Style-transfer Autoencoder for Efficient Deep Voice Conversion

Ilya Makarov
HSE University, Moscow, Russia
Artificial Intelligence Research Institute, Moscow, Russia
iamakarov@hse.ru

Denis Zuenko
HSE University, Moscow, Russia
neron.v2@gmail.com

2021 IEEE 21st International Symposium on Computational Intelligence and Informatics (CINTI) | 978-1-6654-2684-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/CINTI53070.2021.9668528

***Abstract*—We consider the problem of voice cloning, which is desirable in many film-related industries, and developed a new modification of the AutoVC state-of-the-art model in the task of voice conversion. We studied the replacement of recurrent modules with convolutional layers while maintaining the quality of the original model. The result of our work showed the speed improvement on longer voice tracks and faster training with the tiniest deterioration in sound quality, as evidenced by the reconstitution loss and Mel-cepstral distortion.**

***Index Terms*—Deep voice cloning, deep fakes, voice style transfer, deep auto-encoders.**

I. Introduction

Deep learning models have become useful in many fields of applied machine learning. In our studies, we showed the advances of deep learning approaches in various computer science fields, for e.g., in depth estimation deep learning helps to reconstruct depth from various input sources in an end-to-end manner [1]–[7]; in graph machine learning deep neural networks aggregate information from similar nodes [8]–[11]; in deep reinforcement learning agents extract meaningful state representations to propagate reward into action policies or model human behavior in video-games [12]–[20]; in graph recommender systems meaningful features are extracted and fused from attribute and structural information [21]–[29].

In the case of speech processing, deep Conversion or voice cloning (VC) modifies the speaker characteristics of a given speech while preserving the linguistic information. Deep models that would offer more natural-sounding speech than the standard methods. In connection with the widespread use of voice interaction with machines, there is a need to increase the efficiency of high-quality speech synthesis, which has a significant impact on the experience of customer interaction with the system and its preferences. However, the quick building of a high-quality speech remains a challenging task.

We choose this project because we want to expand possible ways to make speech-to-speech translation language. It should be fast to work on mobile or variable devices for different tasks, such as Speech-to-text or Text-to-speech, Generative Models for raw audio, and Deep Voice Conversion.

The goal of this work is to investigate and improve the existing voice Conversion model, which can be an exciting part of action films and science fiction films, and various functional forms, make it suitable for more data, but at the same time maintaining accuracy. This system should take as input two voice samples and produce a style-transferred target voice. We want to make our system as fair as possible. So we aim to try to improve and extend the Autoencoders approach for different kinds of speaker datasets. Our hypothesis is that last computer vision and deep neural networks for wave analysis methods can outperform the state-of-the-art works in voice conversion. The neural networks are implemented within the Pytorch framework In this work, we approach the analysis of the non-visual data using computer vision methods.

The speech is a sequence of varying pressure strengths in a 2D array. Most of the voice data contain unique patterns which are showed to be learnable. However, the data may contain noise, e.g., people talking around, and a bad microphone can capture sounds around the spokesman, that is why preprocessing or clear data is a crucial part.

We assumed that the dataset is labeled. So, we have double tasks: an unsupervised task and a supervised task. Our goal is to teach the neural network to produce embeddings, and another one to produce speech that switched to the target speaker domain.

Representing the speaker as a Mel-spectrogram and stack convolution layers seems like an easy task. But if we look further, there are time-correlated features that should be extracted accurately, while saving computational power. So, that is why we took a state-of-the-art model AutoVC and tried to improve it while holding in mind that sound data clearing can help us to reduce noise in the evaluation step.

II. Related works

Several works offer voice conversion applying Variational Auto-Encoder (VAE) including its combination with adversarial training. Authors of [30] extended VAE-based VC new encoding features of speaker's dynamic components, in addition to the encoded latent features, allow reproducing spectral characteristics at the generation stage. Speaker-code associated with the source (original) speaker is used to approximate the spectrogram; a speaker-code associated with the desired dynamic target is used to a converted spectrogram. In the proposed CycleVAE, the converted characteristics are returned to the system to create a matching cyclically reconstructed spectrogram that can be directly optimized. The cyclic flow

can then be maintained by filling cyclically restored characteristics back into the system. Consequently, the conversion, the estimation of the transformed spectra, is indirectly taken into account when assessing both the recovery loss and the regularization of the hidden space. In experiments, the authors show that the CycleVAE-based VC [30] shows a higher correlation degree of latent features within phonetic space and accurate conversion.

