Global Rhythm Style Transfer Without Text Transcriptions

Kaizhi Qian [*1 2] **Yang Zhang** [*1 2] **Shiyu Chang** [1 2] **Jinjun Xiong** [2] **Chuang Gan** [1 2] **David Cox** [1 2]
Mark Hasegawa-Johnson [3]

Abstract

Prosody plays an important role in characterizing the style of a speaker or an emotion, but most non-parallel voice or emotion style transfer algorithms do not convert any prosody information. Two major components of prosody are pitch and rhythm. Disentangling the prosody information, particularly the rhythm component, from the speech is challenging because it involves breaking the synchrony between the input speech and the disentangled speech representation. As a result, most existing prosody style transfer algorithms would need to rely on some form of text transcriptions to identify the content information, which confines their application to high-resource languages only. Recently, SPEECHSPLIT (Qian et al., 2020b) has made sizeable progress towards unsupervised prosody style transfer, but it is unable to extract high-level global prosody style in an unsupervised manner. In this paper, we propose AUTOPST, which can disentangle global prosody style from speech without relying on any text transcriptions. AUTOPST is an Autoencoder-based Prosody Style Transfer framework with a thorough rhythm removal module guided by self-expressive representation learning. Experiments on different style transfer tasks show that AUTOPST can effectively convert prosody that correctly reflects the styles of the target domains.

1. Introduction

Speech contains many layers of information. Besides the speech content, which can roughly be transcribed to text for many languages, prosody also conveys rich information about the personal, conversational, and world context within which the speaker expresses the content. There are two major constituents of prosody. The first is *rhythm*, which summarizes the sequence of phone durations, and expresses phrasing, speech rate, pausing, and some aspects of prominence. The second is *pitch*, which reflects intonation.

Recently, non-parallel speech style transfer tasks have achieved rapid progress thanks to the advancement of non-parallel deep style transfer algorithms. Speech style transfer refers to the tasks of transferring the source speech into the style of the target domain, while keeping the content unchanged. For example, in voice style transfer, the domains correspond to the speaker identities. In emotion style transfer, the domains correspond to the emotion categories. In both of these tasks, prosody is supposed to be an important part of the domain style — different speakers or emotions have distinctive prosody patterns. However, few of the state-of-the-art algorithms in these two applications can convert the prosody aspect at all. Typically, the converted speech would almost always maintain the same pace and pitch contour shape as the source speech, even if the target speaker or emotion has a completely different prosody style.

The fundamental cause of not converting prosody is that disentangling the prosody information, particularly the rhythm aspect, is very challenging. Since the rhythm information corresponds to how long the speaker utters each phoneme, deriving a speech representation with the rhythm information removed implies breaking the temporal synchrony between the speech utterance and the representation, which has been shown difficult (Watanabe et al., 2017; Kim et al., 2017) even for supervised tasks (*e.g.* automatic speech recognition (ASR)), using state-of-the-art asynchronous sequence-to-sequence architectures such as Transformers (Vaswani et al., 2017).

Due to this challenge, most existing prosody style transfer algorithms are forced to use text transcriptions to identify the content information in speech, and thereby separate out the remaining information as style (Biadsy et al., 2019). However, such methods are language-dependent and cannot be applied to low-resource languages with few text transcriptions. Although there are some sporadic attempts to disentangle prosody in an unsupervised manner (Polyak & Wolf, 2019), their performance is limited. These algorithms

[*]Equal contribution [1]MIT-IBM Watson AI Lab, USA [2]IBM Thomas J. Watson Research Center, USA [3]University of Illinois at Urbana-Champaign, USA. Correspondence to: Yang Zhang <yang.zhang2@ibm.com>, Kaizhi Qian <kaizhiqian@gmail.com >.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

typically consist of an auto-encoder pipeline with a resampling module at the input to corrupt the rhythm information. However, the corruption is often so mild that most rhythm information can still get through. SPEECHSPLIT (Qian et al., 2020b) can achieve a more thorough prosody disentanglement, but it needs access to a set of ground-truth fine-grained local prosody information in the target domain, such as the exact timing of each phone and exact pitch contour, which is unavailable in the aforementioned style transfer tasks that can only provide high-level global prosody information. In short, global prosody style transfer without relying on text transcriptions or local prosody ground truth largely remains unresolved in the research community.

Motivated by this, we propose AUTOPST, an unsupervised speech decomposition algorithm that 1) does not require text annotations, and 2) can effectively convert prosody style given domain summaries (*e.g.* speaker identities and emotion categories) that only provide high-level global information. As its name indicates, AUTOPST is an Autoencoder-based Prosody Style Transfer framework. AUTOPST introduces a much more thorough rhythm removal module guided by the self-expressive representation learning proposed by Bhati et al. (2020), and adopts a two-stage training strategy to guarantee passing full content information without leaking rhythm. Experiments on different style transfer tasks show that AUTOPST can effectively convert prosody that correctly reflects the styles of the target domains.

2. Related Work

Prosody Disentanglement Several prosody disentanglement techniques are found in expressive text-to-speech (TTS) systems. Skerry-Ryan et al. (2018) introduced a Tacotron based speech synthesizer that can disentangle prosody from speech content by an auto-encoder based representation. Wang et al. (2018) further extracts global styles by quantization. Mellotron (Valle et al., 2020) is a speech synthesizer that captures and disentangles different aspects of the prosody information. CHiVE (Kenter et al., 2019) explicitly extracts and utilize prosodic features and linguistic features for expressive TTS. However, these TTS systems all require text transcriptions, which, as discussed, makes the task easier but limits their applications to high-resource language. Besides TTS systems, Parrotron (Biadsy et al., 2019) disentangles prosody by encouraging the latent codes to be the same as the corresponding phone representation of the input speech. Liu et al. (2020) proposed to disentangle phoneme repetitions by vector quantization. However, these systems still require text transcriptions. Polyak & Wolf (2019) proposed a prosody disentanglement algorithm that does not rely on text transcriptions, which attempts to remove the rhythm information by randomly resampling the input speech. However, the effect of their prosody conversion is not very pronounced. SPEECHSPLIT (Qian et al., 2020b) can disentangle prosody with better performance, but it relies on fine-grained prosody ground-truth in the target domain.

