EditSinger: Zero-Shot Text-Based Singing Voice Editing System with Diverse Prosody Modeling

Lichao Zhang[1] , **Zhou Zhao**[1*] , **Yi Ren**[1] and **Liqun Deng**[2]
[1]Zhejiang University
[2]Huawei Noah's Ark Lab
{zju_zlc, zhaozhou, rayeren}@zju.edu.cn, dengliqun.deng@huawei.com

Abstract

Zero-shot text-based singing editing enables singing voice modification based on the given edited lyrics without any additional data from the target singer. However, due to the different demands, challenges occur when applying existing speech editing methods to singing voice editing task, mainly including the lack of systematic consideration concerning prosody in insertion and deletion, as well as the trade-off between the naturalness of pronunciation and the preservation of prosody in replacement. In this paper we propose EditSinger, which is a novel singing voice editing model with specially designed diverse prosody modules to overcome the challenges above. Specifically, 1) a general masked variance adaptor is introduced for the comprehensive prosody modeling of the inserted lyrics and the transition of deletion boundary; and 2) we further design a fusion pitch predictor for replacement. By disentangling the reference pitch and fusing the predicted pronunciation, the edited pitch can be reconstructed, which could ensure a natural pronunciation while preserving the prosody of the original audio. In addition, to the best of our knowledge, it is the first zero-shot text-based singing voice editing system. Our experiments conducted on the OpenSinger prove that EditSinger can synthesize high-quality edited singing voices with natural prosody according to the corresponding operations.

1 Introduction

Recently, text-based speech editing system [Rubin *et al.*, 2013; Jin *et al.*, 2016; Jin *et al.*, 2017; Tang *et al.*, 2021; Morrison *et al.*, 2021; Tan *et al.*, 2021] has received extensive attentions in both research and industrial community. It allows users to edit the speech content just by performing the text-based editing operations (e.g., insertion, deletion and replacement), and thus significantly facilitates the reproducing of various speeches. Similarly, singing voice editing also makes great sense. As people are increasingly enthusiastic about creating videos on the Internet, user-friendly singing voice editing tools that allow them to modify or create some contents of a song record just based on the lyrics modification would be popularly used. However, to the best of our knowledge, this topic has been rarely exploited yet.

Previous speech editing systems [Jin *et al.*, 2016; Jin *et al.*, 2017; Morrison *et al.*, 2021] are based on unit-selection or retrieval of the audio clips. The main drawback of these methods is that a lot of target voices are required for retrieving, which not only imposes great restrictions on editing operations, but also complicates the generation steps. Recently, several works [Tan *et al.*, 2021; Tang *et al.*, 2021] tend to explore the zero-shot manner requiring no extra data from target speaker. The neural text to speech (FastSpeech 2 [Ren *et al.*, 2020a], etc.) is applied to synthesize the target edited speech directly by considering all the input information simultaneously, which avoids audio retrieving and processing. It faces challenges when porting to the singing voice synthesis (SVS) system to construct singing editing system, mainly due to the different demands of singing voice editing and speech editing. Therefore, we summarize the differences and the challenges to be solved in three operations of zero-shot text-based singing editing respectively as follows:

- **Insertion.** Generally, due to the flatter tone and less variability of speech compared with singing, the prosody (pitch etc.) between edited and non-edited region is rarely considered systematically [Tan *et al.*, 2021; Tang *et al.*, 2021] but without causing obvious hearing loss in the insertion task of speech editing. However, in the singing voice editing task, given the importance of the music note to the singing [Ren *et al.*, 2020b; Chen *et al.*, 2020], the construction of the prosody in the editing region should be considered comprehensively.
- **Replacement.** In speech editing, replacement and insertion are generally treated in the same way [Tan *et al.*, 2021]. But in the singing editing, the prosody in the editing region should also be preserved in accordance with the corresponding position of the original audio. An intuitive idea is to apply original music note directly to the lyrics in the editing region after alignment, but it is not effective in our experiments, mainly due to the errors of voiced and unvoiced, which greatly decreases the naturalness of the audio.

*Corresponding Author

- **Deletion.** Previous work [Tan *et al.*, 2021] of speech editing splices the remaining mel-spectrogram segments directly after deletion. Due to the large pitch fluctuations of singing voice, in our experiments, the pronunciation of boundaries is very likely to be discontinuous by direct splicing, resulting in a great hearing loss.

