Many-to-Many Voice Transformer Network
Hirokazu Kameoka, Wen-Chin Huang, Kou Tanaka, Takuhiro Kaneko, Nobukatsu Hojo, and Tomoki Toda

Abstract—This paper proposes a voice conversion (VC) method based on a sequence-to-sequence (S2S) learning framework, which makes it possible to simultaneously convert the voice characteristics, pitch contour and duration of input speech. We previously proposed an S2S-based VC method using a transformer network architecture, which we call the “voice transformer network (VTN)”. While the original VTN is designed to learn only a mapping of speech feature sequences from one domain into another, we extend it so that it can simultaneously learn mappings among multiple domains using only a single model. This allows the model to fully utilize available training data collected from multiple domains by capturing common latent features that can be shared across different domains. On top of this model, we further propose incorporating a training loss called the “identity mapping loss”, to ensure that the input feature sequence will remain unchanged when it already belongs to the target domain. Using this particular loss for model training has been found to be extremely effective in improving the performance of the model at test time. We conducted speaker identity conversion experiments and showed that model obtained higher sound quality and speaker similarity than baseline methods.

Index Terms—Voice conversion (VC), sequence-to-sequence learning, attention, transformer network, many-to-many VC.

I. INTRODUCTION

Voice conversion (VC) is a technique for converting para/non-linguistic information contained in a given utterance such as the perceived identity of a speaker while preserving linguistic information. Potential applications of this technique include speaker-identity modification [1], speaking aids [2], [3], speech enhancement [4]–[6], and accent conversion [7].

Many conventional VC methods use parallel utterances of source and target speech to train acoustic models for feature mapping. A typical way to train an acoustic model consists of extracting acoustic features from source and target utterances, performing dynamic time warping (DTW) to obtain time-aligned parallel data, and training the acoustic model that maps the source features to the target features in a frame-by-frame manner. Examples of the acoustic model include Gaussian mixture models (GMM) [8]–[10] and deep neural networks (DNNs) [11]–[15]. Recently, some attempts have also been made to formulate non-parallel methods, which require no parallel utterances, transcriptions, or time alignment procedures, by using variational autoencoders (VAEs), cycle-consistent generative adversarial networks (CycleGAN), and star generative adversarial networks (StarGAN) [16]–[20].

Although the methods mentioned above are successful in converting the local spectral features, they have difficulty in converting suprasegmental features that reflect long-term dependencies, such as the fundamental frequency ($F_0$) contour, duration and rhythm of the input speech. This is because acoustic models in these methods are designed to describe mappings between local features only. This is why the time alignment process must be performed before and independently of the acoustic model training. However, since these suprasegmental features are as important factors as local features that characterize speaker identities and speaking styles, it would be desirable if these features could also be converted more flexibly. One solution to overcome this limitation would be to adopt a sequence-to-sequence (seq2seq or S2S) model, since it has a powerful ability to learn mappings between sequential data of variable lengths by capturing and utilizing long-range dependencies.

S2S models [21], [22] have recently been applied with notable success in many tasks such as machine translation, automatic speech recognition (ASR) [22] and text-to-speech (TTS) [23]–[29]. They are composed mainly of two elements: an encoder and a decoder. The encoder encodes an input sequence to its latent representation in the form of hidden state vectors whereas the decoder generates an output sequence according to this latent representation. With the original S2S model, all input sequences are encoded into a single context vector of a fixed dimension. One problem with this is that the ability of the model to capture long-range dependencies can be limited especially when input sequences are long. To overcome this limitation, a mechanism called “attention” [30] has been introduced, which allows the network to learn where to pay attention in the input sequence when producing each item in the output sequence.

While recurrent neural networks (RNNs) have initially been used as the default option for designing the encoder and decoder networks in S2S models, recent work has shown that convolutional neural network (CNN)-based architectures also have excellent potential for capturing long-term dependencies [31]. Subsequently, yet another type of architecture called the “transformer” has been proposed [32], which uses neither convolution nor recurrent layers in its network, but only the mechanism of attention. In particular, it uses multi-head self-attention layers to design the two networks. Self-attention is a type of attention mechanism, which offers an efficient way to relate different positions of a given sequence. The multi-head self-attention mechanism splits each element of a sequence into smaller parts and then computes the self-attention over the sequence of each part in parallel. Unlike RNNs, both these architectures have the advantage that they are suitable for parallel computations using GPUs.

Inspired by the success of many attempts that have been made to apply RNN-, CNN- and transformer-based S2S models to ASR and TTS, we have previously proposed VC methods based on these three models [33]–[37]. In our most recent work [36], [37], we have proposed a VC method
based on the transformer architecture called “voice transformer network (VTN)”. Through this work, it transpired that the model trained from scratch did not perform as expected when the amount of training data was limited. To address this, we introduced a TTS pretraining technique, which aims to provide a good initialization for fast and sample-efficient VC model training with the aid of text-speech pair data, thus reducing the parallel data size requirement and training time.

In this paper, we aim to propose several ideas to make VTN perform well even without TTS pretraining. One limitation with regular S2S models including the original VTN is that it can only learn a mapping from one domain to another. Here, examples of the domains include speaker identities and speaking styles including emotional expressions and accents, but for concreteness, we restrict our attention to speaker identity conversion tasks in this paper. When we are concerned with converting speech among multiple speakers, one naive way of applying VTN would be to prepare and train a model for each speaker pair. However, this can be inefficient since the model for a particular pair of speakers fails to use the training data of the other speakers for its training, even though there must be a common set of latent features that can be shared across different speakers, especially when the languages are the same. To fully utilize available training data in multiple speakers, we propose extending VTN so that it can learn mappings among multiple speakers using only a single model. We call this extended version the many-to-many VTN. On top of this model, we further propose several ideas to achieve even better performance, including the identity mapping loss and forward attention algorithm. We also show that the Pre-LN architecture, discussed in [48], [69], is effective in both the pairwise and many-to-many versions of VTN.

II. RELATED WORK

Several VC methods based on S2S models have already been proposed, including the ones we proposed previously. Although regular S2S models usually require large-scale parallel corpora for training, collecting a sufficient number of parallel utterances is not always feasible. Thus, particularly in VC tasks, one challenge would be how best to train S2S models when only a limited amount of training data is available.

One idea involves using text labels as auxiliary information for model training, assuming they are readily available. For example, Miyoshi et al. proposed combining acoustic models for ASR and TTS with an S2S model [40], where the ASR model is used to convert a source speech feature sequence into a context posterior probability sequence, an S2S model is then used to convert the context posterior probability sequence, and the TTS model is finally used to generate a target speech feature sequence according to the converted context posterior probability sequence. Zhang et al. proposed an S2S model-based VC method guided by an ASR system [41]. Subsequently, Zhang et al. proposed a shared model for TTS and VC tasks that allows for joint training of the TTS and VC functions [42]. Recently, Biadsy et al. proposed an end-to-end VC system called “Parrotron”, which is designed to train the encoder and the decoder along with an ASR model based on a multitask learning strategy [43]. Our VTN [36], [57] is another example, which relies on TTS pretraining using text-speech pair data.

