CS230

Voice Conversion using Generative Techniques

Niral Shah*
Department of Computer Science
Stanford University
niral28@stanford.edu

Abstract

Generative approaches have proven to be very effective in style transfer tasks for image data. Gatys *et al.*[3] demonstrated how the skill of convolutional neural networks (CNN) in extracting features could be applied to extract style and content information using pre-trained networks such as VGG19[15]. The result is a flood of interesting re-imaginations of classic pieces of artwork, and interesting applications such as fun camera filters. Similarly Generative Adversarial Models (GANs)[5] and Variational Auto Encoders such as CycleGan[16] have also proven to be effective in style transfer tasks for image data. In the domain of audio, the problem of Voice Conversion could be framed similarly as a style transfer problem. The ability to speak and automatically output audio in someone else's voice such as a famous celebrity remains a recurring theme in popular science fiction movies. Current voice conversion techniques often rely on parallel voice training data where two users say the same sentence, however this obviously limits the number of source and target speakers. This paper explores the feasibility of applying generative techniques to convert the voice of an unseen speaker, building on and comparing existing works including Neural Style Transfer using VGG-like networks, and a Variational Auto Encoder approach using AutoVC[13].

1 Introduction

Voice Conversion or the ability to speak in someone else's voice continues to capture the cultural zeitgeist through film and media. However these depictions often represent practical applications such as speech assistance applications for accessibility, privacy protection for text to speech systems, practical applications in entertainment industry, and potentially serving as a critical building block of addressing the challenge of speech to speech translation.

Voice conversion can be explained as applying the learned mapping on the audio between the input speaker's voice and a target speaker's voice to make it appear as though the input speaker is talking in the target speaker's voice. Existing solutions have produced high quality results when parallel utterances are available (i.e. the input speaker and target speaker have said the same sentence and both are included in the training set). The problem of Zero Shot Voice Conversion is in theory very similar to generative style transfer techniques developed for image translation. This paper will explore applying the Neural Style Transfer algorithm from Gatys *et al.* while modiying the feature extraction networks, and the results will be compared to AutoVC, an auto encoder-decoder approach to address voice conversion, trained on VoxCeleb1[11].

*Code: https://github.com/niral28/generative-vc [2, 12]

CS230: Deep Learning, Fall 2020, Stanford University, CA.

2 Related work

In recent years there have been several papers which have attempted parallel data free voice conversion. The seminal work of Gatys *et al.*.[3] demonstrated the impressive capabilities of convolutional neural networks in extracting style properties and attributes on images. However this approach was limited by only using a single image in each domain and relies heavily on the pre-trained network to extract style and content features. The application of this algorithm on Audio has been very limited and appears to be under explored. Grinstein *et al.*[7] is one of the few published papers that applied the algorithm to audio and found that a framework based on a CNN feature extractor does not generally work well on audio represented as 2D mel-spectrograms. This demonstrates that a carefully designed feature extractor designed for audio data is a pre-requisite to producing any meaningful results.

While 3x3 convolutions work well for extracting style and texture features in traditional image data, this is less applicable to audio data which is 1D in nature. Dmitry Ulyanov[1] demonstrated that 1D Convolutions can be used effectively to extract audio texture and style information.

In a similar vain, Variational Autoencoders(VAE)[9] have been gaining popularity in voice conversion tasks. While VAE's lack the distribution matching properties of GANs they are easier to train due to the fragile convergence tendency of a GAN. CycleGAN uses a encoder-decoder architecture with a GAN to extract style information to train a GAN's generator and discriminator. AutoVC takes an interesting approach by using two encoders one to extract content information from an utterance and another encoder to extract embeddings for speaker voice regardless of an utterance and then a decoder based on the work of Shen *et al.*.[14] with Tacotron 2 to generate output speech. This paper will compare these approaches and understand why one approach may work better than another.

3 Dataset and Features

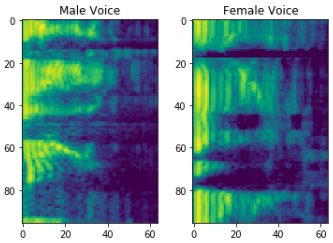

Figure 1: Example of two .wav file processed as a Mel-Spectrogram for input into the VGG19, VGGish and AutoVC encoder networks.

