

SingIt! Singer Voice Transformation

Amit Eliav
*Faculty of Engineering
 Bar-Ilan University
 Ramat-Gan, Israel
 amiteli@biu.ac.il*

Aaron Taub
*Faculty of Engineering
 Bar-Ilan University
 Ramat-Gan, Israel
 aarontaub92@gmail.com*

Renana OPOCHINSKY
*Faculty of Engineering
 Bar-Ilan University
 Ramat-Gan, Israel
 renana.klainman@biu.ac.il*

Sharon Gannot
*Faculty of Engineering
 Bar-Ilan University
 Ramat-Gan, Israel
 sharon.gannot@biu.ac.il*

Abstract— In this paper, we propose a model which can generate a singing voice from normal speech utterance by harnessing zero-shot, many-to-many style transfer learning. Our goal is to give anyone the opportunity to sing any song in a timely manner. We present a system comprising several available blocks, as well as a modified auto-encoder, and show how this highly-complex challenge can be achieved by tailoring rather simple solutions together. We demonstrate the applicability of the proposed system using a group of 25 non-expert listeners. Samples of the data generated from our model are provided.

I. INTRODUCTION

With the latest explosion in the field of generative neural networks, this paper focuses on the “musical capabilities” of deep networks. Machine learning algorithms have already impacted the world of music, mainly in the form of musical arrangements, instrumental backings, sound effects, and sound processing [1], [2]. Here we address the task of speech-to-singing transfer. We aim at a system that can enable us to hear what it would sound like if any given person was to sing any given song, or at least could be fun for those who have no musical talent at all, yet have a strong passion for singing. There are currently some other existing solutions attempting

the singers in the database. Finally, AutoVC: Zero-shot voice style transfer [6] uses an autoencoder architecture model to transfer between the voice signals of two speakers. Although this paper does not mention the use of the network for singing or music, we found that the architectural solution presented in this paper for style conversion of speech is very suitable for the approach we have chosen to adopt, addressing the problem of singing voice transfer. Our work can be regarded as an extension of the work in [6] to the speech-to-singing task. To our best knowledge, such an extension cannot be found in the literature.

Our task and a schematic view of the system can be seen in Fig. 1, where the two input voices, the style transfer block, and the output signal are indicated. Since we aim at a system that can be used in practice, we should be able to transfer the voice of an unknown person. Moreover, the system should execute the style transfer in a timely manner.

II. PROBLEM STATEMENT

We will now rephrase the addressed problem in more specific terms. In this paper, refer to Person A and Person B. Person A refers to the singer in the audio file, where the input consists of singer Person A’s vocals along with the instrumental accompaniment. On the other hand, Person B refers to the *user* whose input is solely speech, unrelated to the content of Person A’s song. The output of our system is the content of the song sung by Person A but in the style of Person B. In other words, it generates the impression of Person A’s song as if it were sung by Person B. More practically, we aim to apply the style transfer based only on the user’s speech. This in itself has many interesting applications, in movies and other media, and more.

III. OUR MODEL

To address the challenge of converting an individual’s style to a given song using only examples of speech, we had to deal with multiple sub-problems: 1) Vocal extraction: extracting the vocal channel from a song mix, 2) Speaker representation: taking one’s voice and describing it in a low-dimensional form, 3) Content embedding: Taking a song to a lower dimension where the singer’s identity is no longer present, and 4) Style transfer: in the song’s lower dimension representation, switching between the singer’s style and the speaker’s style. We tried to solve each sub-problem separately

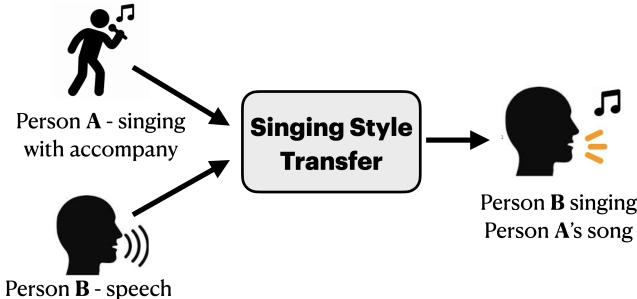


Fig. 1: High level system overview.

to solve similar problems, one of them being the Neural Parametric Singing Synthesizer (NPSS) [3], which has trained a model to learn how a specific singer sounds, and given a new song it produces the input song performed by the singer the network has trained on. There are other existing models for singing synthesis [4] and singing from speech such as WGAN-Sing [5] which presents a deep neural network-based singing voice synthesizer. This network trains on a set of singers and has the ability to exchange styles and content but only within

based on existing solutions and then combine them into one system.