The proposed many-to-many VTN differs from the above methods in that it does not rely on ASR or TTS models and requires no text annotations for model training.

III. VOICE TRANSFORMER NETWORK

A. Feature extraction and normalization

First, we define acoustic features to be converted. Given the recent significant advances in high-quality neural vocoders [44]–[54], we find it reasonable to consider converting acoustic features such as the mel-cepstral coefficients (MCCs) [55] and log \( F_0 \), since we can expect to obtain high-fidelity signals once acoustic features have successfully been converted. If we can design acoustic features in as compact a form as possible, we can expect to reduce the data size requirement for the model training accordingly. Motivated by the above, we choose to use the MCCs, log \( F_0 \), aperiodicity, and voiced/unvoiced indicator of speech as acoustic features as detailed below.

We first use the WORLD analyzer [56] to extract the spectral envelope, the log \( F_0 \), the coded aperiodicity, and the voiced/unvoiced indicator within each time frame of a speech utterance, then compute \( I \) mel-cepstral coefficients (MCCs) from the extracted spectral envelope, and finally construct an acoustic feature vector by stacking the MCCs, the log \( F_0 \), the coded aperiodicity, and the voiced/unvoiced indicator. Thus, each acoustic feature vector consists of \( I + 3 \) elements. Here, the log \( F_0 \) contour is assumed to be filled with smoothly interpolated values in unvoiced segments. At training time, we normalize each element \( x_{i,n} (i = 1, \ldots, I) \) of the MCCs and the log \( F_0 \) at frame \( n \) to \( x_{i,n} \leftarrow (x_{i,n} - \mu_i) / \sigma_i \), where \( i \), \( \mu_i \), and \( \sigma_i \) denote the feature index, the mean, and the standard deviation of the \( i \)-th feature within all the voiced segments of the training samples of the same speaker.

To accelerate and stabilize training and inference, we have found it useful to use a similar trick introduced in [57]. Namely, we divide the acoustic feature sequence obtained above into non-overlapping segments of equal length \( r \) and use the stack of the acoustic feature vectors in each segment as a new feature vector so that the new feature sequence becomes \( r \) times shorter than the original feature sequence.

B. Model

We hereafter use \( X^{(s)} = [x_1^{(s)}, \ldots, x_{N_s}^{(s)}] \in \mathbb{R}^{D \times N_s} \) and \( X^{(t)} = [x_1^{(t)}, \ldots, x_{N_t}^{(t)}] \in \mathbb{R}^{D \times N_t} \) to denote the source and target speech feature sequences of non-aligned parallel utterances, where \( N_s \) and \( N_t \) denote the lengths of the two sequences and \( D \) denotes the feature dimension. VTN [36], [57] has an encoder-decoder structure with a transformer architecture [42] that maps \( X^{(s)} \) to \( X^{(t)} \) (Fig. 1). The encoder is expected to extract contextual information from source speech and the decoder produces the target speech feature sequence according to the contextual information the encoder has generated. Given the fact that unlike RNN-based and
CNN-based S2S models, the transformer model itself does not have any sense of the order of the elements in a sequence, the sinusoidal position encodings \( \mathbf{P}_{N} \in \mathbb{R}^{D \times N} \) and \( \mathbf{P}_{N} \in \mathbb{R}^{D \times N} \) are first added to the source and target feature sequences to make the model be aware of the position at which an each element in the sequences is located. The source and target feature sequences are then passed through convolutional prenets, which we call the source and target prenets, before being fed into the encoder and decoder. The output prenets, which we call the source and target prenets, before the sinusoidal position encodings \( \mathbf{P}_{N} \), have any sense of the order of the elements in a sequence, CNN-based S2S models, the transformer model itself does not have any sense of the order of the elements in a sequence.

The encoder consists of \( \mathbf{X} \in \mathbb{R}^{d \times N} \) and \( \mathbf{X} \in \mathbb{R}^{d \times N} \) to denote the outputs from the source and target prenets, respectively, where \( d \) is the output channel number of each prenet.

1) Encoder: The encoder takes \( \mathbf{X} \) as the input and produces a context vector sequence \( \mathbf{Z} = [z_1, \ldots, z_N] \in \mathbb{R}^{d \times N} \). The encoder consists of \( L \) identical layers, each of which has self-attention (SA) and position-wise fully-connected feed forward network (FFN) sub-layers. Residual connections and layer normalizations are applied in addition to the two sub-layers.

Multi-head self-attention (SA) sub-layer: By using \( \mathbf{X} \in \mathbb{R}^{d \times N} \) and \( \mathbf{Y} \in \mathbb{R}^{d \times N} \) to denote the input and output of an SA sub-layer, the process \( \mathbf{Y} = \text{SA}(\mathbf{X}) \), by which \( \mathbf{Y} \) is produced, is given as

\[
[Q; K; V] = \mathbf{W}_1 \mathbf{X} \in \mathbb{R}^{3d \times N},
\]

\[
\text{where } \begin{cases} Q = [Q_1; \ldots; Q_H] \\ K = [K_1; \ldots; K_H] \\ V = [V_1; \ldots; V_H] \end{cases},
\]

\[
A_h = \text{softmax} \left( \frac{K^{	op} Q}{\sqrt{d}} \right) \quad (h = 1, \ldots, H),
\]

\[
\mathbf{Y} = \mathbf{W}_2 [V_1 A_1; \ldots; V_H A_H],
\]

where \( \mathbf{W}_1 \in \mathbb{R}^{3d \times d} \) and \( \mathbf{W}_2 \in \mathbb{R}^{d \times d} \) are learnable weight matrices, softmax denotes a softmax operation performed on the first axis, \( H \) denotes the number of heads, and \( [\cdot; \cdot] \) denotes vertical concatenation of matrices (or vectors) with compatible sizes. Intuitively, this process can be understood as follows. First, an input vector sequence is converted into three types of vector sequences with the same shape, which can be metaphorically interpreted as the queries and the key-value pairs in a hash table. Each of the three vector sequences is further split into \( H \) homogeneous vector sequences with the same shape. By using the query and key pair, Eq. (3) computes a self-attention matrix, whose element measures how contextually similar each pair of vectors is in the given sequence \( \mathbf{X} \). The splitting into \( H \) heads allows us to measure self-similarity in terms of \( H \) different kinds of context. The \( n \)-th column of \( \mathbf{V}_h A_h \) in Eq. (4) can be seen as a new feature vector given by activating the value vectors at all the positions that are similar to the current position \( n \) in terms of context \( h \) and adding them together. Eq. (4) finally produces the output sequence \( \mathbf{Y} \) after combining all these feature vector sequences using learnable weights.

