ZERO-SHOT SINGING VOICE SYNTHESIS FROM MUSICAL SCORE

Jun-You Wang[1], *Hung-yi Lee*[1], *Jyh-Shing Roger Jang*[1], *Li Su*[2]

[1]National Taiwan University, Taiwan
[2]Institute of Information Science, Academia Sinica, Taiwan

ABSTRACT

Zero-shot singing voice synthesis (SVS), the task to synthesize the singing voice of an arbitrary target singer, has gained increasing attentions in the past few years. Several recently proposed systems have demonstrated promising results on this task. However, these systems require detailed musical features at the frame level as the musical content. To deal with this issue, we propose a model that performs zero-shot SVS with only musical score as the musical content condition. To help model training, we build an acoustic encoder that extracts linguistic features from audio, and train it with the lyrics transcription objective. The output of the acoustic encoder serves as an alternative to the musical score, allowing the SVS model to learn from weakly labeled data. Results suggest that the proposed method outperforms baseline semi-supervised method in both subjective and objective tests.

Index Terms— Singing voice synthesis, zero-shot, semi-weakly-supervised learning

1. INTRODUCTION

Zero-shot singing voice synthesis (SVS) [1–4] is a task that aims to synthesize any unseen target singer's voice. Recent works on zero-shot SVS have demonstrated promising results by adopting advanced model architectures from text-to-speech synthesis (TTS) [5–7] and voice conversion (VC) [8]. Typically, this task is addressed by decomposing the singing voice into 1) musical content such as phoneme, pitch (f0), and energy (volume), and 2) singer identity information such as timbre and pronunciation styles. Models that disentangle the two pieces of information allow one to synthesize a singing voice conditioned on any singer by having the singer identity representation of that singer.

Currently in previous zero-shot SVS works, musical content needs to be sampled framewisely. In practice, the musical content is actually a set of frame-level musical features extracted from a recording of the same song performed by another singer. F0 estimators [9], speech-to-text aligners [10] [1], and energy computing algorithms are necessary for preprocessing [1–4]. This approach, however, cannot be directly applied to the case when no groundtruth recording sung by any singer is available, e.g., when a new song is written. In this case, one has to manually label musical contents at the frame level, which is not realistic in practice. This poses a question that, is it possible to build a zero-shot SVS model that only requires a musical score (i.e., a note sequence) as the musical content?

[1]In [1], their SVS model was trained to do this by itself.

Model	Musical content		Target singer
	Score	Frame-level features	
[11–14]	✓	–	One
[15–17]	–	✓	Many
[18]	✓	–	Many
[1–4]	–	✓	Any
Proposed	✓	–	Any

Table 1. The comparison of the SVS problem formulation in this paper with previous work. The *musical content* column shows the desired input as the musical content, while the *target singer* column shows the number of singers' voice that can be synthesized, ranging from one (single-singer SVS), many (multi-singer SVS) to any (zero-shot SVS).

In this paper, we propose the task of *zero-shot SVS from musical score*, a new zero-shot SVS task in which the musical content is only a musical score rather than the frame-level musical features, while the singer identity condition is still provided by a reference audio of a target singer. The main difference between musical score and frame-level musical features is that musical score is a relatively high-level representation. It consists of a series of notes that indicate the pitch, timings, and lyrics that the singer is expected to sing, and has been widely used in the history of music. However, it does not strictly define the details of the singing at frame level, which is usually determined by the singer's interpretation of the musical score in the actual performance. Considering the granularity of the labels, it is much easier for a composer to create a musical score than frame-level features.

Our model contains a musical score encoder that first predicts frame-level features from the input musical score, and then encodes the predicted frame-level musical features. The main difficulty of training such a model is the lack of training data. Since the musical score is highly correlated to music theory, it requires music experts to label manually or semi-automatically [19]. This hinders the creation of a large-scale singing dataset with musical score labels. Inspired by previous work on SVS [13,18], we introduce an additional acoustic encoder that extracts linguistic features from the audio, which serves as an alternative to the musical score. This allows the model to learn from data without musical score label, which increases the amount of available training data. Furthermore, to ensure that the acoustic encoder extracts correct linguistic features, we propose to use automatic lyrics transcription (ALT) as an auxiliary task. Both subjective and objective results suggest that the proposed method outperforms the baseline that does not use the ALT auxiliary task for training.

2. RELATED WORK

Singing voice synthesis (SVS) focuses on generating singing voice. It can be viewed as the singing version of TTS. However, unlike TTS that explicitly states that the content input is the text, the musical content input of a SVS system varies among previous works, which can be divided into two categories. The SVS systems in the first category take the phonetic, pitch and energy information at frame level as the musical content [1–3, 15–17, 20], which we refer to as frame-level musical features. Other SVS systems take only the musical score [2] as musical content [11–14, 18], which contains a series of notes. Each note indicates the pitch and lyrics that the singer is expected to sing within a certain time interval.

