End-to-End Zero-Shot Voice Style Transfer with Location-Variable Convolutions

Wonjune Kang\(^1,2\)
\(\text{wjkang@mit.edu}\)

Deb Roy\(^1,2\)
\(\text{dkroy@mit.edu}\)

\(^1\)MIT Center for Constructive Communication \(^2\)MIT Media Lab

Abstract

Zero-shot voice conversion is becoming an increasingly popular research direction, as it promises the ability to transform speech to match the voice style of any speaker. However, little work has been done on end-to-end methods for this task, which are appealing because they remove the need for a separate vocoder to generate audio from intermediate features. In this work, we propose Location-Variable Convolution-based Voice Conversion (LVC-VC), a model for performing end-to-end zero-shot voice conversion that is based on a neural vocoder. LVC-VC utilizes carefully designed input features that have disentangled content and speaker style information, and the vocoder-like architecture learns to combine them to simultaneously perform voice conversion while synthesizing audio. To the best of our knowledge, LVC-VC is one of the first models to be proposed that can perform zero-shot voice conversion in an end-to-end manner, and it is the first to do so using a vocoder-like neural framework. Experiments show that our model achieves competitive or better voice style transfer performance compared to several baselines while maintaining the intelligibility of transformed speech much better.

1 Introduction

Voice conversion (VC) is the task of transforming a voice to sound like that of another person without changing the linguistic or prosodic content conveyed in the original speech [1]. It belongs to the general field of speech synthesis, which also includes text-to-speech (TTS), speech vocoding, and the changing of other speech properties such as emotion and accents. VC technologies have many practical applications, such as voice anonymization for privacy preservation [2], communication aids for the speech-impaired [3], and voice dubbing in movies and games. Therefore, it is becoming an increasingly popular research direction in the speech processing and machine learning community.

Advances in deep learning have had a significant impact on voice conversion research in recent years, allowing VC systems to achieve significant improvements in terms of voice quality and similarity to the target speaker, especially in the non-parallel data setting. Many recent approaches have framed the conversion task as a style transfer problem, leading to the alternate name of voice style transfer (VST). Some of these methods have used variants of generative adversarial networks (GANs) [4, 5, 6] or vector-quantized variational autoencoders (VQ-VAEs) [7, 8]. Others have leveraged the latent representations of automatic speech recognition (ASR) models to extract linguistic features from the source speech and decompose it into content and speaker information [9, 10].

Despite many improvements, the naturalness, intelligibility, and speaker similarity of transformed voices even for state-of-the-art VC models still lags behind that of true speech, especially when conversion is applied to new speakers that were previously unseen during training (i.e., in the zero-shot setting). Most zero-shot VC models trained on non-parallel data use a self-supervised learning strategy with a self-reconstruction loss. The idea is to decompose a given training utterance’s speaker and
content information into separate embeddings, and then recombine them using a decoder to reconstruct the original signal. At inference time, conversion can be performed by combining the content embedding from a source utterance with the speaker embedding from a target utterance. The hope is that the model will learn to combine content and speaker embeddings in a coherent way regardless of where each of them comes from. Given this, most models are composed of a content encoder, speaker encoder, and decoder [11, 12, 13, 14, 15]. Traditionally, the information decomposition and conversion are performed on intermediate representations such as mel spectrograms. This is followed by a vocoding step to convert these representations into time-domain audio.

However, little work has been done on developing models that can perform voice conversion and audio synthesis together in an end-to-end manner. End-to-end methods have become popular in deep learning due to their simplicity, elegance, and high performance [16, 17, 18, 19, 20]. In the context of voice conversion (and speech synthesis in general), they are particularly appealing because they do not require a separate vocoder to synthesize time-domain audio. VC methods that produce spectrograms and rely on vocoders to synthesize audio can have highly variable performance depending on the quality of the vocoder. Furthermore, high-quality vocoders can still generate poor quality audio if the spectrogram itself has flaws. VC models that generate spectrograms are also forced to define their training losses in the spectrum domain, which may not align with human perception of audio once converted to the time domain. Consequently, they are prone to producing audio that has artifacts or that sounds muffled due to oversmoothing in the spectrum.

In this work, we introduce **Location-Variable Convolution-based Voice Conversion (LVC-VC)**, a model for performing end-to-end zero-shot voice conversion that is based on the architecture of a neural vocoder. LVC-VC utilizes location-variable convolutions (LVCs) [21] in its generator to efficiently model the time-dependent features that arise in speech using a compact model size. Unlike many other VC models, LVC-VC does not contain a content encoder or decoder. Rather, it uses a set of carefully designed input features that already have disentangled content and speaker style information, and the vocoder-like architecture learns to combine these features to perform voice conversion while synthesizing audio. This significantly streamlines the model’s structure and removes the difficult task of training the model to perform proper disentangled representation learning.

To the best of our knowledge, LVC-VC is one of the first models for end-to-end zero-shot VC that has ever been proposed, and the first to tackle the problem using a vocoder-like neural framework. Our choice of input features, training strategy, and model architecture enable LVC-VC to strike a good balance between achieving good voice style transfer performance while maintaining the clarity and intelligibility of transformed speech. This contrasts with several other baseline VC models, which we find to exhibit a trade-off between audio quality/intelligibility and accurate voice style transfer.

## 2 Related Work

### 2.1 Voice conversion

One of the first zero-shot voice conversion methods was AutoVC [12], an autoencoder-based model that combined a pre-trained speaker encoder with dimensionality bottleneck layers to disentangle content and speaker information. AutoVC has served as the base model for a range of improvements, such as the addition of fundamental frequency (F0) information [13], mutual information-based disentangled representation learning [22], and adversarial voice style mixup [15]. Other approaches have used techniques such as adaptive instance normalization [11], activation function guidance [23], and information perturbation-based training [24] to disentangle the content and speaker information in speech. All of the aforementioned methods produce spectrograms, which necessitate that a separate vocoder be part of the overall system in order to synthesize time-domain audio.

Few models have been proposed that can perform end-to-end voice conversion; to the best of our knowledge, the only two models that can do this are Blow [25] and NVC-Net [14]. Blow is a normalizing flow network for non-parallel raw-audio voice conversion. However, it is not able to perform zero-shot conversion, and like many other flow-based networks, it has a very large number of parameters. NVC-Net is a GAN-based model that performs conversion directly on raw audio waveforms. Unlike LVC-VC, NVC-Net does not deviate significantly from the established VC framework, consisting of a speaker encoder, a content encoder, a decoder, and three discriminators.

---

1 Audio samples are available on our demo page: [https://lvc-vc.github.io/lvc-vc-demo/](https://lvc-vc.github.io/lvc-vc-demo/)
Conceptually, LVC-VC shares some similarities with NANSY [24], a neural analysis and synthesis framework that uses various input features along with information perturbation-based training to flexibly control various attributes of speech. NANSY demonstrated strong performance on zero-shot voice conversion, pitch shift, and time-scale modification. However, it uses intermediate representations from a pre-trained wav2vec 2.0 LARGE model [26] for its speaker and content features, which leads to the overall system having a very large number of parameters. In addition, NANSY relies on a vocoder to generate time-domain audio from the spectrograms that it produces. While LVC-VC is only able to modify vocal identity, it does so in an end-to-end manner, with a significantly smaller model footprint, and using input features that are much simpler to compute.

2.2 Location-variable convolutions (LVCs)

Deep generative models (neural vocoders) have achieved great success in speech synthesis [27, 28, 29, 30]. Many of them are implemented using a WaveNet-like generator network [27], in which mel spectrograms are used as conditioning features and dilated causal convolutions are applied to capture the long-term dependencies of a waveform. This necessitates a large number of convolution kernels to capture the many time-dependent features that arise in speech, since the same kernels must be used for all audio frames. However, in a traditional linear prediction vocoder [31], the coefficients for the all-pole linear filter vary depending on the conditioning acoustic features of the analysis frame. A network with similarly variable kernel coefficients depending on the conditioning features could be able to model long-term dependencies in waveforms more efficiently than fixed-kernel methods.

Inspired by this idea, [21] introduced location-variable convolutions (LVCs), in which different convolutional kernel weights are used to model different intervals in an input sequence depending on the corresponding “local” sections of a conditioning sequence, such as a mel spectrogram. To do this, LVCs utilize kernel predictor networks, which predict the weights of convolution kernels given a local conditioning sequence. Then, each interval of the input sequence has a different convolution operation performed on it depending on the temporally associated section of the conditioning sequence. LVCs have more powerful capabilities for modeling long-term dependencies in audio because they can flexibly generate kernels that correspond in a more tailored way to different conditioning sequences.

3 LVC-VC: Location-Variable Convolution-based Voice Conversion

3.1 Input features

**Content.** The primary content feature used for LVC-VC is inspired by the source-filter model of speech production, in which the excitation of a speaker’s vocal cords (the source) is convolved with a representation of the vocal tract (the filter) [31]. Here, we consider the excitation to contain the speaker information of a spoken utterance and the vocal tract to contain the content information. Indeed, in voiced speech, the excitation contains information on a voice’s fundamental frequency (F0) and its harmonic frequencies, and the vocal tract determines the specific phoneme that is produced by a speech sound. Therefore, separating the source and filter components of an utterance via deconvolution allows us disentangle its speaker and content information to a large extent. In particular, the spectral envelope of an utterance can be extracted by performing low-quefrency liftering on the utterance. Further details on the theory behind this process are described in Appendix A.