**Position-wise feed forward network (FFN) sub-layer:** By using \( \mathbf{X} \in \mathbb{R}^{d \times N} \) and \( \mathbf{Y} \in \mathbb{R}^{d \times N} \) again to denote the input and output of an FFN sub-layer, the process \( \mathbf{Y} = \text{FFN}(\mathbf{X}) \), by which \( \mathbf{Y} \) is produced, is given as

\[
\mathbf{Y} = \mathbf{W}_4 \phi(\mathbf{W}_3 \mathbf{X} + \mathbf{B}_3) + \mathbf{B}_4,
\]

where \( \mathbf{W}_3 \in \mathbb{R}^{d \times d} \) and \( \mathbf{W}_4 \in \mathbb{R}^{d \times d} \) are learnable weight matrices, \( \mathbf{B}_3 = [b_3, \ldots, b_3] \in \mathbb{R}^{d \times d} \) and \( \mathbf{B}_4 = [b_4, \ldots, b_4] \in \mathbb{R}^{d \times d} \) are bias matrices, each consisting of identical learnable column vectors, and \( \phi \) denotes an elementwise nonlinear activation function such as the rectified linear unit (ReLU) and gated linear unit (GLU) functions.
Layer normalization (LN) sub-layers: Recent work has shown that the location of the layer normalization in the transformer architecture affects the speed and stability of the training process as well as the performance of the trained model \[38, 39\]. While the original transformer architecture places layer normalization after the SA and FFN sub-layers, the architectures presented in \[38, 39\] are designed to place it before them, as depicted in Fig. 2 These architectures are called post-layer normalization (Post-LN) and pre-layer normalization (Pre-LN) architectures, respectively. We will show later how differently these architectures actually performed in our experiments. Note that when we say we apply layer normalization to an input vector sequence, say \(X = [x_1, \ldots, x_N]\), we mean to apply layer normalization to all the vectors \(x_1, \ldots, x_N\), treated as mini-batch samples.

If we use \(X_t\) and \(X_{t+1}\) to denote the input and output of the \(t\)-th encoder layer (with the PreLN architecture), the process \(X_{t+1} = Enc_t(X_t)\) of the \(t\)-the layer is given by

\[
U = X_t + \text{SA}(\text{LayerNorm}_1(X_t)), \quad (6)
\]

\[
X_{t+1} = U + \text{FFN}(\text{LayerNorm}_2(U)), \quad (7)
\]

where \(\text{LayerNorm}_1\) and \(\text{LayerNorm}_2\) denote different LN sub-layers. As described above, each layer has learnable parameters in the SA and FFN sub-layers and the two LN sub-layers. The layer implemented as above is particularly attractive in that it is able to relate all the positions in the entire input sequence using only a single layer. This is in contrast to a regular convolution layer, which is only able to relate local positions near each position.

2) Decoder: The decoder takes \(Z^{(s)}\) and \(X^{(t)}\) as the inputs and produces a converted feature sequence \(Y^{(s \rightarrow t)} = [\hat{Y}_1^{(s \rightarrow t)}, \ldots, \hat{Y}_N^{(s \rightarrow t)}] \in \mathbb{R}^{d \times N}\). Similar to the encoder, the decoder consists of \(L\) identical layers, each of which has SA and FFN sub-layers, residual connections and layer normalization sub-layers. In addition to these sub-layers, each layer has a multi-head target-to-source attention (TSA) sub-layer as depicted in Fig. 3 whose role is to find which position in the source feature sequence contextually corresponds to each position in the target feature sequence and convert the context vector sequence according to the predicted corresponding positions.

Multi-head target-to-source attention (TSA) sub-layer: By using \(X \in \mathbb{R}^{d \times N}\) and \(Y \in \mathbb{R}^{d \times N}\) to denote the output from the previous sub-layer and the output of the current TSA sub-layer, the process \(Y = \text{TSA}(X, Z)\), by which \(Y\) is produced, is given in the same way as the SA sub-layer with the only difference being that the key and value pair \((K, V)\) is computed using the output \(Z\) from the encoder:

\[
Q = W_5 X, \quad (8)
\]

\[
[K; V] = W_6 Z, \quad (9)
\]

where

\[
\begin{align*}
Q & = [Q_1; \ldots; Q_H] \\
K & = [K_1; \ldots; K_H] \\
V & = [V_1; \ldots; V_H]
\end{align*}
\]

\[
A_h = \text{softmax} \left( \frac{K^T Q}{\sqrt{d}} \right); \quad (h = 1, \ldots, H), \quad (10)
\]

\[
Y = W_7 [V_1 A_1; \ldots; V_H A_H], \quad (11)
\]

where \(W_5 \in \mathbb{R}^{d \times d}, W_6 \in \mathbb{R}^{2d \times d}\) and \(W_7 \in \mathbb{R}^{d \times d}\) are learnable weight matrices. Analogously to the SA sub-layer, Eq. (11) computes a target-to-source attention matrix using the query and key pair, where the \((n, m)\)-th element indicates the similarity between the \(n\)-th and \(m\)-th frames of source and target speech. The peak trajectory of \(A_h\) can thus be interpreted as a time warping function that associates the frames of the source speech with those of the target speech. The splitting into \(H\) heads allows us to measure the similarity in terms of \(H\) different kinds of context. \(V_h A_h\) in Eq. (12) can be thought of as a time-warped version of \(V_h\) in terms of context \(h\). Eq. (12) finally produces the output sequence \(Y\) after combining all these time-warped feature sequences using learnable weights.

All the other sub-layers are defined in the same way as the encoder. The overall structures of the decoder layers with the PreLN and PostLN architectures are depicted in Fig. 3. If we use \(X_t\) and \(X_{t+1}\) to denote the input and output of the \(t\)-th decoder layer (with the PreLN architecture), the process \(X_{t+1} = \text{Dec}(X_t, Z)\) of the \(t\)-th layer is given by

\[
\begin{align*}
U_1 & = X_t + \text{SA}(\text{LayerNorm}_1(X_t)), \quad (13) \\
U_2 & = U_1 + \text{TSA}(\text{LayerNorm}_2(U_1), Z), \quad (14) \\
X_{t+1} & = U_2 + \text{FFN}(\text{LayerNorm}_3(U_2)). \quad (15)
\end{align*}
\]

Note that each layer has learnable parameters in the SA, FFN and TSA sub-layers and the three LN sub-layers.

3) Autoregressive structure: Since the target feature sequence \(X^{(t)}\) is of course not accessible at test time, we would want to use a feature vector that the decoder has generated as the input to the decoder for the next time step so that feature vectors can be generated one-by-one in a recursive manner. To allow the model to behave in this way, first we must take care that the decoder must not be allowed to use future information about the target feature vectors when producing an output vector at each time step. This can be ensured by simply constraining the convolution layers in the target prenet to be causal and replacing Eq. (3) in all the SA sub-layers in the decoder with

\[
A_h = \text{softmax} \left( \frac{K^T Q}{\sqrt{d}} + E \right), \quad (16)
\]

where \(E\) is a matrix whose \((n, n')\)-th element is given by

\[
e_{n, n'} = \begin{cases} 
0 & (n \leq n') \\
-\infty & (n > n')
\end{cases}, \quad (17)
\]

so that the predictions for position \(n\) can depend only on the known outputs at positions less than \(n\). Secondly, the output sequence \(Y^{(s \rightarrow t)}\) must correspond to a time-shifted version of \(X^{(t)}\), so that at each time step the decoder will be able to predict the target speech feature vector that is likely to appear at the next time step. To this end, we include an \(L_t\) loss

\[
L_{\text{main}} = \frac{1}{M} \| Y^{(s \rightarrow t)};1:M-1 - X^{(t)};2:M \|_1, \quad (18)
\]

in the training loss to be minimized, where we have used the colon operator \(:\) to specify the range of indices of the elements in a matrix we wish to extract (For ease of notation, we use \(:\) itself to represent all elements along an axis. For example, \(X^{(t)};2:M\) denotes a submatrix consisting of the elements in
all the rows and columns 2, 3, \ldots, M of \( X^{(t)} \). Thirdly, the first column of \( X^{(t)} \) must correspond to an initial vector with which the recursion is assumed to start. We thus assume that the first column of \( X^{(t)} \) is always set at an all-zero vector.