As for the input singer identity condition, previous SVS systems can be divided into three categories. The first category is single-singer SVS system that is designed to synthesize only one singer's voice [11–14, 21–23]. The second category is multi-singer SVS system, which is designed to generate a closed set of singers' voice [15–18]. These systems take a singer id as the singer identity condition. It controls which singer's voice should the SVS system synthesize. The last category is zero-shot SVS system, which is designed to synthesize any target singer's voice [1–4]. It takes a reference audio as the singer identity condition and extracts singer related features from it. Then, it synthesizes the singing voice that has the same singer identity as the reference audio. Table 1 shows the comparison of previous works and this work in terms of the problem formulation.

In our problem formulation, we want to build an SVS model that can both achieve zero-shot SVS and perform SVS from musical score. This requires the model to disentangle musical content from singer identity, which has to learn from a large-scale and diverse dataset. However, there is a lack of large-scale paired data of singing voice and musical score, which poses a challenge that has to be addressed. While there

[2]Note that the musical score we define here is a rather generalized definition which is not limited to sheet music. For example, a MIDI file with lyrics label for each note is also a form of "musical score". Here we focus on the granularity of the musical content (either at frame level or note level).

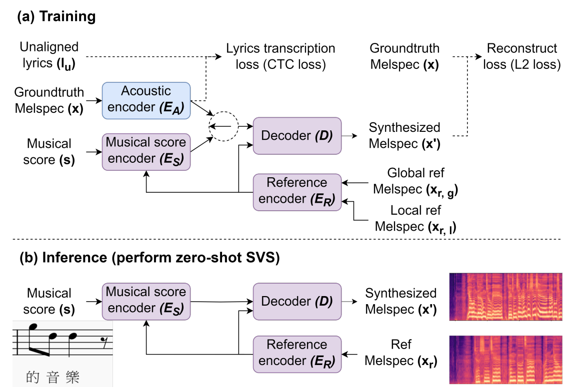

Fig. 1. The diagram of the proposed model in two phases: (a) During training, by adding an acoustic encoder E_A (the blue block), our model can be trained with either weakly labeled data or strongly labeled data. (b) During inference, we drop E_A and perform zero-shot SVS from the musical score $\mathbf{s}$.

are previous works that also take musical score as musical content [11–14, 18], since these models were not designed for zero-shot SVS, they do not suffer from this issue.

However, there are still several SVS works that discussed the use of weakly-labeled data or unlabeled data for training SVS models. Bonada and Blaauw [18] proposed to add an acoustic encoder to map unlabeled audio to the encoding space of musical content, which serves as an alternative to musical score. Choi and Nam [13] first trained a phoneme classifier to provide the phoneme prediction from unlabeled audio, and then used the phoneme prediction as an alternative to train an SVS model. In this work, we apply a similar method to Bonada and Blaauw [18], but add the ALT objective to train the acoustic encoder. This objective encourages the acoustic encoder to produce the correct linguistic features while eliminating features that are not related to musical content, i.e., the singer identity information, which helps the zero-shot SVS model disentangle the two components.

3. PROPOSED METHOD

3.1. Proposed model

As shown in Figure 1, the proposed model contains a musical score encoder E_S, an acoustic encoder E_A, a reference encoder E_R, and a decoder D. It can be trained under two training modes: 1) strongly labeled data, which contains paired data of audio and musical score, and 2) weakly labeled data, which only contains audio and lyrics that are labeled at utterance level. The first mode is the traditional supervised SVS training approach, which only works when the musical score is available. On the other hand, the second mode only requires lyrics label that does not have to be manually aligned with the audio. This increases the amount of data that can be used for training.

During training (Figure 1(a)), the model takes either a musical score $\mathbf{s}$ or the log-Mel spectrogram of the groundtruth audio $\mathbf{x}$ as the musical content, and takes the log-Mel spectrograms of two reference audios, $\mathbf{x_{r,l}}$ and $\mathbf{x_{r,g}}$, as the singer identity condition. In particular, $\mathbf{s}$ contains note labels where each note is labeled with the onset time, offset time, note pitch, and lyrics. When $\mathbf{s}$ is not available, we directly take the groundtruth audio $\mathbf{x}$ as an alternative to $\mathbf{s}$, which also serves as the musical content condition to the model (in the weakly labeled mode). As for the reference audios, $\mathbf{x_{r,g}}$ provides the global singer encoding, while $\mathbf{x_{r,l}}$ provides the local features that are related to singer identity such as the pronunciation of a certain phoneme, which we refer to as the local singer encoding. The effectiveness of this multi-reference training scheme has been proven in previous SVS work [3]. In practice, we set $\mathbf{x_{r,l}}$ to the same as $\mathbf{x}$, and concatenate 5 clips sung by the same singer as $\mathbf{x}$ to form $\mathbf{x_{r,g}}$.