Voice Style Transfer Many style transfer approaches have been proposed for voice conversion. VAE-VC (Hsu et al., 2016) and VAE-GAN (Hsu et al., 2017) directly learns speaker-independent content representations using a VAE. ACVAE-VC (Kameoka et al., 2019) encourages the converted speech to be correctly classified as the target speaker by classifying the output. In contrast, Chou et al. (2018) discourages the latent code from being correctly classified as the source speaker by classifying the latent code. Inspired by image style transfer, Gao et al. (2018) and Kameoka et al. (2018) adapted CycleGan (Kaneko & Kameoka, 2017) and StarGan (Choi et al., 2018) respectively for voice conversion. Later, CDVAE-VC was extended by directly applying GAN (Huang et al., 2020) to improve the degree of disentanglement. Chou & Lee (2019) uses instance normalization to further disentangle timbre from content. StarGan-VC2 (Kaneko et al., 2019) refines the adversarial network by conditioning the generator and discriminator on both the source and the target speaker. AUTOVC (Qian et al., 2019) disentangles the timbre and content by tuning the information-constraining bottleneck of a simple autoencoder. Later, Qian et al. (2020a) fixed the pitch jump problem of AUTOVC by F0 conditioning. Besides, the time-domain deep generative model is also gaining popularity (Niwa et al., 2018; Nachmani & Wolf, 2019; Serrà et al., 2019). However, these methods only focus on converting timbre, which is only one of the speech components.

Emotion Style Transfer Most existing emotion style transfer algorithms disentangle prosody information using parallel data. Early methods (Tao et al., 2006; Wu et al., 2009) use classification and regression tree to disentangle prosody. Later, statistical methods, such as GMM (Aihara et al., 2012) and HMM (Inanoglu & Young, 2009), and deep learning methods (Luo et al., 2016; Ming et al., 2016) are applied. However, these approaches use parallel data. Recently, non-parallel style transfer methods (Zhou et al., 2020a;b) are applied to emotion style transfer to disentangle prosody. However, these methods are unable to explicitly disentangle rhythm information.

3. The Challenges of Disentangling Rhythm

In this section, we will provide some background information on why disentangling rhythm has been difficult.

3.1. Rhythm Information

Figure 1 shows two utterances of the same word *"Please call Stella"*, with the phone segments marked on the x-axis.

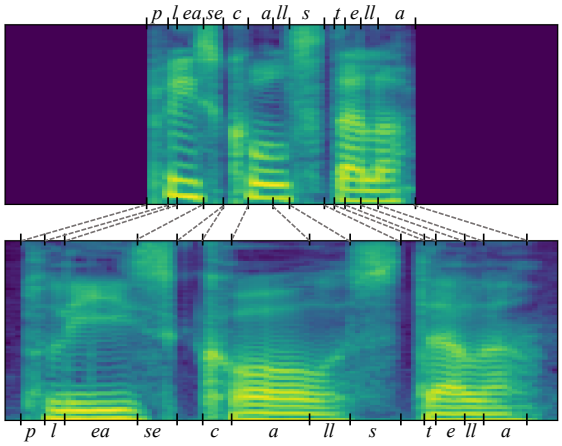


Figure 1. Mel-spectrograms of two utterances of *"Please call Stella"* with different speech rates. The phone segments are marked on the x-axis. The grey dashed lines between the two spectrograms mark the phone alignment, which shows that the changes in duration are disproportional across different phones.

As shown, although the content is almost the same across the two utterances, each phone in the second utterance is repeated for more frames than in the first utterance. In this paper, we will measure rhythm as the number of frame-aligned repetitions of each phone symbol.

Formally, denote $\boldsymbol{X}(t)$ as the spectrogram of a speech utterance, where t represents the frame index. Denote $\boldsymbol{S}$ as the vector of phone symbols contained in an utterance, which we also refer to as the content information. Denote $\boldsymbol{R}$ as a vector of the number of repetitions of each phone symbol in $\boldsymbol{S}$, which we also refer to as the rhythm information. Then $\boldsymbol{S}$ and $\boldsymbol{R}$ represent two information components in $\boldsymbol{X}(t)$. Therefore, the task of disentangling rhythm involves removing phone repetitions while retaining phone identities. Formally, we would like to derive a hidden representation $\tilde{\boldsymbol{Z}}$ from the speech $\boldsymbol{X}$, *i.e.* $\tilde{\boldsymbol{Z}} = f(\boldsymbol{X})$, according to the following objective

$$\min_{\tilde{\boldsymbol{Z}}=f(\boldsymbol{X})} I(\boldsymbol{R}; \tilde{\boldsymbol{Z}}), \tag{1}$$

$$\text{s.t. } I(\boldsymbol{S}; \tilde{\boldsymbol{Z}}) = I(\boldsymbol{S}; \boldsymbol{X}), \tag{2}$$

where $I(\cdot;\cdot)$ denotes mutual information. If the text transcriptions were available, this objective could be achieved by training an asynchronous sequence-to-sequence model, such as the Transformer (Vaswani et al., 2017), to predict the phone sequence from input speech. However, without the phonetic labels, it remains an unresolved challenge to uncover the phonetic units in an unsupervised manner.

3.2. Disentangling Rhythm by Resampling

Existing unsupervised rhythm disentanglement techniques seek to obscure the rhythm information by temporally resampling the speech sequence or its hidden representation sequence. One common resampling strategy, as proposed by Polyak & Wolf (2019), involves three steps. First, the input sequence is separated into segments of random lengths. Second, for each segment, a sampling rate is randomly drawn from some distribution. Finally, the corresponding segment is resampled at this sampling rate.

To gauge how much rhythm information the random resampling can remove, consider two hypothetical speech sequences, $\boldsymbol{X}(t)$ and $\boldsymbol{X}'(t)$, with the same content information $\boldsymbol{S}$, but with different rhythm information $\boldsymbol{R}$ and $\boldsymbol{R}'$ respectively. If random resampling is to reduce the rhythm information in each utterance, there must be a non-zero probability that the random resampling temporally aligns them (hence making them identical because their only difference is rhythm). Thus, the original rhythm information distinguishing the two utterances is removed. Formally, denote $\tilde{\boldsymbol{Z}}(t)$ and $\tilde{\boldsymbol{Z}}'(t)$ as the resampled outputs of the two unaligned but otherwise identical sequences. Then a necessary condition for reduction in rhythm information is that

$$Pr(\tilde{\boldsymbol{Z}}(t) = \tilde{\boldsymbol{Z}}'(t)) > 0. \tag{3}$$

Higher probability removes more rhythm information. If the probability is zero, then $I(\boldsymbol{R}; \tilde{\boldsymbol{Z}})$ will reach its upper bound, which is $I(\boldsymbol{R}; \boldsymbol{X})$. If the probability is one, then $I(\boldsymbol{R}; \tilde{\boldsymbol{Z}})$ would reach its lower bound, which is $I(\boldsymbol{R}; \boldsymbol{S})$. Please refer to Appendix A for a more formal analysis.