StarGAN-VC [31] tackles the problem of voice Conversion using a single generator network. However, even such an in-depth approach does not entirely remove the differences between the original and the reconstructed speech. To fill this gap, the authors [32] rethink the StarGAN-VC conditional methods for training objectives and network architectures, which are critical components for creating non-parallel multi-domain VCs in a specific model, and offer an improved version called StarGAN-VC2. For the above, researchers of StarGAN-VC2 propose a source-and-target conditional adversarial loss that provides all source domain data to be reciprocal to the target domain data.

In a VAEVC (CDVAE-VC) [33] framework, use acoustic characteristics of various properties, to raise the quality of VAE-VC. In their work, they prolong CDVAE-VC structure, combining the notion of competitiveness in learning, to significantly increase the degree of unraveling, thereby improving the quality and identity of the transformed speech. And, in particular, the authors investigated that the effectiveness of combining Generative Adversarial Network (GAN) and their network. Considering the concept of learning in an adversarial field and add specific constraints to the basic latent representation obtained by the speaker's classifier to exclude the speaker's data. Results confirmed that both GANs and the speaker classifier can improve the quality of disentanglement of the learned latent representation. Meanwhile, individual evaluation effects in terms of accuracy and similarity scores show the effectiveness of these methods.

From information theory, success in deciphering a voice timbre is described by the presence of a speaker identification tag that stores almost all of the timbre information so that voice transformations can extract such information from the voice. For example, AutoVC, a state-of-the-art voice conversion system, constructs an autoencoder for speech and input the speaker embedding representation to the decoder. AUTOVC [34] attempts to force the auto-encoder to extract temporal data about unique voice characteristics. But, it is not that efficient Wavenet [35] has used as a vocoder, which decodes it too slow. Another slow part is encoded content, and embedding features decodes with recurrent layers. But it is designed to store lengthened sequence information. Still, the feedforward in Long Short-Term Memory (LSTM) has two problems. The first is that the refresh gradient during backpropagation is small. The second problem is the number of computing operations.

The authors of SpeechSplit [36] pointed out that AutoVC can force the encoder to eliminate the timbre information because the similar information provided to straight the decoder. Within network architecture, researchers decreased the decoder layers and gave more speech components with an additional bottleneck. Also, they have shown that the concrete dimension of the hidden representations can dramatically restrict the information flow.

III. Problem statement

A. Methodology

We aim to solve the problem of voice Conversion using a transformation network and a loss network. Assume some stochastic process generates that speech. Speaker denoted by $\mathbf{U}$ draw from speaker population $p_U(\cdot)$ and a content vector $Z = Z(1 : T)$, where a $(1 : \mathbf{T})$ time indices running drawn from joint content distribution $p_Z(\cdot)$. Content is the information about phonetic and prosodic. Given speaker identity and a content, we got speech segment denoted by $X = X(1 : T)$. Probability density function (PDF) at all times denoted by $p_X(\cdot|Y)$, $\mathbf{X}$ conditional on $\mathbf{Y}$ taking a specific value $\mathbf{y}$. Similarly $\mathbb{E}[X|Y]$, $\mathbb{E}[X|Y = y]$, $\mathbb{E}[X|y]$ denotes the corresponding conditional expectations. $H(\cdot)$ - entropy, $H(\cdot|\cdot)$ - conditional entropy.

B. Data

A signal is a bend to an extent over time. For audio, the size that changes is air pressure. We take examples of air pressure overhead time. The rate at which we inspect the data can vary, but it is commonly 22kHz or 20000 samples per second. Captured a waveform for the signal, and this can be explained, modified. We used [37].

The Mel-spectrum is the amplitude spectrum but taken on the Mel scale with a specific step and window. We set the number of steps in advance of the value 80 used. The transition to the Mel-spectrum saves characteristics essential for the speech signal but reduces the amount of data.

1) The Fourier Transform: This operation in the sequential application of the Fourier transforms to short pieces of the speech signal, multiplied by some window function. The result of applying the window transform is a matrix, where each column is the spectrum of a short section of the original signal.

$$F(k,m) = \sum_{n=0}^{L-1} x[n+m]w[n]e^{-i\frac{2x}{L}kn}$$

Analyses have shown that the human ear is more sensitive to changes in sound at low frequencies than at high. That is, if the frequency of sound changes from 100 Hz to 120 Hz, a person is very likely to notice this change. However, if the frequency changes from 10,000 Hz to 10,020 Hz, we are unlikely to be able to catch this change. So we decided to use a new Mel unit for measuring pitch was introduced: $Mel = 1127.01048 \ln\left(1 + \frac{freq}{700}\right)$.

