UNIFYSPEECH: A UNIFIED FRAMEWORK FOR ZERO-SHOT TEXT-TO-SPEECH AND VOICE CONVERSION

Anonymous authors
Paper under double-blind review

ABSTRACT

The unlabeled speech contains rich speaker style information, which can improve the few-shots modeling capability. This paper proposes UnifySpeech to make use of large amounts of unlabeled data for model training and boost the performance of zero-shot text-to-speech (TTS) and voice conversion (VC) simultaneously. UnifySpeech brings TTS and VC into a unified framework for the first time. The model is based on the assumption that speech can be decoupled into three independent components: content information, speaker information, prosody information. Both TTS and VC can be regarded as mining these three parts of information from the input and completing the reconstruction of speech. For TTS, the speech content information is derived from the text, while in VC it's derived from the source speech, so all the remaining units are shared except for the speech content extraction module in the two pipelines. We applied vector quantization and domain constrain to bridge the gap between the content domains of TTS and VC. Objective evaluation shows UnifySpeech gets higer speaker similarity and pitch prediction accuracy, indicating the improvements of the style modeling ability. Subjective evaluation shows speech generated by UnifySpeech obtains high mean opinion score (MOS) that the audio outperformed baseline model.

Index Terms: decoupling, zero-shot learning, text-to-speech, voice conversion, vector quantization

1 INTRODUCTION

Cloning the voice of the target speaker is an attractive technology, which can be applied to various scenes (Sisman et al., 2020), such as entertainment creation, personalized mobile assistants, security field, etc. The most ideal voice cloning operation is to give only one speech of the unseen target speaker as a reference and then any speech of the target speaker can be synthesized, which is called zero-shot voice clone. In the speech research community, text-to-speech (TTS) and voice conversion (VC) are two mainstream ways to realize voice clone (Sisman et al., 2020). Therefore, a variety of techniques for one-shot TTS and VC have been proposed recently (Gorodetskii & Ozhiganov, 2022; Tang et al., 2021).

However, although TTS and VC techniques are two important ways of voice clone with same output form, the research of TTS and VC is more or less independent. There isn't much interaction between them. But they are both speech synthesis task. In terms of speech generation, we categorize the information of the target speaker's speech into three kinds of information: (1) speech content, the characters of phonemes or phonetic posteriorgram (PPG) in voice conversion, represents the content of the speech. (2) speaker information, which represents the characteristics of speakers, is related to the speaker's articulation organ. (3) prosody information, which covers the intonation, stress, and rhythm of speech. According to FastSpeech2 (Ren et al., 2020), pitch, energy and duration information can reflected them certainly. As TTS extracts speech content directly from text, it is easier to obtain content information irrelevant to speaker than VC. As VC encounters more speakers, it's possible to obtain more robust speaker modeling ability. So, by integrating TTS and VC into a unify framework and combining their training data, it can help the model learn these three kinds of information better.

Unfortunately, investigating TTS and VC in the same framework is challenging, as the speech content are extracted from different input. Specifically, the speech content in TTS is obtained through

text information and the text and speech in TTS are unequal sequences, needing attention mechanism to align them. While the attention mechanism is often affected by the speaker's information, so it is impossible to learn the speech content representation completely irrelevant to the speaker. While in VC, the source speech and target speech are aligned in speech content, so the speech content can be extracted directly from the source speech, which is different from the TTS. In contrast to speech content information, speaker information and prosody information can be modeled using the same network in TTS and VC. Therefore, the difficulty here is to keep speech content for TTS and VC consistent. With the development of speech synthesis, the recently proposed Adaspeech2 (Yan et al., 2021) can combine text information with speech information, which can effectively decouple the speech content from the input. The unified framework becomes possible.