Therefore, we argue that the above random resampling algorithm does not remove much rhythm information, because it has a low probability of aligning any utterance pairs with the same content but different rhythm patterns. Figure 1 is a counterexample, where the duration of each phone changes disproportionally. The vowel "ea" gets more stretched than the consonants. Consider the case where the entire sequence falls into one random segment in the first step of the resampling algorithm. Then, due to the disproportional stretch among the phones, it is impossible to simultaneously align all the phones by uniformly stretching or squeezing the two utterances. If the utterances are broken into multiple random segments, it is possible to achieve the alignment, but this requires the number of random segments is greater than or equal to the number of phones, with at least one segment boundary between each pair of phone boundaries, whose probability of occurring is extremely low. In short, to overcome the limitation of the existing resampling scheme, we need better ways to account for the disproportional variations in duration across different phones.

4. The AUTOPST Algorithm

In this section, we will introduce AUTOPST, which can effectively overcome the aforementioned limitations.

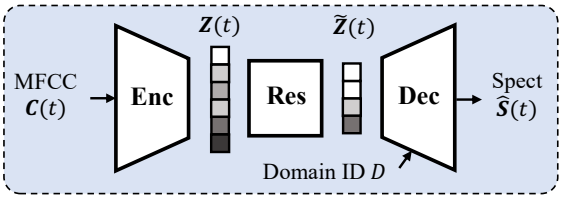


Figure 2. The framework of AUTOPST. "Enc", "Res" and "Dec" denote the encoder, the resampling module, and the decoder respectively; "Spect" represents the spectrogram. Each block in $\boldsymbol{Z}(t)$ and $\tilde{\boldsymbol{Z}}(t)$ represents a frame. The blocks with similar color shades denote that the corresponding frames have a high similarity.

4.1. The Framework Overview

As shown in Figure 2, AUTOPST adopts a similar autoencoder structure as AUTOVC (Qian et al., 2019). The decoder aims to reconstruct speech based on the output of the random resampling module and the domain identifiers, *e.g.*, the speaker identity. AUTOVC has been shown effective in disentangling the speaker's voice via its information bottleneck, but it does not disentangle pitch and rhythm.

Therefore, AUTOPST introduces three changes. First, instead of spectrogram, AUTOPST takes the 13-dimensional MFCC having little pitch information. Second, AUTOPST introduces a novel resampling module guided by self-expressive representation learning (Bhati et al., 2020), which can overcome the challenges mentioned in the previous section. Finally, AUTOPST adopts a two-stage training scheme to prevent leaking rhythm information.

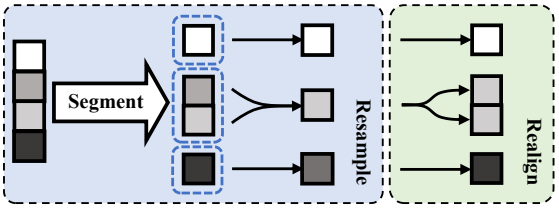
Formally, denote $\boldsymbol{X}(t)$ as the speech spectrogram, $\boldsymbol{C}(t)$ as the input MFCC feature, and D as the domain identifier. The reconstruction process is described as follows

$$\begin{aligned} &\boldsymbol{Z}(t) = \mathrm{Enc}(\boldsymbol{C}(t)),\ \tilde{\boldsymbol{Z}}(t) = \mathrm{Res}(\boldsymbol{Z}(t)), \\ &\hat{\boldsymbol{X}}(t) = \mathrm{Dec}(\tilde{\boldsymbol{Z}}(t), D) \longleftrightarrow \boldsymbol{X}(t) \end{aligned} \tag{4}$$

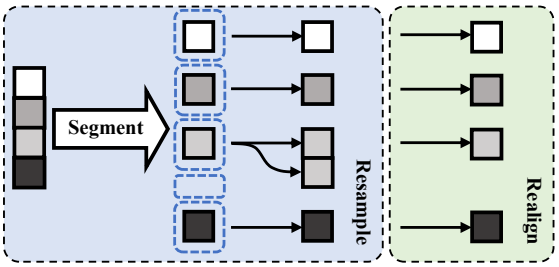
where Enc, Res, Dec stand for the encoder, the resampling module and the decoder, respectively. Sections 4.2 to 4.4 will introduce the random resampling module, and Section 4.6 will discuss the training scheme.

4.2. Similarity-Based Downsampling

Our resampling scheme capitalizes on the observation that the relatively steady segments in speech tend to have more flexible durations, such as the "ea" segment in Figure 1. We thus modify the self-expressive autoencoder (SEA) algorithm proposed by Bhati et al. (2020) into a similarity-based downsampling scheme. SEA derives a frame-level speech representation, which we denote as $\boldsymbol{A}(t)$, that contrastively promotes a high cosine similarity between frames that are similar, and a low cosine similarity between dissimilar frames. We then create a Gram matrix, G to record the



(a) The downsampling case ($\tau \leq 1$)



(b) The upsampling case ($\tau > 1$)

Figure 3. The resampling module (left) and realignment module (right) of AUTOPST. Merging arrows denote mean-pooling. Splitting arrows denote copying the input into multiple replicates. The blocks with similar color shades denote that the corresponding frames have a high similarity. According to the shade, frames 2 and 3 are similar, while the others frames are very dissimilar. (a) When $\tau \leq 1$, the sequence is segmented based on similarity, and each segment is merged to one code by mean-pooling. (b) When $\tau > 1$, each segment contains only one code. In addition, empty segments are inserted where the inter-temporal similarity is high, whose corresponding output positions replicate the previous codes.

cosine similarities between any frame pairs:

$$\mathsf{G}(t, t') = \frac{\boldsymbol{A}^T(t)\boldsymbol{A}(t')}{\|\boldsymbol{A}(t)\|_2 \|\boldsymbol{A}(t')\|_2}. \tag{5}$$

More details of the SEA algorithm can be found in Appendix B.1 and the original paper (Bhati et al., 2020).

As shown in the left panel of Figure 3(a), our downsampling scheme for $\boldsymbol{Z}(t)$ involves two steps. First, we break $\boldsymbol{Z}(t)$ into consecutive segments, such that the cosine similarity of $\boldsymbol{A}(t)$ are high within each segment, and that the cosine similarity drop across the segment boundaries. Second, each segment is merged into one code by mean-pooling.