Mel-spectrogram is a regular spectrogram where the frequency is expressed not in Hz, but in Mel. The transition to chalk was carried out by applying filters to the original spectrogram. These filters are triangular functions evenly distributed on the Mel-scale.

2) Datasets: We took several most popular and big datasets for speech tasks. For experiments, we clear and sorted these three datasets [38], TIMIT [39] which used only for speaker embedder pre-training, VoxCeleb2 [40] and [41]. As we can say from experiments, the most valuable part of speech synthesis or voice conversion and other related tasks is data. It was challenging to converge even for pre-trained networks from related works. We collected statistics about noise, volume, length, and others, deleted all outliers and noise as shown in Table I.

TABLE I
DATASETS STATISTICS

Dataset Name	Number of speakers train	Number of speakers test
CommonVoice	498	74
VCTK corpus	109	20
TIMIT	328	70
VoxCeleb2	200	50

To avoid segments that are mostly silent when sampling partial utterances, we use the python package [42] to perform Voice Activity Detection (VAD). The algorithm sets a binary flag over the audio corresponding to whether or not the segment has a voice or not.

C. Formulation

Let us assume that each speaker produces the same amount of gross information $H(X|U=u) = h_{\text{speech}} = constant$. Two sets of variables (U_1, Z_1, X_1) belong to the source, and (U_2, Z_2, X_2) to the target speaker, are i.i.d., random samples from this process. Our aim to create a speech converter, such that $\hat{X}_{1\to 2}$ and preserves the content in $\hat{X}_1$, but matches features from U_2. Ideally our formal task is $p_{\hat{X}_{1\to 2}}(\cdot|U_2=u_2, Z_1=z_1) = p_X(\cdot|U=u_2, Z=z_1)$. But this task is limited that U_1 and U_2 should be in the training set, which constrains our task. We hope that, the training process will scale our framework to unseen speaker.

D. The Autoencoder Framework

Our framework based on **AutoVC** [34] and **Tacotron2** [43]. It is consist of three networks. $E_c(\cdot)$ represents an Encoder, which create a content vectors, $E_s(\cdot)$ produces speaker embeddings and a decoder $D(\cdot,\cdot)$ outputs a speech from content and a speaker embedding.

1) Conversion : Our goal is to show that during the voice-Conversion that if the source is filled into the content Encoder $E_c(\cdot)$, then when the content vector extracted, the target speech is fed into the speaker encoder $E_s(\cdot)$ which provides target speaker embedding. The decoder $D(\cdot,\cdot)$ produces the converted speech based on the content information and the speaker information in the target: $C_1 = E_c(X_1)$, $S_2 = E_s(X_2)$, $\hat{X}_{1\to 2} = D(C_1, S_2)$, where $\hat{X}_{1\to 2}$ and C_1 are random processes and S_2 is a vector.

2) Training: Through our training process, we assume that the speaker encoder $E_s(\cdot)$ and a content encoder $E_c(\cdot)$ are both pre-trained. The speaker encoder trained in the speaker verification task by minimizing so-called generalized end-to-end (GE2E) [44]. A content encoder is pre-trained by AutoVC [34]. The input to the content encoder is X_1, the input to the voice-Conversion encoder becomes an sample from the same speaker U_1, denoted as $X_1^{'}$. Then for each speaker X_1: $C_1 = E_c(X_1)$, $S_1 = E_s(X_1')$, $\hat{X}_{1\to 1} = D(C_1, S_1)$. The loss function should be able to minimize self-reconstruction error and the content code reconstruction error [34]:

$$L_{\text{rec}} = \mathbb{E}\left\|\hat{X}_{1\to 1} - X_1\right\|_2^2,\ L_{\text{con}} = \mathbb{E}\left\|E_c\left(\hat{X}_{1\to 1}\right) - C_1\right\|_1$$

The final task is $\min_{E_c(\cdot),D(\cdot,\cdot)} L = L_{\text{rec}} + \lambda L_{\text{con}}$.

IV. ARCHITECTURE

According to Figure 1 our network consists of three major modules: a speaker encoder, a content encoder, a decoder connected to Postnet [43]. As a base, we took AutoVC architecture. As an input, we took a speech Mel-spectrogram of the size N-by T, where N is the number of Mel-frequency bins, and T is the time steps (frames). The vocoder used to convert the output Mel spectrogram back to a waveform, also described in detail in this section.