C. Constraints on Attention Matrix

It would be natural to assume that the time alignment between parallel utterances is usually monotonic and nearly linear. This implies that the diagonal region in the attention matrices obtained at each TSA sub-layer in the decoder should always be dominant. We expect that imposing such restrictions can significantly reduce the training effort since the search space can be greatly reduced. To penalize the attention matrices for not having a diagonally dominant structure, we introduce a diagonal attention loss (DAL) \[22\]:

\[
L_{dal} = \frac{1}{N_{MLH}} \sum_l \sum_k \| G_{N, N_t} \odot A_{l,h} \|_1, \tag{19}
\]

where \( A_{l,h} \) denotes the target-to-source attention matrix of the \( h \)-th head in the TSA sub-layer in the \( l \)-th decoder layer, \( \odot \) denotes elementwise product, and \( G_{N, N_t} \in \mathbb{R}^{N \times N_t} \) is a non-negative weight matrix whose \((n,m)\)-th element \( w_{n,m} \) is defined as \( w_{n,m} = 1 - e^{-(n/\sqrt{N}-m/\sqrt{N})^2/\nu^2} \).

D. Training loss

Given examples of parallel utterances, the total training loss for the VTN to be minimized is given as

\[
L = \mathbb{E}_{X^{(c)}, X^{(t)}} \{ L_{main} + \lambda_{dal} L_{dal} \}, \tag{20}
\]

where \( \mathbb{E}_{X^{(c)}, X^{(t)}} \{ \} \) denotes the sample mean over all the training examples and \( \lambda_{dal} \geq 0 \) is a regularization parameter, which weights the importance of \( L_{dal} \) relative to \( L_{dec} \).

E. Conversion process

At test time, a source speech feature sequence \( X \) can be converted to the target speaker via the following recursion:

\[
\begin{align*}
Z & \leftarrow X, Y \leftarrow 0 \\
\text{for } l & \text{ from } 1 \text{ to } L \do \\
Z & \leftarrow \text{Enc}_l(Z) \\
\text{end for} \\
\text{for } m & \text{ from } 1 \text{ to } M \do \\
\text{for } l & \text{ from } 1 \text{ to } L \do \\
Y & \leftarrow \text{Dec}_l(Y, Z) \\
\text{end for} \\
Y & \leftarrow [0, Y] \\
\text{end for} \\
\text{return } Y
\end{align*}
\]

Once \( Y \) has been obtained, we adjust the mean and variance of the generated feature sequence so that they match the pretrained mean and variance of the feature vectors of the target speaker. We can then generate a time-domain signal using the WORLD vocoder or any recently developed neural vocoder \[44\]–\[54\].

However, as Fig. 4 shows, it transpired that with the model trained from scratch, the attended time point did not always move forward monotonically and continuously at test time and can occasionally make a sudden jump to a distant time point, resulting in some segments being skipped or repeated, even though the DAL was considered in training. In \[36\], \[37\], we previously proposed to introduce pretraining techniques exploiting auxiliary text labels to improve the behavior and performance of the conversion algorithm, as mentioned earlier. In the next section, we propose several ideas that can greatly improve the behavior of the VTN even without pretraining using text labels.

IV. MANY-TO-MANY VTN

A. Many-to-Many Extension

The first idea is a many-to-many extension of the VTN, which uses a single model to realize mappings among multiple speakers by allowing the prenets, the postnet, the encoder and the decoder to take source and target speaker indices as additional inputs. The overall structure of the many-to-many VTN is shown in Fig. 5.

Let \( X^{(1)}, \ldots, X^{(k)} \) be examples of the acoustic feature sequences of different speakers reading the same sentence. Given a single pair of parallel utterances \( X^{(k)} \) and \( X^{(k')} \), where \( k \) and \( k' \) denote the source and target speaker indices (integers), the source and target prenets take tuples \((X^{(k)}, k)\) and \((X^{(k')}, k')\) as the inputs and produce modified feature sequences \( \tilde{X}^{(k)} \) and \( \tilde{X}^{(k')} \), respectively. The encoder takes a tuple \((\tilde{X}^{(k)}, k)\) as the input and produces a context vector sequence \( Z^{(k)} \). The decoder takes \((\tilde{X}^{(k')}, Z^{(k')}, k')\) as the input and produces a converted feature sequence \( \hat{Y}^{(k \rightarrow k')} \). The postnet takes \((\hat{Y}^{(k \rightarrow k')}, k')\) as the input and finally produces a
modified version $Y_{(k \rightarrow k')}$ of $Y_{(k \rightarrow k')}$. Each of the networks incorporates the speaker index into its process by modifying the input sequence, say $X$, via

\begin{equation}
S = \text{repeat}(\text{embed}(k)),
\end{equation}

\begin{equation}
X \leftarrow [X; S],
\end{equation}

every time before feeding $X$ into the SA, FFN or TSA sub-layers, where embed denotes an operation that retrieves a continuous vector from an integer input and repeat denotes an operation that produces a vector sequence from an input vector by simply repeating it along the time axis.

The loss functions to be minimized given this single training example are given as

\begin{equation}
L_{\text{main}}^{(k,k')} = \frac{1}{N_{k'}-1} \left\| Y_{(k \rightarrow k')}[:,1:N_{k'}-1] - \left[ X^{(k')}[:,1:N_{k'}] \right] \right\|_1,
\end{equation}

\begin{equation}
L_{\text{dal}}^{(k,k')} = \frac{1}{N_k \times N_{k'}} \sum_h \sum_l \| G_{N_k \times N_{k'}} \odot A_{h,l}^{(k,k')} \|_1,
\end{equation}

where $A_{h,l}^{(k,k')}$ denotes the target-to-source attention matrix of the $h$-th head in the TSA sub-layer in the $l$-th decoder layer.

With the above model, we can also consider the case where $k = k'$. Minimizing the sum of the above losses under $k = k'$ encourages the model to let the input feature sequence $X^{(k)}$ remain unchanged when it already belongs to the target speaker $k'$. We call this loss the “identity mapping loss (IML)”.