Formally, denote the t_m as the left boundary for the m-th segment. Boundaries are sequentially determined. When all the boundaries up to t_m are determined, the next boundary t_{m+1} is set to t if t is the smallest time in (t_m, ∞) where the cosine similarity between t and t_m drops below a threshold:

$$\forall t' \in [t : t+1], \quad \mathsf{G}(t_m, t')) \leq \tau(t). \tag{6}$$

$\tau(t)$ is a pre-defined threshold that can vary across t. Section 4.3 will discuss how to set the threshold. After all the segments are determined, each segment is reduced to one

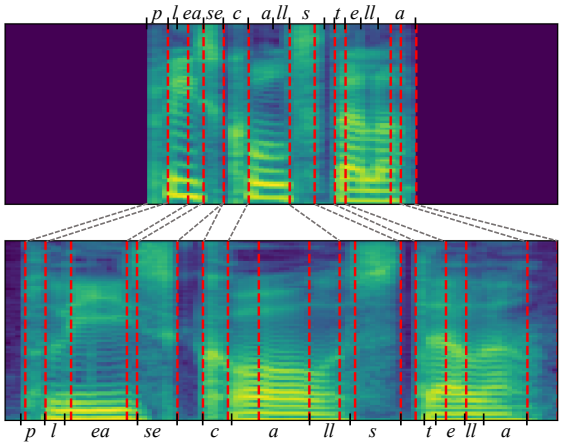


Figure 4. Segmentation results of the downsampling algorithm of the two utterances in Figure 1. The vertical dashed lines represent the segment boundaries. The dashed lines between the two spectrograms show good content alignment among the segments.

code by mean pooling, *i.e.*

$$\tilde{\boldsymbol{Z}}(m) = \text{meanpool}\big(\boldsymbol{Z}(t_m : t_{m+1} - 1)\big). \tag{7}$$

Figure 3(a) shows a toy example, where the input sequence of length four. The second and the third codes are very similar. Then with a proper choice of $\tau(t)$, the downsampling would divide the input sequence into three segments, and collapse each segment into one code by mean-pooling. Note that the threshold $\tau(t)$ governs how tolerant the algorithm is to dissimilarities. If $\tau(t) = 1$, each code will be assigned to an individual segment, leading to no length reduction.

Figure 4 shows the segmentation result of the two utterances shown in Figure 1, where the vertical dashed lines denote the segment boundaries. We can see that despite their significant difference in length, the two utterances are broken into approximately equal number of segments and the segments have a high correspondence in terms of content. Since the downsampled output is obtained by mean-pooling each segment, we can expect that their downsampled output would be very similar and temporally-aligned, which implies that the necessary condition for rhythm information loss (Equation (3)) is approximately satisfied.

4.3. Randomized Thresholding

For any fixed threshold τ in Equation (6), there is a trade-off between rhythm disentanglement and content loss. The lower the τ, the more rhythm information is removed, but the more content is lost as well. Ideally, during testing, we would like to set the threshold to 1 to pass full content information to $\tilde{\boldsymbol{Z}}(t)$, and make the decoder ignore all the rhythm information in $\tilde{\boldsymbol{Z}}(t)$. This can be achieved with a randomized thresholding rule.

To see why, notice that if the decoder were to use the rhythm information in $\tilde{\boldsymbol{Z}}(t)$, it must know the value of τ, because how the decoder (partially) recovers the rhythm information depends on how the rhythm information is collapsed, which is governed by τ. However, the large variations in speech rate, utterance length and rhythm patterns in the training speech would overshadow the variations in τ, making it extremely hard to estimate the value of τ. Thus, the decoder will ignore whatever rhythm information remains in $\tilde{\boldsymbol{Z}}(t)$.

We adopt a double-randomized thresholding scheme. We first randomly draw a global variable $G \sim \mathcal{U}[u_l, u_r]$ that is shared across the entire utterance, where $\mathcal{U}[u_l, u_r]$ denotes the uniform distribution within the interval $[u_l, u_r]$. Then to determine if time t should be the next segment boundary (*i.e.*, t_{m+1} in Equation (6)), we draw a local variable $L(t) \sim \mathcal{U}[G - 0.05, G + 0.05]$. Then

$$\tau(t) = L(t)\text{−quantile}\big[\mathsf{G}(t_m, t_m - b : t_m + b)\big]. \tag{8}$$

q−quantile$[\cdot]$ denotes taking the q-quantile, and b denotes the length of the sliding window within which the threshold is computed, which is set to 20 in our implementation.

The motivation for setting the two levels of randomization is that G can obscure the global speech rate information and $L(t)$ can obscure the local fine-grained rhythm patterns.

4.4. Similarity-Based Upsampling

To further obscure the rhythm information, we generalize our resampling module to accommodate upsampling. Just as downsampling aims to mostly shorten segments with higher similarity (hence decreasing the disproportionality), upsampling aims to mostly lengthen segments with higher similarity (hence increasing the disproportionality).

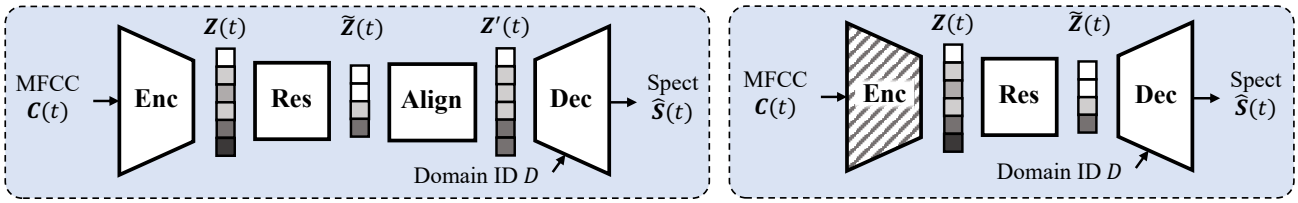
In the downsampling case, $\tau = 1$ implies no length reduction at all. We thus seek to extrapolate the case to $\tau > 1$, where the higher the τ, the more the sequence gets lengthened. Our upsampling algorithm achieves this by inserting new codes in between adjacent codes. Specifically, suppose all the boundaries up to t_m are determined. When $\tau(t) > 1$, according to Equation (6), t_{m+1} will definitely be set to t. In addition to this, we will add yet another sentence boundary to t, *i.e.* $t_{m+2} = t$, if

$$\forall t' \in [t : t + 1], \quad \mathsf{G}(t_m, t') \geq 1 - \tau(t). \tag{9}$$

In other words, we are inserting an empty segment for the $(m + 1)$-th segment (because $t_{m+1} = t_{m+2}$). During the mean-pooling stage, this empty segment will be mapped to the code at its left boundary, *i.e.*,

$$\tilde{\boldsymbol{Z}}(m) = \boldsymbol{Z}(t_m), \quad \text{if } t_m = t_{m+1}. \tag{10}$$

The non-empty segments will still be mean-pooled the same way as in Equation (7).



(a) Synchronous training. (b) Asynchronous training.

Figure 5. The two-stage training scheme of AUTOPST. "Align" denotes the re-alignment module. The striped encoder in (b) means that the parameters in it are frozen. The blocks with similar color shades denote that the corresponding frames have a high similarity.

The left panel of Figure 3(b) illustrates the upsampling process with the length-four toy example. Similar to the case of $\tau = 1$, all the codes are individually segmented. The difference is that a new empty segment is inserted after the third code, which is where the cosine similarity is very high. At the mean-pooling stage, this empty segment turns into an additional code that copies the previous code.