A. The embedding network

As shown in Figure 1(b), the embedding network consists of three LSTM layers and a fully connected layer that projects it to a 256 size vector. The resulting speaker embedding of one speaker (10 speeches of one speaker at least) then we averaging all. The speaker encoder is trained on the GE2E loss [44], we also follow this work with further notations. The embedding vector (d-vector) is defined as the L2 normalization of the network output:

$$\mathbf{e}_{ji} = \frac{f(\mathbf{x}_{ji};\mathbf{w})}{\|f(\mathbf{x}_{ji};\mathbf{w})\|_2}$$

Here $\mathbf{e}_{ji}$ i represents the embedding vector of the j-th speaker's i-th utterance. The centroid of the embedding vectors from the j-th speaker $[\mathbf{e}_{j1},\ldots,\mathbf{e}_{jM}]$ is defined as $\mathbf{c}_j = \mathbb{E}_m[\mathbf{e}_{jm}] = \frac{1}{M}\sum_{m=1}^{M}\mathbf{e}_{jm}$ The similarity matrix $\mathbf{S}_{ji,k}$ is defined as the scaled cosine similarities between each embedding vector $\mathbf{e}_{ji}$ to all centroids $\mathbf{c}_j$ $(1 \le j,k \le N$, and $1 \le i \le M)$, $\mathbf{S}_{ji,k} = w\cdot\cos(\mathbf{e}_{ji},\mathbf{c}_k) + b$, where w and b are learnable parameters. After that authors put softmax on $\mathbf{S}_{ji,k}$ that makes the output equal to 1 iff $k=j$, otherwise makes the output equal to 0. Thus, the loss on each embedding vector $\mathbf{e}_{ji}$ defined as:

$$L(\mathbf{e}_{ji}) = -\mathbf{S}_{ji,j} + \log\sum_{k=1}^{N}\exp(\mathbf{S}_{ji,k})$$

The contrast loss defined on positive pairs and aggressive to negative pairs, as:

$$L(\mathbf{e}_{ji}) = 1 - \sigma(\mathbf{S}_{ji,j}) + \max_{\substack{1\le k\le N\\ k\ne j}}\sigma(\mathbf{S}_{ji,k}), \text{ where}$$

$$\mathbf{S}_{ji,k} = \begin{cases} w\cdot\cos\left(\mathbf{e}_{ji},\mathbf{c}_j^{(-i)}\right)+b & \text{if } k=j \\ w\cdot\cos(\mathbf{e}_{ji},\mathbf{c}_k)+b & \text{otherwise}\end{cases}$$