Strongly labeled mode. When the paired data of $\mathbf{s}$ and $\mathbf{x}$ are available, we run the model as follows:

$$\mathbf{z_g}, \mathbf{z_l} = E_R(\mathbf{x_{r,g}}, \mathbf{x_{r,l}})\,; \tag{1}$$
$$\mathbf{z_c}, \mathbf{t}', \mathbf{p}', \mathbf{l}', \mathbf{e}' = E_S(\mathbf{s}, \mathbf{z_g})\,; \tag{2}$$
$$\mathbf{x_0}', \mathbf{x}' = D(\mathbf{z_c}, \mathbf{z_g}, \mathbf{z_l})\,. \tag{3}$$

First, E_R extracts two sets of singer identity features, $\mathbf{z_g}$ and $\mathbf{z_l}$, from $\mathbf{x_{r,g}}$ and $\mathbf{x_{r,l}}$, respectively. $\mathbf{z_g}$ serves as the global singer encoding, while $\mathbf{z_l}$ serves the local singer encoding. Then, E_S converts $\mathbf{s}$ to the content encoding $\mathbf{z_c}$. It first explicitly predicts four frame-level musical features from $\mathbf{s}$, including 1) time-lag $\mathbf{t}'$, the differences between the onsets in the musical score and the actual performance, as introduced in [11], 2) the phoneme alignment between the phoneme sequence extracted from $\mathbf{s}$ and the predicted frames, which is further converted to the framewise phoneme prediction $\mathbf{l}'$, 3) pitch contour $\mathbf{p}'$, and 4) log-energy contour $\mathbf{e}'$. Then, E_S further encodes these features to form $\mathbf{z_c}$. These features are also optimized by feature prediction losses. Finally, D synthesizes the log-Mel spectrogram $\mathbf{x}'$ from $\mathbf{z_c}$, $\mathbf{z_g}$, and $\mathbf{z_l}$. Similar to [24], we treat the last few layers of D as a PostNet, and also use the output before the PostNet, denoted as $\mathbf{x_0}'$, for optimization to speedup model convergence.

Then, suppose the groundtruth log-Mel spectrogram, time-lag, phoneme alignment, pitch contour, log-energy contour are $\mathbf{x}$, $\mathbf{t}$, $\mathbf{l}$, $\mathbf{p}$, $\mathbf{e}$, respectively, we apply the following loss functions to optimize the model:

$$\mathcal{L}_{\mathrm{r}} = 0.5 \times (\mathrm{L2}(\mathbf{x}', \mathbf{x}) + \mathrm{L2}(\mathbf{x_0}', \mathbf{x}))\,, \tag{4}$$
$$\mathcal{L}_{\mathrm{p}} = \mathrm{L1}(\mathbf{p}', \mathbf{p})\,, \tag{5}$$
$$\mathcal{L}_{\mathrm{a}} = \mathrm{mean}(\max((\mathrm{L1}(\mathbf{l}', \mathbf{l}) - 0.25), 0))\,, \tag{6}$$
$$\mathcal{L}_{\mathrm{e}} = \mathrm{L1}(\mathbf{e}', \mathbf{e}) + \mathrm{L1}(\mathrm{diff}(\mathbf{e}'), \mathrm{diff}(\mathbf{e}))\,, \tag{7}$$
$$\mathcal{L}_{\mathrm{t}} = \mathrm{L1}(\mathbf{t}', \mathbf{t})\,, \tag{8}$$
$$\mathcal{L}_{\mathrm{strong}} = \mathcal{L}_{\mathrm{r}} + \lambda_{\mathrm{p}}\mathcal{L}_{\mathrm{p}} + \lambda_{\mathrm{a}}\mathcal{L}_{\mathrm{a}} + \lambda_{\mathrm{e}}\mathcal{L}_{\mathrm{e}} + \lambda_{\mathrm{t}}\mathcal{L}_{\mathrm{t}}\,, \tag{9}$$

where L2 denotes the L2 loss, L1 denotes the L1 loss, diff denotes the difference between adjacent frames. $\mathcal{L}_{\mathrm{p}}$, $\mathcal{L}_{\mathrm{a}}$, $\mathcal{L}_{\mathrm{e}}$, and $\mathcal{L}_{\mathrm{t}}$ serve as the feature prediction losses. By applying these losses and the log-Mel spectrogram reconstruction loss $\mathcal{L}_{\mathrm{r}}$, E_S is trained to predict both the correct frame-level musical features and content encoding. As for the alignment loss $\mathcal{L}_{\mathrm{a}}$, we give the model a tolerance of 0.25 to allow it to smoothly transfer from one phoneme to another at the boundary. λ_{p}, λ_{a}, λ_{e}, and λ_{t} are weighting factors, which are set to 1.0, 10.0, 1.0 and 0.1 respectively.

Weakly labeled mode. When the musical score label $\mathbf{s}$ is not available, similar to [18], we employ an acoustic encoder E_A to map the groundtruth log-Mel spectrogram to the content encoding space, which serves as an alternative to the musical score $\mathbf{s}$. We run the model as follows:

$$\mathbf{z_g}, \mathbf{z_l} = E_R(\mathbf{x_{r,g}}, \mathbf{x_{r,l}})\,; \tag{10}$$
$$\mathbf{z_c}, \mathbf{l}' = E_A(\mathbf{x})\,; \tag{11}$$
$$\mathbf{x_0}', \mathbf{x}' = D(\mathbf{z_c}, \mathbf{z_g}, \mathbf{z_l})\,, \tag{12}$$

where $\mathbf{l}'$ is the framewise phoneme prediction.