4.5. Summary of the Resampling Algorithm

To sum up, our resampling algorithm goes as follows.

- For each frame, a random threshold $\tau(t)$ is drawn from the distribution specified in Equation (8).
- If $\tau(t) < 1$, the current frame would be either merged into the previous segment, or start a new segment, depending on whether its similarity with the previous segment exceeds the τ, hence achieving downsampling (as elaborated in Section 4.2).
- If $\tau(t) \geq 1$, the current frame would form either one new segment or two new segments (by duplicating the current frame), depending on whether its similarity with the previous segment exceeds $1 - \tau(t)$, hence achieving upsampling (as elaborated in Section 4.4).
- Move onto the next frame and repeat the previous steps.

Because the threshold for each frame is random, an utterance could be downsampled at some parts, while upsampled at others. This would ensure the rhythm information is sufficiently scrambled. As a final remark, the random threshold distribution (Equation (8)) is governed by the percentile of the similarity, because the percentile has a direct correspondence to the length of the utterance after resampling. Appendix D.1 provides a visualization of how different thresholds affect the length of the utterance after resampling.

4.6. Two-Stage Training

Despite the resampling module, it is still possible for the encoder and decoder to find alternative ways to communicate the rhythm information that is robust against temporal resampling. Thus we introduce a two-stage training scheme to prevent any possible collusion.

The first stage of training, called the *synchronous training*, realigns $\tilde{\boldsymbol{Z}}(m)$ with $\boldsymbol{Z}(m)$, as shown in the right panels of Figure 3. Specifically, for the downsampling case, we copy each $\tilde{\boldsymbol{Z}}(m)$ to match the length of the original segment from which the code is mean-pooled; for the upsampling case, we delete the newly inserted $\tilde{\boldsymbol{Z}}(m)$. The network is then trained end-to-end to reconstruct the input with the realignment module, as shown in Figure 5(a). Since the decoder has full access to the rhythm information, the encoder will be trained to pass the content information and not the rhythm information. The second stage, called *asynchronous training*, removes the realignment module, freezes the encoder, and only updates the decoder, as shown in Figure 5(b).

5. Experiments

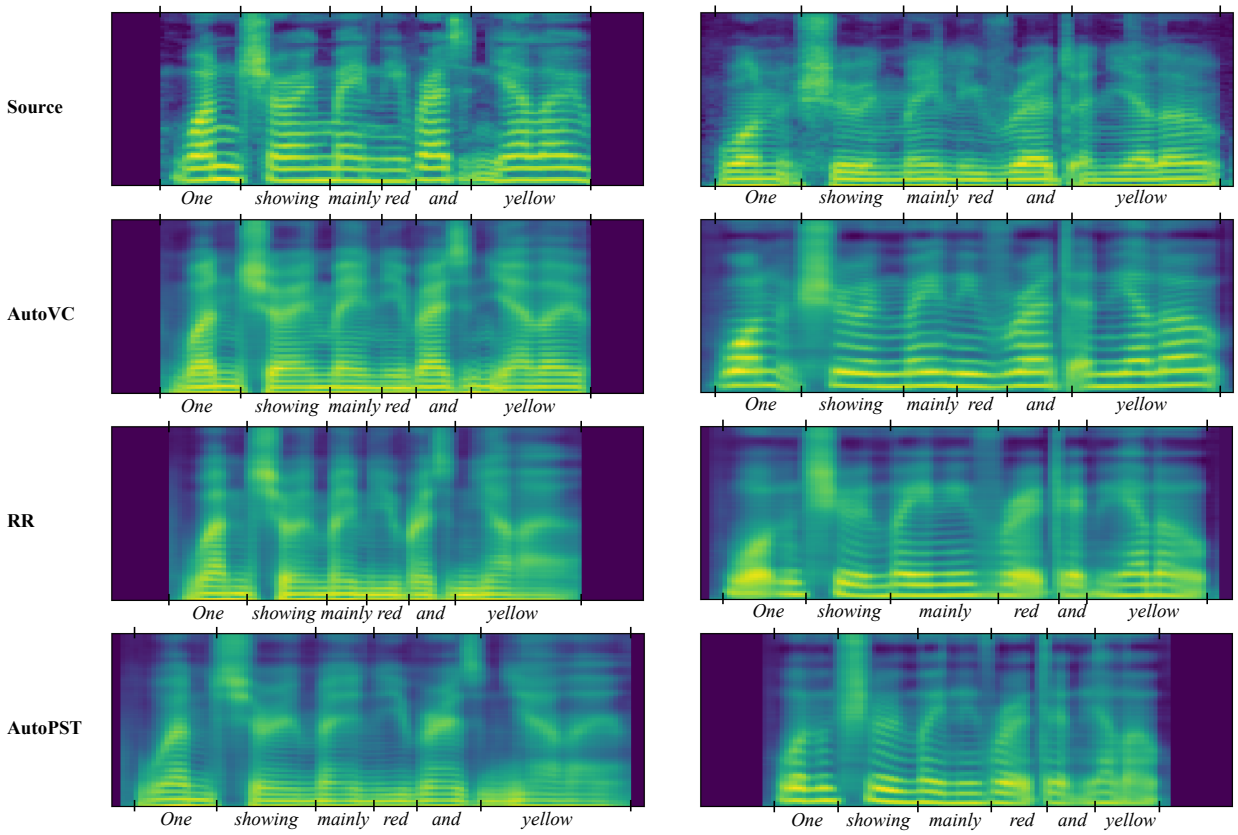
We evaluate AUTOPST on speech style transfer tasks. Additional experiment results can be found in Appendix D. We encourage readers to listen to our online demo audios[1].

5.1. Configurations

Architecture The encoder consists of eight 1×5 of convolution layers with group normalization (Wu & He, 2018). The encoder output dimension is set to four. The decoder is a Transformer with four encoder layers and four decoder layers. The spectrogram is converted back to waveform using a WaveNet vocoder (Oord et al., 2016). More hyperparameters setting details can be found in Appendix C.

Dataset Our dataset is VCTK (Veaux et al., 2016), which consists of 44 hours of speech from 109 speakers. We use this dataset to perform the voice style transfer task, so the domain ID is the speaker ID. We use 24 speakers for training and follow the same train/test partition as in (Qian et al., 2020b). We select the two fastest speakers and two slowest speakers from the seen speakers for evaluating rhythm style transfer. For further evaluation, we select two other speaker pairs with smaller rhythm differences. The

[1] https://auspicious3000.github.io/AutoPST-Demo



(a) Fast-to-slow conversion. (b) Slow-to-fast conversion.

