STREAMING VOICE CONVERSION THROUGH CHUNK-WISE TRAINING AND LOOKAHEAD LOSS

Baotong Tian
University of Rochester
baotong.tian@rochester.edu

ABSTRACT

Voice conversion systems have recently achieved impressive results in non-streaming scenarios, but their application to real-time settings remains challenging. This paper explores different training schemes for streaming voice conversion, with a particular focus on chunk-wise processing and temporal consistency. We demonstrate that training on chunked audio input can improve the performance in streaming scenarios compared to training on complete utterances. We also analyze the impact of various inference parameters, including chunk size and overlap ratio, on system performance.

1. INTRODUCTION

Voice conversion (VC), which aims to transform the speaker identity of a source utterance while preserving its linguistic content, has witnessed remarkable progress in recent years. Existing VC systems primarily focus on converting the timbre and style of the source speaker to those of a target speaker [1]. While non-streaming voice conversion approaches have achieved impressive performance in both speech quality and speaker similarity [2,3], with some systems achieving fast inference speeds in offline scenarios [4–6], they are insufficient for emerging real-time applications.

The growing demand for real-time voice conversion in applications such as live broadcasting, video conferencing, and real-time communication systems (RTC) necessitates streaming solutions. This need is further amplified by the increasing importance of voice anonymization for protecting speaker identity against potential attacks. However, simply applying non-streaming voice conversion systems trained on whole sentences to streaming scenarios leads to significant performance degradation, resulting in poor audio quality, noticeable artifacts, and lack of coherency between output chunks, but most of the works still focus on training whole utterances. Streaming voice conversion systems, which process speech in a chunk-wise manner with minimal latency, face unique challenges, particularly at the onset where the model has limited context about the linguistic content. These systems must maintain temporal consistency across variable-length chunks while ensuring seamless transitions, all while operating under strict latency constraints. Current methods often struggle to match the quality of non-streaming systems, especially in maintaining speaker consistency and speech naturalness across chunk boundaries. These challenges highlight the need for specialized training schemes and loss functions that explicitly account for the streaming nature of the task.

We conduct comparative analyses of different training and inference settings, including the adaptation of the lookahead loss proposed in [7]. Our experiments demonstrate that it's necessary to adjust training schemes for this task to preserve the continuity and the quality of the converted speech.

The rest of the paper is organized as follows. We start by reviewing related works on streaming voice conversion in Section 2. In Section 3, we outline the non-streaming framework we adopted. In Section 4, we introduce the dataset, training settings, metrics and present our experimental results to evaluate our algorithm. Finally, we conclude in Section 5 and offer directions and insight for future work.

2. RELATED WORKS

Voice conversion systems have predominantly adopted an encoder-decoder architecture, where content information is obtained through various means. One approach utilizes Automatic Speech Recognition (ASR) systems to obtain features such as Phonetic Posteriorgrams (PPG) [8, 9] and Intermediate Bottleneck Features (IBF) [10] which are extracted from the last layer and the intermediate layers of the ASR Encoder respectively. Another approach leverages self-supervised representations like HUBERT [4, 6]. The timbre information is extract by a speaker encoder, which is either trained during the process or directly use a pre-trained model. The decoder typically follows either a Codec architecture, such as SoundStream [11] and Encodec [12], or employs vocoders like Hifi-GAN [13] to convert the latent representations directly to audio or spectrograms [7, 14].

Streaming voice conversion presents unique challenges due to the absence of future audio information, which can impact the coherence and quality of the generated output. Previous works have addressed this problem in different ways. Chen et al. [10] and Ning et al. [15, 16] applied teacher guidance to the system. Specifically, Chen et al. [10] employed a non-streaming teacher to generate the parallel data so that the student can learn to convert voice rather than reconstruct the original source audio during training, and Ning et al. [15,16] developed a dual-mode system that performs both non-streaming and streaming

voice conversion, with the non-streaming mode providing representational guidance to the streaming mode. They also proposed the Hybrid Predictive Coding (HPC) method which combines 2 losses from CPC [17] and Autoregressive Predictive Coding (APC) [18] to predict the future information through unsupervised representation learning. More recent approaches include Wang et al.'s [19] use of context-aware language models for teacher-guided context foresight, and Ning et al.'s [20] implementation of language models to generate pseudo-context, effectively increasing the input chunk size and reducing the system's dependence on large chunk sizes.