After obtaining $\mathbf{x_0}'$ and $\mathbf{x}'$, it is possible to directly apply the reconstruction loss on them. However, this does not guarantee that $\mathbf{z_c}$ only contains features related to musical content and does not contain any singer identity feature. Therefore, we propose to add a lyrics transcription objective to train the acoustic encoder. Suppose the lyrics label is $\mathbf{l_u}$, we use the following losses for model optimization:

$$\mathcal{L}_{\mathrm{r}} = 0.5 \times (\mathrm{L2}(\mathbf{x}', \mathbf{x}) + \mathrm{L2}(\mathbf{x_0}', \mathbf{x}))\,, \tag{13}$$
$$\mathcal{L}_{\mathrm{ALT}} = \mathrm{CTC}(\mathbf{l}', \mathbf{l_u})\,, \tag{14}$$
$$\mathcal{L}_{\mathrm{weak}} = \mathcal{L}_{\mathrm{r}} + \lambda_{\mathrm{ALT}}\mathcal{L}_{\mathrm{ALT}}\,, \tag{15}$$

where CTC denotes the CTC loss [25], which serves as the ALT objective. This encourages E_A to produce the correct linguistic feature and drops the singer identity features. The entire loss function $\mathcal{L}_{\mathrm{weak}}$ is the weighted sum of the reconstruction loss and the CTC loss. λ_{ALT} is the weighting factor, which is set to 1.0 in practice.

Inference. During inference, as shown in Figure 1(b), we drop E_A and perform zero-shot SVS from musical score by running Equation (1)–(3). Based on the problem definition, there is only one reference audio available. Therefore, we set both $\mathbf{x_{r,g}}$ and $\mathbf{x_{r,l}}$ to the same audio, which we denote as $\mathbf{x_r}$.

3.2. Implementation details

The architecture of the proposed model is shown in Figure 2. In this subsection, we briefly introduce the four main components of the model.

Acoustic encoder E_A. Figure 2 (a) shows the architecture of E_A. The Conv, $\mathrm{Strided\ conv}$, $\mathrm{Residual\ conv}$, Deconv blocks all consist of two 1-D convolution layers with kernel size of 5, with group normalization and ReLU activation function between them. The difference is that the second convolution layer of $\mathrm{Strided\ conv}$ has the stride size of 2, while

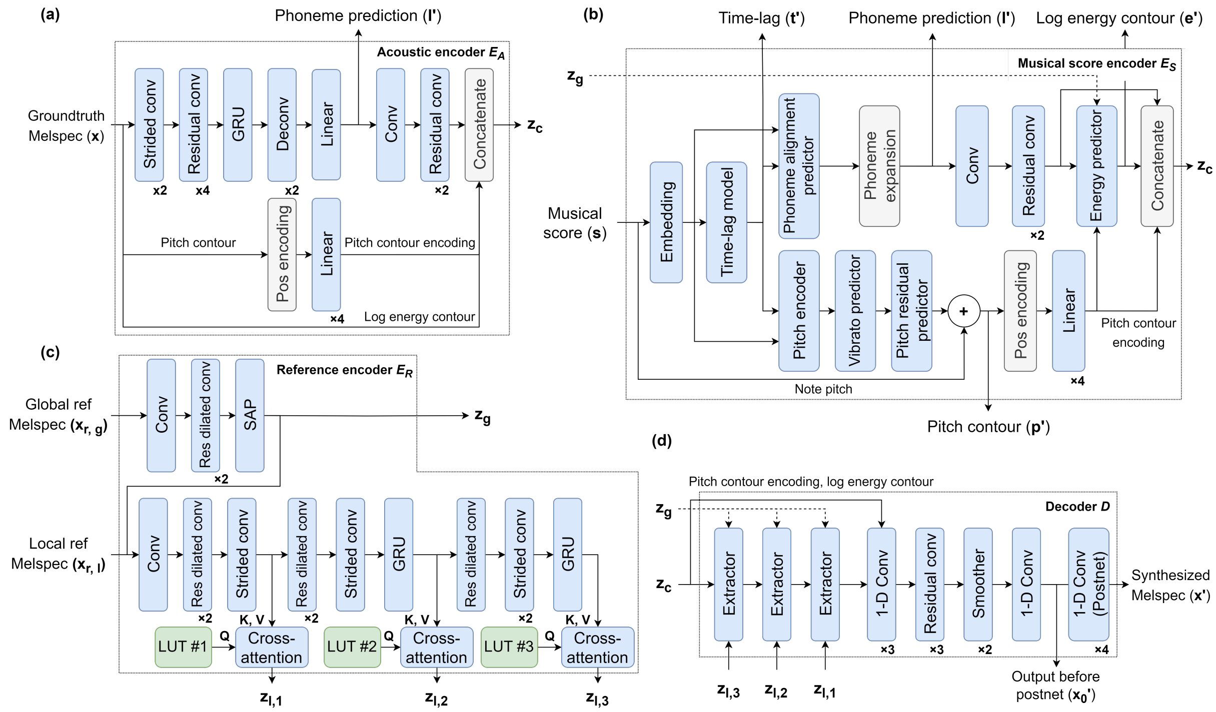

Fig. 2. The detailed architecture of the proposed model. "LUT" stands for a trainable lookup table which serves as queries (the green blocks); "SAP" stands for the self-attentive pooling; "Pos encoding" stands for the positional encoding. Blue blocks denote trainable building blocks of the model. Grey blocks denote operations without trainable parameters.