Figure 6. Mel-spectrograms of the converted utterances of *"One showing mainly red and yellow"* between a fast speaker and a slow speaker. AUTOPST is the only algorithm that can significantly change the rhythm to match the target speakers' styles.

first pair consists of speakers whose speech rates are at 25% and 75% percentiles among all the test speakers; the second pair at 40% and 60%, respectively. More details can be found in Appendix C.

Baselines We introduce two baselines. The first is the F0-assisted AUTOVC (Qian et al., 2020a), an autoencoder-based voice style transfer algorithm. For the second baseline, we replace the AUTOPST's random resampling module with that of the SPEECHSPLIT (as introduced in Section 3.2). We refer to this baseline as RR (random resample).

5.2. Spectrogram Visualization

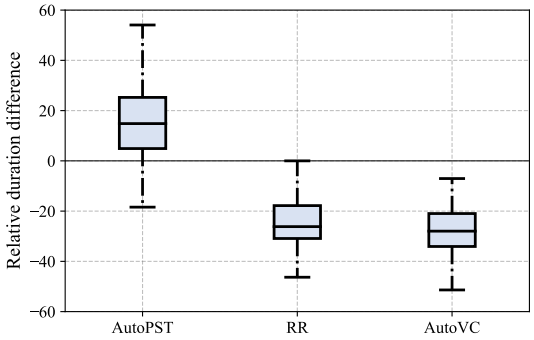
For a fast-slow speaker pair and one of their parallel sentences, *"One showing mainly red and yellow"*, Figure 6 shows the spectrograms of conversions from the slow speaker to the fast (left panel), and from the fast speaker to the slow (right panel). The word alignment is marked on the x-axis. As shown, all the algorithms can change the voice to the target speaker, as indicated by the average F0 and formant frequencies. However, only AUTOPST significantly changes the rhythm towards the desired direction. AUTOVC and RR barely change the rhythm. It is also worth noting that AUTOPST indeed learns the disproportionality in duration changes across phones, most duration changes occur in the steady vowel segments, *e.g.*, *"ow"* in *"yellow"*. This verifies that our similarity-based resampling scheme can effectively obscure the relative length of each phone.

5.3. Relative Duration Difference

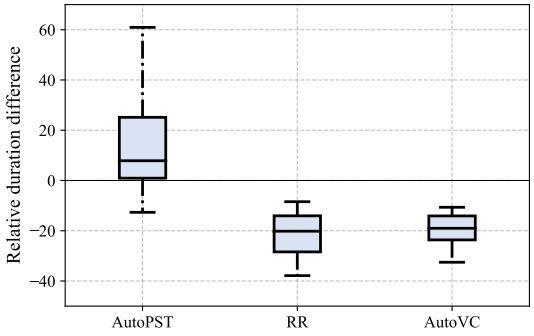
To objectively measure the extent to which each algorithm modifies rhythm to match the style of the target speaker, for each test sentence and each fast-slow speaker pair (the test set is a parallel dataset), we generate a fast-to-slow and a slow-to-fast conversion. If an algorithm does not change the rhythm at all, then the fast-to-slow version should have a shorter duration than its slow-to-fast counterpart; in contrast, if an algorithm can sufficiently obscure and ignore the rhythm information in the source speech, then it can flip the ordering. We compute relative duration difference as

$$\text{Relative Duration Difference} = (L_{\text{F2S}} - L_{\text{S2F}})/L_{\text{S2F}}, \quad (11)$$

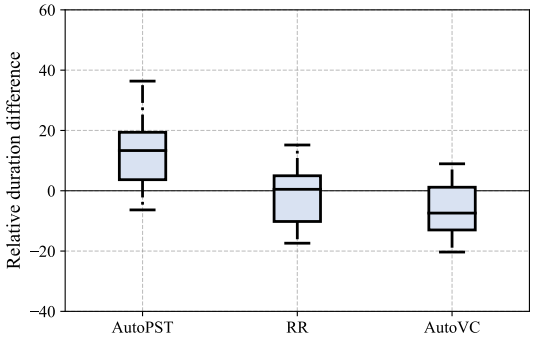
where L_{F2S} and L_{S2F} denote the lengths of fast-to-slow and slow-to-fast conversions, respectively. If the rhythm disen-



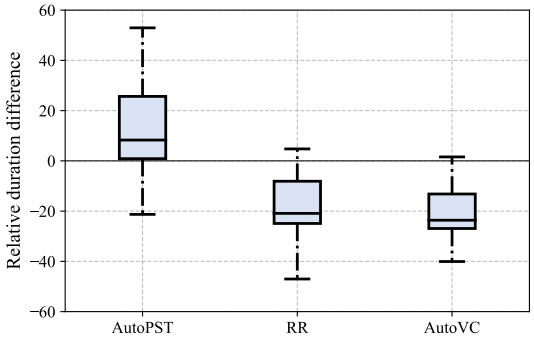
(a) VCTK (100% vs 0%)



(b) VCTK (75% vs 25%)



(c) VCTK (60% vs 40%)



(d) Emo-VDB

Figure 7. The box plot of the relative duration difference between low-to-fast conversion and fast-to-slow conversion of utterance pairs with the same content. Positive duration differences indicate more sufficient rhythm disentanglement. Percentiles in the subcaptions denote rankings in speech rate from fast to slow.

Table 1. Subjective evaluation results.

	AUTOPST	RR	AUTOVC
Timbre	**4.29 ± 0.032**	4.07 ± 0.037	4.26 ± 0.034
Prosody	**3.61 ± 0.053**	2.97 ± 0.063	2.64 ± 0.066
Overall	**3.99 ± 0.036**	3.63 ± 0.045	3.49 ± 0.052

tanglement is sufficient, this difference should be positive.

Figure 7(a) shows the box plot of the relative duration differences across all test sentences and all the top four fastest-slowest speaker pairs. Figure 7(b) and Figure 7(c) show the results on speaker pairs with smaller rhythm differences (75% vs 25% and 60% vs 40% respectively). As shown, only AUTOPST can achieve a positive average relative duration difference, which verifies its ability to disentangle rhythm. AUTOVC gets the most negative relative duration differences, which is expected because it does not modify duration at all. RR achieves almost the same negative duration differences, which verifies that its rhythm disentanglement is insufficient.

5.4. Subjective Evaluation

To better evaluate the overall quality of prosody style transfer, and whether prosody style transfer improves the perceptual similarity to the target speaker, we performed a subjective evaluation on Amazon Mechanical Turk. Specifically, in each test unit, the subject first listens to two randomly ordered reference utterances from the source and target speaker respectively. Then the subject listens to a converted utterance from the source to the target speaker by one of the algorithms. Note that the content of the converted utterance is different from that in the reference utterances. Finally, the subject is asked to assign a score of 1-5 to describe the similarity to the target speaker in one of the three aspects: *prosody similarity*, *timbre similarity*, and *overall similarity*. A score of 5 means entirely like the target speaker; 1 means completely like the source speaker; 3 means somewhat between the two speakers. Each algorithm has 79 utterances, where each utterance is assigned to 5 subjects.