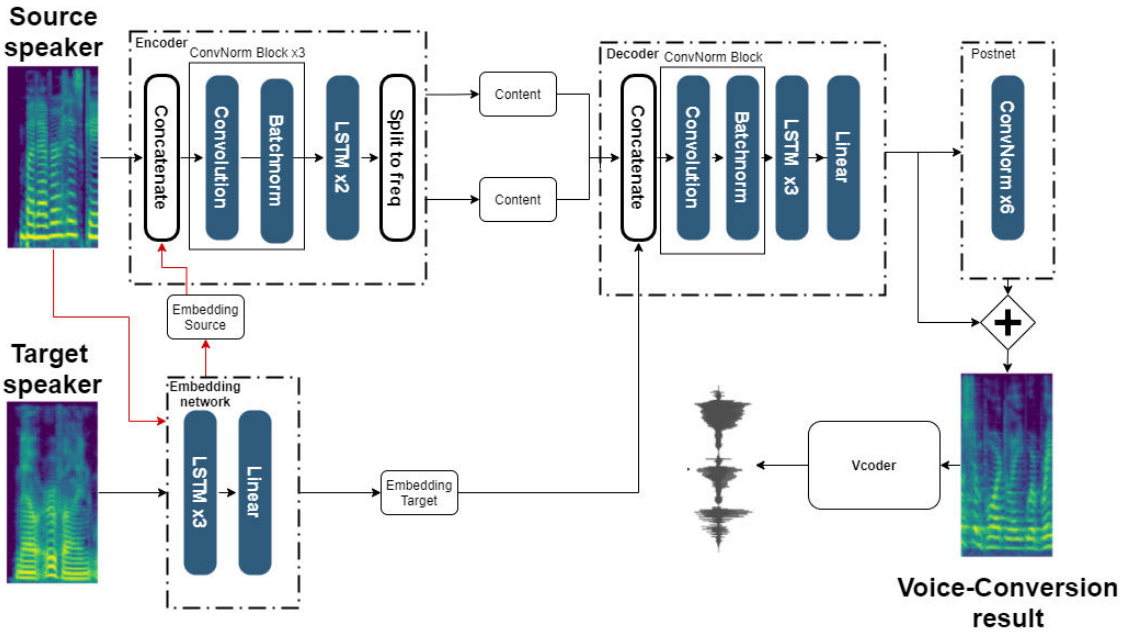

Fig. 1. Our proposed model for voice style-transfer

From a computing perspective, the cosine similarity of two L2-normed vectors is simply their dot product. Combining Equations above the final GE2E loss L_G is the sum of all losses over the similarity matrix $\mathbf{c}_j$ ($1 \leq j,k \leq N$, and $1 \leq i \leq M$): $L_G(\mathbf{x};\mathbf{w}) = L_G(\mathbf{S}) = \sum_{j,i} L\left(\mathbf{e}_{ji}\right)$. This loss allows us to produce embeddings from speech efficiently.

B. The Encoder (content encoder)

In Figure 1, the input vector is a Mel-spectrogram concatenated with the speaker representation from the Embedding network at each time step. These features are fed into the ConvNorm block with ReLU activation and then into bidirectional LSTM. Bidirectional helps the model to read the sequence from left to right and also from right to left. In that way, we obtain information about the past and future of each feature. The resulting content is a set of two 32-by-T/32 matrices, which are denoted as $C_{1\rightarrow}$, $C_{1\leftarrow}$. This information bottleneck is that downsampling defines downsampling along the temporal axis, which is common in the Autoencoders architectures handles important properties:

- Data-specific. We can not use autoencoder compress photos if it was trained on the spectrograms because this compression can not be efficient.
- Lossy. While compressing input in the Encoder part, we lose information. However, in the end, it should be close.
- Self-supervised. Because Autoencoder generates their labels from the training data.

C. The Decoder

First, the two content and one speaker embeddings are Upsampled with Convolution layer to restore the original resolution. Formally, we define the upsampling features as $U_{\rightarrow}$, $U_{\leftarrow}$ and then $U_{\rightarrow}(:,t) = C_{1\rightarrow}(:,\lfloor t/32 \rfloor)$, $U_{\leftarrow}(:,t) = C_{1\leftarrow}(:,\lfloor t/32 \rfloor)$. The intuition of this formula is that each representation vector at the time step should contain information about the past and future. For speaker embedding, we copy the vector T times. We concatenate them before passing them to the Convolution layer, as shown in Figure 1. A convolutional layer can be useful to represent local correlations between contents and embeddings by extracting features for RNN modeling sequences. Which gives the Decoder model a long-term context. Formally, (from f_1 to f_{T_x}) and a backward RNN $\overleftarrow{f}$ which reads the input sequence in reverse order (from f_{T_x} to f_1). Their respective outputs are $U_{\rightarrow}$ and $U_{\rightarrow}$. $\overrightarrow{f}$ and $\overleftarrow{f}$ functions in LSTM. After upsampling, we fed the resulting matrix to the ConvNorm block, which consists of the Convolution layer and BatchNorm activated with the ReLU layer. Then passed to three bidirectional LSTM layers with cell, outputs projected to down with Linear layer. Resulting vector is an estimation of converted speech $\tilde{X}_{1\rightarrow 2}$.

D. The Postnet

Last but not least, once the decoder did decoding, the predicted Mel-spectrogram is passed through a stack of convolutional layers with tanh nonlinearities to increase the overall output quality [43]. Due to the ability of convolutional layers to obtain features in images or sequences, especially that they see both past and the future context in already decoded spectrograms. A final linear projection layer follows these convolutions to the output space. Formally, the PostNet can be computed in a residual manner: $R_{1\rightarrow 2} = \text{Postnet}(\tilde{X}_{1\rightarrow 2}) = W_{postnet} f_{postnet} + b_{postnet}$, where $f_{postnet}$, the Postnet convolutional features computed as a stack of 5 convolutions with tanh activations $F_{postnet,i} * x$ with x the output of

the previous convolutional layer or from the decoder at the initial step. To obtain Mel-spectrogram, we sum the decoder spectrogram outputs with the residual to get the final outputs $\hat{X}_{1\to2} = \tilde{X}_{1\to2} + R_{1\to2}$

E. The Vocoder (Spectrogram Inverter)

We convert Mel-spectrogram to the waveform. We apply WaveNet [35], because the authors of the AutoVC framework already trained it, so it is more comfortable to compare results. WaveNet is a type of network, where the convolutional layers have various dilation factors that allow its receptive field to grow and cover thousands of timesteps.