Deconv has the stride size of 0.5. The $\mathrm{Residual\ conv}$ block has a residual connection to the output.

First, $\mathbf{x}$ is passed through several convolution and GRU blocks. The output $\mathbf{l}'$ is viewed as the phoneme prediction. Then, we apply 1 Conv and 2 $\mathrm{Residual\ conv}$ blocks to encode $\mathbf{l}'$. Meanwhile, the pitch contour and log-energy contour are directly extracted from $\mathbf{x}$. Then, the encoding of these features are concatenated with the encoding of $\mathbf{l}'$ to form $\mathbf{z_c}$.

Musical score encoder E_S**.** Figure 2 (b) shows the architecture of E_S. The musical score $\mathbf{s}$ is first passed through an embedding layer. Then, similar to [11], we apply a time-lag model to predict the time-lag of each note, and a phoneme alignment predictor to predict the phoneme alignment. Each phoneme is modeled by a Gaussian distribution along the temporal axis, whose mean, variance and amplitude are predicted by the predictor. The phoneme alignment is then normalized and expanded to the phoneme prediction $\mathbf{l}'$ at frame level. Finally, we apply one Conv and two $\mathrm{Residual\ conv}$ blocks on $\mathbf{l}'$ to generate the linguistic features.

As for the pitch contour, first, a pitch encoder processes the embeddings and $\mathbf{z_g}$. Then, a vibrato predictor predicts the vibrato period and amplitude. Then, a pitch residual predictor predicts the pitch residual of each frame. We add the predicted pitch residual by the note pitch to form the pitch contour. The pitch contour is then encoded by a sinusoidal positional encoding and 4 linear layers.

Finally, we feed the linguistic features, pitch contour encoding and $\mathbf{z_g}$ to a log-energy predictor to obtain the log-energy contour, which is then concatenated with the above-mentioned two features to form $\mathbf{z_c}$.

Reference encoder E_R**.** Figure 2 (c) shows the architecture of E_R. First, $\mathbf{x_{r,g}}$ is passed through 1 Conv and 2 $\mathrm{Res\ dilated\ conv}$ blocks. A $\mathrm{Res\ dilated\ conv}$ contains 3 1-D convolution blocks with the dilation rate of 1, 2, and 4. Then, a self-attentive pooling (SAP) layer is applied to generate $\mathbf{z_g}$.

Then, $\mathbf{x_{r,l}}$ and $\mathbf{z_g}$ are concatenated and fed into a series of convolution and GRU layers to extract local features. Then, similar to [26], we use three trainable lookup tables as queries. Through a cross-attention module, we condense the local features to form a fix-dimension local singer encoding $\mathbf{z_l} := (\mathbf{z_{l,1}}, \mathbf{z_{l,2}}, \mathbf{z_{l,3}})$. We adopt such an architecture to reduce the computation cost and avoid model overfitting [26], especially when the duration of $\mathbf{x_{r,l}}$ is long.

Decoder D**.** Figure 2 (d) shows the architecture of D. The design of D is similar to FragmentVC's decoder [8]. Taking $\mathbf{z_c}$, $\mathbf{z_g}$ and $\mathbf{z_l}$ as the input, the extractor concatenates $\mathbf{z_c}$ and $\mathbf{z_g}$, and applies a linear layer and a self-attention layer on it to create the query. Then, it applies a cross-attention layer to extract the local singer features by setting the key and value to the corresponding $\mathbf{z_{l,i}}$, where $\mathbf{i} \in (1, 2, 3)$. These cross-attention layers encourage the decoder to obtain singer-related features from $\mathbf{z_l}$. Finally, we feed the output of the third extractor, pitch contour encoding and log-energy contour to several $\mathrm{Residual\ conv}$ blocks, 1-D convolution layers and Smoother [8] blocks to obtain the output log-Mel spectrogram. The last 4 1-D convolution layers are treated as the PostNet similar to Tacotron 2 [24].

4. EXPERIMENTS

4.1. Datasets

We test the proposed model on Mandarin SVS in a zero-shot manner. Three datasets are used for training and testing, including the MPOP600 dataset [19], the OpenSinger dataset [17], and the Musdb-V dataset [1,27]. The MPOP600 dataset contains 10 hours of audio sung by 4 distinct singers with musical score labels. We segment the audios based on the rest symbols in the musical score. The OpenSinger dataset contains 51.8 hours of audio sung by 76 distinct singers with unaligned lyrics labels. The Musdb-V dataset is the collection of the Musdb-18 dataset's vocal tracks [27] in which the silence parts were manually removed [1]. It contains 2.3 hours of audio sung by 86 singers without musical content labels.