Table 1 shows the subjective similarity scores. As shown, AUTOPST has a significant advantage in terms of prosody similarity over the baselines, which further verifies that AUTOPST can generate a prosody style that is perceived as similar to the target speaker. In terms of timbre similarity, AUTOPST performs on-par with AUTOVC, and the gaps among the three algorithms are small because all three algorithms apply the same mechanism to disentangle timbre.

For the overall similarity, it is interesting to see how the subjects weigh the different aspects in their decisions. Specifically, although AUTOVC can satisfactorily change the timbre, it is still perceived as only very slightly similar to the target speaker, because the prosody is not converted at all.

In contrast, the AUTOPST results, with all aspects transferred, are perceptually much more similar to the target speaker. This result shows that prosody indeed plays an important role in characterizing a speaker's style and should be adequately accounted for in speech style transfer.

5.5. Restoring Abnormal Rhythm Patterns

So far, our experiments have mostly focused on the overall speech rate. One question we are interested in is whether AUTOPST can recognize fine-grained rhythm patterns, or can only adjust speech rate globally. To study this question, we modify the utterance *"One showing mainly red and yellow"* in Figure 6(a) by stretching *"yellow"* by two times, creating an abnormal rhythm pattern. We then let RR and AUTOPST reconstruct the utterance from this abnormal input. If these algorithms can recognize fine-grained rhythm patterns, they should be able to restore the abnormality.

Figure 8 shows the reconstructed spectrogram from the abnormal input. As shown, RR attempts to reduce the overall duration, but it seems unable to reduce the abnormally long word *"yellow"* more than the other words. In contrast, AUTOPST not only restores the overall length, but also largely restores the word *"yellow"* to its normal length. This shows that AUTOPST can indeed capture the fine-grained rhythm patterns instead of blindly adjusting the speech rate.

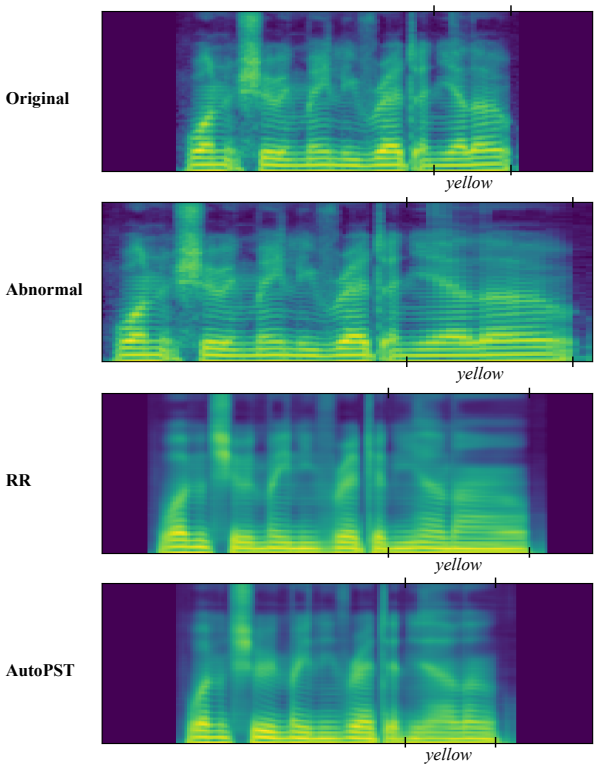


Figure 8. Mel-spectrograms of reconstruction from *"One showing mainly red and yellow"* with an abnormally long *"yellow"*. AUTOPST can largely restore the duration to its original length.

5.6. Emotion Style Transfer

Although AUTOPST is designed for voice style transfer, we nevertheless also test AUTOPST on the much harder non-parallel emotion style transfer to investigate the generalizability of our proposed framework. We use the EmoV-DB dataset (Adigwe et al., 2018), which contains acted expressive speech of five emotion categories (amused, sad, neutral, angry, and sleepy) from four speakers. During training, two emotion categories are randomly chosen for each speaker and held out, for the purpose of evaluating generalization to unseen emotion categories for each speaker.

Among the five emotions, neutral has the fastest speech rate and sleepy has the slowest. We thus follow the same step in Section 5.3 to compute the relative duration difference between fast-to-slow and slow-to-fast emotion conversions for each speaker. Figure 7(d) shows the box plot. Consistent with the observations in Section 5.3, AUTOPST can bring most of the relative duration differences to positive numbers, whereas the baselines cannot. This result shows that AUTOPSTcan generalize to other domains. Additional results can be found in Appendix D.

6. Conclusion

In this paper, we propose AUTOPST an autoencoder based prosody style transfer algorithm that does not rely on text transcriptions or fine-grained local prosody style information. We have empirically shown that AUTOPST can effectively convert prosody, particularly the rhythm aspect, to match the target domain style. There are still some limitations of AUTOPST. Currently, in order to disentangle the timbre information, AUTOPST introduces a very harsh limitations on the dimension of the hidden representations. However, this would compromise the quality of the converted speech. How to strike a better balance between timbre disentanglement, prosody disentanglement, and audio quality remains a challenging future direction. Also, it has been shown by Deng et al. (2021) that AUTOVC performs poorly when given in-the-wild audio examples. Since AUTOPST inherits the basic framework of AUTOVC, it is unlikely to generalize well to in-the-wild examples either. Improving the generalizability to audios recorded in different environments is a future direction.

Acknowledgment

We would like to give special thanks to Gaoyuan Zhang from MIT-IBM Watson AI Lab, who has helped us a lot with building our demo webpage.

References

Adigwe, A., Tits, N., Haddad, K. E., Ostadabbas, S., and Dutoit, T. The emotional voices database: Towards controlling the emotion dimension in voice generation systems. *arXiv preprint arXiv:1806.09514*, 2018.

Aihara, R., Takashima, R., Takiguchi, T., and Ariki, Y. Gmm-based emotional voice conversion using spectrum and prosody features. *American Journal of Signal Processing*, 2(5):134–138, 2012.

Bhati, S., Villalba, J., Żelasko, P., and Dehak, N. Self-Expressing Autoencoders for Unsupervised Spoken Term Discovery. In *Proc. Interspeech 2020*, pp. 4876–4880, 2020. doi: 10.21437/Interspeech.2020-3000. URL http://dx.doi.org/10.21437/Interspeech.2020-3000.