Note that the filter still contains some amount of speaker information; intuitively, this is because different speakers have different vocal tract shapes, which causes variations in spectral envelopes and formants even when the same phonemes are being pronounced. In Section 3.3 we describe an information perturbation strategy to prevent the model from using this information during training.

Formally, let the source utterance in the time domain be \(x\) and its log-mel spectrogram be \(X\). We convert \(X\) to the cepstral domain, perform low-quefrency liftering, and re-convert back to the spectral domain to obtain the spectral envelope of the utterance \(H\). \(H\) serves as the primary content feature that is fed into the model, and is intended to contain the content information of \(X\) but with the excitation of the vocal cords (the harmonics) removed.

We also compute the normalized F0 contour of \(x\) and use it as an additional input feature for the model. Specifically, we use the per-frame normalized quantized log F0 feature \(p_{\text{norm}}\) that was introduced and used previously in [13] and [32]. To compute \(p_{\text{norm}}\), we first extract the log F0 from all of a speaker’s voiced samples using a pitch tracking algorithm [33]. Then, we compute the speaker’s log
F0 mean $\mu$ and variance $\sigma^2$. We use the same analysis window and hop sizes as when computing $X$ to ensure that the number of extracted F0 frames matches up with the number of spectrogram frames. Then, for each voiced frame, we normalize the raw log F0 $p_{\text{raw}}$ as follows:

$$p_{\text{norm}, n} = \frac{p_{\text{raw}, n} - \mu}{4\sigma} + \frac{1}{2},$$

where $n$ denotes the index of an analysis frame. This operation roughly constrains values of $p_{\text{norm}}$ to be within the range $[0, 1]$; any values falling outside this range are clipped. We then quantize the range $[0, 1]$ into 256 bins and one-hot encode the $p_{\text{norm}}$ values. Finally, we add another bin to represent unvoiced frames, resulting in a 257-dimensional one-hot encoded feature for each frame. The concatenation of these features across all frames then becomes $p_{\text{norm}}$.

**Speaker.** For speaker-related conditioning features, we use embeddings extracted from a speaker encoder $E_s$ that has been pre-trained on a speaker recognition task. The architecture and training details of $E_s$ are described more in-depth in Section 3.2. We denote speaker embeddings using the variable $s$. For an utterance $x$, the speaker embedding is then: $s = E_s(x)$.

We also include the quantized median log F0 of a speaker as an additional conditioning feature. This is computed by extracting the log F0 from all of a speaker’s voiced speech samples using the same pitch tracking algorithm as for $p_{\text{norm}}$, and computing the median for a speaker. Then, we quantize the range log(65.4) Hz to log(523.3) Hz (corresponding to the notes ‘C2’ and ‘C5’) into 64 bins and and one-hot encode the median log F0 value. As before, any values falling outside the quantized range are clipped. This results in a 64-dimensional vector $m$ that encodes a speaker’s F0 information.

### 3.2 Model architecture

LVC-VC consists of a generator $G$, a speaker encoder $E_s$, and a set of discriminators for GAN-based training. The input features are fed into the kernel predictors of the generator’s LVC layers, which combine the information from the various features and pass it to the generator to synthesize audio.

**Generator.** The generator $G$ is based on the c16 variant of UnivNet [30]. It has a channel size of 16 in each of its convolutional layers and contains around 4.5 million parameters. Figure 1 shows a diagram of the overall architecture. It is a fully convolutional neural network that takes random noise $z$ as an input sequence and the content and speaker features described in Section 3.1 as conditions, and outputs a raw audio waveform $\hat{x}$. Its main body consists of a series of 1D transposed convolutional layers to upsample the input waveform $x$, which is specified to have the same length as the low-quefrency lifted log-mel spectrogram $H$. In our experiments, spectrograms are at a $256 \times$ lower resolution compared to raw audio. Therefore, there are three transposed convolutional layers with upsampling factors of $8 \times$, $8 \times$, and $4 \times$ to perform the total $256 \times$ upsampling.

Each transposed convolutional layer is followed by a stack of four residual blocks, each of which consists of a dilated 1D convolution, a 1D location-variable convolution (LVC), and a gated activation unit (GAU) [34]. The four dilated convolutions in each stack have dilation factors of $[1, 3, 9, 27]$. Leaky ReLU [35] with $\alpha = 0.2$ is used as the activation before the dilated convolutions and LVCs. The kernels of the LVC layers are determined by kernel predictor networks that take the conditioning features $H$, $p_{\text{norm}}$, $s$, and $m$ as input (Figure 1b). Each residual stack has its own kernel predictor network, for a total of three kernel predictors. Each kernel predictor consists of a residual stack of 1D convolutions with Leaky ReLU activations ($\alpha = 0.2$), and simultaneously predicts the kernels of all of the LVC layers in the stack that it is associated with. The output waveform is thus a result of feeding the input noise sequence and all of the conditioning features through the generator:

$$\hat{x} = G(z, H, p_{\text{norm}}, s, m).$$

**Speaker encoder.** For the speaker encoder $E_s$, we use the Fast ResNet-34 speaker recognition model from [36]. The model was pre-trained using angular prototypical loss on the development set of the VoxCeleb2 dataset [37] and uses self-attentive pooling [38] to aggregate frame-level features into an utterance-level representation. The model takes 40 dimensional log-mel spectrograms as input and outputs speaker embeddings of dimension 512.

**Discriminators.** LVC-VC uses two discriminators for GAN-based training: a multi-resolution spectrogram discriminator (MRSD) and a multi-period waveform discriminator (MPWD).
Figure 1: Overall LVC-VC architecture. (a) Generator. (b) Kernel predictor network for LVC layers.

The MRSD was originally introduced in [30]. It is a mixture of $M$ sub-discriminators that evaluates a synthesized waveform at multiple frequency resolution scales using linear magnitude spectrograms and makes a decision as to whether it is real or not. The spectrograms are computed using short-time Fourier transform (STFT) parameter sets $\{\text{FT}_m(\cdot)\}_{m=1}^M$. Here, $\text{FT}_m(\cdot)$ denotes the Fourier transform performed by the $m$-th sub-discriminator. Each STFT parameter set consists of: (number of Fourier transform points, window length (in seconds), hop length (in seconds)). Formally, the MRSD computes the following:

$$\{s_m = |\text{FT}_m(x)|, \hat{s}_m = |\text{FT}_m(\hat{x})|\}_{m=1}^M.$$  \hfill (3)

In our experiments, $M = 3$ and the STFT parameter sets for the sub-discriminators were $[(512, 0.025, 0.005), (1024, 0.05, 0.01), (256, 0.01, 0.002)]$.

The MPWD, originally introduced in [29], is also a mixture of sub-discriminators, each of which takes as input equally spaced samples of a time-domain audio waveform at a different period $p$ and evaluates whether it is real or not. The periods are set to the prime numbers $p \in \{2, 3, 5, 7, 11\}$ in order to avoid overlaps in analysis between the sub-discriminators as much as possible. The MPWD is designed to model and capture implicit structures in the periodic patterns of audio at multiple temporal resolutions, thereby guiding the generator to synthesize more realistic waveforms.

### 3.3 Training

Recall that we use a speaker encoder $E_s$ that has already been pre-trained to extract speaker information. Therefore, we keep the weights of $E_s$ fixed and only train $G$ and the discriminators.

**Self-reconstruction.** The training losses for self-reconstruction largely follow those used by UnivNet [30]. The content and speaker features described in Section 3.1 are extracted from training utterances and used to reconstruct the original audio.

Recall that the content feature $H$ still contains some amount of speaker information. This could lead to the residual speaker information in $H$ leaking into the synthesized audio, resulting in poor voice conversion at inference time. To prevent this from happening, we randomly warp $H$ by stretching or compressing it along the frequency axis during training; we denote the warped version of the feature $H'$. This step removes most of the residual speaker information in low-quefrency liftered log-mel spectrograms while still preserving the content information in the original spectral envelope.
For the speaker embeddings, rather than using the embedding extracted from a given training utterance, we sample an embedding \( s' \) from a GMM with 1 component that we fit on the corresponding speaker's training utterances. This strategy ensures that a similar, but different speaker embedding is used to reconstruct an utterance every time. It also helps the model generalize better to unseen speakers.