Overall, the main contributions of this paper are:

- We propose UnifySpeech, a unified framework for TTS and VC. VC enables unlabeled data to join training process, making TTS encounters more speaker. TTS enhance the ability for voice content decoupling in VC. Thus, both pipeline benefits from the other one.
- We apply vector quantization and domain constrain to bridge the gap between the content domains of TTS and VC. This ensures that TTS and VC can effectively use unlabeled data for training, safeguarding the proper operation of the model.
- We perform extensive experiments on two tasks: zero-shot TTS and zero-shot VC. Objective evaluation shows that unlabeled data enhances the model's speaker style modeling capability. Subjective evaluation indicates that the generated speech is as natural as human voice.

2 BACKGROUND

In this section, we will briefly review the background of this work, including neural TTS and VC models, and the zero-shot learning for TTS and VC tasks.

2.1 TEXT-TO-SPEECH TASK

TTS task is to model the mapping between text and speech, which is a modeling problem between two unequal length sequences. According to the alignment mechanism (Battenberg et al., 2020) between text and speech in the model, the end-to-end TTS model can be divided into two categories: 1) Using a neural network to learn the alignment information between text and speech, such as local sensitive attention in Tacotron (Wang et al., 2017b). Various improvements to the attention mechanism have been proposed. In addition, inspired by CTC (Kim et al., 2017) in ASR, glow-TTS (Kim et al., 2020), VITS (Kim et al., 2021) can automatically learn the alignment information. 2) By introducing the duration information (Ren et al., 2019) of phonemes as prior knowledge, the text is upsampled to achieve alignment (McAuliffe et al., 2017). Since the upsampled information based on text is independent of the speaker, the speech content can be well separated from the speech. Therefore, we introduce the duration information to build the TTS model in this paper.

2.2 VOICE CONVERSION TASK

Voice conversion can be seen as two steps (Sisman et al., 2020). Firstly, extract the speaker-independent speech content information from the source speech, and then embed the target speaker information to the speech content to reconstruct the speech of the target speaker. According to the way of extracting speech content, the VC model can be divided into two categories: (1) text-based approach. (2) Information bottleneck approach. The first approach requires an additional pre-trained ASR model. Since the ASR is trained in a supervised manner, it demands a lot of paired text and speech. Additionally, pipeline modeling is easy to accumulate errors and affects the performance of the system. Therefore, a lot of research work is focused on the latter. By adding some restrictions to the information bottleneck, different kinds of information can be decoupled. However, if the information bottleneck can not be decoupled well, the performance of the model will be significantly reduced.

2.3 ZERO SHOT LEARNING FOR TTS AND VC

The research of zero-shot learning based on TTS and VC focused on how to extract effective speaker information and then embed it into TTS or VC model for joint training or segmented training. Typical speaker features include i-vector (Wang et al., 2017a), d-vector (Variani et al., 2014), x-vector (Snyder et al., 2018) and so on. In addition, the modules that extract speaker-style information through the specially designed network structure, such as GST (Wang et al., 2018), VAE (Van Den Oord et al., 2017), can also achieve good results.

3 UNIFYSPEECH

In this section, we introduce the details of UnifySpeech for zero-shot TTS and VC tasks. We first give the key idea of UnifySpeech: speech factorization, and then introduce the formulation of UnifySpeech. Finally, we will describe the model structure of UnifySpeech.

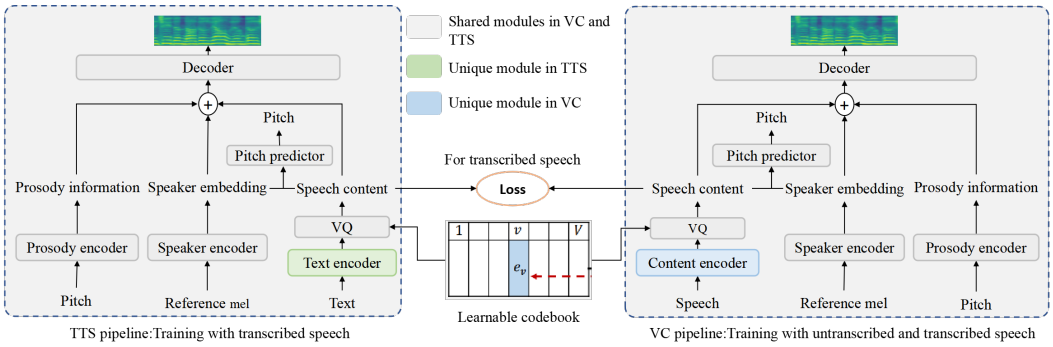

Figure 1: Structure of UnifySpeech.

