SPEECH STYLE TRANSFER

Wei Luo

ABSTRACT

This is our project for Fund Speech Recognition at Columbia University. In this project, we build a speech style transfer model that learns the transformation from one speech to another. Specifically, we use a encoder-decoder network architecture to learn transformation from male speech to female speech using CELP features.

***Key Words*—** Style Transfer, Neural Networks, Seq2Seq

1. INTRODUCTION

Style transfer has been a topic of great interest for many years. From as early as the 1980s, researchers in classical signal processing have been using simple edge detectors to apply style effects to images [1]. With respect to images, style transfer means moving the style from one image to another, thereby creating new content but retaining the style. With respect to audio, style transfer can open up exciting opportunities and fields such as automatic music generation, sound design, and reliving the performances that historical musicians leave behind. Similarly, with respect to speech, style transfer means transferring how text is spoken from one content to another. For example, with style transfer, we can extract "style" from reader of *Snow White*, and synthesize speech of the same person reading *Cinderella*. Then we can theoretically obtain sound of anyone we want reading any text. Also, such research opens up opportunities in automatic dubbing in the movie production industry – the actors can put their voices into the movie later on by reading any text and letting the computer perform the style transfer.

With the resurgence of deep learning and promising results from neural style transfer in computer vision, researchers in speech and audio have been inspired to use similar neural network models on spectrograms to perform speech style transfer [2]. However, such approach does not naturally arise from foundations of speech recognition, and there are techniques more naive to the field that can solve the problem. For example, linear predictive coding (LPC) coefficients can approximate signals in compressed form. In our project, our focus is to use features such as LPC that are native to speech processing to build a speech style transfer model.

2. RELATED WORK

For audio, there are global homogeneity features that can describe the sound texture, and the spectral shapes can be indicative of sound color [3]. These in combination define a sound texture. Then one way of performing style transfer is to synthesize sound color with new sound texture [4]. Another related work for mixing two sounds is to extract short term spectrum from one sound, short term spectral envelope from another, and multiply them together [5].

With the resurgence of deep networks, there were promising results in style transfer in computer vision [6]. In his work, Gatys et al. [6] starts with a style image and a content image as input to features of a convolutional neural network (CNN). Starting from an output image of random noise, the model is tuned with respect to a style and a content loss to eventually generate a combined image. This combined image holds content represented by the content image while retaining style from the style image.

In audio style transfer, similar frameworks have been adapted. In their work, Ulyanov and Lebedev [7] start with a style audio and a content audio. These audio are transformed into 2D spectrograms with phase discarded, and viewed as 2D images. Then the researchers use a shallow network with 4096 filters to start from random noise and synthesize incrementally better combined spectrograms with respect to content and style. Similarly, the work of [3] view 2D spectrograms obtained from short time Fourier transform (STFT) as images, and apply a convolutional model. Instead of using both content and style loss, they simplify the loss to be only related to style. To retain the style, instead of starting with random noise, they start with the content audio and incrementally apply style to the signal.

Although previous work shows that applying techniques from computer vision can achieve proof-of-concept results, there is still much room for improvement. For example, the approach does not naturally arise from foundations of speech recognition, and there are techniques more naive to the field that can solve the problem. For example, linear predictive coding (LPC) coefficients can approximate signals in compressed form. In our project, our focus is to use features such as LPC that are native to speech processing to build a speech style transfer model. In particular, we will focus on learning to transfer style from male voices to female voices. Specifically, we are interested in the following problem: *Given audio*

of female speaker F reading text T_1, can we synthesize audio of speaker F reading a new text T_2? To do this, we will rely on an anchor male voice M. If we have audio of F reading T_1 and M reading T_1, we can learn the mapping from M to F, let M read new text T_2, and apply our learned mapping to obtain audio of F reading T_2.