Formally, let an input utterance for training be \( x \) and the associated conditioning features be \( H', p_{\text{norm}}, s', m \). Then, the reconstructed output is produced by \( \hat{x} = G(z, H', p_{\text{norm}}, s', m) \). In addition to the GAN losses defined by the discriminators, we use multi-resolution STFT loss \(^{[9]}\) as an auxiliary training criterion. The full auxiliary loss \( \mathcal{L}_{\text{aux}} \), which is comprised of the spectral convergence loss \( \mathcal{L}_{\text{sc}} \) and log STFT magnitude loss \( \mathcal{L}_{\text{mag}} \), is defined as follows:

\[
\mathcal{L}_{\text{aux}}(x, \hat{x}) = \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}_{x, \hat{x}} \left[ \mathcal{L}_{\text{sc}}(s_m, \hat{s}_m) + \mathcal{L}_{\text{mag}}(s_m, \hat{s}_m) \right],
\]

\[
\mathcal{L}_{\text{sc}}(s, \hat{s}) = \frac{\|s - \hat{s}\|_F}{\|s\|_F},
\]

\[
\mathcal{L}_{\text{mag}}(s, \hat{s}) = \frac{1}{N} \|\log s - \log \hat{s}\|_1.
\]

\( N \) denotes the number of frames in a spectrogram and \( \| \cdot \|_F \) and \( \| \cdot \|_1 \) denote the Frobenius and L1 norms, respectively. \( M \) is the number of MRSD sub-discriminators. \( s \) and \( \hat{s} \) are defined in Equation \( \[3\] \) and each \( m \)-th \( \mathcal{L}_{\text{sc}} \) and \( \mathcal{L}_{\text{mag}} \) reuse the \( s_m \) and \( \hat{s}_m \) used for the \( m \)-th MRSD sub-discriminator.

**Speaker similarity.** Self-reconstructive training on its own does not explicitly force the converted audio to take on the vocal characteristics of the target speaker. Therefore, we utilize an additional loss which induces LVC-VC to generate audio that more closely matches the characteristics of the target speaker. We call this loss the speaker similarity criterion (SSC). Let the original utterance used for self-reconstructive training be \( x_0 \) and its associated features be \( (H_0, p_{\text{norm}}_0, s_0, m_0) \). For each reconstructive training sample, we sample \( N \) more utterances from different speakers \( x_1, \ldots, x_N \) with associated content features \( (H_n, p_{\text{norm}}_n), \forall n \in [1, \ldots, N] \). We designate \( x_0 \) to be the target utterance for performing conversion. Then, the SSC loss \( \mathcal{L}_{\text{ssc}} \) is:

\[
\hat{x}_{n \rightarrow 0} = G(z, H_n, p_{\text{norm}}_n, s'_0, m_0),
\]

\[
\mathcal{L}_{\text{ssc}} = \frac{1}{N} \sum_{n=1}^{N} \cos \left( E_s(\hat{x}_{n \rightarrow 0}), s'_0 \right),
\]

where \( \cos(x_1, x_2) \) denotes the cosine similarity between \( x_1 \) and \( x_2 \).

**GAN-based training.** The generator and discriminator losses for training follow the least-squares GAN objective functions \(^{[40]}\). The overall losses are defined as follows:

\[
\mathcal{L}_G = \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}_{z, c} \left[ (D_k(G(z, H', p_{\text{norm}}, s', m)) - 1)^2 \right] + \lambda_{\text{aux}} \mathcal{L}_{\text{aux}}(x, G(z, H', p_{\text{norm}}, s', m)) + \lambda_{\text{ssc}} \mathcal{L}_{\text{ssc}},
\]

\[
\mathcal{L}_D = \frac{1}{K} \sum_{k=1}^{K} \left( \mathbb{E}_{x} \left[ (D_k(x) - 1)^2 \right] + \mathbb{E}_{z, c} \left[ D_k(G(z, H', p_{\text{norm}}, s', m)) \right] \right),
\]

where \( K \) denotes the number of all sub-discriminators across the MRSD and MPWD and \( D_k \) denotes the \( k \)-th sub-discriminator across all sub-discriminators. \( \lambda_{\text{aux}} \) and \( \lambda_{\text{ssc}} \) are weighting factors that balance the contributions of the auxiliary loss and SSC loss for the generator, respectively.

### 3.4 Inference

At inference time, LVC-VC performs voice conversion by combining the content features from the source utterance and the speaker features from the target utterance to generate audio. Given source utterance \( x_1 \) with content features \( (H_1, p_{\text{norm}}_1) \) and target utterance \( x_2 \) with speaker features \( (s_2, m_2) \), the converted utterance \( \hat{x}_{1 \rightarrow 2} \) is produced by:

\[
\hat{x}_{1 \rightarrow 2} = G(z, H_1, p_{\text{norm}}_1, s_2, m_2).
\]
4 Experiments

4.1 Configurations

We used the VCTK Corpus\(^2\) for training and evaluation, which consists of around 44 hours of
audio spoken by 109 speakers. All audio was resampled to 16 kHz. We randomly partitioned
the dataset into 99 “seen” speakers, which the model would be exposed to during training, and 10
“unseen” speakers. The utterances of the 99 seen speakers were further randomly split into train and
test sets in a 9-1 ratio. Only utterances from the seen speakers’ train set were used for training.

To obtain \(H\) from a time-domain utterance \(x\), we started off with an 80-dimensional log-mel spectro-
gram \(X\) computed using a 1024 point Fourier transform, with a Hann window of size 1024 and hop
length of size 256. We then took the 20 lowest quefrency coefficients for low-quefrency liftering. To
compute \(H'\), we chose the warping factor along the frequency axis from a uniform distribution over
the range \([0.85, 1.15]\) for each training sample.

Training was done on four NVIDIA GeForce GTX 1080 Ti GPUs. We used the AdamW optimizer \(^{42}\)
with a learning rate of 1e-4 and \(\beta_1 = 0.5, \beta_2 = 0.9\). All input features were cropped or padded to
correspond to 16,384 samples (1 second) for batch processing. For the SSC loss, we set \(N = 8\).
Following \(^{30}\), we set \(\lambda_{aux} = 2.5\), and we empirically set \(\lambda_{ssc} = 0.9\). LVC-VC was first trained
with only self-reconstructive loss using a batch size of 32 for 1.8 million iterations. Then, we halved
the learning rate to 5e-5 and continued training with the SSC loss included for 5,000 more iterations,
using a decreased batch size of 16 due to GPU memory constraints. We found that this strategy
ensured that the model learned to produce high-quality audio first, before then being guided to
perform better voice conversion without compromising audio quality significantly. \(\lambda_{ssc}\) was linearly
annealed from 0 to its final value for the first 2,000 steps in which the SSC loss was used.

4.2 Evaluation metrics

Subjective metrics. We conducted subjective listening tests for evaluating naturalness, intelligibil-
ity, and speaker similarity on Amazon Mechanical Turk. For naturalness (mean opinion score, MOS)
and intelligibility, subjects provided a score on a scale from 1 to 5. For similarity, we used the metric
introduced in \(^{43}\); subjects evaluated whether two utterances sounded like they could be from the
same speaker, and we computed the binary decision percentage of “same” or “different.” The full
description of the evaluation task is provided in Appendix C.

Objective metrics. We utilized four objective metrics: word error rate (WER) and character error
rate (CER) on an automatic speech recognition (ASR) task, equal error rate (EER) on an automatic
speaker verification (ASV) task, and NISQA \(^{44}\) score, which estimates the MOS for overall audio
quality that would be assigned to an utterance on a scale from 1 to 5. For ASR, we used a wav2vec
2.0 Base model \(^{45}\) that was trained to perform ASR on 960 hours of the LibriSpeech corpus \(^{46}\).
For ASV, we used a ResNet-34-based model from \(^{47}\) that uses attentive statistics pooling \(^{48}\) to
aggregate temporal frames and was trained on the development split of the VoxCeleb2 dataset \(^{57}\).

4.3 Results

We compared LVC-VC against six other models: AdaIN-VC \(^{11}\), AGAIN-VC \(^{23}\), AutoVC \(^{12}\),
AutoVC-F0 \(^{13}\), Blow \(^{25}\), and NVC-Net \(^{14}\). We used the official implementations of all models
except for AutoVC-F0, which we implemented according to the paper since an official implementation
was not available. All models were trained from scratch on the same train-test split of the VCTK
dataset as LVC-VC. For a fair comparison, models taking spectrograms as input features were trained
using the same spectrogram configuration as LVC-VC, and all time-domain audio was synthesized
using a UnivNet-c16 vocoder \(^{30}\) that was trained on the LibriTTS dataset \(^{59}\).

We considered three different source-to-target voice conversion settings for evaluation: seen-to-
seen, unseen-to-seen, and unseen-to-unseen. For every voice conversion setting, we considered
four gender-to-gender conversion combinations: male-to-male, male-to-female, female-to-male, and
female-to-female. For the seen-to-seen setting, we sampled 25 speakers from the 99 seen speakers
as source speakers and randomly assigned one speaker from each gender to act as a target speaker.

\(^2\)The VCTK Corpus is available under a Creative Commons Attribution 4.0 License.
We find that most of the previously proposed models fall under two categories: 1) those that perform voice style transfer (VST) reasonably well, but produce low-quality or less intelligible audio (AdaIN-VC, AGAIN-VC, NVC-Net), and 2) those that produce high-quality and intelligible audio, but do not perform VST very well (AutoVC, AutoVC-F0). In other words, there is a trade-off between producing high-quality audio and achieving good VST performance. LVC-VC is able to manage these trade-offs much better than the other models. It achieves MOS and NISQA scores that are competitive with the other best models in those categories, especially in the seen-to-seen and unseen-to-seen settings. Notably, it does very well in terms of Intelligibility and has the lowest WER and CER in all three settings, showing that it is able to maintain the linguistic content of source utterances very well. LVC-VC also performs quite well in terms of VST performance. Although it does not quite outperform AdaIN-VC and AGAIN-VC in terms of Similarity, it obtains a better EER than them in all three settings. It also obtains the best EER overall in the unseen-to-unseen setting.