The total training loss including the IML thus becomes

\begin{equation}
L = \sum_{k,k' \neq k} \mathbb{E}_{X^{(k)}, X^{(k')}} \left\{ L_{\text{all}}^{(k,k')} \right\} + \lambda_{\text{iml}} \sum_{k} \mathbb{E}_{X^{(k)}} \left\{ L_{\text{all}}^{(k,k')} \right\},
\end{equation}

where $L_{\text{all}}^{(k,k')} = L_{\text{main}}^{(k,k')} + \lambda_{\text{dal}} L_{\text{dal}}^{(k,k')}.$

\[ \mathbb{E}_{X^{(k)}, X^{(k')}} \{ \cdot \} \] and $\mathbb{E}_{X^{(k)}} \{ \cdot \}$ denote the sample means over all the training examples of parallel utterances of speakers $k$ and $k'$, and $\lambda_{\text{iml}} \geq 0$ is a regularization parameter, which weighs the importance of the IML. The significant effect of the IML will be shown later.

Fig. 6 shows examples of the target-to-source attention matrices predicted using the many-to-many VTN from the same test samples used in Fig. 4. As these examples show, the predicted attention matrices obtained with the many-to-many VTN exhibit more monotonic and continuous trajectories than the ones with the original VTN, thus demonstrating the impact of the many-to-many extension.

### B. Forward Attention

Here, we present another idea that can be used alone or combined with the many-to-many extension to improve the original VTN. To assist the attended point to move forward monotonically and continuously at test time, we propose to modify the algorithm presented in Subsection III-E. Specifically, we limit the paths through which the attended point is allowed to move by forcing the attentions to all the time points distant from the previous attended time point to zeros. Here, we assume the attended time point to be the peak of the attention distribution, given as the mean of all the target-to-source attention matrices in the TSA sub-layers in the decoder.

This can be implemented by replacing Eq. (11) in the TSA sub-layer in each decoder layer $l$ at the $m'$-th iteration of the for-loop for $m = 1, \ldots, M$ in the conversion process with

\begin{equation}
\hat{A}_{l,h} = \text{softmax}(\frac{K^T Q}{\sqrt{d}} + F) \quad (h = 1, \ldots, H),
\end{equation}

where the $(n, m)$-th element $f_{n,m}$ of $F$ is given by

\begin{equation}
f_{n,m} = \begin{cases} -\infty & (m = m', n = 1, \ldots, \hat{n} - N_0) \\ -\infty & (m = m', n = \hat{n} + N_1, \ldots, N) \\ 0 & \text{(otherwise)} \end{cases}
\end{equation}
an arbitrary speaker or an arbitrary speaking style that is not included in the training dataset. Another important advantage of the many-to-many extension presented above is that it can be modified to handle any-to-many VC tasks by not allowing the source pretynet and the encoder to take the source speaker index $k$ as inputs. Namely, with the modified version, the output sequence of each layer in these networks is directly passed to the next layer without going through Eqs. (21) and (22).

We show later how well this modified version performs on an any-to-many VC task in which the source speaker is unseen in the training dataset.

D. Real-Time System Settings

It is important to be aware of real-time requirements when building VC systems. To let the VTN work in real-time, we need to make two modifications. Firstly, we must not let the source pretynet and the encoder use future information as with the target pretynet, the decoder, and the postnet during training. This requirement can easily be implemented by constraining the convolution layers in the source pretynet to be causal and replacing Eq. (3) with Eq. (10) also for all the sub-layers in the encoder. Secondly, since the speaking rate and rhythm of input speech cannot be changed drastically at test time, we simply set all the target-to-source attention matrices to identity matrices so that the speaking rate and rhythm will be kept unchanged.

V. Experiments

A. Experimental Settings

To confirm the effects of the ideas presented in Section IV, we conducted objective and subjective evaluation experiments involving a speaker identity conversion task. For the experiment, we used the CMU Arabic database [58], which consists of recordings of 1132 phonetically balanced English utterances spoken by four US English speakers. We used all the speakers, “clb” (female), “bdl” (male), “slt” (female) and “rms” (male), for training and evaluation. Thus, in total there were twelve different combinations of source and target speakers. The audio files for each speaker were manually divided into 1000 and 132 files, which were provided as training and evaluation sets, respectively. All the speech signals were sampled at 16 kHz. As already detailed in Subsection III-A, for each utterance, the spectral envelope, log $F_0$, coded aperiodicity, and voiced/unvoiced information were extracted every 8 ms using the WORLD analyzer [56]. 28 mel-cepstral coefficients (MCCs) were then extracted from each spectral envelope using the Speech Processing Toolkit (SPTK) [59]. The reduction factor $r$ was set to 3. Hence, the dimension of the acoustic feature was $D = (28 + 3) \times 3 = 93$. Adam optimization was used for model training.

B. Network Architecture Details

Dropouts with rate 0.1 were applied to the input sequences before being fed into the source and target pretnets and the postnet only at training time. For the nonlinear activation function $\phi$ in each FFN sub-layer, we chose to use the GLU
function since it yielded slightly better performance than the ReLU function. The two prenets and the postnet were each designed using three 1D dilated convolution layers with kernel size 5, each followed by a GLU activation function, where weight normalization [61] was applied to each layer. The channel number d was set to 256 for the pairwise version and 512 for the many-to-many version, respectively. The middle channel number $d'$ of each FFN sub-layer was set at 512 for the pairwise version and 1024 for the many-to-many version, respectively.

C. Hyperparameter Settings

$\lambda_{dal}$ and $\lambda_{ml}$ were set to 2000 and 1, respectively. $\nu$ was set to 0.3 for both the vanilla and many-to-many VTNs. The $L_1$ norm $\|X\|_1$ used in Eqs. (13) and (23) were defined as a weighted norm

$$\|X\|_1 = \sum_{n=1}^{N} \frac{1}{r} \sum_{j=1}^{r} \sum_{i=1}^{31} \gamma_i |x_{ij,n}|,$$

where $x_{1,j,n}, \ldots, x_{28,j,n}, x_{29,j,n}, x_{30,j,n}$ and $x_{31,j,n}$ denote the entries of $X$ corresponding to the 28 MCCs, log $F_0$, coded aperiodicity and voiced/unvoiced indicator at time $n$, and the weights were set at $\gamma_1 = \cdots = \gamma_{28} = \frac{1}{28}, \gamma_{29} = \frac{1}{10}, \gamma_{30} = \gamma_{31} = \frac{1}{30}$, respectively.

All the networks were trained simultaneously with random initialization. Adam optimization [60] was used for model training where the mini-batch size was 16 and 30,000 iterations were run. The learning rate and the exponential decay rate for the first moment for Adam were set at $1.0 \times 10^{-4}$ and 0.9 for the many-to-many version with the PreLN architecture and at $5.0 \times 10^{-5}$ and 0.9 otherwise.

D. Objective Performance Measures

The test dataset consisted of speech samples of each speaker reading the same sentences. Thus, the quality of a converted feature sequence could be assessed by comparing it with the feature sequence of the reference utterance.