3. METHOD

Figure 1 illustrates our general setup. Let $a(M, T_1)$ denote audio of male speaker M reading text T_1. Our task is to learn the mapping $M \rightarrow F$ from pairs of audio $\{a(M, T_i), a(F, T_i)\}_i$, where M is the anchor male voice, and F is the target female voice. Instead of directly learning mapping from audio to audio, we extract features related to code excited linear predicting coding (LPC) [8] from each audio. These features are used in telephone signals and can approximate the signal and be inverted back to the signal if needed [9]. Also, these features are time-related. Then we can obtain pairs of LPC features corresponding to pairs of speakers. We then learn the mapping from features for audio of speaker M to features for audio of speaker F using a encoder-decoder network [10], a general-purpose neural network architecture for machine translation. During training, we let the model learn the mapping, and during testing, given speaker M reading a new text, we apply the mapping to obtain features of audio for speaker M reading the new text. Then we use the coefficients to synthesize complete audio of speaker F reading the new text.

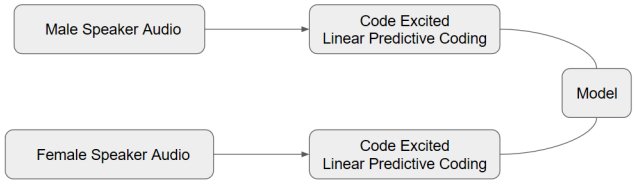


Fig. 1: General framework of our approach.

3.1. Signal Compression and Reconstruction

In this section, we discuss the method we use for transforming a speech signal into a compressed representation of LPC features and reconstructing the signal from the compression. Throughout this part, our notation is inspired by notes from Stanford University [11] and tutorial from Speex [12].

3.1.1. Linear Predictive Coding

Linear Predictive Coding (LPC) is a technique that is frequently used to compress speech signals in applications such as telephone signals. As shown in Figure 2, the core idea behind LPC is that a sound signal can be decomposed into two parts: an excitation or source, and a filter. Given an excitation, the filter can be applied to obtain the signal. Mathematically, the filtering action can be represented as

$$e(n) * h(n) = x(n).$$

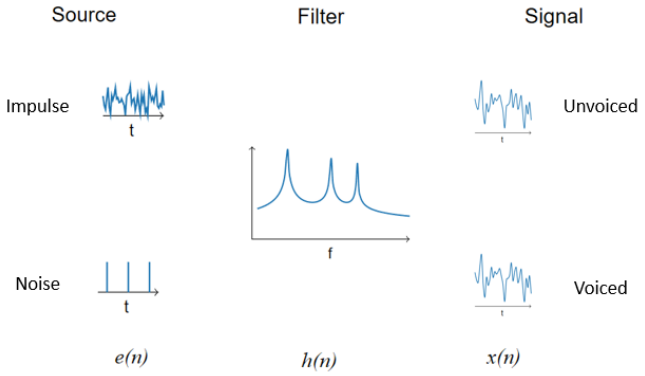


Fig. 2: The LPC filtering model.

In real-world applications, the challenge is that we do not have the excitation and the filter that can exactly reproduce the signal available to us. Thus given a signal to compress, we have to produce our best estimates of both the excitation and the filter.

For the excitation, if it is a random Gaussian noise, then non-vocal sound will be produced after filtering. On the other hand, an impulse train with period k such as $[1, 0, 0, ..., 0, 1, 0, 0, ...]$ can be filtered to obtain vocal signal. Note that because both the impulse train and the random white noise are spectrally flat, we can include the spectral information in our filter. We can perform pitch detection on the original signal to determine whether a segment is voiced or unvoiced, and select the corresponding excitation.

For the filter, we assume that it is a p-th order all-pole filter. Then we can formulate our problem as

$$\begin{aligned} X(z) = H(z)E(z) &= \frac{1}{1 - \sum_1^p a_k z^{-k}} E(z) \\ \implies X(z) - \sum_1^p a_k z^{-k} X(z) &= E(z) \\ \implies X(Z) = \sum_1^p a_k z^{-k} X(z) &+ E(z) \\ \implies x(n) = \sum_1^p a_k x(n-k) &+ e(n), \end{aligned}$$

where we call $a_{i i=1,...,p}$ the p-th order LPC coefficients.