Table 1: Seen-to-seen voice conversion evaluation results.

| Model       | MOS ± | Intelligibility ± | Similarity | WER | CER | EER | NISQA ± |
|-------------|-------|-------------------|------------|-----|-----|-----|--------|
| Ground Truth| 4.40  | 4.64 ± 0.06       | 91.75      | 11.27| 3.94| 1.50| 4.50 ± 0.05 |
| UnivNet     | 4.33  | 4.47 ± 0.07       | 90.50      | 12.26| 4.47| 2.00| 4.45 ± 0.06 |
| LVC-VC      | 3.54  | 4.17 ± 0.10       | 46.00      | 22.69| 9.55| 18.50| 4.00 ± 0.08 |
| AdaIN-VC    | 2.33  | 3.05 ± 0.13       | 53.00      | 43.06| 22.48| 33.00| 3.75 ± 0.09 |
| AGAIN-VC    | 2.04  | 2.88 ± 0.13       | 44.00      | 47.64| 25.22| 24.00| 3.70 ± 0.10 |
| AutoVC      | 3.78  | 4.15 ± 0.09       | 22.25      | 24.24| 10.82| 40.50| 4.13 ± 0.06 |
| AutoVC-F0   | 3.59  | 4.03 ± 0.10       | 34.50      | 25.16| 11.81| 33.00| 4.10 ± 0.07 |
| Blow        | 1.85  | 3.19 ± 0.13       | 35.00      | 31.01| 14.86| 53.00| 3.07 ± 0.11 |
| NVC-Net     | 2.96  | 3.40 ± 0.13       | 67.75      | 48.91| 27.25| 15.00| 4.31 ± 0.07 |

Table 2: Unseen-to-seen voice conversion evaluation results.

| Model       | MOS ± | Intelligibility ± | Similarity | WER | CER | EER | NISQA ± |
|-------------|-------|-------------------|------------|-----|-----|-----|--------|
| Ground Truth| 4.43  | 4.83 ± 0.07       | 93.75      | 9.69| 2.93| 0.00| 4.42 ± 0.09 |
| UnivNet     | 4.34  | 4.55 ± 0.12       | 93.75      | 11.70| 3.77| 0.00| 4.38 ± 0.10 |
| LVC-VC      | 3.31  | 4.31 ± 0.14       | 41.88      | 17.37| 7.03| 20.00| 3.89 ± 0.14 |
| AdaIN-VC    | 2.42  | 3.23 ± 0.18       | 53.75      | 36.56| 17.86| 36.25| 3.85 ± 0.13 |
| AGAIN-VC    | 2.43  | 3.34 ± 0.20       | 45.63      | 43.33| 21.62| 33.75| 3.72 ± 0.18 |
| AutoVC      | 3.50  | 4.33 ± 0.14       | 28.75      | 23.03| 10.97| 30.00| 4.18 ± 0.11 |
| AutoVC-F0   | 3.52  | 4.06 ± 0.16       | 38.13      | 22.12| 9.67| 26.25| 4.04 ± 0.12 |
| NVC-Net     | 3.17  | 3.44 ± 0.21       | 60.63      | 48.45| 26.91| 11.25| 4.14 ± 0.13 |

Table 3: Unseen-to-unseen voice conversion evaluation results.

| Model       | MOS ± | Intelligibility ± | Similarity | WER | CER | EER | NISQA ± |
|-------------|-------|-------------------|------------|-----|-----|-----|--------|
| Ground Truth| 4.41  | 4.73 ± 0.08       | 93.75      | 12.06| 3.58| 0.00| 4.37 ± 0.10 |
| UnivNet     | 4.36  | 4.67 ± 0.09       | 91.25      | 14.34| 4.85| 0.00| 4.37 ± 0.09 |
| LVC-VC      | 3.08  | 4.06 ± 0.16       | 29.38      | 20.10| 8.29| 26.25| 3.50 ± 0.13 |
| AdaIN-VC    | 2.41  | 3.28 ± 0.21       | 50.63      | 41.43| 20.53| 35.00| 3.55 ± 0.17 |
| AGAIN-VC    | 2.18  | 2.90 ± 0.20       | 30.00      | 51.57| 26.86| 32.50| 3.35 ± 0.16 |
| AutoVC      | 3.39  | 4.00 ± 0.16       | 5.63       | 27.97| 12.41| 66.25| 4.09 ± 0.10 |
| AutoVC-F0   | 3.21  | 4.08 ± 0.16       | 12.50      | 28.15| 12.55| 63.75| 3.94 ± 0.12 |
| NVC-Net     | 3.09  | 3.44 ± 0.20       | 35.63      | 50.51| 26.27| 37.50| 4.24 ± 0.11 |

For each source-target speaker pair, we randomly sampled two utterances from each speaker for performing conversion. This resulted in 200 (= 25 × 2 × 4) utterance pairs. For the unseen-to-seen setting, we used the 10 unseen speakers as source speakers and randomly sampled target speakers and utterances from the 99 seen speakers in the same way as above. This resulted in 80 (= 10 × 2 × 4) utterance pairs. Finally, for the unseen-to-unseen setting, we used the 10 unseen speakers as source speakers and randomly sampled target speakers and utterances from the other 9 unseen speakers, resulting in 80 (= 10 × 2 × 4) utterance pairs. Tables 1-3 show the results on the seen-to-seen, unseen-to-seen, and unseen-to-unseen settings, respectively. We report the results for ground truth speech and speech reconstructed using the UnivNet vocoder to provide baseline values for reference. We also report 95% confidence intervals for MOS, Intelligibility, and NISQA.
Table 4: Unseen-to-unseen voice conversion evaluation results for various ablations of LVC-VC.

| Model                              | WER   | CER   | EER   | NISQA  |
|------------------------------------|-------|-------|-------|--------|
| LVC-VC                            | 20.10 | 8.29  | 26.25 | 3.50 ± 0.13 |
| w/o GMM embeddings                 | 26.92 | 11.11 | 25.00 | 2.89 ± 0.14 |
| w/o SSC loss                       | 16.78 | 6.64  | 68.75 | 3.83 ± 0.13 |
| w/o warping H                      | 21.33 | 8.60  | 32.50 | 3.36 ± 0.18 |
| w/o  p_{norm}                      | 22.90 | 9.90  | 28.75 | 3.47 ± 0.14 |
| w/o m                              |       |       |       |         |

4.4 Ablation studies

We conducted ablation studies on the various training strategies and input features used in LVC-VC; the results are shown in Table 4. For brevity and convenience, we only report scores from objective metrics and show results on unseen-to-unseen voice conversion. Full results for all three conversion settings are in Appendix D. Training LVC-VC on fixed speaker embeddings instead of sampling from GMMs causes audio quality to decrease significantly. This suggests that training the model on audio from more diverse speaker embeddings helps it combine speaker information with other input features more coherently, which also leads to better generalization to unseen speakers. Training the model without SSC loss or without warping $H$ causes VST performance to decrease, demonstrating the importance of explicitly guiding the model to perform conversion rather than only relying on self-reconstructive training, as well as perturbing the source speaker information in $H$. Finally, $p_{norm}$ and $m$ contribute to relatively small, but significant performance gains in all measured metrics.

5 Ethical Considerations and Broader Impact

Voice conversion is a field that is unfortunately fraught with potential misuse. “Audio deepfakes” can be used to deceive people by synthetically generating speech and attributing them to certain individuals; this can lead to problems such as voice spoofing [50] or the spread of fake news and misinformation [51]. To the extent that this work enables more realistic manipulation of speaker identities, it could potentially exacerbate these misuses if used by a malicious party. A wide variety of work has sought to address these issues by developing techniques for anti-spoofing [52,53,54] and general fake audio detection [55,56]. Recently, there have also been several public challenges to encourage the development of systems that can detect fake audio [57,58,59]. Going forward, the speech machine learning community should continue to encourage these research directions and raise awareness of the potential problems with high-fidelity synthetic audio.

One application of voice conversion is speaker anonymization, which can enable privacy preservation by masking the identity of a speaker’s voice [2]. Traditionally, using VC methods for anonymization necessitated changing the original voice to another existing person’s voice. To avoid the issues that arise with this, we propose a method for performing un-targeted voice anonymization using LVC-VC, which obviates the need for specifying a target speaker. The details are described in Appendix E.

6 Conclusion

In this work, we presented Location-Variable Convolution-based Voice Conversion (LVC-VC), an end-to-end model for performing zero-shot voice conversion that is based on a neural vocoder. Rather than disentangling speaker and content information like a standard zero-shot VC model, LVC-VC utilizes a set of input features that already have disentangled content and speaker style information and combines them within a vocoder-like neural framework. This allows it to simultaneously perform voice conversion while generating time-domain audio. LVC-VC achieves voice style transfer performance that is competitive with or better than several baselines while maintaining the intelligibility of transformed speech significantly better. This confirms the effectiveness of the chosen input features for encoding the relevant content and speaker information for use during speech synthesis.