Given the differences and challenges above, directly applying previous speech editing methods to singing voice editing can not address the issues simultaneously. Therefore, in this article, we develop EditSinger, a zero-shot text-based singing voice editing system and we mainly focus on the challenges of prosody modeling mentioned above.

The pipeline of EditSinger integrates three editing operations above: the original audio and lyrics are first converted into the music score (phoneme, duration and pitch, etc.), while the operation on the lyrics is converted into the mask tokens and edited phoneme according to the type, region and content of the operation. These singing elements with the mask tokens are then fed into FastSpeech 2-based singing model with our specially designed diverse prosody modules for different singing editing operations introduced as follows to get the final edited audio:

- For the comprehensive prosody modeling of the inserted lyrics and the transition of deletion boundary, a general masked variance adaptor based on feed-forward Transformer blocks with mask tokens is introduced where the prosody of the editing region is generated in a context-aware manner and is fed directly into the acoustic model.
- To fix the errors of voiced/unvoiced (V/UV) in replacement, we design a fusion pitch predictor. The F0 sequence is disentangled from the original pitch, and then fused with the V/UV information obtained from the linguistic features of the new lyrics to get the pitch of the edited singing voice.

We conduct experiments on high-quality singing dataset OpenSinger [Huang *et al.*, 2021], which consists of 50 hours of Chinese singing voices recorded in a professional recording studio. Experiments prove that EditSinger can synthesize high-quality edited singing voices with natural prosody according to operating demands[1]. The contributions of this paper are summarized as follows:

- To the best of our knowledge, EditSinger is the first text-based singing editing system that can support a variety of operations. We dive into the challenges and carefully design for different operations with diverse prosody modules.
- The FastSpeech 2-based SVS model with our general masked variance adaptor for the comprehensive prosody modeling of inserted lyrics and the transition of deletion boundary can cooperate to synthesize a coherent and natural edited singing voice.
- By fusing the reference pitch and the predicted pronunciation information, the fusion pitch predictor is capable of ensuring a natural pronunciation while preserving the prosody of the original audio.
- EditSinger is a non-autoregressive, zero-shot system that does not require any additional singing data from the target singers, which can quickly perform singing editing with high quality and natural prosody.

[1]Audio samples can be listened in https://editsinger.github.io/.

2 Background

In this section, we briefly introduce the background of this work, including singing voice synthesis and text-based speech editing.

Singing voice synthesis. With the rapid development of text to speech (TTS) [Wang *et al.*, 2017; Li *et al.*, 2019; Ren *et al.*, 2020a], it has been applied to various scenarios and has been the basic technology in singing voice synthesis [Ren *et al.*, 2020b; Chen *et al.*, 2020; Lu *et al.*, 2020; Gu *et al.*, 2021]. However, SVS has distinct features compared with TTS. In addition to the given lyrics (text) , SVS has a high dependence on music note (note pitch and note duration), which is usually given directly to synthesize singing voices.

Text-based speech editing. The development of text-based speech editing can be roughly divided into two stages. In the first stage, the systems [Jin *et al.*, 2016; Jin *et al.*, 2017; Morrison *et al.*, 2021] leverage retrieved audio clips to perform speech editing tasks. The main drawback of these methods is that a lot of target voices are required for retrieving, which not only imposes great restrictions on editing operations, but also complicates the generation steps. Therefore, even if context awareness of prosody [Morrison *et al.*, 2021] is considered, they apply the generated prosody to correct the retrieved audio segments, making the steps complicated and no quality assurance when the data of target speaker is limited. In the second stage, several works [Tang *et al.*, 2021; Tan *et al.*, 2021] tend to explore the zero-shot technique. But they does not take prosody into consideration systematically and the settings of operations could not be suitable for singing editing. Reference duration is considered in zero-shot text-based speech insertion [Tang *et al.*, 2021], but modeling of pitch is ignored. In another work [Tan *et al.*, 2021], partial inference and bidirectional fusion are proposed to incorporate the contextual information. However, only a simple linear method is applied to modify the predicted duration in this work, what's more, they also ignore the modeling of the pitch, which results in a unnatural transition of the prosody between the editing region and the non-editing region. When dealing with the replacement task, they treat the replacement as a type of insertion [Tan *et al.*, 2021], ignoring the effect of the corresponding position of the replacement region, which is not feasible in singing voice replacement.