3.1 SPEECH REPRESENTATION DISENTANGLEMENT

The core of the controllable and migratable speech generation task is to decouple the components of the generated speech first, and then control and transfer each component. Although some end-to-end models can directly model the relationship between text and speech (TTS) or speech-to-speech task (VC), due to the mutual coupling of various components of the end-to-end model, it brings great difficulties to the transfer learning of the model. Therefore, we first decouple the speech generation task into three independent components and then input them into the decoder to synthesize speech. This idea is also the main architecture of UnifySpeech. The three components and their sources will be described in detail below.

Speech Content: To generate intelligible speech signals, it is important to model accurate speech content information. Speech content is linguistic information, which is irrelevant to the speaker. Due to the different types of input signals of TTS and VC, the ways of extracting speech content are different. For TTS, the source of speech content is text. Firstly, to learn the context information of the text, a text encoder is used to encode the text to obtain the context representation. The context representation is up-sampled according to the phoneme's duration information to obtain the speech content information. For VC, since the source speech is aligned with the target speech, we directly use a content encoder to extract speech content information from the source speech.

Speaker: The speaker information includes the speaker's characteristics, such as the timbre, volume, etc. We can extract the speaker information from the given speech of the target speaker. This is common for TTS and VC tasks, so the speaker extraction network can be shared in the two tasks. Since the speaker extraction network can directly extract information from speech without text, so a large number of data without text annotation in the VC task can be used for training, which can help to improve the transfer learning ability of the model.

Prosody: The prosody information represents how the speaker says the content information. It is independent of the speaker information and related to the way of expression. Since the pitch information can reflect the rhythm of speech, therefore, the pitch information is used to extract prosody information. The prosody information, like speaker information, can be shared by both TTS and VC. In addition, in the training process, we can obtain pitch information from the ground truth speech, but there is no ground truth in the inference stage. Therefore, a pitch prediction module (Ren et al., 2020) is proposed in the training stage, which takes the speech content information and speaker information as the input to predict the pitch information.

3.2 SPEECH CONTENT WITH VECTOR QUANTIZATION

Since there are different ways to obtain the speech content in TTS and VC, it is very easy to deviate between the two domains. If this deviation occurs, it will cause devastating damage to some downstream shared modules (such as pitch predictor and decoder). Therefore, to ensure that the consistency of the two speech content domains as close as possible, we reconstruct them from two aspects:

- First, we use the shared codebook to quantify the continuous feature space.
- Second, we use the labeled data to narrow two discrete feature spaces.

The detailed process is described below. Suppose that the vector obtained by the text encoder in TTS pipeline is $C_p = (C_p^1, C_p^2, \cdots, C_p^T)$ with length T, the vector obtained by the content encoder in VC pipeline is $C_s = (C_s^1, C_s^2, \cdots, C_s^{T'})$ with length T'. It should be noted that we add a length regulator module in the text encoder to solve the problem of length mismatch between the text and speech sequence, which is introduced in FastSpeech (Ren et al., 2019). Therefore, if text and speech are paired, $T' = T$. The vector C_p and C_s is a sequence of continuous vector in Eq. Due to the large representation range of continuous features, C_p and C_s are difficult to match. We borrow the discretization method for latent variables from Vector Quantized Variational AutoEncoder (Van Den Oord et al., 2017). Specifically, for each time step t, the continuous latent representations C_p^t in C_p can be mapped into $\overline{C}_p^t$ by finding the nearest pre-defined discretized embedding in the dictionary as:

$$\overline{C}_p^t = e_k, \quad k = argmin_j \left\| C_p^t - e_j \right\|_2 \tag{1}$$

where e_j is the j-th embedding in the codebook dictionary, and $j \in 1, 2, \cdots, V$. Since selecting the entry with the minimum distance will cause the operation to be non-differentiable, the straight-through gradient estimator can be used to approximate the gradient, which can be expressed as:

$$\bar{h}_t = h_t + e_v - \mathrm{sg}\left(h_t\right), \quad v = \arg\min_k \left\| h_t - e_k \right\|_2 \tag{2}$$

where sg$(\cdot)$ is the stop-gradient (Van Den Oord et al., 2017) operation that treats its input as constant during back-propagation.