While there is still a gap between the audio produced by LVC-VC and true speech in terms of naturalness and speaker similarity, the model’s performance could likely be improved by training it on data from a much larger number of speakers; exposure to a wider space of speaker embeddings would help it generalize better to new speakers during inference. Additionally, audio quality could be improved by using a larger model with more channels in its convolutional layers. By removing the need for a vocoder, we believe that the proposed framework opens up many avenues in efficient end-to-end speech synthesis, not just in voice conversion, but also in related fields such as text-to-speech.
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, 2020.

[2] N. Tomashenko, B. M. L. Srivastava, X. Wang, E. Vincent, A. Nautsch, J. Yamagishi, N. Evans, J. Patino, J.-F. Bonastre, P.-G. Noé, et al., “Introducing the voiceprivacy initiative,” in INTERSPEECH 2020, 2020.

[3] J. Yamagishi, C. Veaux, S. King, and S. Renals, “Speech synthesis technologies for individuals with vocal disabilities: Voice banking and reconstruction,” Acoustical Science and Technology, vol. 33, no. 1, pp. 1–5, 2012.

[4] C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang, “Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks,” arXiv preprint arXiv:1704.00849, 2017.

[5] T. Kaneko and H. Kameoka, “Cyclegan-vc: Non-parallel voice conversion using cycle-consistent adversarial networks,” in 2018 26th European Signal Processing Conference (EUSIPCO), pp. 2100–2104, IEEE, 2018.

[6] T. Kaneko, H. Kameoka, K. Tanaka, and N. Hojo, “Cyclegan-vc2: Improved cyclegan-based non-parallel voice conversion,” in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6820–6824, IEEE, 2019.

[7] B. van Niekerk, L. Nortje, and H. Kamper, “Vector-quantized neural networks for acoustic unit discovery in the zerospeech 2020 challenge,” in INTERSPEECH, 2020.

[8] K. Kobayashi, W.-C. Huang, Y.-C. Wu, P. L. Tobing, T. Hayashi, and T. Toda, “crank: An open-source software for nonparallel voice conversion based on vector-quantized variational autoencoder,” in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5934–5938, IEEE, 2021.

[9] L.-J. Liu, Z.-H. Ling, Y. Jiang, M. Zhou, and L.-R. Dai, “Wavenet vocoder with limited training data for voice conversion,” in Interspeech, pp. 1983–1987, 2018.

[10] J.-X. Zhang, L.-J. Liu, Y.-N. Chen, Y.-J. Hu, Y. Jiang, Z.-H. Ling, and L.-R. Dai, “Voice conversion by cascading automatic speech recognition and text-to-speech synthesis with prosody transfer,” in Proc. Joint Workshop for the Blizzard Challenge and Voice Conversion Challenge 2020, pp. 121–125, 2020.

[11] J.-c. Chou and H.-Y. Lee, “One-shot voice conversion by separating speaker and content representations with instance normalization,” Proc. Interspeech 2019, pp. 664–668, 2019.

[12] 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, pp. 5210–5219, PMLR, 2019.

[13] K. Qian, Z. Jin, M. Hasegawa-Johnson, and G. J. Mysore, “F0-consistent many-to-many non-parallel voice conversion via conditional autoencoder,” in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6284–6288, IEEE, 2020.

[14] B. Nguyen and F. Cardinaux, “Nvc-net: End-to-end adversarial voice conversion,” in ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 7012–7016, IEEE, 2022.

[15] S.-H. Lee, J.-H. Kim, H. Chung, and S.-W. Lee, “Voicemixer: Adversarial voice style mixup,” Advances in Neural Information Processing Systems, vol. 34, 2021.

[16] D. Silver, H. Hasselt, M. Hessel, T. Schaul, A. Guez, T. Harley, G. Dulac-Arnold, D. Reichert, N. Rabinowitz, A. Barreto, et al., “The predictron: End-to-end learning and planning,” in International Conference on Machine Learning, pp. 3191–3199, PMLR, 2017.

[17] S. Dieleman and B. Schrauwen, “End-to-end learning for music audio,” in 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6964–6968, IEEE, 2014.
18. D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al., “Deep speech 2: End-to-end speech recognition in english and mandarin,” in International Conference on Machine Learning, pp. 173–182, PMLR, 2016.

19. T. Boros, S. D. Dumitrescu, and R. Burtica, “Nlp-cube: End-to-end raw text processing with neural networks,” in Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pp. 171–179, 2018.

20. A. Gordo, J. Almazan, J. Revaud, and D. Larlus, “End-to-end learning of deep visual representations for image retrieval,” International Journal of Computer Vision, vol. 124, no. 2, pp. 237–254, 2017.

21. Z. Zeng, J. Wang, N. Cheng, and J. Xiao, “Lvcnet: Efficient condition-dependent modeling network for waveform generation,” in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6054–6058, IEEE, 2021.

22. S. Yuan, P. Cheng, R. Zhang, W. Hao, Z. Gan, and L. Carin, “Improving zero-shot voice style transfer via disentangled representation learning,” in International Conference on Learning Representations, 2020.

23. Y.-H. Chen, D.-Y. Wu, T.-H. Wu, and H.-y. Lee, “Again-vc: A one-shot voice conversion using activation guidance and adaptive instance normalization,” in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5954–5958, IEEE, 2021.

24. H.-S. Choi, J. Lee, W. Kim, J. Lee, H. Heo, and K. Lee, “Neural analysis and synthesis: Reconstructing speech from self-supervised representations,” Advances in Neural Information Processing Systems, vol. 34, 2021.

25. J. Serrà, S. Pascual, and C. Segura Perales, “Blow: a single-scale hyperconditioned flow for non-parallel raw-audio voice conversion,” Advances in Neural Information Processing Systems, vol. 32, pp. 6793–6803, 2019.

26. A. Conneau, A. Baevski, R. Collobert, A. Mohamed, and M. Auli, “Unsupervised cross-lingual representation learning for speech recognition,” arXiv preprint arXiv:2006.13979, 2020.

27. 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.

28. K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, “Melgan: Generative adversarial networks for conditional waveform synthesis,” Advances in Neural Information Processing Systems, vol. 32, 2019.

29. 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, 2020.

30. W. Jang, D. Lim, J. Yoon, B. Kim, and J. Kim, “UnivNet: A Neural Vocoder with Multi-Resolution Spectrogram Discriminators for High-Fidelity Waveform Generation,” in Proc. Interspeech 2021, pp. 2207–2211, 2021.

31. B. S. Atal and S. L. Hanauer, “Speech analysis and synthesis by linear prediction of the speech wave,” The Journal of the Acoustical Society of America, vol. 50, no. 2B, pp. 637–655, 1971.

32. 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, pp. 7836–7846, PMLR, 2020.

33. M. Mauch and S. Dixon, “pyin: A fundamental frequency estimator using probabilistic threshold distributions,” in 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 659–663, IEEE, 2014.

34. N. Van den Oord, Aaron anModed Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al., “Conditional image generation with pixelcnn decoders,” Advances in Neural Information Processing Systems, vol. 29, 2016.

35. A. L. Maas, A. Y. Hannun, A. Y. Ng, et al., “Rectifier nonlinearities improve neural network acoustic models,” in Proc. ICML, vol. 30, p. 3, Citeseer, 2013.
[36] J. S. Chung, J. Huh, S. Mun, M. Lee, H.-S. Heo, S. Choe, C. Ham, S. Jung, B.-J. Lee, and I. Han, “In defence of metric learning for speaker recognition,” Proc. Interspeech 2020, pp. 2977–2981, 2020.

[37] J. S. Chung, A. Nagrani, and A. Zisserman, “Voxceleb2: Deep speaker recognition,” arXiv preprint arXiv:1806.05622, 2018.

[38] W. Cai, J. Chen, and M. Li, “Exploring the encoding layer and loss function in end-to-end speaker and language recognition system,” in Proc. Odyssey 2018 The Speaker and Language Recognition Workshop, pp. 74–81, 2018.

[39] R. Yamamoto, E. Song, and J.-M. Kim, “Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram,” in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6199–6203, IEEE, 2020.

[40] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, “Least squares generative adversarial networks,” in Proceedings of the IEEE International Conference on Computer Vision, pp. 2794–2802, 2017.

[41] J. Yamagishi, C. Veaux, K. MacDonald, et al., “Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit (version 0.92),” 2019.

[42] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in International Conference on Learning Representations, 2018.

[43] M. Wester, Z. Wu, and J. Yamagishi, “Analysis of the voice conversion challenge 2016 evaluation results,” Interspeech 2016, pp. 1637–1641, 2016.

[44] G. Mittag, B. Naderi, A. Chehadi, and S. Möller, “Nisqa: A deep cnn-self-attention model for multidimensional speech quality prediction with crowdsourced datasets,” Proc. Interspeech 2021, pp. 2127–2131, 2021.

[45] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” in Advances in Neural Information Processing Systems, vol. 33, pp. 12449–12460, 2020.

[46] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: an asr corpus based on public domain audio books,” in 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5206–5210, IEEE, 2015.

[47] H. S. Heo, B.-J. Lee, J. Huh, and J. S. Chung, “Clova baseline system for the voxceleb speaker recognition challenge 2020,” arXiv preprint arXiv:2009.14153, 2020.