The model is based on an autoencoder with self-reconstruction losses. That means that the model is not restricted to using parallel data which has both the speaker’s speech and the speaker’s singing and allows us to use existing datasets. A detailed description of the proposed system is depicted in Fig. 2. We will now elaborate on each of these building blocks.

A. System Components

Vocal extraction: For the vocal separator, we used ‘Spleeter’ [7], [8], a Python package that receives a song (mix of vocals and instruments) as an input and can separate the song into different stems (channels). In our case, we are separating each song into two stems: ‘vocal’ and ‘instrumental’.

Speaker representation: We chose to use ‘Resemblyzer’,¹ [9] a deep learning network that derives a high-level representation of a voice in the form of a 256-sized embedding vector. It should represent only the identity/style of the speaker regardless of the speech content and should summarize the characteristics of the given voice.

Content embedding: We chose to use an encoder block with an architecture that follows the encoder block from [6]. The content embedding model is a key part of our system, the bottleneck size and the input size were tested with many different values and the results were significantly different from one to another.

The Short-time Fourier Transform (STFT) log-spectrogram was preferred rather than the mel-spectrogram which is commonly used in many other speaker-to-speaker methods presented by other researchers. The reason is that the mel-scale tends to emphasize the speech-related frequencies, but in our case, singing voice, the log-spectrogram performed better.

The input to the encoder is a concatenation of the following two: 1) A log-spectrogram of Person A’s song, with a shape of $256 \times T$, which are the number of Time-Frequency (TF) bins in the spectrogram, and 2) A 256-sized embedding vector, representing the target style. This embedding vector is replicated and concatenated to each time bin of the spectrogram. This result with an input of the size $512 \times T$.

Following [6], the encoder itself consists of 3 sets of a 1-D convolution layer, a batch normalization, and a Relu activation function, which is then followed by a Bidirectional Long Short-Term Memory (BLSTM) layer. The output of these layers is then down-sampled by a factor of 32 and considered as the encoder’s bottleneck. We experimented with the downsampling factor to achieve the best results.

Style transfer: The style transfer is carried out in the decoder block, which also follows the decoder block architecture presented in [6]. The decoder is responsible for reversing the process of the encoder.

The input to the decoder is a concatenation of two inputs, namely the encoder output, and the 256-sized embedding vector, representing the target style. This input passes a BLSTM layer followed by 3 blocks, as in the encoder, namely a 1-D convolution layer, a batch normalization, and a ReLu activation function. Finally, we apply a second BLSTM layer and a linear projection layer, which results in a spectrogram with the same TF output size as the encoder’s input spectrogram. This output is a converted spectrogram. To complete the process, all that is needed is to convert the spectrogram back to a waveform, as explained in Sec. III-C.

Postnet Similar to the model described in [6], [10], we also apply another Convolutional Neural Networks (CNN)-based model to further enhance the quality of the spectrogram generated by the decoder.

B. Loss Functions

We denote the log-spectrogram input as X , the encoder, decoder, and Postnet as $E(\cdot)$, $D(\cdot)$, $P(\cdot)$, respectively. We also define codes = $E(X)$, $\hat{X} = D(\text{codes})$, and $\tilde{X} = P(\hat{X})$. Three loss functions were used to train the model.

Loss 1: An MSE loss calculated between X and the decoder output \hat{X} :

$$L_1 = \text{MSE}(X, \hat{X}) \quad (1)$$

Loss 2: An MSE loss, calculated between X and the Postnet model output \tilde{X} :

$$L_2 = \text{MSE}(X, \tilde{X}) \quad (2)$$

Loss 3: An ℓ_1 loss, calculated between the output of the encoder with the input data and the output of the encoder with the input taken from Postnet model output:

$$L_3 = \ell_1(E(X), E(\tilde{X})) \quad (3)$$

The **total loss** is given by

$$L_{\text{Total}} = L_1 + L_2 + \lambda \cdot L_3 \quad (4)$$

which is trained with $\lambda = 10000$.