We did a couple of experiments with effective Vocoder implementation. In [45], SqueezeWave model shows efficiency and quality compared to the WaveNet model selected by the authors. But unfortunately, after long weeks of training, we were not able to achieve a good result applicable to the WaveNet model.

V. Experiments

Hyper-parameters for Mel-spectrograms are as follows: Fast furrier transforms length equals 1024, # of Mel-spectrograms is 80 with hop length equaled to 256, and $f_{min} = 90, f_{max} = 7600$, sample rate equaled to 16000.

Before setting up an experiment, we prepared the data using the method described in the Data section. The first in line for training with us is the embedder encoder based on [46]. The only difference is in the data. We wanted the domain of all our data to match. To confirm our success in training, we should define the metric Equal Error Rate (EER). But before we should determine False Positive Rate (FPR), False Negative Rate (FNR)

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{FN}+\mathrm{TP}}; \mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}},$$

where TN - True Negative, FP - False Positive, FN - False Negative, TP - True Positive. During training, reconstruction loss is applied to both the initial and final reconstruction. Formally, the loss function:

$$L_{\mathrm{rec}} = \mathbb{E}\left\|\hat{X}_{1\to1} - X_1\right\|_2^2,\ L_{\mathrm{con}} = \mathbb{E}\left\|E_c\left(\hat{X}_{1\to1}\right) - C_1\right\|_1,$$

$$L_{\mathrm{rec}_0} = \mathbb{E}\left\|\tilde{X}_{1\to1} - X_1\right\|_2^2$$

$$\min_{E_c(\cdot),D(\cdot,\cdot)} L = L_{\mathrm{rec}} + L_{\mathrm{rec}_0} + \lambda L_{\mathrm{con}}$$

To evaluate the algorithm, we primarily focus on the common metric in Auto-Encoders is the reconstruction loss described above. Secondly, we consider Mel-Cepstral Distortion (MCD), which measures the spectral distortion, we compute

$$MCD = \frac{10}{\log 10}\sqrt{2\sum_{d=1}^{K}\left(mcc^{(c)} - mcc^{(t)}\right)^2}$$

VI. Results

This network trained quite easily, at some point the loss hung, and after testing, we got the following result: the best training loss is 0.177; ERR is 0.0945. In our opinion, this is enough to get a good vector representation of the speaker.

To make sure that we are doing everything right, we trained AutoVC. Then we took the encoder weights and began to train our version. Server machine configuration: Nvidia 1080 Ti, Intel i7-7700K, SSD.

The results are very close. But if we talk about the time of training, then the AutoVC [34] trained for about 1 month, while our network took 3 weeks to reach a plateau. The tricks, as described above, this is a pre-trained encoder and scheduler for learning rate.

TABLE II
Comparison of AutoVC and Our Model

Framework	Reconstruction loss	MCD	Forward time
AutoVC	0.0087	3.2494	0.0617
Our	0.0097	3.5020	0.0554

Table II shows a comparison of a speaker from a test dataset on a trained AutoVC and our model. As we can see, our implementation turned out to be a little noisier and also does not have such excellent sharpness. Unfortunately, such small details play an essential role in the sound. If you look at the original, we see a clear structure that our model does not hold. But, we won at speed.

VII. Conclusions

This research aimed to identify capable model architecture compare to AutoVC. Based on an MCD, reconstruction loss, and speed analysis of our implementation and AutoVC, it can be concluded that replacing LSTM with a convolution layer can lead us to better performance while still holding excellent speaker characteristics which are essential factors to consider when designing and implementing voice-conversion module. The results indicate that potential usage and experiments can help us to research useful layers that are more receptive to sound portraying a small and compelling architecture.

References

[1] I. Makarov, V. Aliev, and O. Gerasimova, "Semi-dense depth interpolation using deep convolutional neural networks," in *Proceedings of the 2017 ACM on Multimedia Conference*, ACM. NY, USA: ACM, 2017, pp. 1407–1415.

[2] I. Makarov, V. Aliev, O. Gerasimova, and P. Polyakov, "Depth map interpolation using perceptual loss," in *Mixed and Augmented Reality (ISMAR-Adjunct), 2017 IEEE International Symposium on*, IEEE. NY, USA: IEEE, 2017, pp. 93–94.

[3] I. Makarov, A. Korinevskaya, and V. Aliev, "Fast semi-dense depth map estimation," in *Proceedings of the 2018 ACM Workshop on Multimedia for Real Estate Tech*. ACM, 2018, pp. 18–21.