This paper uses the VoxCeleb1 dataset[11] which consists of over 100,000 utterances for 1,251 celebrities extracted from videos on YouTube. This dataset was chosen to due the diversity of examples and in addition due to potential interesting application of converting a user's voice into that of an iconic celebrity or movie character. For training the data, specifically the AutoVC network, the data was split into a training/test split of 97% and 3% respectively. Given the complexity of the problem, it was thought that giving the majority of data for training purposes would lead to better results.

Moreover, in order to aid the network in its ability to extract speaker style and content information, the raw audio data needs to be processed. Audio data extracted from a waveform file is 1-dimensional, however in order to help CNN's better extract information from an audio signal, it is practice to convert this into a 2-dimensional representation using *Short Time Forier Transform (STFT)* applied to the log scale to extract a 2D representation of sound with time and frequency. This representation is further processed by applying Mel filter banks which are used to apply non linearity to the frequency spectrum. See Figure 1(3) for an example of a mel-spectrogram. Given the audio data fed in all vary in length the data is chunked into bins of size NXT where N represents the number of mel-frequency bins and T the number of time steps.

4 Methods/Approach

To evaluate Voice Conversion using Generative Techniques, two types of methods are compared the application of Neural Style Transfer algorithm with VGG-based architectures (using 3x3 and 1D convolutional blocks) and the AutoVC auto-encoder decoder framework.

4.1 Neural Style Transfer

Neural Style Transfer for images, originally outlined by Gatys *et al.*, takes as input two $224x224x3$ images A, B representing the style and content with the goal of outputting another image that applies the style of image A to the content of image B. To review, the Neural Style Transfer generates an image by performing gradient descent on a a randomly initialized image, $\overrightarrow{x}$. The algorithm first minimizes loss on the mean square loss between the $\overrightarrow{x}$ and content image $\overrightarrow{p}$, $\mathcal{L}_{content}(\overrightarrow{p}, \overrightarrow{x}, l) = \frac{1}{2}\sum_{i_j}(F^l_{i_k} - P^l_{i_k}$. In addition to minimizing the content loss, $\mathcal{L}_{content}$ the generated image should minimize style loss between the generated image, $\overrightarrow{x}$ and the style image, $\overrightarrow{a}$. The definition of style is calculated using Gram matrices, where $G^{[}l]_{i_j}$ is the inner product between the vectorized feature maps i and$j$ in layer l and $F^{[}l]_{i_j}$ corresponds to the activation of the i^{th} filter at position j in layer l.

$$G^l = \sum_k F^l_{i_k} F^l_{j_k}$$

Give $\overrightarrow{a}$ and $\overrightarrow{x}$ correspond to the original image and the generated image, and A^l and G^l correspond to their respective style representation in layer l. The contribution of layer l to the total style loss is: $E_l = \frac{1}{4N_l^2 M_l^2}\sum_{i_j}(G^l_{i_j} - A^l_{i_j})^2$ This means total style loss is, $\mathcal{L}_{style}(\overrightarrow{a}, \overrightarrow{x}) = \sum_{l=0}^{l} w_l E_l$ And total loss, used for optimization is defined as:

$$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style}$$

(3)).

Finally, in order to reconstruct the audio from the spectrogram, the Griffin-Lim Algorithm[6] was for several iterations to get a 1D audio signal.

4.1.1 Using VGG-19 with ImageNet weights

To get a baseline understanding of how well generative approaches with Convolutional Neural Networks might work for audio data, we start off by testing how well Neural Style transfer with pre-trained VGG on ImageNet works on Mel-Spectrogram Images.

In order to match the expected input size of VGG, the 2D audio mel-spectrogram was tiled into three channels and resized to match the (224 x 224 x 3) image size. To calculate style gradients, similar to the original paper, the activation from VGG's $block1\_conv1$,$block2\_conv1$, $block3\_conv1$, $block4\_conv1$, and $block5\_conv1$ were taken; while to calculate content gradients the activation from $block5\_conv2$. This was chosen based on the principles that higher level layers have learned more complex features while lower level layers have learned specific attributes of an image.