C. Vocoder: The Griffin-Lim Method

The output of the decoder is a spectrogram. In order to convert the signal back to the time domain, the phase of the signal is required. As the processed spectrogram is totally different from the input spectrogram, there is no sense in using the original phase, as commonly done in enhancement tasks.

In order to reconstruct the time signal, we used the Griffin-Lim algorithm [11]. The gist of this algorithm is to use an initial phase for the signal, whether the original phase, zero-phase, or random phase, and then apply an iterative update until convergence. An alternative is to use a deep learning-based algorithm, e.g., Hifi-GAN [12], Mel-GAN [13] or other similar solutions [14]. In our tests, we found that the results using the Griffin-Lim algorithm were just as good as the ones using deep-learning methods. We, therefore, preferred this solution due to its relatively low computational resource requirements.

¹<https://github.com/resemble-ai/Resemblyzer>

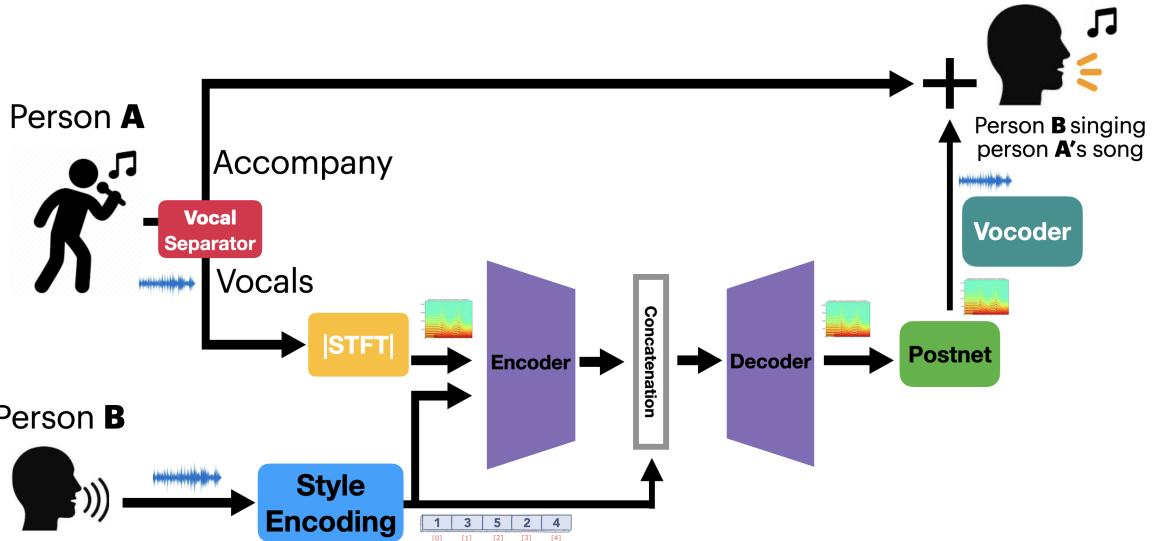


Fig. 2: Detailed Solution Architecture.

IV. DATASETS

A. Existing Datasets

The corpus used to train our model is NUS-HLT Speak-Sing (NHSS) [15], a database of parallel recordings of speech and singing. The audio recordings in the NHSS database correspond to a total of 100 songs sung and spoken by 10 singers, 5 male, and 5 female, resulting in a total of 7 hours of audio data. Although this dataset is presented as a parallel dataset, we did not use this property in our training but rather arbitrarily introduced both singing and speaking audio to the model. Another corpus used for our model is the Voice Cloning Toolkit (VCTK) Corpus [16]. It includes speech recordings by 109 native English speakers. We use this dataset in inference time as a pool of unseen speakers.