From the above equations, we see that if we can ensure that $e(n)$ is close to impulse or white noise and if we can estimate the coefficients a_i, we can successfully reproduce $x(n)$. Thus we can formulate the problem as a least squares

problem, assuming we have N samples. Define the signal

$$x_i = [x(n-1-i), ..., x(n-p-i)]^T,$$

and let the coefficients be

$$a = [a_1, a_2, ..., a_p]^T.$$

Then we get the equations

$$\begin{aligned} x(n) &= x_0 \cdot a + e(n) \\ x(n+1) &= x_1 \cdot a + e(n+1) \\ &\vdots \\ x(n+N) &= x_N \cdot a + e(n+N). \end{aligned}$$

We can then combine variables and write this system as

$$b = A \cdot a + e,$$

where

$$\begin{aligned} b &= [x(n), x(n+1), ..., x(n+N)]^T, \\ A &= [x_0, x_1, ..., x_N]^T, \\ e &= [e(n), e(n+1), ..., e(n+N)]^T. \end{aligned}$$

After rearranging, we obtain

$$A \cdot a - b = e,$$

where e can be viewed as the residual and we can view the system as a least squares, where we minimize $||e(n)||^2$. This system has solution

$$a = A^+ b,$$

where A^+ is the pseudo-inverse of A.

Having obtained the coefficients, we can calculate the excitation, which will be ideally impulse train or white noise. Then, we can use pitch detection to determine whether the signal is vocal. These methods combined give us the ability to compress and re-synthesize a short signal. To use the method for a longer signal such as a spoken word, we can process the signal in small chunks by using a windowed version of the signal:

$$x_m(n) = w(n)x(n + mR),$$

where $w(n)$ is a window function such that

$$\sum_{m=-\infty}^{\infty} w(n)x(n + mR) = x(n)$$

and is constant zero outside the window of range N.

3.2. Code-Excited Linear Predictive Coding

In vanilla LPC, we start with either impulse train or white noise as the excitation. However, it has been shown that there is much room for improvement in terms of quality under this method. To resolve this, we use Code-Exicited Linear Predictive Coding (CELP), a more complex excitation, consisted of pitch prediction and signal taken from a codebook.

For pitch prediction, we use

$$p(n) = \beta e(n - T)$$

, where β is a pitch gain constant and $T >> N$. For the other signal c(n), we take from a codebook generated from vector quantization. Our new excitation can then be written as

$$e(n) = p(n) + c(n).$$

3.3. Sequence-To-Sequence Modeling

In the previous section, we discussed that during the LPC modeling process, we divide an utterance into several small chunks and perform LPC analysis on each chunk. This results in a sequence of LPC features for each utterance. Then our mapping from one speech style to another has been turned into a mapping from one sequence to another. In this section, we discuss seq2seq [10], a neural network architecture specially purposed for working with sequence translation.

The general architecture is shown in Figure 3. The network can be viewed as having an encoder and a decoder part, where in the encoder part, the network learns meaningful representations of the input sequence, and in the decoder part, the network uses the representations to make sequential predictions.

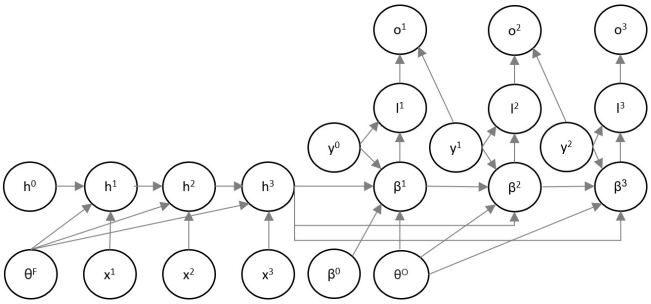


Fig. 3: The encoder-decoder seq2seq model.