In the experiments, we use the MPOP600 dataset as strongly labeled data, and the Opensinger dataset as weakly labeled data. For the MPOP600 dataset, we leave 5 songs of each singer as the test set. For the OpenSinger dataset, we select 3 male and 3 female singers and leave their data as the test set. The Musdb-V dataset is only used for testing.

4.2. Training and testing details

We set the dimension of all hidden layers to 256, and train the model with the AdamW optimizer with the initial learning rate of 10^{-5} and weight decay of 10^{-9}. Similar to [8], cosine annealing is used for learning rate scheduling, which decreases the learning rate to 2×10^{-6}. The model is trained for $1.2\mathrm{M}$ steps with a batch size of 2 (one strongly labeled data and one weakly labeled data). The total loss is the unweighted sum of $\mathcal{L}_{\text{strong}}$ and $\mathcal{L}_{\text{weak}}$.

As for feature extraction, all the audios are resampled to 24KHz and normalized. The number of Mel bands is set to 80. The frame rate of all acoustic features are set to 200Hz. The pitch contour is extracted by CREPE [9] [3].

To convert the log-Mel spectrogram back to waveform, we train a Parallel WaveGAN (PWG) vocoder [28] [4] from scratch on the OpenSinger training set for 400K steps with the batch size of 3. All the other hyper-parameters are remained the same as in the original paper.

During testing, we use the musical scores from the MPOP600 test set as the musical content condition, and use the Musdb-V test set and the OpenSinger test set as the reference audio. We concatenate all the clips with the same song name and singer name. The average duration of reference audios is 4.6 minutes for the OpenSinger test set, and 1.4 minutes for the Musdb-V test set. The source code of our experiments is available at `https://github.com/york135/zero_shot_svs_ASRU2023`.

4.3. Baseline and topline systems

To test the effectiveness of the proposed zero-shot SVS model, several systems are used for comparison, which are listed as follows:

Proposed. The proposed model trained with the proposed method discussed in Section 3 and 4.2.

Proposed (w/o ALT). Use the same model as *Proposed*, but does not use the ALT loss $\mathcal{L}_{\text{ALT}}$. This is similar to Bonada and Blaauw's work [18] which does not use lyrics labels for training. We regard this setting as a baseline.

Proposed - local. Similar to *Proposed*, but only use the speaker embedding $\mathbf{z_g}$. All the cross-attention layers in D are removed.

Wu et al. (topline). We use Wu et al.'s model [1] as a topline, which performs zero-shot SVS with a source audio that provides frame-level musical features.

4.4. Evaluation metrics

To evaluate the performance of zero-shot SVS models, two main aspects are considered, including 1) the singer identity similarity between the synthesized audio and the reference audio, and 2) the naturalness of the synthesized audio. In the remainder of this section, we directly refer to these two aspects as *similarity* and *naturalness*.

We conduct both subjective and objective tests. For the subjective test, in each question group, we provide the subjects with one reference audio and several synthesized audios generated by different systems. We ask them to rank the two aspects of the synthesized audios. Then, we convert the ranking data to the preference test results.

For the objective test, we train automatic speaker verification (ASV) models to quantify the similarity of the synthesized audio. We first use the ResNetSE34L pretrained model in [29] to extract fix-dimension features. Then, we train a 3-layer DNN with hidden dimension of 64 using the NT-Xent loss [30]. Then, we determine the threshold value that leads to the equal error rate (EER), and use it as the threshold to compute the speaker verification acceptance rate (SVAR) [8]. We report both the SVAR and the cosine similarity between the reference audio and the synthesized audio. In practice, we train one ASV model for each test dataset. The EER and threshold are 0.80% and 0.6514 for the OpenSinger test set model, 12.75% and 0.2957 for the Musdb-V test set model.

Besides the performance of zero-shot SVS, for models that use ALT loss, we also report the phoneme error rate (PER) of E_A's prediction on the OpenSinger test set. We use greedy search to decode the prediction.

[3] https://github.com/maxrmorrison/torchcrepe
[4] https://github.com/kan-bayashi/ParallelWaveGAN

Model	OpenSinger test		Musdb-V test	
	SVAR	Sim	SVAR	Sim
Proposed	86.7%	0.864	80.9%	0.553
Proposed (w/o ALT)	85.3%	0.836	68.5%	0.443
Proposed - local	43.2%	0.490	42.6%	0.232
Wu et al. (topline)	**95.5%**	**0.928**	**93.9%**	**0.753**

Table 2. Results of the objective similarity test. *Sim* denotes the average cosine similarity. *SVAR* denotes the speaker verification acceptance rate [8].