3 EditSinger

In this section, we introduce EditSinger, a zero-shot text-based singing editing system. We first define the tasks of the three operations, and then we describe the pipeline of EditSinger and briefly introduce each step. Then the architecture of singing model based on the FastSpeech 2 [Ren *et al.*, 2020a] will be introduced. After that, we introduce our masked variance adaptor and fusion pitch predictor in detail, and finally describe the training and inference process.

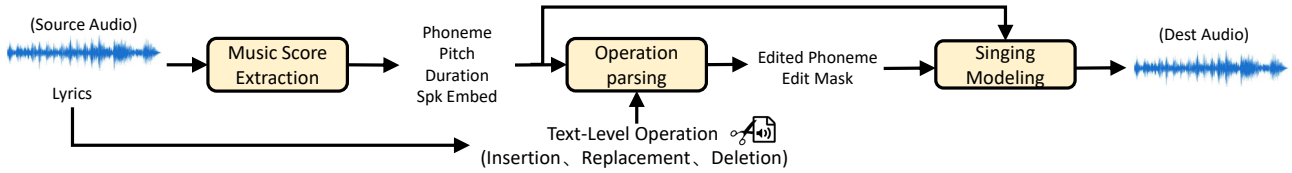


Figure 1: The pipeline of EditSinger.

3.1 Overview

In this section we will introduce the pipeline of EditSinger. Firstly, we define the tasks to be solved by text-based singing editing system:

- Singing insertion: Not only the prosody of the inserted words should be coherent and natural, but also the timbre is consistent with the original audio.
- Singing replacement: The prosody of the original audio should be preserved in the editing region, and be in natural coherence with the surrounding lyrics while the timbre remains unchanged.
- Singing deletion: The content after deletion should be cohered naturally.

To integrate these text-based singing editing operations into our system, as shown in Figure 1, EditSinger consists of several data processing, operation analysis, and modeling steps, including:

Music score extraction. The inputs of this step are a piece of singing voice and the lyrics corresponding to it. In order to facilitate the operation, we leverage tools to extract and process the various music elements required firstly, including phoneme, pitch, duration and speaker embedding, 1) Use the character analysis tool to convert the input lyrics to phoneme. Pypinyin[2] is used to convert Chinese lyrics to phoneme, besides it can also be replaced with other tools when dealing with lyrics in other languages. 2) Use montreal forced alignment (MFA) [McAuliffe *et al.*, 2017] to align original lyrics and audio, and extract the duration of each phoneme. 3) Use parselmouth[3] to extract the pitch information of the audio. 4) Use resemblyzer[4] to extract the speaker embedding.

Operation parsing. Three operations are supported in EditSinger. And each operation includes three parts: operation type, operation region and operation content. After the operation parsing step, the edited phoneme and the mask of the music score will be fed into the singing model to synthesize the edited audio together with the extracted score of the original audio.

Singing modeling. We design a FastSpeech 2-based singing model that takes edited phoneme, duration, pitch and edit mask information processed in the previous steps as inputs to generate the edited singing voices. The details of the singing model will be illustrated in Section 3.2, followed by our general masked variance adaptor and fusion pitch predictor designed for editing operations.

[2]https://github.com/mozillazg/python-pinyin

[3]https://github.com/YannickJadoul/Parselmouth

[4]https://github.com/resemble-ai/Resemblyzer

3.2 Singing Model

The overall model architecture of EditSinger is shown in Figure 2(a). Linguistic features are obtained through the phoneme encoder, and are fed into MVA-D to gain the duration information of the entire edited sentence. After being stretched by the length regulator, they are fed into the edit pitch predictor. Relying on the type of operation, the edit pitch predictor selects different pitch module to predict the pitch of the whole sentence, and enters the mel decoder together with the linguistic features to obtain the synthesized mel-spectrogram, which is fed to the vocoder together with the pitch information obtained by the edit pitch predictor to synthesize the final audio. In the following parts, we present our design of masked variance adaptor and fusion pitch predictor in detail.