[48] K. Okabe, T. Koshinaka, and K. Shinoda, “Attentive statistics pooling for deep speaker embedding,” Proc. Interspeech 2018, pp. 2252–2256, 2018.

[49] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu, “Libritts: A corpus derived from librispeech for text-to-speech,” Proc. Interspeech 2019, pp. 1526–1530, 2019.

[50] T. Kinnunen, Z.-Z. Wu, K. A. Lee, F. Sedlak, E. S. Chng, and H. Li, “Vulnerability of speaker verification systems against voice conversion spoofing attacks: The case of telephone speech,” in 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4401–4404, IEEE, 2012.

[51] S. Singhal, R. R. Shah, T. Chakraborty, P. Kumaranagu, and S. Satoh, “Spotfake: A multimodal framework for fake news detection,” in 2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM), pp. 39–47, IEEE, 2019.

[52] Q. Wang, X. Lin, M. Zhou, Y. Chen, C. Wang, Q. Li, and X. Luo, “Voicepop: A pop noise based anti-spoofing system for voice authentication on smartphones,” in IEEE INFOCOM 2019-IEEE Conference on Computer Communications, pp. 2062–2070, IEEE, 2019.

[53] M. R. Kamble, H. B. Sailor, H. A. Patil, and H. Li, “Advances in anti-spoofing: from the perspective of asvspoof challenges,” APSIPA Transactions on Signal and Information Processing, vol. 9, 2020.

[54] H. Tak, J. Patino, M. Todisco, A. Nautsch, N. Evans, and A. Larcher, “End-to-end anti-spoofing with rawnet2,” in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6369–6373, IEEE, 2021.
[55] T. Mittal, U. Bhattacharya, R. Chandra, A. Bera, and D. Manocha, “Emotions don’t lie: An audio-visual deepfake detection method using affective cues,” in *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 2823–2832, 2020.

[56] H. Dhamyal, A. Ali, I. A. Qazi, and A. A. Raza, “Fake audio detection in resource-constrained settings using microfeatures,” *Proc. Interspeech 2021*, pp. 4149–4153, 2021.

[57] M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. Evans, T. Kinnunen, and K. A. Lee, “Asvspoof 2019: Future horizons in spoofed and fake audio detection,” in *INTERSPEECH 2019-20th Annual Conference of the International Speech Communication Association*, 2019.

[58] J. Yamagishi, X. Wang, M. Todisco, M. Sahidullah, J. Patino, A. Nautsch, X. Liu, K. A. Lee, T. Kinnunen, N. Evans, *et al.*, “Asvspoof 2021: accelerating progress in spoofed and deepfake speech detection,” in *ASVspoof 2021 Workshop-Automatic Speaker Verification and Spoofing Countermesures Challenge*, 2021.

[59] J. Yi, R. Fu, J. Tao, S. Nie, H. Ma, C. Wang, T. Wang, Z. Tian, Y. Bai, C. Fan, *et al.*, “Add 2022: the first audio deep synthesis detection challenge,” *arXiv preprint arXiv:2202.08433*, 2022.

[60] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[61] C. Olah, A. Mordvintsev, and L. Schubert, “Feature visualization,” *Distill*, vol. 2, no. 11, p. e7, 2017.

[62] D. Stanton, M. Shannon, S. Mariooryad, R. Skerry-Ryan, E. Battenberg, T. Bagby, and D. Kao, “Speaker generation,” in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7897–7901, IEEE, 2022.

[63] A. Nagrani, J. S. Chung, and A. Zisserman, “Voxceleb: a large-scale speaker identification dataset,” *arXiv preprint arXiv:1706.08612*, 2017.

[64] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
A Theory Behind Low-Quefrency Liftered Spectrogram as Content Feature

From a signal processing point of view, the physical speech production process can be modeled by a linear system in which the excitation of the vocal cords (the source) is convolved with a representation of the vocal tract (the filter) [31]. Here, the excitation is modeled using either an impulse train (for voiced speech) or white noise (for unvoiced speech), represented by a signal $e(n)$ with Fourier transform $E(z)$. Meanwhile, the vocal tract can be modeled using a discrete time-varying linear filter with impulse response $h(n)$ and transfer function $H(z)$. Therefore, an output speech signal $x(n)$ and its Fourier transform $X(z)$ can be described as follows:

$$x(n) = e(n) * h(n),$$
$$X(z) = E(z)H(z),$$

where $*$ denotes the convolution operation.

At a given point in time, the vocal tract determines the spectral envelope of the voice, which determines the specific phoneme that is produced via formant frequencies. The source-filter model provides a way of modeling the spectral envelope: it can be approximated by the transfer function of the filter, $H(z)$ [31].

We can consider the excitation of the vocal cords $E(z)$ to contain some of the speaker information in a spoken utterance. Indeed, $E(z)$ contains information on a voice’s fundamental frequency (F0) as well as its harmonic frequencies. Meanwhile, we can consider the spectral envelopes and formants $H(z)$ over time to contain a significant portion of the content information of an utterance. By separating $H(z)$ from $E(z)$ via deconvolution, we can disentangle the content information an utterance from a large portion of its speaker information. (As we note in the main text of the paper, $H(z)$ still contains some amount of speaker information on its own, which motivates our warping strategy for performing information perturbation during training.)

### A.1 Deconvolution in the cepstrum

Recall from Equation [12] that a speech signal $x(n)$ can be expressed as a convolution between an excitation signal $e(n)$ and the impulse response of the vocal tract filter $h(n)$. In the frequency domain, this convolution becomes equivalent to the multiplication of their respective Fourier transforms, as shown in Equation [13]. Taking the logarithms of the absolute values of the Fourier transforms to compute the log magnitude spectra converts the multiplication operation to addition:

$$\log |X(z)| = \log |E(z)H(z)|$$
$$= \log |E(z)| + \log |H(z)|.$$  \hfill (14)

If we apply a Fourier transform (in practice, actually a discrete cosine transform (DCT) since the log magnitude spectrum only has real components) to the above, we obtain the cepstrum ($C$), which is a frequency distribution of the fluctuations in the curve of the spectrum:

$$C = \text{DCT}(\log |X(z)|)$$
$$= \text{DCT}(\log |E(z)|) + \text{DCT}(\log |H(z)|).$$ \hfill (15)

If we assume that the source (excitation) spectrum has only rapid fluctuations (since the excitation signal is a stable, regular oscillation), its contribution to the cepstrum will be concentrated in the higher quefrency bins of $C$. Conversely, the filter (vocal tract) will contribute slow fluctuations to the spectrum of $X$ and will be concentrated in the lower quefrency bins.

Therefore, the separation of $E(z)$ and $H(z)$ becomes straightforward: we simply have to perform liftering and select the desired quefrency region by multiplying the entire cepstrum by a window at the appropriate position. Low-quefrency liftering, where the quefrency coefficients below a certain point are extracted, allows us to obtain the vocal tract characteristics in the quefrency domain. High-quefrency liftering, the opposite, allows us to obtain the excitation characteristics. Once we have performed liftering, it is a simple matter of performing the inverse DCT to obtain the deconvolved spectral envelope and excitation. Figure 2 illustrates the results of performing low-quefrency liftering on a sample log-mel spectrogram.
B Analysis of the Speech Synthesis Process

LVC-VC synthesizes audio by combining together various content and speaker-related input features within a vocoder-like framework. However, we do not know how exactly the audio is generated. What is actually happening inside the model? We performed several explorations of the intermediate representations of LVC-VC in order to better understand how it works.

B.1 Time-Domain Audio Generation

To investigate how LVC-VC generates audio, we performed spectral analyses of the intermediate outputs of the model after each transposed convolutional block. Specifically, we looked at spectrograms of each of these intermediate outputs in order to gain an intuition of what is happening at each step as the model upsamples the input noise sequence to eventually produce the output audio signal.

Recall that LVC-VC starts with an input noise sequence that is at a $\frac{1}{256} \times$ temporal resolution compared to the final output signal. It contains three 1D transposed convolutional layers that upsample this input sequence by $8 \times$, $8 \times$, and $4 \times$ to produce the final time domain waveform. Therefore, we can essentially consider the outputs of the three transposed convolutional blocks to be downsampled versions of the final output signal with temporal resolutions that are $\frac{1}{32} \times$, $\frac{1}{4} \times$, and $1 \times$ that of real-time. Taking this into account, we computed the STFTs of these downsampled signals using the following Fourier transform parameters:

- $\frac{1}{32} \times$ downsampled signal: 32 point Fourier transform, 32 sample window length, 8 sample hop length, corresponding to audio sampled at 500 Hz.
- $\frac{1}{4} \times$ downsampled signal: 256 point Fourier transform, 256 sample window length, 64 sample hop length, corresponding to audio sampled at 4 kHz.
- $1 \times$ downsampled signal: 1024 point Fourier transform, 1024 sample window length, 256 sample hop length, corresponding to audio sampled at 16 kHz.

Note that because the model uses 16 channels in its convolutional layers, the output of each transposed convolutional block also has 16 channels.