4.1.2 Using VGGish with AudioSet weights

Moving further, a hypothesis was made that an CNN based Audio Classifier may produce even better results, given the architecture of VGG-19 is designed for images which are more complex and spatial in nature compared to audio spectrograms. The VGGish[8] architecture pre-trained on the Audio Set dataset, which contains audio from thousands of YouTube Videos, [4] fit a good description of a network with layers that might produce features that could more meaningfully represent style and content while using the same loss function. While, VGGish only has 6 convolutional layers, the same principle applied in VGG19 based Style Transfer was applied here, where style gradients were calculated using layers $conv1$, $conv1$, $conv3/conv3\_1$, $conv3/conv3\_2$, $conv4/conv4\_1$, while the final activation from the final convolutional layer $conv4/conv4\_2$ was used to calculate content gradients.

4.2 Shallow Randomly Initialized CNN

While the above methods rely on Deep Neural Networks with 3x3 convolutional layers designed for classification purposes, the primary goal of these input networks is to extract a meaningful representation of style and content information. Audio by nature is a one dimensional signal, converting it using a STFT represents it as a 2D image like representation. Ulyanov's method modifies this approach by treating the 2D spectrogram as a 1xT image with 4096 Filters, where T represents the timesteps of the spectrogram. Once the input feature is reshaped in this way, a shallow single layer 1D convolutional network with 4096 filters could extract style and content information that could be fed to Gatys' original Neural Style Transfer algorithm to produce meaningful results. One of the challenges of this approach is that since vocal audio is subtly different across speakers compared to music or image data, it becomes harder to extract style content.

4.3 AutoVC: Autoencoder-Decoder

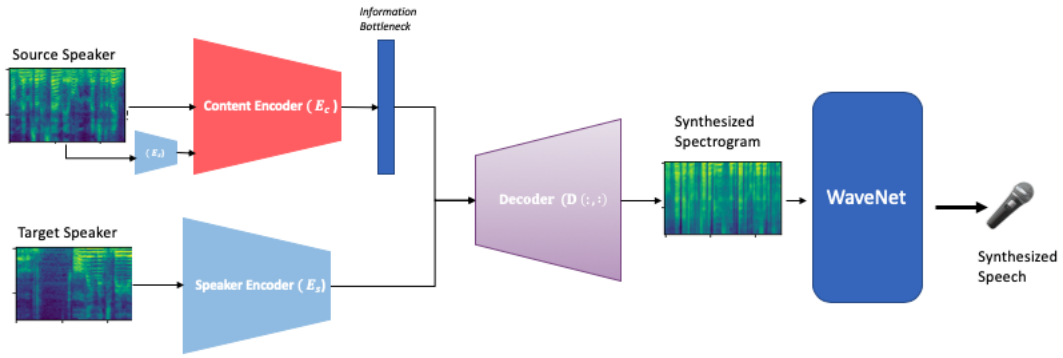

Figure 2: AutoVC Auto Encoder-Decoder Conceptual Architecture [13]

While the Neural Style Transfer algorithm takes on the task of Image Translation well, it appears that without major changes the algorithm would not work effectively for the task of Voice Conversion. Taking some of the same principles, such as using 1D convolutional layers, AutoVC rather than trying to calculate style attempts to calculate the direct mapping between a source speaker and a target speaker with an encoder-decoder architecture (see Figure 2). The architecture consists of a content encoder E_C, and a speech encoder E_S. The speech encoder is a pre-trained network on VoxCeleb1 and Librispeech for a total of 3549 speakers, which make network embeddings generalizable. While the content encoder takes as input the mel-spectrogram of source speech X_1 concatenated with the speaker embedding $E_S(X_1)$.

During the training process, the content and decoder are trained to minimize the identity loss between two input different utterances U, from the same speaker X and the generated output from the decoder D:

$$C_1 = E_c(X_1), S_1 = E_c(X_1^{'}), \hat{X}_{1->1} = D(C_1, S_1)$$

$$min_{E_c(\cdot),D(\cdot,\cdot)} L = L_{recon} + \lambda L_{content}$$

where L_{recon} is the MSE (L2) of the generated output $\hat{X}$ and X while $L_{content}$ is the the euclidean distance (L1) between $E_c(\hat{X})$ and C_1.