After vector quantization, the quantized sequence $\overline{C_p} = (\overline{C}_p^1, \overline{C}_p^2, \cdots, \overline{C}_p^T)$ and $\overline{C_s} = (\overline{C}_s^1, \overline{C}_s^2, \cdots, \overline{C}_s^T)$ can be obtained. It should be noted that when C_p and C_s are quantized into $\overline{C_p}$ and $\overline{C_s}$, they share the same codebook e. The advantage of this is that since the speech content features $\overline{C_p}$ in the TTS pipeline are independent of the speaker, sharing the same codebook can help learn the speech content features $\overline{C_s}$ independent of the speaker in the VC pipeline, which is essential for the VC task.

Although both pipeline use the same codebook for coding, there is no guarantee that there is no deviation between the two fields after coding. Therefore, to further eliminate the deviation between the two domains, we use the labeled speech in the TTS pipeline to supervise the training of the two domains. Specifically, for paired text and speech data, we constrain the feature distance between the quantized sequence $\overline{C_p}$ and $\overline{C_s}$:

$$\mathcal{L}_{\text{pair}} = \left\| \overline{C_p} - \overline{C_s} \right\|_2^2 \tag{3}$$

In this way, we can efficiently minimize the domain discrepancy by using limited labeled data.

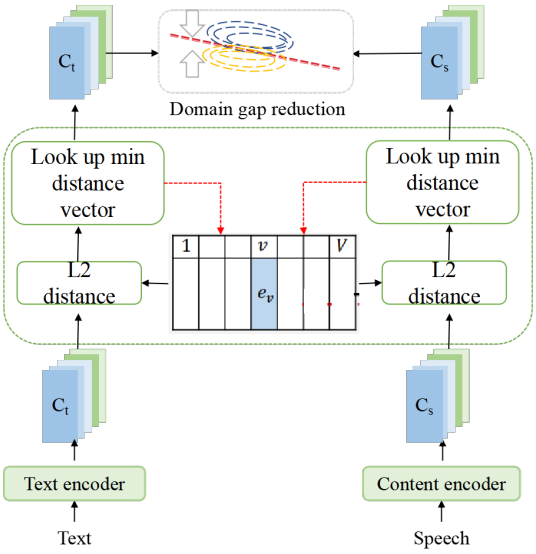

Figure 2: Structure of vector quantized operation.

3.3 UNIFYSPEECH PIPELINE

An overview of our proposed UnifySpeech architecture is illustrated in Fig. 2. It consists of a sequence-to-sequence TTS, and a sequence-to-sequence VC. The key idea is to share most of the module parameters (speaker encoder, prosody encoder, decoder and pitch predictor) and map the speech content in TTS and VC to the same space. As mentioned above, the UnifySpeech allows us to train the model on the concatenation of both the labeled and unlabeled data. For supervised training with labeled data, both models can be trained independently by minimizing the loss between their predicted speech and the ground truth. For unsupervised training with unlabeled data, the VC pipeline can be trained, and the parameters are shared with TTS.

To further clarify the training process, we unrolled the framework as follows.

3.3.1 TTS PIPELINE

Denote the text and speech sequence pair $(x, y, F0) \in D$, where D is the paired text and speech corpus. Each element in the text sequence x represents a phoneme or character, while each element in the speech sequence y represents a frame of speech. $F0$ is the pitch information of y. The representation obtained after three encoders are speech content C, speaker S and prosody P.

