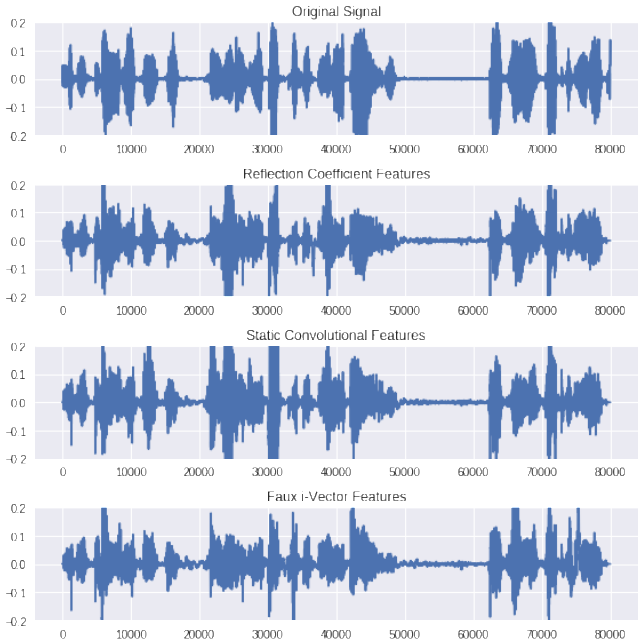


**Figure 3: Audio signal from the original sample and resynthesized after modification through each of the three neural nets**

Recreating the chosen feature profile was the first step, but the question was whether matching these features corresponded with matching the target 'style', a subjective question judged by listening to the resynthesized audio.

Unfortunately, while the resynthesized audio segments were all intelligible, they contained a noticeable buzz and none could be said to sound especially like Alan Rickman.

Subjectively, the speaking style in the new audio was slower and slightly drawn-out, indicating that it may have been moving in the correct direction – particularly the audio generated by matching faux i-Vector gram matrices. However, the changes were not dramatic enough to definitively state that one set of features was bet

### 4.2 RESULTS: VOCAL CORD

The fundamental problem with this project's approach is that the excitation frequency and amplitude were no longer aligned with the filter in the appropriate way. Even if the new filter modeled a vocal tract moving more slowly – perhaps because Alan Rickman speaks at a lower word-per-minute rate – the excitation signal did not change along with it. The result was many time frames of incorrect unvoiced or voiced source noise.

Additionally, a primary identifying feature of Alan Rickman's voice is the bass-y, low-frequency pitch he generates as he speaks. As the vocal cords from the LibriSpeech speaker remained unchanged, they failed to produce the lower frequencies and resonances the project was intended to recreate.

In an attempt to better align the excitation signal to the new filter coefficients, a dynamic time warping algorithm was used to compare the old reflection coefficients over time to the new ones. This would address the problem of a voiced/unvoiced source excitation alignment, but not the pitch issue, and was ultimately not used in the final project.

## 5: FUTURE WORK

In order to achieve better results in speech style transfer, any model would need to incorporate both the source and filter aspect of audio generation. The advantage of focusing solely on the vocal tract is that an autoregressive model allows a compact representation of the information – this is why Linear Predictive Coefficients and their corollaries were designed for audio transmission. But even during transmission, the LPCs were not enough – the excitation signal needed to be sent as well.

One particularly promising direction would be to use the TensorFlow Signal module, which is capable of taking a long series of points as input and perform signal processing functions on it during the neural net training process with a GPU or the Google-specific TPU processor.

This would allow a neural net to operate directly on the signal, extracting any desired features for a gram matrix and adjusting the signal itself without separating it into the source and filter components. In doing so, a neural net could avoid the problems of aligning excitation to filter as well as the question of how to represent the excitation signal for processing.

Grinstein et al. [3] mention representing the audio in a number of ways, including the spectrogram and pure signal, with promising results.

## 6: LINKS

The Python notebook with this project's code can be found (and run) at:
https://colab.research.google.com/drive/157CErdbKP68nU86g5K645Czn5irG8gKL

Audio files used in training can be found at:
https://drive.google.com/drive/folders/10-PJABVj6p69yGLzD99SnmzqctfQFPVT

Five examples of audio – Original and resynthesized from each of the three neural nets – can be found at:
https://drive.google.com/open?id=1b2rfXjqwFue5DQUeTtNzTrM8iS1JkQjj

## REFERENCES

[1] Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style. arXiv:1508.06576 (2015)
[2] Justin Johnson, Alexandre Alahi, Li Fei-Fei: Perceptual Losses for Real-Time Style Transfer and Super-Resolution. arXiv:1603.08155v1 (2016)
[3] Eric Grinstein, Ngoc Q. K. Duong, Alexey Ozerov and Patrick Perez: Audio Style Transfer, arXiv:1710.11385v2 [cs.SD] (2018)
[4] Josh H. McDermott and Eero P. Simoncelli: Sound texture perception via statistics of the auditory periphery: Evidence from sound synthesis. Neuron, vol. 71, no. 5, pp. 926 – 940, 2011
[5] Dmitry Ulyanov and Vadim Lebedev: Audio texture synthesis and style transfer. tinyurl.com/y844x8qt. (2016)
[6] Anthony Perez, Chris Proctor, and Archa Jain: Style transfer for prosodic speech. Tech. Rep., Stanford University, 2017.