While the intricacies of the details of the AutoVc Architecture are in the Appendix, it is worth noting that a key aspect which the success of the results rely on is the Bottleneck layer. The output embeddings of the Content Encoder are down sampled in this bottleneck layer in an attempt to remove information about the source speaker. If the embeddings are overly downsampled then too much information is removed leading to noiser results; if the bottleneck is too wide then the reconstruction is imperfect.

5 Experiments & Evaluation

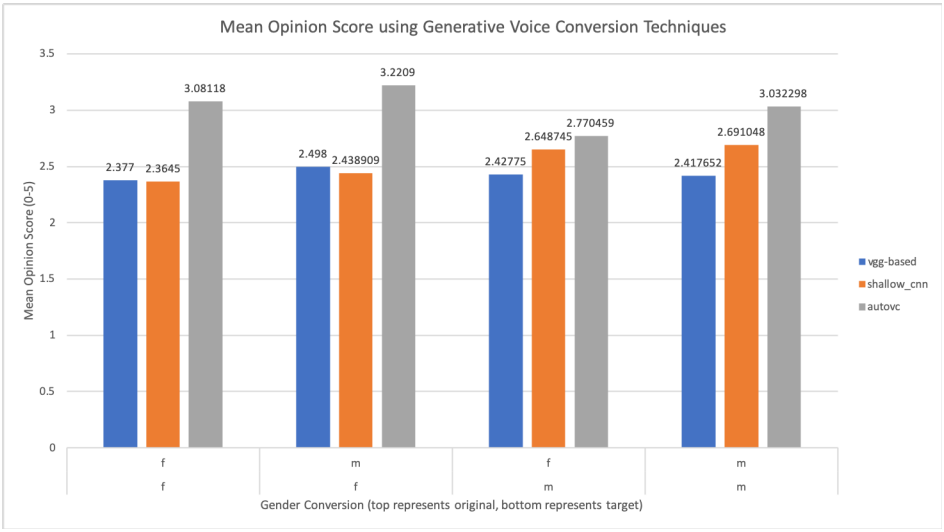

Figure 3: Average Mean Opinion Score for each Approach. [13]

To quantitatively evaluate and compare the results of each of the algorithms explored, this paper uses Mean Opinion Score to rank an audio clip from 1 to 5 the "realness" of the generated speech. Typically this is by surveying large crowds, using something like Mechanical Turk, however since this was not feasible, MOSNet[10] was used to evaluate the audio files. The data was evaluated on a test set of 40 speakers, with 310 voice conversion pairs that were generated across the test speaker set.

5.1 Neural Style Transfer using VGG-19 & VGGish

Both the VGG based experiments were run for 1000 iterations using pretrained weights. VGG-19 used weights pre-trained on Image Net, while VGGish used weights pre-trained on Audio Set. In order for the 2D spectrogram to fit into the 3D Convolutional layers, the spectrogram is tiled for each channel and rescaled to fit the network's inputshape. For faster convergence, the generated spectrogram was initialized to that of the content spectrogram. Ideally this would make the sound resemble the original speech. The generated image was optimized using the Adam Optimizer with parameters learning_rate=0.02, beta_1=0.99, epsilon=1e-1. The audio is reconstructed using the Griffin Lim algorithm.

As seen in Figure 3 above, across all gender pairs, the VGG based methods produced dismal MOS scores, an average of about 2.4/5. Despite being initialized to the content speech, the algorithm produced very noisy clips. Some changes in volume and sound patterns indicated the algorithm was attempting to extract some information.

For choosing the style and content layers some experimentation occurred. The content layer, following the original Neural Style Transfer algorithm, was chosen to be the second to last Conv output (before ReLU activation).

5.2 Neural Style Transfer using Shallow 1D-CNN

The Randomly Initialized CNN with a single 1D Convolutional layer approach was fairly straight forward. The number of filters chosen for the 1D Convolutional layer was found to perform best when 4096 filters were used. This is necessary to extract enough information out of the 1D audio signal. The style audio STFT image is fed into the network to get the style vectors; while the content audio STFT image is fed into the network to get the content vectors to create a gram matrices. In this approach, it expects the size of the content and style matrices to have the same shape. In order to address this, the data was zero padded.