Let $x = [x^1, x^2, x^3]$ denote an input sequence of length 3, where each x^1 is again a vector of features $x^1 = (x_1^1, x_2^1, ..., x_k^1)$. The network maintains encoder hidden states $h^0, h^1, ..., h^3$, with the intuition that each h^i would learn a representation of the sequence up to x^i. The set of parameters such as weight matrices are represented by θ^F. At the last hidden state h^3, information about the entire sequence x is represented.

In the decoder part of the network, there are states $\beta^0, \beta^1, ..., \beta^3$ that represent the network's sequence prediction. In training, each state β^i takes as input the previous ground truth sequence element y^{i-1}, the final encoder hidden state h_3, and parameters θ^O. The network uses these information to calculate current sequence element prediction and calculates loss. In testing, because there is no ground truth available, the network uses its own previous prediction as input to the current sequence element. For example, β^2 would have β^1 instead of y^1 as the previous element input.

4. DATA

Our data comes from the CMU Arctic Speech Synthesis Database [13]. It contains audio of US Speakers reading eBooks from Project Gutenberg, with around 1200 sentence utterances of male and female speakers each, totaling 2400 utterances and 1GB worth of space. Each utterance comes in .wav form and there is clean transcription corresponding to each utterance. The distributions have 16 KHz waveform and EGG signals.

5. EXPERIMENTS AND RESULTS

In this section, we discuss the experiments that we performed and their results. Figure 4 shows the end-to-end pipeline that we use. This includes first transforming the raw data into a form suitable for performing more precise LPC modeling, extracting features from the transformed data, training our neural network, and evaluating our results. Each step is discussed in detail in the subsections below.

Fig. 4: Our pipeline for performing and evaluating the experiments.

5.1. Data Transformation

As discussed in Section 3, because sentences are too long of an utterance for the LPC modeling, we split each sentence into words. Also, the data does not come with alignment of the audio and the transcription, so that we have to produce our own. We choose to not perform automatic alignment using Kaldi [14], because Kaldi uses machine learning models that are tolerate to alignment errors. Instead, we use a semi-manual approach using EasyAlign [15], a tool that allows users to visualize spectrograms, drag word-level alignments, and produce corresponding TextGrid files. The sentence-to-word process produces 9990 words, 2826 of which are unique, and the semi-manual process aligns the 9990 words to their audio. The word-level segments along with their aligned text take up 178M total.

5.2. Features

We extract features related to 10-th, 15-th, and 20-th order CELP modeling respectively, including LPC coefficients and codebook features. We will compare the performance of using different number of LPC coefficients in our experiments. We use frame length of 160 samples and divide each frame into 4 subframes for LPC modeling. This transforms each word audio into a sequence of subframe features. Because word audio have different lengths, we pad each sequence to the longest word audio's length by zeros. For order 10 CELP, this yields a tensor of pickled size 5M and shape $(9990, 51, 19)$ for male and female voices each, where 10117 is the number of sequences, 51 is the length of each sequence, and 19 is the number of features per point in a sequence. Similarly, the tensor for 15-th order CELP features has size 5M and shape $(9990, 51, 24)$ and the tensor for 20-th order CELP features has size 5M and shape $(9990, 51, 29)$.

5.3. Experiments

For our experiments, we start with data of shape $(9990, 51, k)$, where k is dependent upon the specific order of LPC we use. We divide the data into training set, validation set, and test set using an $6 - 2 - 2$ split. This gives us a training set of shape $(5994, 51, k)$, validation set of shape $(1998, 51, k)$, and test set of shape $(1998, 51, k)$.