[4] A. Korinevskaya and I. Makarov, "Fast depth map super-resolution using deep neural network," in *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. IEEE, 2018, pp. 117–122.

[5] I. Makarov, D. Maslov, O. Gerasimova, V. Aliev, A. Korinevskaya, U. Sharma, and H. Wang, "On reproducing semi-dense depth map reconstruction using deep convolutional neural networks with perceptual loss," in *Proceedings of the 27th ACM International Conference on Multimedia*. ACM, 2019, pp. 1080–1084.

[6] D. Maslov and I. Makarov, "Online supervised attention-based recurrent depth estimation from monocular video," *PeerJ Computer Science*, vol. 6, p. e317, 2020.

[7] ——, "Fast depth reconstruction using deep convolutional neural networks," in *International Work-Conference on Artificial Neural Networks*. Springer, 2021, pp. 456–467.

[8] P. Zolnikov, M. Zubov, N. Nikitinsky, and I. Makarov, "Efficient algorithms for constructing multiplex networks embedding," in *Proceedings of EEML conference*, 2019.

[9] I. Makarov, D. Kiselev, N. Nikitinsky, and L. Subelj, "Survey on graph embeddings and their applications to machine learning problems on graphs," *PeerJ Computer Science*, 2021.

[10] I. Makarov, M. Makarov, and D. Kiselev, "Fusion of text and graph information for machine learning problems on networks," *PeerJ Computer Science*, vol. 7, 2021.

[11] I. Makarov, K. Korovina, and D. Kiselev, "Jonnee: Joint network nodes and edges embedding," *IEEE Access*, pp. 1–15, 2021.

[12] M. Ilya, T. Mikhail, and T. Lada, "Imitation of human behavior in 3d-shooter game," *AIST'2015 Analysis of Images, Social Networks and Texts*, p. 64, 2015.

[13] I. Makarov, M. Tokmakov, P. Polyakov, P. Zyuzin, M. Martynov, O. Konoplya, G. Kuznetsov, I. Guschenko-Cheverda, M. Uriev, I. Mokeev *et al.*, "First-person shooter game for virtual reality headset with advanced multi-agent intelligent system," in *Proceedings of the 24th ACM international conference on Multimedia*. ACM, 2016, pp. 735–736.

[14] I. Makarov, Z. Peter, P. Pavel, T. Mikhail, G. Olga, G.-C. Ivan, and U. Maxim, "Modelling human-like behavior through reward-based approach in a first-person shooter game," in *Proceedings of the Third Workshop on Experimental Economics and Machine Learning (EEML 2016), Moscow, Russia, July 18, 2016*. CEUR Workshop Proceedings, 2016, pp. 24–33.

[15] I. Makarov and P. Polyakov, "Smoothing voronoi-based path with minimized length and visibility using composite bezier curves." in *AIST (Supplement)*, 2016, pp. 191–202.

[16] I. Makarov, A. Kashin, and A. Korinevskaya, "Learning to play pong video game via deep reinforcement learning: Tweaking deep q-networks versus episodic control," in *Supp. Proc. of International Conference on Analysis of Images, Social Networks and Texts (AIST-SUP'17), Moscow, Russia, July 27-29, 2017*. CEUR-WS. org, 2017, pp. 1–6.

[17] I. Makarov, D. Savostyanov, B. Litvyakov, and D. I. Ignatov, "Predicting winning team and probabilistic ratings in "dota 2" and "counter-strike: Global offensive" video games," in *International Conference on Analysis of Images, Social Networks and Texts*. Springer, 2017, pp. 183–196.

[18] D. Akimov and I. Makarov, "Deep reinforcement learning with vizdoom first-person shooter," in *CEUR Workshop Proceedings*, vol. 2479, 2019, pp. 3–17.

[19] I. Kamaldinov and I. Makarov, "Deep reinforcement learning in match-3 game," in *2019 IEEE Conference on Games (CoG)*. IEEE, 2019, pp. 1–4.

[20] M. Bakhanova and I. Makarov, "Deep reinforcement learning in vizdoom via dqn and actor-critic agents," in *International Work-Conference on Artificial Neural Networks*. Springer, 2021, pp. 138–150.

[21] I. Makarov, O. Bulanov, and L. E. Zhukov, "Co-author recommender system," in *International Conference on Network Analysis*. Springer, 2016, pp. 251–257.