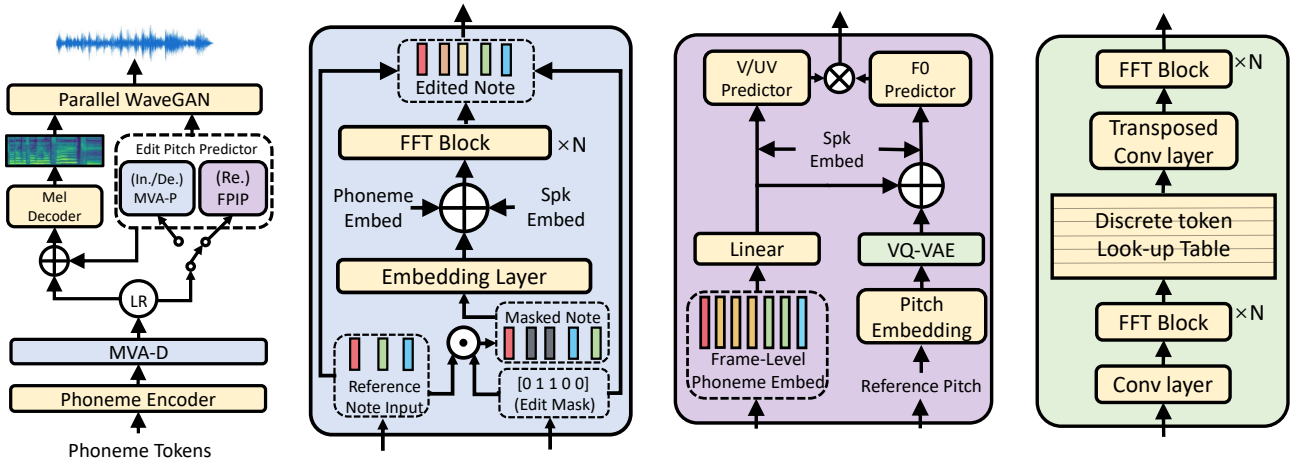
Masked variance adaptor. Different from the variance adaptor for TTS in FastSpeech 2 [Ren *et al.*, 2020a], which feeds phoneme hidden sequence, speaker embedding and positional encoding into variance adaptor, singing voice synthesis is not only related to linguistic features, music note information is also crucial for prosody modeling. Based on this, our variance model is designed as shown in Figure 2(b): The binary variance mask sequence $M = \{m_1, m_2, ..., m_n\}$, which denotes the editing/non-editing (one/zero) region and the reference note (i.e., the ground truth of prosody) are fed to the module. A shared, learned embedding variable $[Mask]$ is used to represent the mask region while the non-edit uses the extracted variance directly, and the masked reference variance sequence $V^* = \{v_1^*, v_2^*, ..., v_n^*\}$ is obtained after the mask process. Then the masked variance sequence is added to the phoneme hidden sequence and speaker embedding after passing through the embedding layer. Then after the interaction of feed-forward Transformer blocks, the predicted variance sequence V is obtained. And the final output V^o is formulated as:

$$v_i^o = \begin{cases} v_i, & \text{if } m_i = 1, \\ v_i^*, & \text{if } m_i = 0, \end{cases} \tag{1}$$

which means that the module is only used to predict the variance in the editing region, so as to preserve the prosody of the original audio as much as possible. Inspired by BERT [Devlin *et al.*, 2018], we calculate loss only on masked region, which is different from the previous work [Tang *et al.*, 2021] that calculates the loss of the entire sequence. The loss of masked variance adaptor (MVA) L_{MVA} can be written as:

$$L_{MVA} = \sum_{i=1}^{n} \left\| (v_i - v_i^*) * m_i \right\|. \tag{2}$$

It is worth noting that we leverage MVA in the modeling of duration and pitch, and it is a general module which can also



(a) EditSinger (b) Masked variance adaptor (MVA) (c) Fusion pitch predictor (FPIP) (d) VQ-VAE

Figure 2: The overall architecture of EditSinger. LR in subfigure (a) denotes the length regulator proposed in FastSpeech [Ren *et al.*, 2019]. FFT denotes the feed-forward Transformer. MVA-D and MVA-P denote MVA for the prediction of duration and pitch respectively. In./Re./De. denotes insertion/replacement/deletion, which means that MVA is used for pitch prediction in insertion and deletion, while the pitch is constructed with FPIP in replacement. $\odot$ in subfigure (b) denotes the mask operation performed on reference note according to edit mask, where the black blocks represent the masks while others represent acoustic variance. Phoneme Embed and Spk Embed in subfigure (b) and (c) denote the linguistic features and speaker embedding respectively. $\otimes$ in subfigure (c) denotes the frame-wise multiplication operation between F0 and V/UV.

be extended to the modeling of other acoustic variance.