1) Mel-Cepstral Distortion (MCD): Given two mel-cepstra, $\hat{x} = [\hat{x}_1, \ldots, \hat{x}_{28}]^T$ and $x = [x_1, \ldots, x_{28}]^T$, we can use the mel-cepstral distortion (MCD):

$$\text{MCD}[\text{dB}] = \frac{10}{\ln 10} \sqrt{2 \sum_{i=2}^{28} (\hat{x}_i - x_i)^2},$$

(29)

to measure their difference. Here, we used the average of the MCDs taken along the dynamic time warping (DTW) path between converted and reference feature sequences as the objective performance measure for each test utterance. Note that a smaller MCD indicates better performance.

2) Log $F_0$ Correlation Coefficient (LFC): To evaluate the log $F_0$ contour of converted speech, we used the correlation coefficient between the predicted and target log $F_0$ contours [62] as the objective performance measure. Since the converted and reference utterances were not necessarily aligned in time, we must compute the correlation coefficient after properly aligning them. Here, we used the MCC sequences $X_{1,28,1:N}, X_{1,28,1:M}$ of converted and reference utterances to find phoneme-based alignment, assuming that the predicted and reference MCCs at the corresponding frames were sufficiently close. Given the log $F_0$ contours $X_{29,1:N}, X_{29,1:M}$ and the voiced/unvoiced indicator sequences $X_{31,1:N}, X_{31,1:M}$ of converted and reference utterances, we first warp the time axis of $X_{29,1:N}$ and $X_{31,1:N}$ according to the DTW path between the MCC sequences $X_{1,28,1:N}, X_{1,28,1:M}$ of the two utterances and obtain their time-warped versions, $X_{29,1:M}, X_{31,1:M}$. We then extract the elements of $X_{29,1:M}$ and $X_{29,1:N}$ at all the time points corresponding to the voiced segments such that $\{m|X_{31,m} = X_{31,n} = 1\}$. If we use $y = [y_1, \ldots, y_M]$ and $\tilde{y} = [\tilde{y}_1, \ldots, \tilde{y}_M]$ to denote the vectors consisting of the elements extracted from $X_{29,1:M}$ and $X_{29,1:N}$, we can use the correlation coefficient between $\tilde{y}$ and $y$

$$R = \frac{\sum_{m=1}^{M} (\tilde{y}_m - \bar{\phi})(y_m - \phi)}{\sqrt{\sum_{m=1}^{M} (\tilde{y}_m - \bar{\phi})^2 \sum_{m=1}^{M} (y_m - \phi)^2}}$$

(30)

where $\bar{\phi} = \frac{1}{M} \sum_{m=1}^{M} \tilde{y}_m$ and $\phi = \frac{1}{M} \sum_{m=1}^{M} y_m$, to measure the similarity between the two log $F_0$ contours. In the current experiment, we used the average of the correlation coefficients taken over all the test utterances as the objective performance measure for log $F_0$ prediction. Thus, the closer it is to 1, the better the performance. We call this measure the “log $F_0$ correlation coefficient (LFC)”.

3) Local Duration Ratio (LDR): To evaluate the speaking rate and the rhythm of converted speech, we used the local slopes of the DTW path between converted and reference utterances to determine the objective performance measure. If the speaking rate and the rhythm of the two utterances are exactly the same, all the local slopes should be 1. Hence, the better the conversion, the closer the local slopes become to 1. To compute the local slopes, we undertook the following process. Given the MCC sequences $X_{1,28,1:N}, X_{1,28,1:M}$ of converted and reference utterances, we first performed DTW on $X_{1,28,1:N}$ and $X_{1,28,1:M}$. If we use $\{p_1, q_1\}, \ldots, \{p_M, q_M\}$ to denote the obtained DTW path where $(p_1, q_1) = (1, 1)$ and $(p_M, q_M) = (M, N)$, we computed the slope of the regression line fitted to the 33 local consecutive points for each j:

$$s_j = \frac{\sum_{j'=0}^{j+16} (p_{j'} - \bar{p}_j)(q_{j'} - \bar{q}_j)}{\sum_{j'=0}^{j+16} (p_{j'} - \bar{p}_j)^2},$$

(31)

where $\bar{p}_j = \frac{1}{33} \sum_{j'=0}^{j+16} p_{j'}$ and $\bar{q}_j = \frac{1}{33} \sum_{j'=0}^{j+16} q_{j'}$, and then computed the median of $s_1, \ldots, s_N$. We call this measure the “local duration ratio (LDR)”. The greater this ratio, the longer the duration of the converted utterance is relative to the reference utterance. In the following, we use the mean absolute difference between the LDRs and 1 (in percentage) as the overall measure for the LDRs. Thus, the closer it is to zero, the better the performance. For example, if the converted speech is 2 times faster than the reference speech, the LDR will be 0.5 everywhere, and so its mean absolute difference from 1 will be 50%.
E. Baseline Methods

1) sprocket: We chose the open-source VC system called “sprocket” [63] for comparison with our experiments. To run this method, we used the source code provided by the author [64]. Note that this system was used as a baseline system in the Voice Conversion Challenge (VCC) 2018 [65].

2) RNN-S2S-VC and ConvS2S-VC: To compare different types of network architectures, we also tested the RNN-based S2S model [34], inspired by the architecture introduced in a S2S model-based TTS system called “Tacotron” [23], and the CNN-based model, presented in [34], [35]. We refer to these models as RNN-S2S-VC and ConvS2S-VC, respectively.

RNN-S2S-VC: Although the original Tacotron employed mel-spectra as the acoustic features, the baseline system was designed to use the same acoustic features as our system. The architecture was specifically designed as follows. The encoder consisted of a bottleneck fully-connected prenet followed by a stack of $1 \times 1$ 1D GLU convolutions and a bi-directional LSTM layer. The decoder was an autoregressive content-based attention network, consisting of a bottleneck fully-connected prenet followed by a stateful LSTM layer producing the attention query, which was then passed to a stack of 2 uni-directional residual LSTM layers, followed by a linear projection to generate the features.

ConvS2S-VC: Fig. 8 shows the overall architecture of the ConvS2S model we implemented for this experiment. The model consisted of source/target encoders and a decoder, each of which had eight 1D GLU dilated convolution layers with kernel size 5. We used single-step single-head scaled dot-product attention to compute attention distributions from the outputs of the source/target encoders. The convolutions in the target encoder and the decoder were constrained to be causal as with the target prenet and the postnet in the VTN. A residual connection and weight normalization were applied to each layer in the three networks.

![ConvS2S architecture](image)

Fig. 8. Overall ConvS2S architecture.

We also designed and implemented many-to-many extensions of the above RNN-based and CNN-based models.