Then, the three parts are added and input into a decoder to obtain the predicted speech y'. In addition, to obtain the pitch information in the interference stage, we use the content information and speaker information to predict the F0. These processes can be expressed as:

$$y' = \text{decoder}(C + S + P) \tag{4}$$

$$F0' = \text{pitch\_predictor}(C + S) \tag{5}$$

Therefore, the reconstruction loss in TTS process includes two parts:

$$\mathcal{L}_{rec}^{VC} = MSE(y, y') + MSE(F0, F0') \tag{6}$$

In addition, we use the content encoder in the VC pipeline to extract the content representation C_s for the training speech in TTS, and close the distance between the two domains by calculating the distance loss of C_s and C_p, which is explained in Sec. 3.2.

$$\mathcal{L}_{\text{pair}} = \left\| \overline{C_p} - \overline{C_s} \right\|_2^2 \tag{7}$$

The loss of TTS pipeline can be expressed as:

$$\mathcal{L}^{TTS} = \mathcal{L}_{rec}^{TTS} + \mathcal{L}_{\text{pair}} \tag{8}$$

where MSE denotes the mean squared errors.

3.3.2 VC PIPELINE

Denote all the unlabeled or labeled speech $(y, F0) \in Y$. We first extract three information from $(y, F0)$, which are speech content C_s, speaker S_s and prosody P_s.

Then, similar to the TTS pipeline, the shared decoder and pitch predictor module is used to predict the speech signal y' and $F0'$, which is similar to the Eq.1 and Eq.2.

The loss of VC pipeline only includes reconstruction loss, which can be expressed as:

$$\mathcal{L}^{VC} = MSE(Y, Y') + MSE(F0, F0') \tag{9}$$

where MSE denotes the mean squared errors.

3.3.3 TRAINING PROCESS

With such a unified framework, TTS and VC can learn from each other through joint training. The details of the algorithm can be found below.

Procedure 1 UnifySpeech training algorithm

1: **Input**: Paired speech and text dataset (x, y), speech only dataset $y^{'}$
2: **repeat**
3: # A. TTS pipeline with speech-text data pairs
   1. Sample paired speech and text (x, y)
   2. Extract speech content information from x for domain loss
   3. Generate the predict speech y, pitch F0 and speech content from text
   4. Calculate the loss for TTS $\mathcal{L}_{rec}^{TTS}$
4: # B. VC pipeline with speech-only data
   1. Sample paired speech and text (x, y)
   2. Extract speech content information from y for domain loss
   3. Calculate the domain loss for the two pipeline $\mathcal{L}_{pair}$
   4. Sample speech $y^{'}$ in speech only dataset
   5. Generate the predict speech y, pitch F0 and speech content from speech $y^{'}$
   6. Calculate the loss for VC $\mathcal{L}^{VC}$
5: # C. Loss combination:
   1. Combine all loss ($\mathcal{L}_{rec}^{TTS}$, $\mathcal{L}_{pair}$, $\mathcal{L}^{VC}$) into a single loss variable
   2. Calculate TTS and VC parameters gradient
   3. Update TTS and VC parameters with gradient descent optimization
6: **until** convergence

4 EXPERIMENTS AND RESULTS ANALYSIS

In this section, we conduct experiments to evaluate our proposed methods. The experiments are carried out from two aspects: zero-shot TTS, zero-shot VC.

4.1 DATASETS

Two datasets are used to simulate labeled data and unlabeled data, respectively. VCTK dataset, an English language dataset containing 44 hours of speech and 109 speakers is used as labeled data. Each speaker has approximately 400 sentences. LibriTTS (Zen et al., 2019) are used as unlabeled data, which consists of 585 hours of speech data from 2484 speakers. We only use the speech data in LibriTTS and discard the text for unsupervised training. In this way, it can simulate the scene where a large number of speech that we can obtain are unlabeled. We use a 16-bit, 22050 Hz sampling rate for all experiments. The 80-dim Mel spectrogram is extracted with Hann windowing, frame shift of 12.5-ms, frame length of 50-ms, and 1024-point Fourier transform. In this experiment, we use hifigan (Lei et al., 2022) as vocoder.