As seen in Figure 3 above, the Shallow CNN method performed comparably to the vGG-19/VGGish based method, albeit slightly better.

Qualitatively, it was observed that in this method the audio sounded more meaningful, but generally a noisier version of the original content audio clip. Style information did not get transferred much, perhaps because the approach has a hard time differentiating between the vocal qualities of the two speakers. See appendix for a visualization of the style transfer process, its clear from the figure that the generated STFT image is very close to the original speaker's STFT but with lots of noise added.

5.3 AutoVC

AutoVC was run for 300k iterations with and used an Adam Optimizer with parameters batch_size: 2, adam_beta1': 0.9, adam_beta2: 0.999, adam_epsilon: 1e-8. The algorithm was re implemented in Tensorflow 2.0. It achieved a final loss of around 0.0001, which is comparable to the loss achieved in the original paper. The bottleneck size hyper parameter was experimented with between 16 and 32, it was found that a bottleneck size of 32 performed the best. It was observed when the bottleneck size is too small it results in an overly smooth signal, suggesting too much down sampling occurred at the bottleneck.

As seen in Figure 3, the AutoVC algorithm performs the best achieving an average MOS score of around 3.1/5. This matches the qualitative observations of the audio clip. The algorithm does a good job in transferring speaker style information (i.e. converting a male voice to female), however the content information is heavily lost, resulting in a lack of similarity to the original content audio clip or source speaker's utterance. Furthermore, the algorithm performs the worst in male to female voice conversion and female to male voice conversion tasks, suggesting that maybe a lack of female speakers were present in the speaker embedding dataset, the speaker embedding network used was pretrained.

6 Conclusion & Future Work

Overall three different approaches were explored to see if voice conversion could be addressed using generative techniques. First this paper explored the potential of treating the problem of voice conversion as a style transfer problem. To establish a baseline, the Neural Style Transfer algorithm, designed for images; both VGG-19 pretrained on ImageNet and VGGish pretrained on AudioSet were explored as feature extractors. However, this approach produced very noisy results, with an average MOS of approximately 2.5/5. Next, rather than treating audio as 3D images, the next approach attempted was a Shallow randomly initialized CNN with 1D convolutional layer with many filters. This produced slightly better results, doing better in some voice conversion types such as female to male , and male to male voice conversion. An auto encoder-decoder framework was explored based on AutoVC, which was re-implemented in TensorFlow. This algorithm produced much better results, with a MOS range from 2.8-3.2 across all voice conversion tasks, performing the worst on female to male voice conversion tasks. It was observed that in many of these clips the voice style transfer completed successful but the content information was lost, so the actual similarity between the generated audio and the original content speaker's audio was lacking.

To further improve upon the results, better fine tuning of the content encoder in the AutoVC network could be completed. Clearly the down sampling, which occurs in the network removes too much content information in its attempt to strip the source speaker style, is too narrow. Widening the bottleneck to sizes like 64 (vs the current 32 and 16 tried), might produce better results. Similarly, recently in many audio and text related tasks, the Attention mechanism using LSTM has shown promising results. Since audio information is sequential by nature, this could be a useful improvement in the content encoder. Given more time, this may be the key to addressing the shortfalls of the AutoVC network.

7 Contributions

I was the only member of the team.

References

[1] audio-texture-synthesis-and-style-transfer.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

[3] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016. doi: 10.1109/CVPR.2016.265.

[4] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[6] D. Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984. doi: 10.1109/TASSP.1984.1164317.

[7] E. Grinstein, N. Q. K. Duong, A. Ozerov, and P. Perez. Audio style transfer. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr 2018. doi: 10.1109/icassp.2018.8461711. URL `http://dx.doi.org/10.1109/ICASSP.2018.8461711`.

[8] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson. Cnn architectures for large-scale audio classification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017. URL `https://arxiv.org/abs/1609.09430`.

[9] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2014.