To demonstrate the results of our experiments, we use a sample utterance from the VCTK corpus to illustrate the results of the spectral analyses we performed on the internal representations of LVC-VC. The utterance is of the phrase “Please call Stella” and is spoken by a female voice. Figure 2 shows the linear spectrogram corresponding to this utterance. Figure 4 shows the results of computing the STFT on the 16 channels of the output of the first transposed convolution stack when LVC-VC is tasked with reconstructing the sample utterance. We can see that the model immediately begins to construct content information, as demonstrated by the emergence of three clear voiced segments in the STFT outputs (“Please”, “call”, “Stella”). In these voiced segments, we also see the emergence of the first harmonic band of the speaker’s F0 around the 200 Hz mark. In addition, we can notice that individual channels appear to model different aspects of the audio signal. Some channels appear to model the voiced segments of the utterance, while others appear to model the unvoiced segments, background noise, or silence.
Figure 3: Spectrogram of the sample utterance that we use to illustrate spectral analyses of the internal representations of LVC-VC.

Figure 4: Results of computing the STFT on the output of the first transposed convolutional stack of LVC-VC. Signals are at a $\frac{1}{32}$ × temporal resolution compared to real-time.

Figures 5 and 6 show the corresponding STFT visualizations for the outputs of the second and third transposed convolution stacks, respectively. We see that similar patterns emerge in terms of different channels appearing to correspond to different aspects of the final time domain audio signal: voiced segments, unvoiced segments, background noise, and silence. In Figure 5, we can notably see one channel on the bottom-right that appears to encode the spectral envelope and the formant frequencies of the utterance over time. We also see the formation of more harmonic frequencies and the overall F0 contour, indicating the gradual addition of more detailed speaker and content information as the signal propagates through the model.

Overall, these results indicate that the different channels of LVC-VC’s convolutional layers encode the various aspects of a speech signal, such as voiced and unvoiced segments, vocal cord harmonics, formants, silence, and noise. LVC-VC thus appears to generate audio by starting off with incorporating high-level speaker and content features in the lower layers of the model, and then gradually adding in more fine-grained speaker and content information as it dilates the signal across the time domain and spectrum. These results are perhaps not entirely surprising; we can see them as analogous to the way in which convolutional filters in deep computer vision models learn to encode different aspects of images in their layers, such as edges, colors, and patterns [60, 61].
B.2 Incorporation of Speaker and Content Information

Now that we have gained an intuition of how LVC-VC synthesizes audio overall, we would like to see how it incorporates speaker and content information during the audio generation process. To do this, we performed experiments where we ablated certain input features and performed spectral analyses of the resulting outputs.
Speaker information. To analyze how LVC-VC incorporates speaker information in the speech generation process, we zeroed out the speaker embedding $s$ and quantized log median F0 $m$ and made the model generate audio using only content features. Then, we performed spectral analyses in the same way as above by computing the STFTs of the intermediate outputs of the transposed convolution stacks as well as of the final output signal. Figure 7 illustrates the final output of the model when it generates audio with no speaker information, and Figure 8 illustrates the intermediate outputs of the transposed convolution stacks. For brevity, we only include visualizations for 4 out of the 16 channels for each of the intermediate outputs.

We see that the spectral envelope and formants of the utterance are well preserved at each intermediate layer and in the final output signal, indicating that the content information has been passed through the model properly. However, we also see that the speaker’s F0 band and its harmonics do not form at any point, indicating that the speaker’s vocal characteristics have not been imparted onto the output signal. Intuitively, this makes sense; since the model does not have any conditioning information about the speaker’s identity, there is no way for it to determine the characteristics of the speaker’s voice. These results demonstrate that the content-related features we feed into LVC-VC properly transfer the content information of the source utterance to the output of the model without allowing any speaker information through.

Content information. To analyze how LVC-VC incorporates content information, we made the model generate audio after zeroing out the low-quefrency liftered mel spectrogram $H$. However, we did continue to provide the normalized F0 contour $p_{\text{norm}}$ in order to see how it would interact with the speaker information in $s$ and $m$. Figure 9 illustrates the final output of the model when it generates audio with no content information, and Figure 10 illustrates the intermediate outputs of the transposed convolution stacks.

The outputs are essentially the reverse of what we see when we zero out the speaker information. Voiced segments, which are specified by the normalized F0 contour $p_{\text{norm}}$, still appear to be generated more or less properly. The shape and contour of the F0 and its harmonic frequencies still form somewhat normally, demonstrating that $p_{\text{norm}}$ is being successfully “un-normalized” by combining it with the speaker information. However, as expected, the spectral envelope and formants do not form at all, resulting in a situation in which the F0 contour is present but there is no content information for it to correspond to. In addition, we see that in the absence of the knowledge of which unvoiced frames are silent, the model appears to fill in the gaps with white noise-like artifacts up to around 6,000 Hz (as seen in Figure 9). These results indicate that the speaker-related features that are fed into LVC-VC allow the model to effectively express the speaker’s vocal characteristics in the output audio without passing any content information through.

Summary. Overall, the results support the following intuition: LVC-VC combines speaker information with the content features in such a way that “un-warps” the low-quefrency liftered spectrogram $H$ and “un-normalizes” the normalized F0 contour $p_{\text{norm}}$. This indicates that our strategy for LVC-VC’s design—combining carefully designed input features in an end-to-end vocoder-like framework, rather than training the model to explicitly disentangle and recombine the information in an utterance—is an effective way of synthesizing audio using the speaker and content information taken from different utterances.
Figure 7: Spectrogram computed from the output signal generated by LVC-VC when the speaker embedding $s$ and quantized log median F0 $m$ have been zeroed out.

Figure 8: Results of computing the STFT on the intermediate outputs of LVC-VC after each transposed convolutional stack when the speaker embedding $s$ and quantized log median F0 $m$ have been zeroed out.
Figure 9: Spectrogram computed from the output signal generated by LVC-VC when the low-quefrency liftered mel spectrogram $H$ has been zeroed out.

(a) Output of first transposed convolution stack.

(b) Output of second transposed convolution stack.

(c) Output of third transposed convolution stack.

Figure 10: Results of computing the STFT on the intermediate outputs of LVC-VC after each transposed convolutional stack when the low-quefrency liftered mel spectrogram $H$ has been zeroed out.
C Design of Amazon Mechanical Turk Surveys for Subjective Evaluations

C.1 Subjective listening test for Naturalness (MOS)

For the subjective listening test for evaluating the naturalness of utterances, subjects were asked to assign a score from 1–5 on the naturalness of the audio. 1 meant that the utterance did not sound natural at all and 5 meant that the utterance sounded completely natural. Each utterance was evaluated by two subjects. Participants were paid USD $0.05 per response; the estimated hourly wage was USD $12.00. The full instructions given to the subjects were as follows:

Listen to the sample of speech, which may or may not have been generated by a computer, and assess the quality of the audio based on how close it is to natural speech.
You should wear headphones and work in a quiet environment.

The rubric for evaluation was:

- Excellent (5) – Completely natural speech
- Good (4) – Mostly natural speech
- Fair (3) – Equally natural and unnatural speech
- Poor (2) – Mostly unnatural speech
- Bad (1) – Completely unnatural speech

Figure 11: Amazon Mechanical Turk instructions for subjective evaluations of naturalness.

C.2 Subjective listening test for Intelligibility

For the subjective listening test for evaluating the intelligibility of utterances, subjects were asked to assign a score from 1–5 on the intelligibility of the audio. 1 meant that the utterance was not understandable at all and 5 meant that the utterance was completely understandable. Each utterance was evaluated by two subjects. Participants were paid $0.05 per response; the estimated hourly wage was USD $12.00. The full instructions given to the subjects were as follows:

Listen to the sample of speech, which may or may not have been generated by a computer, and assess how understandable the words being spoken are. Some of the audio samples may sound somewhat degraded or distorted. Please try to listen beyond the audio quality and make your rating based on the clarity of the pronunciation of the words.
You should wear headphones and work in a quiet environment.
The rubric for evaluation was:

- Excellent (5) – Completely intelligible speech
- Good (4) – Mostly intelligible speech
- Fair (3) – Somewhat intelligible speech
- Poor (2) – Mostly unintelligible speech
- Bad (1) – Completely unintelligible speech

Figure 12: Amazon Mechanical Turk instructions for subjective evaluations of intelligibility.

C.3 Subjective listening test for Similarity

For the subjective listening test for evaluating the similarity of two utterances, subjects were asked to indicate whether the two voices sounded like the could have come from the same speaker. Each utterance pair was evaluated by two subjects. Participants were paid $0.10 per response; the estimated hourly wage was USD $12.00. The full instructions given to the subjects were as follows:

Listen to the two speech samples, which may or may not have been generated by a computer. Please give an assessment as to whether you think the two samples could have been said by the same speaker. Some of the audio samples may sound somewhat degraded or distorted. The speed and accent with which the speech was spoken may also be different. Please try to listen beyond these differences and concentrate on deciding whether the voices themselves sound similar or not. You should wear headphones and work in a quiet environment.

The rubric for evaluation was:

- Same speaker – Absolutely sure
- Same speaker – Not sure
- Different speaker – Not sure
- Different speaker – Absolutely sure
Listen to the two speech samples, which may or may not have been generated by a computer. Please give an assessment as to whether you think the two samples could have been said by the same speaker.

Some of the audio samples may sound somewhat degraded or distorted. The speed and accent with which the speech was spoken may also be different. Please try to listen beyond these differences and concentrate on deciding whether the voices themselves sound similar or not.