3. METHOD

Our system builds upon NeuralVC [6], a fast inference non-streaming voice conversion framework based on VITS [21]. We adapted this architecture for streaming scenarios by exploring various training schemes and implementing a 2-frame lookahead loss to maintain chunk consistency through better utilization of historical information.

3.1 NeuralVC System

The NeuralVC architecture consists of three main components: a prior encoder, a speaker encoder, and a neural decoder. The prior encoder consists of HuBERT [22], a content encoder, and a normalization flow. The HuBERT module takes audio as input and extracts 256-dimension Hubert features, which are the information of the speech content. Then the content encoder learns the prior distribution $\mathcal{N}(z'; \mu_\theta, \sigma_\theta^2)$. The Flow model is conditioned on the speaker embedding g and is set to output a more complex distribution to enhance the complexity the model learns. Another benefit of Flow is that it is revertible, meaning that we can compute the loss on the content encoder output side. After the prior encoder module, we can get a compact linguistic feature z. We use a pre-trained speaker encoder which is a speaker verification model trained on a large amount of speakers. We can directly extract 256-dimensional speaker embedding by feeding the source audio to the speaker encoder. The neural decoder takes the input of the linguistic feature z and the speaker embedding g and directly generates the waveform rather than generating the spectrogram and using vocoders such as Hifi-GAN [13] to convert the spectrogram to audio. The decoder is based on the SEANet architecture from Encodec [12], enhanced with two additional one-dimensional convolutional layers to better process the combined speaker and content latent representations. Their experiment shows that this design improves speaker similarity and increases inference speed, while performing better than Hifi-GAN and MS-iSTFT [4].

3.2 NeuralVC Losses

The overall loss is denoted as:

$$\mathcal{L}_G = L_{rec} + L_{kl} + L_{adv}(G) + L_{fm}(G) + L_{spk}, \quad (1)$$

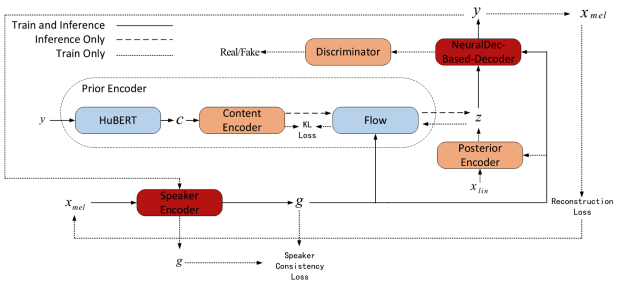

Figure 1. Training and inference procedure of NeuralVC from [6].

$$\mathcal{L}_D = L_{adv}(D), \quad (2)$$

where the reconstruction loss computes the L_1 loss between the predicted and target mel-spectrogram:

$$L_{recon} = \|x_{mel} - \hat{x}_{mel}\|_1; \quad (3)$$

the kl loss L_{kl} is the KL divergence between the prior distribution $p_{theta}(z|c) = N(z; \mu_\phi, \sigma_\phi^2)$ and the posterior distribution $q_\theta(z|x_{lin}) = N(z'; \mu_\theta, \sigma_\theta^2)|\det \frac{\partial z'}{\partial z}|$; the adversarial loss is the generator loss in the GAN formulation,

$$L_{adv}(G) = \mathbb{E}_{(z,g)}[(D(G(z,g)) - 1)^2], \quad (4)$$

$$L_{adv}(D) = \mathbb{E}_{(y,z,g)}[(D(y) - 1)^2 + (D(G(z,g)))^2], \quad (5)$$

where y is the ground truth waveform, z is the linguistic latent, g is the speaker latent. We refer D as the multi period discriminator and G as the neural decoder; the feature matching loss is proposed in [23], which is added as a reconstruction loss in the hidden layers to further improve the performance of adversarial training:

$$L_{fm}(G) = \mathbb{E}_{(y,z,g)}\left[\sum_{L=1}^{T} \frac{1}{N_l}\|D^l(y) - D^l(G(z,g))\|\right], \quad (6)$$

where T is the number of layers in the discriminator and D^l gives the features map of the l-th layer with N_l number of features. The speaker loss is the speaker consistency loss, which measures the L_1 distance of the speaker embedding of the generated speech and the input speech.