B. New Dataset: ‘Songlist’

We created our own dataset and handpicked over 600 songs performed by the top music artists of the past century. We then used ‘Spleeter’ [7], [8], which, as explained above, is a tool to separate the song into ‘vocal’ and ‘instrumental’ channels. The vocal channel will be used for the speech-to-singing conversion and later will be re-mixed with the ‘instrumental’ channel. The songs mainly contain only a single singer, so the model could make a conversion properly between a single speaker and a single singer.

V. EXPERIMENTAL STUDY

A. Training Process

In order to avoid the need for a parallel dataset (speech and singing of the same speaker), the model is trained as an autoencoder model with self-reconstruction loss. For the train set, we chose to use only the NHSS dataset, which combines both speech and singing audio. During training, we randomly choose an utterance (either speaking or singing) and train only the autoencoder block. This utterance, during training, is

simultaneously treated as both the style and the content. We do not make use of the parallelism in the dataset but rather provide both the speaking and singing examples during train time. This will, hopefully, make the model familiar with both and encourage it to generate a singing-like output at inference time. It is important to note, that during the training stage, the concatenated embedding vector (to both the encoder and decoder) and the input spectrogram are of the same, randomly chosen, person.

In the test stage of our model, when performing a style transfer, the embedding vector, concatenated to the inputs of the encoder and decoder, is of person B, the speaker, while the input spectrogram is of person A, the singing person. We stress that, at inference time, we are not limited by the type or pool of inputs, for both the songs and the speakers. For the results we present here, we used a trained model only over the NHSS dataset but took a style and content from the VCTK, NHSS datasets, and our newly created dataset, the ‘Songlist’.

B. Results

One of the challenges we faced with the evaluation of our results was the fact that we have not seen any published work which attempts to perform the same task, namely applying the style of an unknown speaker to any given song.

We, therefore, resort to a subjective evaluation, based on non-intrusive quality measures. For the evaluation, we used a group of 25 random listeners, both male and female between the ages of 18-60, who are not experts in music or sound generation. Most of the participants are friends or family, or a second circle to them. During the listening test, the participants listened to two audio clips: first, “Audio A - Content” and second, “Audio B - Style”. Lastly, they listened to the output of our network: “Audio C” an audio clip with the content from “Audio A - Content” in the style of “Audio B - Style”. Then they were asked to answer the following questions:

- 1) From ‘Audio C’ - how clear were the words to you?

- 2) From ‘Audio C’ - how much does the melody sound like the melody in ‘Audio A’?
- 3) How much did ‘Audio C’ sound like it was sung in the style of the person in the ‘Audio A’?
- 4) How much did ‘Audio C’ sound like it was sung in the style of the person in ‘Audio B’?
- 5) How life-like did ‘Audio C’ sound (does it sound like a real human being)?

TABLE I: Listening test survey results. ↓ designate the lower is better, and ↑ the higher is better.

Question	Result
1 ↑	3.55±0.386
2 ↑	3.87±0.356
3 ↓	3.06±0.448
4 ↑	3.17±0.404
5 ↑	2.39±0.413

The results are presented in Table I. We report on the listening test survey, displayed on a 1–5 scale with their respective 95% confidence intervals.

In general, we observe that the main goal was accomplished. We were able to synthesize a result that sounds closer to the style of the target person (3.17) than to the style of the original person (3.06), namely, it sounds more like Person B than Person A. Moreover, the quality of the sound produced was good enough for listeners to clearly hear the words and to perceive that the melody has not changed much throughout the entire process. That being said, the style change attempted is probably not distinctive enough, especially on a singular test, when the listener is not comparing results to other existing networks, but to their ground truths.