You should wear headphones and work in a quiet environment.

![Image](image_url)

Figure 13: Amazon Mechanical Turk instructions for subjective evaluations of similarity.

## D Full Results of Ablation Studies on LVC-VC

### Table 5: Seen-to-seen voice conversion evaluation results for various ablations of LVC-VC.

| Model                      | WER | CER | EER  | NISQA     |
|----------------------------|-----|-----|------|-----------|
| LVC-VC                     | 22.69 | 9.55 | 18.50 | 4.00 ± 0.08 |
| w/o GMM embeddings         | 23.33 | 10.18 | 15.50 | 3.86 ± 0.09 |
| w/o SSC loss               | 15.79 | 5.97 | 68.00 | 4.16 ± 0.08 |
| w/o warping $H$            | 19.80 | 8.51 | 41.50 | 3.96 ± 0.09 |
| w/o $p_{\text{norm}}$      | 23.11 | 10.22 | 18.50 | 3.71 ± 0.10 |
| w/o $m$                    | 22.34 | 9.05 | 21.00 | 3.91 ± 0.09 |

### Table 6: Unseen-to-seen voice conversion evaluation results for various ablations of LVC-VC.

| Model                      | WER | CER | EER  | NISQA     |
|----------------------------|-----|-----|------|-----------|
| LVC-VC                     | 17.37 | 7.03 | 20.00 | 3.89 ± 0.14 |
| w/o GMM embeddings         | 16.64 | 7.10 | 16.25 | 3.69 ± 0.15 |
| w/o SSC loss               | 12.25 | 4.38 | 71.25 | 4.04 ± 0.13 |
| w/o warping $H$            | 19.56 | 8.40 | 42.50 | 3.81 ± 0.17 |
| w/o $p_{\text{norm}}$      | 19.01 | 7.61 | 25.00 | 3.66 ± 0.15 |
| w/o $m$                    | 18.28 | 7.46 | 20.00 | 3.82 ± 0.13 |

### Table 7: Unseen-to-unseen voice conversion evaluation results for various ablations of LVC-VC.

| Model                      | WER | CER | EER  | NISQA     |
|----------------------------|-----|-----|------|-----------|
| LVC-VC                     | 20.10 | 8.29 | 26.25 | 3.50 ± 0.13 |
| w/o GMM embeddings         | 26.92 | 11.11 | 25.00 | 2.89 ± 0.14 |
| w/o SSC loss               | 16.78 | 6.64 | 68.75 | 3.83 ± 0.13 |
| w/o warping $H$            | 19.76 | 7.39 | 51.25 | 3.62 ± 0.17 |
| w/o $p_{\text{norm}}$      | 21.33 | 8.60 | 32.50 | 3.36 ± 0.18 |
| w/o $m$                    | 22.90 | 9.90 | 28.75 | 3.47 ± 0.14 |
Extension: Un-targeted Speaker Anonymization

A beneficial application of VC technologies is speaker anonymization; the vocal identity of a speaker can be masked by transforming a speaker's voice to sound like someone else. Although anonymization can be performed by using voice conversion to change the perceived identity of a speaker to another individual, this requires that a specific target speaker—a real person—be chosen. This brings up a variety of potential issues. First, target speakers must give permission for their voices to be used for the purposes of VC-based anonymization. In addition, there likely needs to be a reasonably large number of speakers in the pool of possible target voices in order to have enough options to satisfactorily anonymize a wide variety of source speakers. Finally, there is a major ethical issue regarding the potential of VC technologies to impersonate individuals (audio deepfakes). Given this, a VC-based anonymization methodology that could synthesize speech in a rich variety of non-existent speakers’ voices would be an attractive prospect.

Inspired by the recently introduced task of speaker generation [62], we introduce a methodology to use VC models for speaker anonymization without the need for specifying a target speaker. Although we discuss this approach in the context of LVC-VC, it can feasibly be used to extend the capabilities of any VC model that incorporates a speaker encoder.

E.1 Sampling arbitrary speaker embeddings

At a high level, our idea for performing un-targeted speaker anonymization is straightforward: it involves modeling the distribution of speaker embeddings generated by the speaker encoder of a VC model and then sampling from that distribution to obtain an arbitrary speaker embedding. That embedding is then used as the “target” speaker embedding for the VC model in order to change the vocal characteristics of a given source utterance.

To do this, we use the speaker encoder \( E_s \) of LVC-VC to extract embeddings from a large number of speakers. Specifically, we do this for the speakers in the development set of the VoxCeleb1 dataset [63]; we randomly sample up to 50 utterances from each speaker, for a total of 60,402 embeddings from the 1,211 speakers in the dataset. Then, we fit a Gaussian mixture model (GMM) with 8 mixture components to the extracted embeddings; we refer to this GMM as \( S \). To perform anonymization, we sample from \( S \) to extract an arbitrary speaker embedding \( \tilde{s} \), which can then be fed into LVC-VC to transform and anonymize a source utterance.

E.2 Selecting F0

Recall that in addition to the speaker embedding, one of the speaker-related features that is fed into LVC-VC is a one-hot quantized representation of the target speaker’s log F0. Therefore, randomly sampling a speaker embedding is not sufficient on its own to perform anonymization using LVC-VC; we must also specify the target voice’s median F0 value in order to transform the source utterance.

It is possible to select a target F0 by randomly choosing a value from the range that is quantized. However, this could result in an F0 that is very different from the expected voice that the speaker embedding encodes. For example, the speaker embedding might correspond to a male-sounding voice, but the selected F0 value could be very high, corresponding to a female-sounding voice. We found during experiments that this mismatch could result in anonymized speech that sounded noisy, buzzy, or otherwise unnatural compared to performing conversion on a specified target speaker.

To solve this issue, we trained a model \( F \) to predict the F0 of a voice from its corresponding speaker embedding \( s \in \mathbb{R}^{512} \). The model is a feedforward neural network with one hidden layer of 512 units using ReLU activation and an output layer with 1 unit using sigmoid activation. Dropout [64] is used with \( p = 0.5 \). \( F \) is trained to predict the raw median F0 value of the voice corresponding to a speaker embedding. Specifically, it predicts a value between the minimum and maximum frequencies that are quantized by \( m \) (65.4 Hz (‘C2’) and 523.3 Hz (‘C5’)), which are normalized to be in the range \([0, 1]\).

We trained our F0 predictor model on the speaker embeddings of 40,017 utterances from 99 speakers in the VCTK corpus [41] and tested on the 4,225 utterances from the remaining 10 speakers. This was the same speaker-wise train-test split as we used for training LVC-VC and other baseline voice conversion models (see Section 4.1). We used the AdamW optimizer [42] with a learning rate of 1e-4 and \( \beta_1 = 0.9, \beta_2 = 0.999 \). On utterances from the test set, the model achieved a mean absolute F0
error of 12.43 Hz, with a standard deviation of 10.58 Hz. We determined that this level of performance was satisfactory for our purposes, since the objective of the F0 predictor was simply to approximate F0 values that would somewhat match random speaker embeddings that are sampled for performing anonymization.

E.3 Inference

To perform un-targeted speaker anonymization with LVC-VC, we first sample an embedding \( \hat{s} \) from \( S \). Then, we use the F0 predictor network \( F \) to estimate the median F0 of the voice that would correspond to \( \hat{s} \) and convert it into the one-hot quantized feature \( \hat{m} \). Given a source utterance \( x_1 \) with associated content features \( (H_1, \text{p}^{\text{norm,1}}) \), an anonymized version of that utterance \( \tilde{x}_1 \) can be generated as follows:

\[
\tilde{x}_1 = G(z, H_1, \text{p}^{\text{norm,1}}, \hat{s}, \hat{m}).
\]  

(18)

E.4 Experiments and results

We performed un-targeted voice anonymization on the same 80 utterances that were used for evaluating unseen-to-seen targeted voice conversion on the VCTK corpus. Here, because the goal is to perform anonymization, lower Similarity and higher EER scores are better; 0.00% Similarity and 50.00% EER would indicate perfect anonymization for all utterances. Table 8 shows the results of performing anonymization in this way.

We find that our anonymization method appears to work quite well for masking speaker identity. Similarity decreased from 93.75% to 26.88% and EER increased from 0.00% to 31.25%, indicating that speakers’ voices were successfully masked in most cases for both human listeners and the ASV model. We note that we did not put any particular constraints when sampling speaker embeddings for anonymization (e.g., by enforcing that the sampled embeddings have a minimum cosine distance from the source utterance’s embedding); adding these constraints would likely result in an even greater degree of anonymization on average.

As a side effect of anonymization, however, we do see that the naturalness, intelligibility, and general audio quality of anonymized speech decreases compared to the original ground truth audio (although WER and CER are not as adversely affected). This is likely a result of LVC-VC being fed speaker embeddings that have been sampled from a latent space of embeddings that it has not seen before. We hypothesize that this quality degradation could be mitigated to some extent if LVC-VC were trained on a larger dataset with many more speakers; being exposed to a wider space of speaker embeddings during training could help the model generalize better to arbitrary speaker embeddings during inference. The latent space from which speaker embeddings are sampled could also be learned jointly along with the speech synthesis model. This remains a potential direction of future work.