3.3 Lookahead Loss

We adopted the 2-frame lookahead loss computation from StreamVC [7]. In their work, they paired the output frame of the t-th time step o_t with the input frame of $(t-2)$-th time step s_{t-2} rather than the current input frame s_t. This design serves two purposes: it enables the model to better leverage historical information and provides additional time for accumulating sufficient linguistic content before generating the initial output. Specifically, the model begins output generation only after receiving the third chunk of input, effectively providing a two-chunk buffer period for processing.

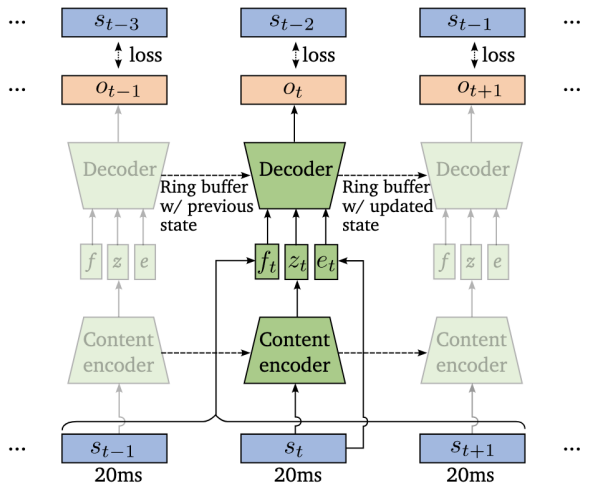

Figure 2. Lookahead loss from [6].

4. EXPERIMENTS

4.1 Dataset

We used VCTK [24] as the dataset for training and testing. It includes 44 hours of clean speech audio uttered by 110 English speakers with various accents. All the recordings are 16kHz sample rate. We follow the NeuralVC preprocessing step to divide the whole datasets. For each speaker, 2 of the utterances are used for validation, 10 of the utterances are used for testing, and the others are used for training.

4.2 Training Settings

We used the soft speech version of HuBERT [25] instead of the discrete version, which can learn more speaker independent information and preserve content information as much as possible because the space of speech sounds is not discrete, thus increase intelligibility. We replaced all the non-causal convolution blocks to causal ones, and also provided buffers for each convolutional layer in the framework for use in inference.

For training data, we prepared overlapped chunks of 3200 samples (200ms) with the overlap ratio of 25%. The overlapping part can give the model context information in chunk-wise inference. Since it takes a long time for the model to train from scratch, due to time limit of the project, we fine-tuned the model on the pre-trained NeuralVC model on the same dataset but with shorter segments to improve its performance in streaming scenarios. For the 2-frame lookahead loss, as they are using short chunks of 20ms, the loss is equivalent to 2-chunk lookahead loss.

During fine-tuning, we used the AdamW optimizer with the learning rate of 2^{-6}, $\beta_1 = 0.8$ and $\beta_2 = 0.99$, batch size of 20. We assigned the weight of the reconstruction loss to be 45 and other weights to be 1 following the NeuralVC configuration to focus more on the reconstruction part during training. For the chunk-wise output generation, given that the output dimension matches the input dimension, we implemented direct concatenation of previous outputs with the non-overlapping segments of the current output, which means that the overlapping regions serve solely to provide contextual information to the model without directly influencing the final output structure.

4.3 Metrics

We used 3 metrics to evaluate the performance. The first is Real-time Factor (RTF), which is a metric for measuring the speed of a speech processing system in the inference time. We followed the implementation of Quamer et al. [26]. The second one is speaker similarity, which is a metric to measure the speaker similarity between the target speech and the generated speech. We follow [4] and use Resemblyzer [1] voice encoder to get the speaker embedding and compare them. The third one is Word Error Rate (WER) which is originally designed for ASR tasks, we can also use it here to transcribe the generated speech to check the word error. We use jiwer [2] package to implement the WER computation.