F. Objective Evaluations

1) Ablation Studies: We conducted ablation studies to confirm the individual effects of the many-to-many extension, the IML, and the FA algorithm, and compare the performance obtained with the PostLN and PreLN architectures. It should be noted that the models trained without the DAL were unsuccessful in producing recognizable speech, possibly due to the limited amount of training data. For this reason, we omit the results obtained when $\lambda_{dal} = 0$

Tab. I shows the average MCDs, LFCs and LDRs over the test samples obtained with the pairwise and many-to-many

| Versions | FA Settings | Measures |
|----------|-------------|----------|
|         | L | H | MC | Dom | LFC | LDR |
| PostLN- |   |   | 7.09 | 0.710 | 4.97  |
| PreLN-   |   |   | 7.12 | 0.705 | 5.75  |
|          | ✓ |   | 6.96 | 0.734 | 5.45  |
|          | ✓ |   | 6.94 | 0.738 | 4.17  |
| pairwise |   | ✓ | 6.89 | 0.721 | 4.81  |
| PreLN-   |   | ✓ | 6.89 | 0.721 | 4.81  |
|          |   | ✓ | 6.89 | 0.721 | 4.81  |
| many-to- | ✓ |   | 6.71 | 0.725 | 4.50  |
|          | ✓ |   | 6.70 | 0.712 | 4.03  |
|          | ✓ |   | 6.71 | 0.725 | 4.50  |
|          | ✓ |   | 6.70 | 0.712 | 4.03  |
|          | ✓ |   | 6.71 | 0.725 | 4.50  |
|          | ✓ |   | 6.70 | 0.712 | 4.03  |
|          | ✓ |   | 6.71 | 0.725 | 4.50  |
|          | ✓ |   | 6.70 | 0.712 | 4.03  |

Table I: Performance of the pairwise and many-to-many VTN with PostLN and PreLN architectures with and without the FA process under different L and H settings.
versions with the PostLN and PreLN architectures with and without the FA process under different L and H settings. The number in bold face indicates the best performance among all the L and H settings. We observe from these results that the effect of the many-to-many extension was noticeable. Comparisons between with and without the FA process revealed that while the FA process showed a certain effect in improving the pairwise version in terms of all the measures, it was found to be only slightly effective for the many-to-many version. This may imply that the prediction of attentions by the many-to-many version was already so successful that no correction by the FA process was necessary. As for the PostLN and PreLN architectures, the latter performed consistently better than the former especially for the pairwise version.

Tab. II shows the average MCDs, LFCs and LDRs over the test samples obtained with the many-to-many version trained with and without the IML. As these results show, the IML had a significant effect on performance improvements in terms of the MCD and LFC measures.

2) Comparisons with Baseline Methods: Tabs. III, IV and V show the MCDs, LFCs and LDRs obtained with the proposed and baseline methods. It should be noted that sprocket is designed to only adjust the mean and variance of the log F₀ contour of input speech and keep the rhythm unchanged. Hence, the performance gains over sprocket in terms of the LFC and LDR measures show how well the competing methods are able to predict the F₀ contours and the rhythms of target speech. As the results show, all the S2S models performed better than sprocket in terms of the LFC and LDR measures, thus demonstrating the ability to properly convert the prosodic features in speech. They also performed better than or comparably to sprocket in terms of the MCD measure. It is worth noting that the many-to-many extension was found to be significantly effective for all the architecture types of S2S models. It is interesting to compare the performance of the many-to-many versions of RNN-S2S, ConvS2S and VTN. The many-to-many ConvS2S performed best in terms of the MCD and LFC measures whereas the many-to-many VTN performed best in terms of the LDR measure. This may indicate that the strengths of S2S models can vary depending on the type of architecture.

As mentioned earlier, one important advantage of the transformer architecture over its RNN counterpart is that it can be trained efficiently thanks to its parallelizable structure. In fact, while it took about 30 hours and 50 hours to train the pairwise and many-to-many versions of the RNN-S2S model, it only took about 3 hours and 5 hours to train the two versions of the VTN under the current experimental settings. We implemented all the algorithms in PyTorch and used a single Tesla V100 GPU with a 32.0 GB memory for training each model.

3) Performance of any-to-many VTN: Our many-to-many conversion model can handle any-to-many VC tasks by using the modifications described in Subsection IV-C. We evaluated the performance of the any-to-many model under an open-set condition where the speaker of the test utterances are unseen in the training data. We used the utterances of the speaker “Inh” (female) as the test input speech. The results are shown in Tab. VI (a). For comparison, Tab. VI (b) shows results of sprocket performed on the same speaker pairs under a speaker-dependent closed-set condition. As these results show, the any-to-many VTN performed still better than sprocket, even though sprocket had an advantage in both the training and test conditions.

4) Performance with Real-Time System Settings: We evaluated the MCDs and LFCs obtained with the many-to-many VTN under the real-time system setting described in Subsection IV-D. The results are shown in Tab. VII. As the results show, it is worth noting that it performed only slightly worse than the default setting despite the restrictions related to the real-time system settings and performed still better than sprocket in terms of the MCD and LFC measures.

G. Subjective Listening Tests

We conducted mean opinion score (MOS) tests to compare the sound quality and speaker similarity of the converted speech samples obtained with the proposed and baseline methods.

With the sound quality test, we included the speech samples synthesized in the same way as the proposed and baseline methods (namely, the WORLD synthesizer) using the acoustic features directly extracted from real speech samples. Hence, the scores of these samples are expected to show the upper limit of the performance. We also included speech samples produced using the pairwise and many-to-many versions of RNN-S2S-VC, ConvS2S-VC and VTN, and sprocket in the stimuli. Speech samples were presented in random orders to eliminate bias as regards the order of the stimuli. Ten listeners participated in our listening tests. Each listener was asked to evaluate the naturalness by selecting “5: Excellent”, “4: Good”, “3: Fair”, “2: Poor”, or “1: Bad” for each utterance.

| Versions | Settings | Measures |
|----------|----------|----------|
|          | IML/ | L | H | MCD | LFC | LDR |
| PostLN   |        |   |   | MCD |    |    |
| −        | 4     | 1 |   | 0.81 | 0.683 | 3.96 |
|          | 2     |   |   | 0.96 | 0.659 | 5.73 |
|          | 4     |   |   | 0.94 | 0.659 | 4.12 |
|          | 1     |   |   | 7.13 | 0.652 | 3.69 |
|          | 2     |   |   | 7.02 | 0.654 | 4.45 |
|          | 4     |   |   | 7.72 | 0.576 | 5.17 |
| PreLN    |        |   |   | MCD |    |    |
| −        | 4     | 1 |   | 0.65 | 0.739 | 3.70 |
|          | 2     |   |   | 6.35 | 0.753 | 3.53 |
|          | 4     |   |   | 6.35 | 0.761 | 3.90 |
|          | 1     |   |   | 0.32 | 0.722 | 4.17 |
|          | 2     |   |   | 6.38 | 0.736 | 4.04 |
|          | 4     |   |   | 6.40 | 0.754 | 3.81 |
|          | 1     |   |   | 0.51 | 0.706 | 3.37 |
|          | 2     |   |   | 6.53 | 0.698 | 3.51 |
|          | 4     |   |   | 6.57 | 0.650 | 4.12 |
|          | 1     |   |   | 6.58 | 0.716 | 3.43 |
|          | 2     |   |   | 6.53 | 0.702 | 3.78 |
|          | 4     |   |   | 6.62 | 0.661 | 3.87 |
|          | 1     |   |   | 6.44 | 0.715 | 3.59 |
|          | 2     |   |   | 6.34 | 0.758 | 3.83 |
|          | 4     |   |   | 6.28 | 0.792 | 2.51 |
|          | 1     |   |   | 6.40 | 0.752 | 3.05 |
|          | 2     |   |   | 6.34 | 0.763 | 3.63 |
|          | 4     |   |   | 6.35 | 0.761 | 3.35 |
The results are shown in Fig. 9. As the results show, the pairwise VTN performed better than sprocket and the pairwise versions of the other S2S-based methods. We also confirmed that the many-to-many extension had a significant effect in improving the audio quality of all the S2S-based methods. It is worth noting that the many-to-many VTN performed better than all the competing methods including the many-to-many ConvS2S-VC, even though the many-to-many ConvS2S-VC was found to outperform the many-to-many VTN in terms of the MCD and LFC measures through the objective evaluation experiments, as reported earlier.