4.2 MODEL DETAILS

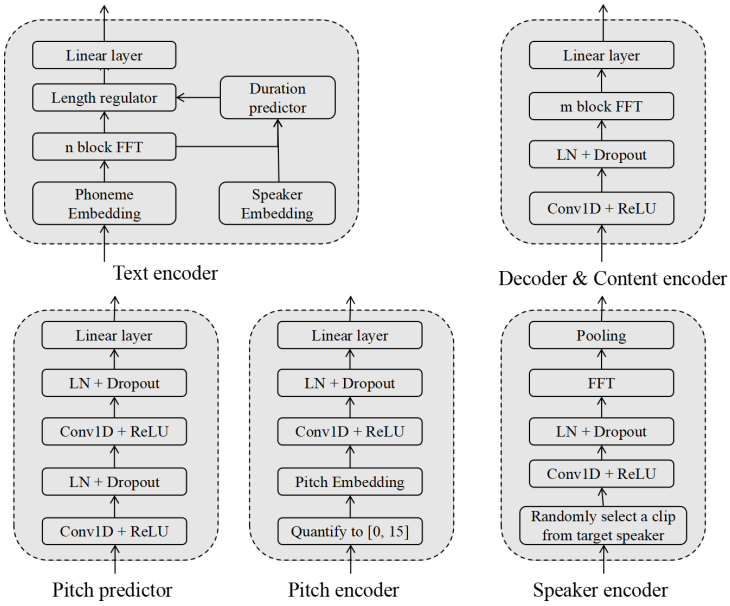

Figure 3: Structure of each module in UnifySpeech. FFT (Ren et al., 2019) means feed-forward Transformer.

The detail of each module in our proposed method is shown in the Fig. 3. Specifically, to make the sequence of speech content extracted from the text encoder and content encoder equal, a length regulator is added to the text encoder, which is inspired by the FastSpeech (Ren et al., 2019). The structure of the duration predictor is the same as that in FastSpeech. The structure of the decoder and content encoder is similar, but the dimensions of input and output are opposite. For the prosody encoder, we first quantize F0 of each frame to 15 possible values and encode them into a learnable embedding vector according to the value. By this way, pitch prediction becomes a classification task, reducing the difficulty of frame-level pitch prediction. For the speaker encoder, we first randomly select a speech from the speaker as the reference speech. The linear layer of each encoder is to unify the output dimensions to the same dimension so that the addition operation can be performed.

The number of feed-forward Transformer (Vaswani et al., 2017) (FFT) blocks in the text encoder is 2 and it is 6 in the decoder module. In each FFT block, the dimension of hidden states is 256. The kernel sizes of all the 1D-convolution are set to 3. The dropout rate is set to 0.5. The dimension of the last linear layer in the decoder is 80, which is consistent with Mel spectral dimension. The dimension of last linear layer in encoders (text encoder, pitch encoder, content encoder) is 256. An Adam optimizer (Kingma & Ba, 2014) is used to update the parameters. The init learning rate is 0.001 and the learning rate decreased exponentially.

4.3 SPEAKER SIMILARITY EVALUATION

We first evaluate the speaker similarity of zero-shot tasks. The speaker embedding cosine similarity here is adopted for evaluating. For both task, we separately calculate the Average Cosine Similarity (ACS) (Lei et al., 2022) for the embedding from same (S-ACS) and different speakers (D-ACS). And then their ratio is used as an evaluation metric.

When the TTS pipeline is jointly trained with the VC pipeline, it obtains more accurate speaker information from reference speech. This shows that the unlabeled data applied only in the VC pipeline improves the speaker style modeling capability.

Table 1: Ratio of ACS from same and different speaker. Unseen or seen means whether the speaker is from test set. With or without VC means whether the TTS pipeline is trained with VC pipeline.

Model	$\frac{S-ACS}{D-ACS}$(unseen)	$\frac{S-ACS}{D-ACS}$(seen)
TTS (with VC)	**2.8**	**6.0**
TTS (without VC)	2.1	4.9

For the VC pipeline, we randomly choose some target speeches of different speakers from both training set and testing set. Then we use the Speaker Encoder to get their speaker embedding and calculating the ration of ACS from same and different speakers. Results are shown in Table 2.