4.4 Experiment Results

Table 1 shows the 3 different settings on the non-streaming scenario. We didn't train the model using chunks as input and adopted the lookahead loss to convergence. As there is no chunk-wise inference in the process, we can only compute the inference time instead of RTF. We can see that after fine-tuning on chunks, the real-time factor is comparable, but the speaker similarity and WER is slightly worse, due to the fact that it is seeing less context compared to training on the whole utterance. Table 2 shows the experimental result of streaming inference with the configuration of 3200 of chunk size, 25% of overlapping ratio (800 of chunk overlap) and using ring buffer for inference. We can see that the RTF for training on whole utterance is higher. The second row shows that training and inference on the same length of data can provide increase of performance on the speaker similarity of WER. The third training setting may have similar effects after training of convergence, but since we are fine-tuning the original model trained on whole utterance, the 2-frame difference may confuse the model. We also explored different inference settings including the chunk size, overlap ratio and buffer usage, which is shown in table 3. From the comparison we can see that using the buffer inside the model cam improve the WER, regardless of the overlap ratio, but when the overlap ratio decreases, the WER will slightly decrease. In addition, using the same inference setting as the training setting gives the best WER score, and when the segment size is smaller, it takes longer to inference, and the audio is more discontinuous, leading to low speaker similarity score and high WER.

We also show an example of the streaming voice conversion result shown in fig. 3, the first row is the result of whole utterance inference, the second row is the result of chunk-wise inference. The 3 columns from the left to the

[1] https://github.com/resemble-ai/Resemblyzer

[2] https://github.com/jitsi/jiwer

right is 3 different strategies on training on whole utterances, chunked audio, and chunked audio using lookahead loss.

Table 1. Non-streaming inference for different training settings.

Training Settings	Inference Time	Similarity	WER
Whole	0.1384	0.7839	0.0336
Chunk	0.1277	0.7502	0.133
Chunk+Lookahead	0.1342	0.7404	0.4344

Table 2. Streaming inference for different training settings.

Training Settings	RTF	Similarity	WER
Whole	0.4943	0.7371	0.6298
Chunk	0.3584	0.7644	0.2686
Chunk+Lookahead	0.3629	0.7253	0.6074

Table 3. Streaming performance with different inference settings.

Chunk Size, Overlap Ratio, Buffer	RTF	Similarity	WER
3200, 0, True	0.3644	0.7529	0.3719
3200, 0, False	0.178	0.7851	0.4269
3200, 25%, True	0.3584	0.7644	0.2686
3200, 25%, False	0.3492	0.7842	0.3553
3200, 12.5%, True	0.3282	0.7632	0.3072
320, 25%, True	2.706	0.5079	1.009

5. CONCLUSION

In this paper we show that the training scheme is important for streaming voice conversion. While models trained on complete utterances show degraded performance when applied to streaming scenarios, we found that fine-tuning on chunked audio input significantly improves their effectiveness. Future work will focus on maintaining intelligibility and temporal consistency when processing extremely short chunks (e.g., 20ms), which is crucial for real-time applications. This advancement would represent a significant step toward robust, real-time voice conversion systems.

6. REFERENCES

[1] B. Sisman, J. Yamagishi, S. King, and H. Li, "An overview of voice conversion and its challenges: From statistical modeling to deep learning," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 132–157, 2020.

[2] S. Zhao, H. Wang, T. H. Nguyen, and B. Ma, "Towards natural and controllable cross-lingual voice conversion based on neural tts model and phonetic posteriorgram," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5969–5973.

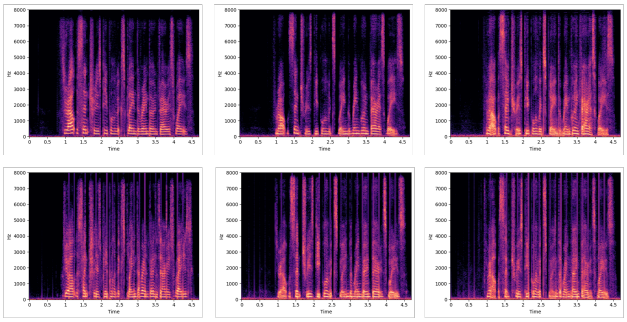

Figure 3. Conversion results.