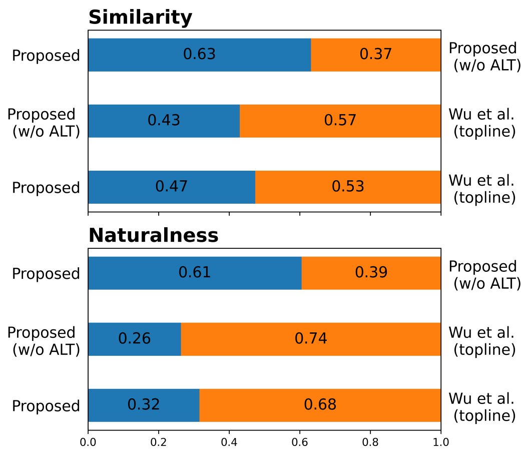

Fig. 3. Results of the subjective test.

4.5. Results

Objective results. Table 2 shows the results in terms of objective similarity. Among the proposed models, *Proposed* performs the best, suggesting that both the use of the ALT loss and the local singer encoding improve the similarity.

As for the comparison with the topline, Wu et al.'s model [1] still outperforms the proposed model, showing that the proposed model still has room of improvement in terms of similarity. However, based on the results, *Proposed* still makes the ASV model believe that the synthesized audio and reference audio are sung by the same singer in more than 80% of the cases on both datasets in the zero-shot setting, showing that the proposed model is capable of generating any target singers' voice with the guide of the reference audio.

As for the ALT performance, the PER of *Proposed* is 25.60%, while the PER of *Proposed - local* is 24.69%. Although the object similarity scores of the two models differ a lot, we do not observe clear difference between the ALT performance of their corresponding E_A.

Subjective results. Based on the objective results, we select *Proposed*, *Proposed (w/o ALT)* and *Wu et al. (topline)* for the subjective test. In this test, we only use the OpenSinger test set as reference audios. We recruited 19 subjects who are fluent in Mandarin for the test. Each subject is presented with 6 question groups. Figure 3 shows the subjective results. In terms of similarity, *Proposed* outperforms *Proposed (w/o ALT)* significantly ($p \approx 0.003$), showing that the use of ALT loss does help the model learn to imitate unseen singer's voice better. For the two other pairs, *Wu et al. (topline)* is slightly preferred over both *Proposed* and *Proposed (w/o ALT)*, but without statistical significance ($p \approx 0.32$ and 0.08 respectively), showing that the proposed model performs similarly to the topline in terms of similarity.

As for the naturalness, *Proposed* outperforms *Proposed (w/o ALT)* significantly ($p \approx 0.015$), showing that the use of ALT loss also improves the naturalness of the synthesized audio. We assume that with the use of the ALT loss, E_A learns to produce better $\mathbf{z_c}$, which further guides other parts of the model to learn to synthesize more natural voice. For the two other pairs, *Wu et al. (topline)* is preferred significantly over both *Proposed* ($p \approx 5 \times 10^{-5}$) and *Proposed (w/o ALT)* ($p \approx 2 \times 10^{-7}$), showing that the proposed model still has much room for improvement.

To examine the underlying reason that leads to such results, we run the *Proposed* model, but with the groundtruth frame-level musical features as the musical content, which is similar to previous work [1–4]. By listening to the results of the two settings, we found that the audio generated with the frame-level musical features has higher expressiveness in terms of pitch fluctuation and word pronunciation. This implies that 1) the proposed task in this work is more challenging than the zero-shot SVS task in previous work, and 2) in terms of bridging the information gap between musical score and frame-level musical features, our model, or more specifically, E_S, which is expected to produce the content encoding from musical score, does not achieve it perfectly. Unlike other components, E_S can only be trained using the data with musical score label, which may be the main reason that leads to such results. We believe that this reflects the limitation of this work, and that developing a method to train E_S better is crucial in future work.

5. CONCLUSION

In this paper, we propose the task of zero-shot SVS from musical score, which generates singing voice of unseen singers with only musical score as the musical content. By adding an acoustic encoder and the lyrics transcription loss, the proposed model learns from weakly labeled singing data with lyrics labels at utterance level efficiently. Experiment results suggest that the proposed model outperforms the baseline without the lyrics transcription loss. As a future work, we would like to work on training a better musical score encoder that generates the content encoding from the musical score, which is shown to be a crucial component that limits the overall performance of the proposed model.

6. REFERENCES

[1] Jui-Te Wu, Jun-You Wang, Jyh-Shing Roger Jang, and Li Su, "A unified model for zero-shot singing voice conversion and synthesis," in *Proceedings of the 23rd International Society for Music Information Retrieval Conference, ISMIR 2022*, 2022, pp. 809–816.

[2] Yi Ren, Xu Tan, Tao Qin, Jian Luan, Zhou Zhao, and Tie-Yan Liu, "Deepsinger: Singing voice synthesis with data mined from the web," in *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020, pp. 1979–1989.

[3] Shoutong Wang, Jinglin Liu, Yi Ren, Zhen Wang, Changliang Xu, and Zhou Zhao, "MR-SVS: Singing voice synthesis with multi-reference encoder," *arXiv preprint arXiv:2201.03864*, 2022.

[4] Kai Shen, Zeqian Ju, Xu Tan, Yanqing Liu, Yichong Leng, Lei He, Tao Qin, Sheng Zhao, and Jiang Bian, "Naturalspeech 2: Latent diffusion models are natural and zero-shot speech and singing synthesizers," *CoRR*, vol. abs/2304.09116, 2023.