Another aspect that should be improved is the “livelihood” of the result. What we have seen throughout our work is really how sensitive the human sense of listening is, and how even a good output (both in terms of melody and words), which does sound similar to the target, can still be far from persuading that this is actually a human singing. Some samples of the results from our style transfer model can be found online.²

VI. SUMMARY

In this paper, we have presented a new way of generating singing from one’s voice while only being familiar with the user’s speech. Current solutions deal with the complexity of this problem by requiring a singing utterance from the user. Hence they are only addressing the problem of generating a singing voice from another singing voice or the problem of generating only specific singers or songs that have been learned by the network. Our proposed method involves separating the vocals, then using an encoder and a decoder in the frequency domain and reconstructing the signal using a Griffin-Lim decoder. The results presented show how we managed to transfer the style from a given speech and apply

²<https://thesignitproject-singit-streamlit-streamlit-code-zgx4p0.streamlit.app/>

it to a song. Our results show alignment with the original song both in pitch and tempo. These results still have much to improve in the “livelihood” aspect of the output audio, as well as the capability to succeed with the style transfer in more complex cases, e.g., more singers, and inferior quality input. The implementation of this work can be relevant in a wide spectrum of fields from art to science, to health and pleasure, and will hopefully invite much more work to be done in this specific field.

REFERENCES

- [1] B. L. Sturm, O. Ben-Tal, Ú. Monaghan, N. Collins, D. Herremans, E. Chew, G. Hadjeres, E. Deruty, and F. Pachet, “Machine learning research that matters for music creation: A case study,” *Journal of New Music Research*, vol. 48, no. 1, pp. 36–55, 2019.
- [2] R. Fiebrink and B. Caramiaux, “The machine learning algorithm as creative musical tool,” in *The Oxford handbook of algorithmic music*, A. Mclean and R. T. Dean, Eds. Oxford University Press, 2018, pp. 181–208.
- [3] M. Blaauw and J. Bonada, “A neural parametric singing synthesizer modeling timbre and expression from natural songs,” *Applied Sciences*, vol. 7, no. 12, p. 1313, 2017.
- [4] X. Zhuang, T. Jiang, S.-Y. Chou, B. Wu, P. Hu, and S. Lui, “Litesing: Towards fast, lightweight and expressive singing voice synthesis,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 7078–7082.
- [5] P. Chandna, M. Blaauw, J. Bonada, and E. Gomez, “WGANSing: A multi-voice singing voice synthesizer based on the wasserstein-GAN,” in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, sep 2019. [Online]. Available: <https://doi.org/10.23919/Eusipco.2019.8903099>
- [6] K. Qian, Y. Zhang, S. Chang, X. Yang, and M. Hasegawa-Johnson, “AutoVC: Zero-shot voice style transfer with only autoencoder loss,” in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019, pp. 5210–5219.
- [7] R. Hennequin, A. Khelif, F. Voituret, and M. Moussallam, “Spleeter: a fast and efficient music source separation tool with pre-trained models,” *Journal of Open Source Software*, vol. 5, 2020.
- [8] L. Ansal and C. Ancy, “Research on DNN methods in music source separation tools with emphasis to Spleeter,” *International Research Journal on Advanced Science Hub*, vol. 3, no. Special Issue 6S, pp. 24–28, 2021.
- [9] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, “Generalized end-to-end loss for speaker verification,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4879–4883.
- [10] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, R. A. Saurous, Y. Agiomvrgianakis, and Y. Wu, “Natural TTS synthesis by conditioning Wavenet on mel spectrogram predictions,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4779–4783.
- [11] N. Perraudin, P. Balazs, and P. L. Søndergaard, “A fast Griffin-Lim algorithm,” in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2013.
- [12] J. Kong, J. Kim, and J. Bae, “Hifi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 17022–17033, 2020.
- [13] L. Sheng, D.-Y. Huang, and E. N. Pavlovskiy, “High-quality speech synthesis using super-resolution mel-spectrogram,” *arXiv preprint arXiv:1912.01167*, 2019.
- [14] L. Wyse, “Audio spectrogram representations for processing with convolutional neural networks,” in *Proceedings of the First International Conference on Deep Learning and Music*, 2017, pp. 37–41.
- [15] B. Sharma, X. Gao, K. Vijayan, X. Tian, and H. Li, “NHSS: A Speech and Singing Parallel Database,” *Speech Commun.*, vol. 133, no. C, pp. 9–22, Oct. 2021.
- [16] M. K. Veaux Christophe, Yamagishi Junichi, “CSTR VCTK corpus: English multi-speaker corpus for CSTR voice cloning toolkit,” University of Edinburgh. The Centre for Speech Technology Research (CSTR), 2017.