It can be found that when VC pipeline is trained alone, its performance is poor. In other words, it doesn't have the ability to discriminate. But jointly training with TTS improve its speaker style modeling ability.

Table 2: Ratio of ACS from same and different speaker. Unseen or seen means whether the speaker is from test set. With or without TTS means whether the VC pipeline is trained with TTS pipeline.

Model	$\frac{S-ACS}{D-ACS}$(unseen)	$\frac{S-ACS}{D-ACS}$(seen)
VC (with TTS)	**2.8**	**6.0**
VC (without TTS)	1.0	1.0

4.4 Speaker Embedding Visualization

To further investigate the speaker embedding similarity of the generated speeches, we apply the t-SNE method (Van der Maaten & Hinton, 2008) to reduce the dimension of speaker embedding from 256 to 2. And then show the results in the same coordinate system. Here, we only show the results of the model which is jointly trained with both parts of UnifySpeech.

Fig. 4 shows the speaker visualization. For the seen speaker (x) and unseen speaker (o), the Corresponding speaker embedding form a cluster and distinct from others. The boundary between different speakers is clear. This shows that the Speaker Encoder performs well.

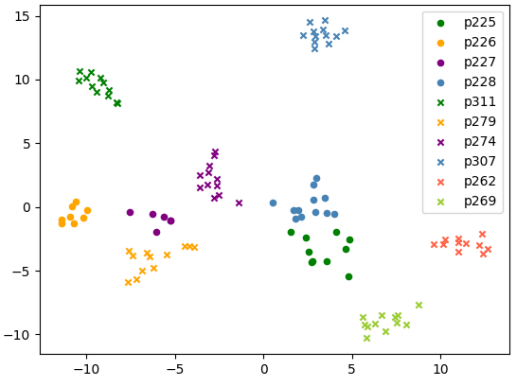

Figure 4: Speaker visualization of generated speech,where different shape represent whether the speaker is in training set.The circle indicate unseen speaker,while x indicate seen speaker.

4.5 Naturalness

Objective evaluation was conducted to compare the speech quality from the TTS pipeline of UnifySpeech trained with or without the VC pipeline. F0 RMSE (root of mean square errors of F0), MCD (Kubichek, 1993) (Mel-cepstrum distortion), V/UV (the error rate of voicing/unvoicing flags) and F0 CORR (correlation factor of F0) are adopted as metrics and are calculated on the test set of VCTK. It can be found that TTS trained with VC performed better on pitch prediction. It illustrates that unlabeled data fed in VC improves the model's style modeling ability.

Table 3: Objective evaluation results on the VCTK dataset.

Model	F0 RMSE (Hz)	MCD (db)	V/UV	F0 CORR
TTS (trained with VC)	**22.52**	3.77	29.95%	**0.90**
TTS (trained without VC)	26.26	**3.75**	**28.81**%	0.86

We also use mean opinion score (MOS) to evaluate the quality of synthesized speech. Here we separately conduct the test on TTS (trained with and without VC) and VC (trained without and with TTS). The results are as follows.

Table 4: Mean opinion score (MOS) of the models.

Model	MOS
TTS (trained with VC)	**3.87**
TTS (trained without VC)	3.83
VC (trained with TTS)	3.61
VC (trained without TTS)	**3.92**
GT (VCTK)	**4.21**
GT (HifiGAN)	4.12

MOS shows UnifySpeech realize high naturalness. But it should be noted that VC (without TTS) is a simple reconstruction of the source speech, not changing its style at all. So it's still not of good quality.

4.6 Result analysis

It can be found that joint training has improve both pipeline's speaker style modeling ability, especially for VC.

For TTS, sharing modules with VC enables unlabeled data to participate in its training process. Along with the richer speaker style pattern, the speaker style modeling capability has been enhanced.