[22] M. K. Rustem, I. Makarov, and L. E. Zhukov, "Predicting psychology attributes of a social network user," in *Proceedings of the Fourth Workshop on Experimental Economics and Machine Learning (EEML 2017), Dresden, Germany, September 17-18, 2017*. CEUR Workshop Proceedings, 2017, pp. 1–7.

[23] I. Makarov, O. Bulanov, O. Gerasimova, N. Meshcheryakova, I. Karpov, and L. E. Zhukov, "Scientific matchmaker: Collaborator recommender system," in *International Conference on Analysis of Images, Social Networks and Texts*. Springer, 2017, pp. 404–410.

[24] I. Makarov, O. Gerasimova, P. Sulimov, and L. E. Zhukov, "Recommending co-authorship via network embeddings and feature engineering: The case of national research university higher school of economics," in *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries*. ACM, 2018, pp. 365–366.

[25] I. Makarov, O. Gerasimova, P. Sulimov, K. Korovina, and L. E. Zhukov, "Joint node-edge network embedding for link prediction," in *International Conference on Analysis of Images, Social Networks and Texts*. Springer, 2018, pp. 20–31.

[26] I. Makarov, O. Gerasimova, P. Sulimov, and L. E. Zhukov, "Co-authorship network embedding and recommending collaborators via network embedding," in *International Conference on Analysis of Images, Social Networks and Texts*. Springer, 2018, pp. 32–38.

[27] ——, "Dual network embedding for representing research interests in the link prediction problem on co-authorship networks," *PeerJ Computer Science*, vol. 5, p. e172, 2019.

[28] I. Makarov and O. Gerasimova, "Predicting collaborations in co-authorship network," in *2019 14th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP)*. IEEE, 2019, pp. 1–6.

[29] ——, "Link prediction regression for weighted co-authorship networks," in *International Work-Conference on Artificial Neural Networks*. Springer, 2019, pp. 667–677.

[30] P. L. Tobing, Y.-C. Wu, T. Hayashi, K. Kobayashi, and T. Toda, "Non-parallel voice conversion with cyclic variational autoencoder," *arXiv preprint arXiv:1907.10185*, 2019.

[31] H. Kameoka, T. Kaneko, K. Tanaka, and N. Hojo, "Stargan-vc: Non-parallel many-to-many voice conversion using star generative adversarial networks," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 266–273.

[32] T. Kaneko, H. Kameoka, K. Tanaka, and N. Hojo, "Stargan-vc2: Rethinking conditional methods for stargan-based voice conversion," *arXiv preprint arXiv:1907.12279*, 2019.

[33] W.-C. Huang, H. Luo, H.-T. Hwang, C.-C. Lo, Y.-H. Peng, Y. Tsao, and H.-M. Wang, "Unsupervised representation disentanglement using cross domain features and adversarial learning in variational autoencoder based voice conversion," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 4, pp. 468–479, 2020.

[34] K. Qian, Y. Zhang, S. Chang, X. Yang, and M. Hasegawa-Johnson, "Autovc: Zero-shot voice style transfer with only autoencoder loss," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5210–5219.

[35] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[36] K. Qian, Y. Zhang, S. Chang, M. Hasegawa-Johnson, and D. Cox, "Unsupervised speech decomposition via triple information bottleneck," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7836–7846.

[37] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8. Citeseer, 2015, pp. 18–25.

[38] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," *arXiv preprint arXiv:1912.06670*, 2019.

[39] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, N. L. Dahlgren *et al.*, "Darpa timit acoustic-phonetic continuous speech corpus cd-rom {TIMIT}," 1993. [Online]. Available: https://catalog.ldc.upenn.edu/LDC93S1

[40] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," 2018. [Online]. Available: http://www.robots.ox.ac.uk/~vgg/data/voxceleb/vox2.html

[41] C. Veaux, J. Yamagishi, K. MacDonald *et al.*, "Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit," *University of Edinburgh. The Centre for Speech Technology Research (CSTR)*, 2017.

[42] A. Sehili, "Audio activity detection," 2021. [Online]. Available: https://github.com/amsehili/auditok

[43] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4779–4783.

[44] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized end-to-end loss for speaker verification," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4879–4883.

[45] B. Zhai, T. Gao, F. Xue, D. Rothchild, B. Wu, J. E. Gonzalez, and K. Keutzer, "Squeezewave: Extremely lightweight vocoders for on-device speech synthesis," *arXiv preprint arXiv:2001.05685*, 2020.

[46] HarryVolek, "Pytorch speaker verification," 2017. [Online]. Available: https://github.com/HarryVolek/PyTorch_Speaker_Verification