[5] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu, "FastSpeech: Fast, robust and controllable text to speech," in *Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, 2019, pp. 3165–3174.

[6] 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," in *9th International Conference on Learning Representations, ICLR 2021*, 2021.

[7] Dongchan Min, Dong Bok Lee, Eunho Yang, and Sung Ju Hwang, "Meta-stylespeech: Multi-speaker adaptive text-to-speech generation," in *International Conference on Machine Learning*, 2021, pp. 7748–7759.

[8] Yist Y. Lin, Chung-Ming Chien, Jheng-Hao Lin, Hung-yi Lee, and Lin-Shan Lee, "FragmentVC: any-to-any voice conversion by end-to-end extracting and fusing fine-grained voice fragments with attention," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021*, 2021, pp. 5939–5943.

[9] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello, "Crepe: A convolutional representation for pitch estimation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018*, 2018, pp. 161–165.

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

[11] Yukiya Hono, Shumma Murata, Kazuhiro Nakamura, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda, "Recent development of the DNN-based singing voice synthesis system - Sinsy," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA ASC 2018*, 2018, pp. 1003–1009.

[12] Peiling Lu, Jie Wu, Jian Luan, Xu Tan, and Li Zhou, "XiaoiceSing: A high-quality and integrated singing voice synthesis system," in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association*, 2020, pp. 1306–1310.

[13] Soonbeom Choi and Juhan Nam, "A melody-unsupervision model for singing voice synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022*, 2022, pp. 7242–7246.

[14] Yongmao Zhang, Jian Cong, Heyang Xue, Lei Xie, Pengcheng Zhu, and Mengxiao Bi, "Visinger: Variational inference with adversarial learning for end-to-end singing voice synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022*, 2022, pp. 7237–7241.

[15] Pritish Chandna, Merlijn Blaauw, Jordi Bonada, and Emilia Gómez, "WGANSing: A multi-voice singing voice synthesizer based on the wasserstein-gan," in *27th European Signal Processing Conference, EUSIPCO 2019, A Coruña, Spain, September 2-6, 2019*, 2019, pp. 1–5.

[16] Juheon Lee, Hyeong-Seok Choi, Junghyun Koo, and Kyogu Lee, "Disentangling timbre and singing style with multi-singer singing synthesis system," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020*, 2020, pp. 7224–7228.

[17] Rongjie Huang, Feiyang Chen, Yi Ren, Jinglin Liu, Chenye Cui, and Zhou Zhao, "Multi-singer: Fast multi-singer singing voice vocoder with A large-scale corpus," in *MM '21: ACM Multimedia Conference*, 2021, pp. 3945–3954.

[18] Jordi Bonada and Merlijn Blaauw, "Semi-supervised learning for singing synthesis timbre," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 7083–7087.

[19] Chan-Chuan Chu, Fu-Rong Yang, Yi-Jhe Lee, Yi-Wen Liu, and Shan-Hung Wu, "Mpop600: A mandarin popular song database with aligned audio, lyrics, and musical scores for singing voice synthesis," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA 2020*, 2020, pp. 1647–1652.

[20] Merlijn Blaauw and Jordi Bonada, "A neural parametric singing synthesizer modeling timbre and expression from natural songs," *Applied Sciences*, vol. 7, no. 12, 2017.

[21] Jordi Bonada and Xavier Serra, "Synthesis of the singing voice by performance sampling and spectral models," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 67–79, 2007.

[22] June Sig Sung, Doo Hwa Hong, Shin Jae Kang, and Nam Soo Kim, "Factored MLLR adaptation for singing voice generation," in *INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association*, 2011, pp. 2789–2792.

[23] Yu Gu, Xiang Yin, Yonghui Rao, Yuan Wan, Benlai Tang, Yang Zhang, Jitong Chen, Yuxuan Wang, and Zejun Ma, "ByteSing: A chinese singing voice synthesis system using duration allocated encoder-decoder acoustic models and wavernn vocoders," in *12th International Symposium on Chinese Spoken Language Processing, ISCSLP 2021*, 2021, pp. 1–5.

[24] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, Rif A. Saurous, Yannis Agiomvrgiannakis, and Yonghui Wu, "Natural TTS synthesis by conditioning WaveNet on MEL spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4779–4783.

[25] Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006)*, 2006, vol. 148, pp. 369–376.

[26] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and João Carreira, "Perceiver: General perception with iterative attention," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, 2021, vol. 139, pp. 4651–4664.

[27] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner, "The MUSDB18 corpus for music separation," Dec. 2017.

[28] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim, "Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6199–6203.

[29] Joon Son Chung, Jaesung Huh, Seongkyu Mun, Minjae Lee, Hee-Soo Heo, Soyeon Choe, Chiheon Ham, Sunghwan Jung, Bong-Jin Lee, and Icksang Han, "In defence of metric learning for speaker recognition," in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association*, 2020, pp. 2977–2981.

[30] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, 2020, vol. 119, pp. 1597–1607.