For VC, when trained alone, it doesn't have the speaker modeling ability. As the self-supervised training process aims at reducing the reconstruction loss of source audio, the reference audio of the target speaker isn't crucial in the process. Only the content encoder and the decoder are enough for the process, that's possibly why the speaker embeddings are all similar, though they are from different speakers. When jointly trained, the text loss plays a role of regularization factor, resulting that the content encoder just extracting the content information from the source speech (speaker information is reserved and others are discarded). This makes the reference speech with rich speaker information become indispensable for the reconstruction process. Thus the model's speaker modeling ability has been improved.

5 Conclusions

In this paper, we propose UnifySpeech, a unified framework for TTS and VC. Both task benefits from the other one. Due to training with large amounts of unlabeled data, their few-shot modeling ability makes progress as well as the synthesized speech's quality. In the future, further improving the synthesized speech's quality and making the generated speech's style more similar to target speaker will be our endeavor.

References

Eric Battenberg, RJ Skerry-Ryan, Soroosh Mariooryad, Daisy Stanton, David Kao, Matt Shannon, and Tom Bagby. Location-relative attention mechanisms for robust long-form speech synthesis. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6194–6198. IEEE, 2020.

Artem Gorodetskii and Ivan Ozhiganov. Zero-shot long-form voice cloning with dynamic convolution attention. *arXiv preprint arXiv:2201.10375*, 2022.

Jaehyeon Kim, Sungwon Kim, Jungil Kong, and Sungroh Yoon. Glow-tts: A generative flow for text-to-speech via monotonic alignment search. *Advances in Neural Information Processing Systems*, 33:8067–8077, 2020.

Jaehyeon Kim, Jungil Kong, and Juhee Son. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In *International Conference on Machine Learning*, pp. 5530–5540. PMLR, 2021.

Suyoun Kim, Takaaki Hori, and Shinji Watanabe. 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.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Robert Kubichek. Mel-cepstral distance measure for objective speech quality assessment. In *Proceedings of IEEE pacific rim conference on communications computers and signal processing*, volume 1, pp. 125–128. IEEE, 1993.

Yi Lei, Shan Yang, Jian Cong, Lei Xie, and Dan Su. Glow-wavegan 2: High-quality zero-shot text-to-speech synthesis and any-to-any voice conversion. *arXiv preprint arXiv:2207.01832*, 2022.

Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: Trainable text-speech alignment using kaldi. In *Interspeech*, volume 2017, pp. 498–502, 2017.

Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: Fast, robust and controllable text to speech. *Advances in Neural Information Processing Systems*, 32, 2019.

Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech 2: Fast and high-quality end-to-end text to speech. *arXiv preprint arXiv:2006.04558*, 2020.

Berrak Sisman, Junichi Yamagishi, Simon King, and Haizhou Li. An overview of voice conversion and its challenges: From statistical modeling to deep learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:132–157, 2020.

David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5329–5333. IEEE, 2018.

Chuanxin Tang, Chong Luo, Zhiyuan Zhao, Dacheng Yin, Yucheng Zhao, and Wenjun Zeng. Zero-shot text-to-speech for text-based insertion in audio narration. *arXiv preprint arXiv:2109.05426*, 2021.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez. Deep neural networks for small footprint text-dependent speaker verification. In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4052–4056. IEEE, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Shuai Wang, Yanmin Qian, and Kai Yu. What does the speaker embedding encode? In *Interspeech*, pp. 1497–1501, 2017a.

Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017b.

Yuxuan Wang, Daisy Stanton, Yu Zhang, RJ-Skerry Ryan, Eric Battenberg, Joel Shor, Ying Xiao, Ye Jia, Fei Ren, and Rif A Saurous. 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.

Yuzi Yan, Xu Tan, Bohan Li, Tao Qin, Sheng Zhao, Yuan Shen, and Tie-Yan Liu. Adaspeech 2: Adaptive text to speech with untranscribed data. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6613–6617. IEEE, 2021.

Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. Libritts: A corpus derived from librispeech for text-to-speech. *arXiv preprint arXiv:1904.02882*, 2019.