Biadsy, F., Weiss, R. J., Moreno, P. J., Kanvesky, D., and Jia, Y. Parrotron: An end-to-end speech-to-speech conversion model and its applications to hearing-impaired speech and speech separation. *Proc. Interspeech 2019*, pp. 4115–4119, 2019.

Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., and Choo, J. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8789–8797, 2018.

Chou, J.-C. and Lee, H.-Y. One-shot voice conversion by separating speaker and content representations with instance normalization. *Proc. Interspeech 2019*, pp. 664–668, 2019.

Chou, J.-C., Yeh, C.-C., Lee, H.-Y., and Lee, L.-S. Multi-target voice conversion without parallel data by adversarially learning disentangled audio representations. *Interspeech*, pp. 501–505, 2018.

Deng, K., Bansal, A., and Ramanan, D. Unsupervised audiovisual synthesis via exemplar autoencoders. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=43VKWxg_Sqr.

Gao, Y., Singh, R., and Raj, B. Voice impersonation using generative adversarial networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2506–2510. IEEE, 2018.

Hsu, C.-C., Hwang, H.-T., Wu, Y.-C., Tsao, Y., and Wang, H.-M. Voice conversion from non-parallel corpora using variational auto-encoder. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pp. 1–6. IEEE, 2016.

Hsu, C.-C., Hwang, H.-T., Wu, Y.-C., Tsao, Y., and Wang, H.-M. Voice conversion from unaligned corpora using variational autoencoding Wasserstein generative adversarial networks. In *Interspeech*, pp. 3364–3368, 2017.

Huang, W.-C., Luo, H., Hwang, H.-T., Lo, C.-C., Peng, Y.-H., Tsao, Y., and Wang, H.-M. Unsupervised representation disentanglement using cross domain features and adversarial learning in variational autoencoder based voice conversion. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2020.

Inanoglu, Z. and Young, S. Data-driven emotion conversion in spoken english. *Speech Communication*, 51(3):268–283, 2009.

Kameoka, H., Kaneko, T., Tanaka, K., and Hojo, N. StarGAN-VC: Non-parallel many-to-many voice conversion using star generative adversarial networks. In *IEEE Spoken Language Technology Workshop (SLT)*, pp. 266–273. IEEE, 2018.

Kameoka, H., Kaneko, T., Tanaka, K., and Hojo, N. ACVAE-VC: Non-parallel voice conversion with auxiliary classifier variational autoencoder. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27 (9):1432–1443, 2019.

Kaneko, T. and Kameoka, H. Parallel-data-free voice conversion using cycle-consistent adversarial networks. *arXiv preprint arXiv:1711.11293*, 2017.

Kaneko, T., Kameoka, H., Tanaka, K., and Hojo, N. StarGAN-VC2: Rethinking conditional methods for StarGAN-based voice conversion. *Proc. Interspeech 2019*, pp. 679–683, 2019.

Kenter, T., Wan, V., Chan, C.-A., Clark, R., and Vit, J. CHiVE: Varying prosody in speech synthesis with a linguistically driven dynamic hierarchical conditional variational network. In *International Conference on Machine Learning*, pp. 3331–3340, 2019.

Kim, S., Hori, T., and Watanabe, S. Joint ctc-attention based end-to-end speech recognition using multi-task learning. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4835–4839. IEEE, 2017.

Liu, A. H., Tu, T., Lee, H.-y., and Lee, L.-s. Towards unsupervised speech recognition and synthesis with quantized speech representation learning. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7259–7263. IEEE, 2020.

Luo, Z., Takiguchi, T., and Ariki, Y. Emotional voice conversion using deep neural networks with mcc and f0 features. In *2016 IEEE/ACIS 15th International Conference on*

Computer and Information Science (ICIS), pp. 1–5. IEEE, 2016.

Ming, H., Huang, D., Xie, L., Wu, J., Dong, M., and Li, H. Deep bidirectional lstm modeling of timbre and prosody for emotional voice conversion. In *Interspeech 2016*, pp. 2453–2457, 2016. doi: 10.21437/Interspeech.2016-1053. URL http://dx.doi.org/10.21437/Interspeech.2016-1053.

Nachmani, E. and Wolf, L. Unsupervised singing voice conversion. *Proc. Interspeech 2019*, pp. 2583–2587, 2019.

Niwa, J., Yoshimura, T., Hashimoto, K., Oura, K., Nankaku, Y., and Tokuda, K. Statistical voice conversion based on WaveNet. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5289–5293. IEEE, 2018.

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

Polyak, A. and Wolf, L. Attention-based wavenet autoencoder for universal voice conversion. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6800–6804. IEEE, 2019.

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

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

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

Serrà, J., Pascual, S., and Perales, C. S. Blow: A single-scale hyperconditioned flow for non-parallel raw-audio voice conversion. In *Advances in Neural Information Processing Systems*, pp. 6790–6800, 2019.

Skerry-Ryan, R., Battenberg, E., Xiao, Y., Wang, Y., Stanton, D., Shor, J., Weiss, R., Clark, R., and Saurous, R. A. Towards end-to-end prosody transfer for expressive speech synthesis with Tacotron. In *International Conference on Machine Learning*, pp. 4693–4702, 2018.

Tao, J., Kang, Y., and Li, A. Prosody conversion from neutral speech to emotional speech. *IEEE transactions on Audio, Speech, and Language processing*, 14(4):1145–1154, 2006.

Valle, R., Li, J., Prenger, R., and Catanzaro, B. Mellotron: Multispeaker expressive voice synthesis by conditioning on rhythm, pitch and global style tokens. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6189–6193. IEEE, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Veaux, C., Yamagishi, J., MacDonald, K., et al. Superseded-CSTR VCTK corpus: English multi-speaker corpus for CSTR voice cloning toolkit, 2016.

Wang, Y., Stanton, D., Zhang, Y., Ryan, R.-S., Battenberg, E., Shor, J., Xiao, Y., Jia, Y., Ren, F., and Saurous, R. A. Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis. In *International Conference on Machine Learning*, pp. 5180–5189. PMLR, 2018.

Watanabe, S., Hori, T., Kim, S., Hershey, J. R., and Hayashi, T. Hybrid ctc/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253, 2017.

Wu, C.-H., Hsia, C.-C., Lee, C.-H., and Lin, M.-C. Hierarchical prosody conversion using regression-based clustering for emotional speech synthesis. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1394–1405, 2009.

Wu, Y. and He, K. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533. PMLR, 2020.

Zhou, K., Sisman, B., and Li, H. Transforming spectrum and prosody for emotional voice conversion with non-parallel training data. *arXiv preprint arXiv:2002.00198*, 2020a.

Zhou, K., Sisman, B., and Li, H. Vaw-gan for disentanglement and recomposition of emotional elements in speech. *arXiv preprint arXiv:2011.02314*, 2020b.