[3] J. Li, W. Tu, and L. Xiao, "Freevc: Towards high-quality text-free one-shot voice conversion," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[4] H. Guo, C. Liu, C. T. Ishi, and H. Ishiguro, "Quickvc: Any-to-many voice conversion using inverse short-time fourier transform for faster conversion," *arXiv preprint arXiv:2302.08296*, 2023.

[5] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, M. Kudinov, and J. Wei, "Diffusion-based voice conversion with fast maximum likelihood sampling scheme," *arXiv preprint arXiv:2109.13821*, 2021.

[6] D. Cao, Z. Zhang, and J. Zhang, "Neuralvc: Any-to-any voice conversion using neural networks decoder for real-time voice conversion," *IEEE Signal Processing Letters*, 2024.

[7] Y. Yang, Y. Kartynnik, Y. Li, J. Tang, X. Li, G. Sung, and M. Grundmann, "Streamvc: Real-time low-latency voice conversion," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 11 016–11 020.

[8] X. Zhao, F. Liu, C. Song, Z. Wu, S. Kang, D. Tuo, and H. Meng, "Disentangling content and fine-grained prosody information via hybrid asr bottleneck features for voice conversion," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 7022–7026.

[9] S. Liu, Y. Cao, D. Wang, X. Wu, X. Liu, and H. Meng, "Any-to-many voice conversion with location-relative sequence-to-sequence modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1717–1728, 2021.

[10] Y. Chen, M. Tu, T. Li, X. Li, Q. Kong, J. Li, Z. Wang, Q. Tian, Y. Wang, and Y. Wang, "Streaming voice conversion via intermediate bottleneck features and non-streaming teacher guidance," in *ICASSP 2023-2023*

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2023, pp. 1–5.

[11] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "Soundstream: An end-to-end neural audio codec," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 495–507, 2021.

[12] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High fidelity neural audio compression," *arXiv preprint arXiv:2210.13438*, 2022.

[13] 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. 17 022–17 033, 2020.

[14] M. Chen and Z. Duan, "Controlvc: Zero-shot voice conversion with time-varying controls on pitch and speed," *arXiv preprint arXiv:2209.11866*, 2022.

[15] Z. Ning, Y. Jiang, P. Zhu, J. Yao, S. Wang, L. Xie, and M. Bi, "Dualvc: Dual-mode voice conversion using intra-model knowledge distillation and hybrid predictive coding," *arXiv preprint arXiv:2305.12425*, 2023.

[16] Z. Ning, Y. Jiang, P. Zhu, S. Wang, J. Yao, L. Xie, and M. Bi, "Dualvc 2: Dynamic masked convolution for unified streaming and non-streaming voice conversion," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 11 106–11 110.

[17] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[18] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. Glass, "An unsupervised autoregressive model for speech representation learning," *arXiv preprint arXiv:1904.03240*, 2019.

[19] Z. Wang, Y. Chen, X. Wang, Z. Chen, L. Xie, Y. Wang, and Y. Wang, "Streamvoice: Streamable context-aware language modeling for real-time zero-shot voice conversion," *arXiv preprint arXiv:2401.11053*, 2024.

[20] Z. Ning, S. Wang, P. Zhu, Z. Wang, J. Yao, L. Xie, and M. Bi, "Dualvc 3: Leveraging language model generated pseudo context for end-to-end low latency streaming voice conversion," *arXiv preprint arXiv:2406.07846*, 2024.

[21] J. Kim, J. Kong, and J. Son, "Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5530–5540.

[22] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 29, pp. 3451–3460, 2021.

[23] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *International conference on machine learning*. PMLR, 2016, pp. 1558–1566.

[24] a. C.Veaux, J.Yamagishi, "Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit (version 0.92)," *Centre Speech Technol. Res., Univ. Edinburgh*, 2019.

[25] B. Van Niekerk, M.-A. Carbonneau, J. Zaïdi, M. Baas, H. Seuté, and H. Kamper, "A comparison of discrete and soft speech units for improved voice conversion," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6562–6566.

[26] W. Quamer and R. Gutierrez-Osuna, "End-to-end streaming model for low-latency speech anonymization," *arXiv preprint arXiv:2406.09277*, 2024.