[10] C.-C. Lo, S.-W. Fu, W.-C. Huang, X. Wang, J. Yamagishi, Y. Tsao, and H.-M. Wang. Mosnet: Deep learning based objective assessment for voice conversion, 2019.

[11] A. Nagrani, J. S. Chung, and A. Zisserman. Voxceleb: a large-scale speaker identification dataset. In *INTERSPEECH*, 2017.

[12] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[13] K. Qian, Y. Zhang, S. Chang, X. Yang, and M. Hasegawa-Johnson. Autovc: Zero-shot voice style transfer with only autoencoder loss, 2019.

[14] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. J. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu. Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions. *CoRR*, abs/1712.05884, 2017. URL `http://arxiv.org/abs/1712.05884`.

[15] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[16] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020.

A Appendix

This is a detailed figure of the AutoVC network:

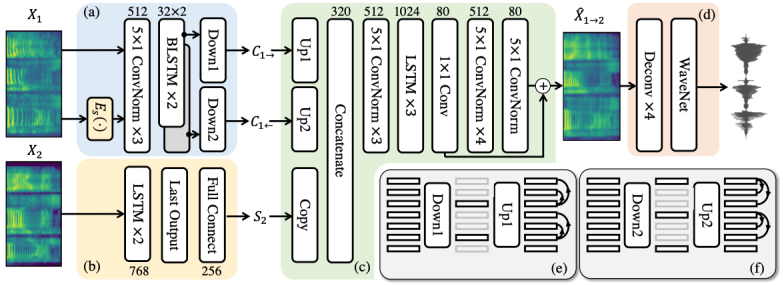

Figure 3. AUTOVC architecture. The number above each block represents the cell/output dimension of the structure. ConvNorm denotes convolution followed by batch normalization. BLSTM denotes bi-directional LSTM, whose white block denotes forward direction, and grey block denotes backward direction. (a) The content encoder. The $E_s(\cdot)$ module is of the same architecture as in (b). (b) The style encoder. (c) The decoder. (d) The spectrogram inverter. (e) and (f) demonstrate the downsampling and upsampling of the forward and backward outputs of the Bi-directional LSTM, using a up/downsampling factor of 3 as an example. The real up/downsampling factor is 32. The lightened feature denotes that they are removed; the arrows denote copying the feature at the arrow origin to the destination.

Figure 4: The AutoVC Network.

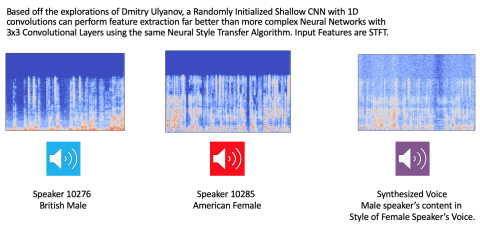

Figure 5: Input and Outputs from Neural Style Transfer applied using the Randomly Initialized CNN.

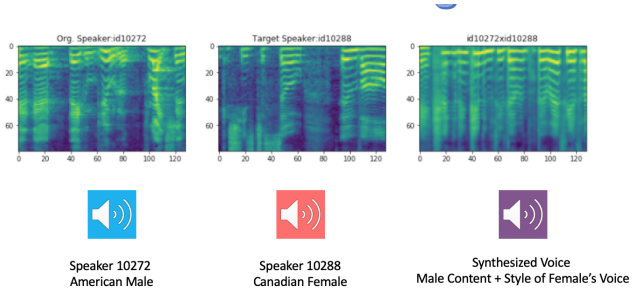

Figure 6: Inputs & Outputs from the AutoVC Network

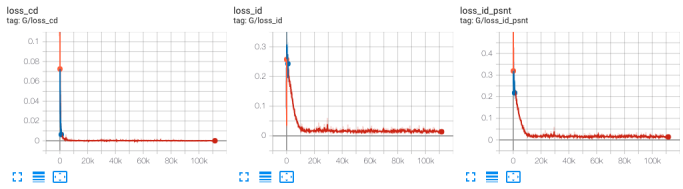

Figure 7: *From left to right*: Content Loss, Initial Reconstruction Loss, Final Reconstruction Loss. The AutoVC network was trained for 300k iterations. The original paper achieved a total summed loss of 0.0001.