With the speaker similarity test, each subject was given a converted speech sample and a real speech sample of the corresponding target speaker and was asked to evaluate how likely they are to be produced by the same speaker by selecting “5: Definitely”, “4: Likely”, “3: Fair”, “2: Not very likely” or “1: Unlikely”. We used converted speech samples generated by the pairwise and many-to-many versions of RNN-S2S-VC and ConvS2S-VC, and sprocket for comparison as with the sound quality test. Each listener was presented $5 \times 10$ pairs of utterances. The results are shown in Fig. 10. As the results show, the many-to-many versions of ConvS2S-VC and VTN
performed comparably to each other, and performed slightly better than all other methods.

VI. CONCLUSIONS

This paper has proposed several extensions of VTN, which provide the flexibility of handling many-to-many, any-to-many and real-time VC tasks without relying on ASR models and text annotations. Through ablation studies, we confirmed the individual effect of each of the ideas introduced in the proposed method. Objective and subjective evaluation experiments on a speaker identity conversion task showed that the proposed method could perform better than baseline methods.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI 17H01763 and JST CREST Grant Number JPMJCR19A3, Japan.

REFERENCES

[1] A. Kain and M. W. Macon, “Spectral voice conversion for text-to-speech synthesis,” in Proc. ICASSP, 1998, pp. 285–288.
[2] A. B. Kain, J.-P. Hosom, X. Niu, J. P. van Santen, M. Fried-Oken, and J. Staeleny, “Improving the intelligibility of dysarthric speech,” Speech Commun, vol. 49, no. 9, pp. 743–759, 2007.
[3] K. Nakamura, T. Toda, H. Saruwatari, and K. Shikano, “Speaking-aid systems using GMM-based voice conversion for electrolaryngeal speech,” Speech Commun, vol. 54, no. 1, pp. 134–146, 2012.
[4] Z. Inanoglu and S. Young, “Data-driven emotion conversion in spoken English,” Speech Commun, vol. 51, no. 3, pp. 268–283, 2009.
[5] O. Turk and M. Schröder, “Evaluation of expressive speech synthesis with voice conversion and copy resynthesis techniques,” IEEE Trans. ASLP, vol. 18, no. 5, pp. 965–973, 2010.
[6] T. Toda, M. Nakagiri, and K. Shikano, “Statistical voice conversion techniques for body-conducted unvoiced speech enhancement,” IEEE Trans. ASLP, vol. 20, no. 9, pp. 2505–2517, 2012.
[7] D. Felps, H. Bortfeld, and R. Gutierrez-Osuna, “Foreign accent conversion in computer assisted pronunciation training,” Speech Communication, vol. 51, no. 10, pp. 920–932, 2009.
[8] Y. Stylilou, O. Cappé, and E. Moulines, “Continuous probabilistic transform for voice conversion,” IEEE Trans. SAP, vol. 6, no. 2, pp. 131–142, 1998.
[9] T. Toda, A. W. Black, and K. Tokuda, “Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory,” IEEE Trans. ASLP, vol. 15, no. 8, pp. 2222–2235, 2007.
[10] E. Helander, T. Virtanen, J. Nurmisen, and M. Gabbouj, “Voice conversion using partial least squares regression,” IEEE Trans. ASLP, vol. 18, no. 5, pp. 912–921, 2010.
[11] S. Desai, A. W. Black, B. Yegnarayanan, and K. Prahalad, “Spectral mapping using artificial neural networks for voice conversion,” IEEE Trans. ASLP, vol. 18, no. 5, pp. 954–964, 2010.
[12] S. H. Mohammadi and A. Kain, “Voice conversion using deep neural networks with speaker-independent pre-training,” in Proc. SLT, 2014, pp. 19–23.
[13] Y. Saito, S. Takamichi, and H. Saruwatari, “Voice conversion using input-to-output highway networks,” IEICE Trans Inf. Syst., vol. E100-D, no. 8, pp. 1925–1928, 2017.
[14] L. Sun, S. Kang, K. Li, and H. Meng, “Voice conversion using deep bidirectional long short-term memory based recurrent neural networks,” in Proc. ICASSP, 2015, pp. 4869–4873.
[15] T. Kaneko, H. Kameoka, K. Hiramatsu, and K. Kashino, “Sequence-to-sequence voice conversion with similarity metric learned using generative adversarial networks,” in Proc. Interspeech, 2017, pp. 1283–1287.
[16] C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang, “Voice conversion from non-parallel corpora using variational auto-encoder,” in Proc. APSIPA, 2016.
[17] ——, “Voice conversion from unaligned corpora using variational autoencoding Wasserstein generative adversarial networks,” in Proc. Interspeech, 2017, pp. 3364–3368.
[18] H. Kameoka, T. Kaneko, K. Tanaka, and N. Hojo, “ACVAE-VC: Non-parallel voice conversion with auxiliary classifier variational autoencoder,” IEEE Trans. ASLP, vol. 27, no. 9, pp. 1432–1443, 2019.
[19] T. Kaneko and H. Kameoka, “Non-parallel voice conversion using cycle-consistent adversarial networks,” in Proc. EUSIPCO, 2018, pp. 2114–2118.
[20] H. Kameoka, T. Kaneko, K. Tanaka, and N. Hojo, “StarGAN-VC: Non-parallel many-to-many voice conversion using star generative adversarial networks,” in Proc. SLT, 2018, pp. 266–273.
[21] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in Adv. NIPS, 2014, pp. 3104–3112.
[22] J.chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-based models for speech recognition,” in Adv. NIPS, 2015, pp. 577–585.
[23] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, “Tacotron: Towards end-to-end speech synthesis,” in Proc. Interspeech, 2017, pp. 4006–4010.
[24] S. O. Ark, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, A. Ng, J. Raiman, S. Sengupta, and M. Shoeybi, “Deep voice: Real-time neural text-to-speech,” in Proc. ICML, 2017.
[25] S. O. Ark, G. Diamos, A. Gibiansky, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou, “Deep voice 2: Multi-speaker neural text-to-speech,” in Proc. NIPS, 2017.
[26] J. Sotoelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. Courville, and Y. Bengio, “Char2Wav: End-to-end speech synthesis,” in Proc. ICLR, 2017.
[27] H. Tachibana, K. Uenoiyama, and S. Aihara, “Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention,” in Proc. ICASSP, 2018, pp. 4784–4788.
Fig. 9. Results of the MOS test for sound quality.

Fig. 10. Results of the MOS test for speaker similarity.