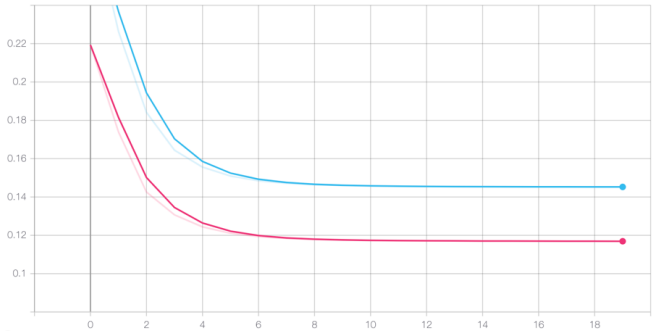
We use an encoder-decoder network architecture with two recurrent layers, where the first layer has 16 hidden units, and the second layer has 8 hidden units. In the end there is a dense layer with linear activation to broadcast the prediction output to the correct dimension. The network uses MSE loss and L2 regularization with $\lambda = 0.001$. We train the network using learning rate of 0.001 with learning rate decay of 0.0001. We use a batch size of 8 and train for 20 epochs. We trained networks with this setup for our experiments, in which we vary the order of LPC modeling in our features.

5.4. Results

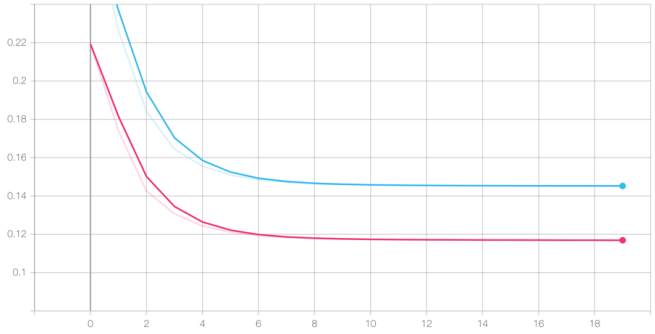
Figure 5 shows the training and validation loss over 20 epochs for our network that learns the features with order 10, 15, and 20 LPC modeling respectively. In the figure, blue denotes the training loss, and red denotes the validation loss [1].

As expected, both the training and validation losses decrease rapidly in the beginning, and eventually plateau due to our regularization. For each experiment, we choose the model that leads to the lowest validation loss. This means that we use model that achieves 0.1169 validation loss on the order 10 LPC features, the model that achieves 0.1085 validation loss on the order 15 LPC features, and the model that achieves 0.1082 validation loss on the order 20 LPC features.

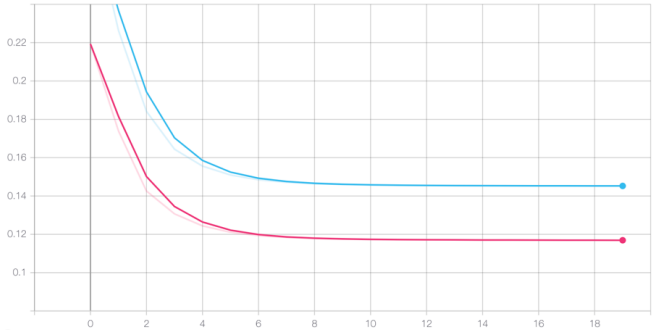
[1] Note that the fact that the validation loss is below the training loss is reasonable, since the recorded quantity is the total loss, and there are fewer samples in the validation set.



(a) Loss curves for network using order 10 LPC features.



(b) Loss curves for network using order 15 LPC features.



(c) Loss curves for network using order 20 LPC features.

Fig. 5: Loss curves over epochs for network using order 10, 15, and 20 LPC features. Blue indicates training loss, and red indicates validation loss.

We run each of these models on the test set, and the test set errors are shown in Table 1.

LPC Order	Validation Loss	Test Loss
10	0.1169	0.1538
15	0.1085	0.1017
20	0.1082	0.1015

Table 1: Validation and test set losses for network using various orders of LPC modeling.

Observe that regardless of the LPC order, the test loss is higher than the validation loss. This is expected, since we've tuned our model according to the validation set. Here the validation set and test set losses are comparable, because now they do have the same size. Also, note that as LPC order increases, both the validation and test losses decrease. This is reasonable, given that a higher order LPC modeling uses more descriptors for approximating the signal.

Although we have quantified our model's performance using loss measurements, it is not an objective assessment of the final synthesized speech with the transferred style. In fact, this is a difficult task, as assessing how well a style has transferred from one speech to another is mostly subjective. Therefore, we do not provide numerical measurements for this. Instead, we let the models produce LPC feature predictions for the style-transferred signal, and synthesize the signal with new style using the generated features. These new speech can be found at our soundcloud [2], where the first wav is a male speaker saying the word "forever", and the second wav is the synthesized female speech. Observe that there is resemblance of "forever" in the synthesized speech. Also, the voice sounds like a female speaker, and the word is spoken in a faster, more compressed manner. However, synthesized speech suffers from quality degradation, and it is not immediately clear what word the speaker is saying without a transcription known. These open up opportunity for future work.

6. FUTURE WORK

In our work, our contribution is as follows:

1. We develop a method of using neural networks to learn the transformation from LPC features of one style to LPC features of another style, thereby synthesizing speech with new style.
2. Our method is novel in that we do not rely on techniques from computer vision, and use features native to speech processing.
3. We train and test our models on data with order 10, 15, and 20 features, thereby obtaining a proof-of-concept model.

[2] https://soundcloud.com/user-941333416/sets/fund-speech-samples/s-v1NkG

Continuing from our work, there is much room for improvement and opportunities for future work. For example, in our work, we extracted order $10, 15, 20$ LPC features from the data. It would be worthwhile to investigate how the model's performance varies if we increase the order of LPC even further. In addition, we specified the scope of the style transfer to the word level. Although this allows for more precise LPC calculation, style "flow" between words is lost. Therefore another direction of research is to work with complete sentences, or even paragraphs. Yet another possibility for future work is to develop a much more sophisticated model, for our work is only a proof-of-concept. For example, in our work we relied on recurrent connections to capture time-series dependencies between frames. However, as have noted by [16], when working with text, it is possible to replace recurrent components of the network by using attention and transformers. It would be interesting to see whether such architecture would work for style transfer with speech.

7. REFERENCES

[1] Paul Rosin and John Collomosse, *Image and video-based artistic stylisation*, vol. 42, Springer Science & Business Media, 2012.

[2] Prateek Verma and Julius O Smith, "Neural style transfer for audio spectograms," *arXiv preprint arXiv:1801.01589*, 2018.

[3] Eric Grinstein, Ngoc QK Duong, Alexey Ozerov, and Patrick Pérez, "Audio style transfer," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 586–590.

[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] Xavier Serra, "A system for sound analysis/transformation/synthesis based on a deterministic plus stochastic decomposition," 1989.

[6] Leon A Gatys, Alexander S Ecker, and Matthias Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[7] Dmitry Ulyanov and Vadim Lebedev, "Neural style transfer for audio," 2017.

[8] Richard L Zinser, "Method for improving speech quality in code excited linear predictive speech coding," Dec. 25 1990, US Patent 4,980,916.

[9] Douglas O'Shaughnessy, "Linear predictive coding," *IEEE potentials*, vol. 7, no. 1, pp. 29–32, 1988.

[10] I Sutskever, O Vinyals, and QV Le, "Sequence to sequence learning with neural networks," *Advances in NIPS*, 2014.

[11] Hyung-Suk Kim, "Linear predictive coding is all-pole resonance modeling," 2008.

[12] Jean-Marc Valin, "Speex: A free codec for free speech," *arXiv preprint arXiv:1602.08668*, 2016.

[13] John Kominek, Alan W Black, and Ver Ver, "Cmu arctic databases for speech synthesis," Tech. Rep., 2003.

[14] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. Dec. 2011, IEEE Signal Processing Society, IEEE Catalog No.: CFP11SRW-USB.

[15] Jean-Philippe Goldman, "Easyalign: an automatic phonetic alignment tool under praat," 2011.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.