Fusion pitch predictor. The replacement of singing voice is extremely different from speech as the edited speech only needs to keep the prosody natural and continuous while the edited singing should preserve the prosody of the original audio in the edited region additionally as we defined in Section 3.1. Intuitively, the prosody (duration, pitch, etc.) of the original audio can be directly transferred to the new lyrics. In our experiments, this approach will cause serious V/UV errors (i.e., migrating the original unvoiced regional prosody to the voiced phoneme will make the sense of hearing unnatural, which is presented in method analyses). Therefore, the module is required to combine the pitch of the original audio to predict the pitch of the new lyrics, so that the listening feel is natural and the original prosody is preserved as much as possible.

On the other hand, FastSpeech 2 [Ren *et al.*, 2020a] leverages the pitch predictor to predict the pitch (F0 and V/UV) of the phoneme based on the phoneme hidden sequence. In the singing scene, F0 has little relationship with phoneme (F0 can be specified manually), but V/UV is generally determined by the linguistic feature of phoneme. Therefore, our approach is to predict F0 and V/UV separately, and then fuse them together. As shown in Figure 2(c), the V/UV part takes linguistic features as input, and learns the pronunciation of the phoneme through ground truth. For the F0 part, we feed the reference F0 and leverage feed-forward Transformer blocks for interpolation in the UV region, which can also be seen as the disentanglement of pitch between F0 and V/UV, through which we obtain a sequence with only F0 information. Finally, the information of F0 and V/UV are fused by frame-wise multiplication operation, so that the pitch of edited lyrics can be reconstructed. In addition, we add VQ-VAE [van den Oord *et al.*, 2017] to the F0 part to help model the presentation of F0 and improve the stability of the modeling, as shown in Figure 2(d). The loss of fusion pitch predictor (FPIP) L_{FPIP} can be written as:

$$L_{FPIP} = \lambda_1 L_{F0} + \lambda_2 L_{commit} + \lambda_3 L_{V/UV}, \qquad (3)$$

where L_{F0} denotes the *reconstruction* loss of the F0 in the voiced region, $L_{V/UV}$ is the *binary cross entropy* loss of the V/UV prediction and the ground truth, and L_{commit} is the *commitment* loss encouraging the encoder to produce vectors lying close to the prototypes in the codebook. λ is a hyperparameter that weighs the importance of the three terms, which are all set to 1 in our experiments.

3.3 Training and Inference

The training and inference processes of the singing model are described as follows.

Training. Process the training data through the *music score extraction* step and transform the audio into mel-spectrograms.

1) We minimize the MAE and SSIM [Wang *et al.*, 2004] loss between the output mel-spectrograms and the ground truth mel-spectrograms to optimize the phoneme encoder and mel decoder.

2) We randomly mask out 15% of the words in the lyrics, utilizing the pitch and duration of the surrounding lyrics to predict the masked parts, and only calculate the loss in the mask region to optimize the masked variance adaptors.

3) We set a discrete token look-up table for VQ-VAE, and calculate the loss on the entire sentence of V/UV and F0 separately to optimize the fusion pitch predictor.

4) In addition, we leverage multi-length GAN (ML-GAN) [Chen *et al.*, 2020] on mel-spectrograms.

Inference. Process the target data similar to that used in the training. The edited phoneme and masked music score are obtained through the *operation parsing* step. We set different inference settings for the three operations:

- Insertion. The duration and pitch of the insertion region are set to mask, while non-editing region are directly fed ground truth.
- Replacement. Since the original and edited phoneme can be one-to-one correspondence, we use the ground truth duration directly. In addition, the edited phoneme hidden sequence and original pitch are fed into the fusion pitch predictor together to predict the pitch of the edited lyrics.
- Deletion. We mask the pitch of the phoneme closest to the deletion boundary and use the masked pitch predictor to predict the new pitch of the boundary phoneme.

4 Experiments and Results

4.1 Experimental Setup

Datasets. We conduct experiments on the singing datasets OpenSinger [Huang *et al.*, 2021] which consists of 50 hours of Chinese singing voices recorded in a professional recording studio and split OpenSinger randomly by singer into the test set (songs of 3 females and 3 males) and the training set (all the songs of the remaining singers). We pick up 10 sentences randomly from the songs of each test singer. Finally, a total of 60 sentences from these 6 test singers are collected for experimental evaluation.

Configuration Detail. We choose FastSpeech 2 [Ren *et al.*, 2020a] as the basic acoustic model and Parallel WaveGAN (PWG) [Yamamoto *et al.*, 2020] as the vocoder, both of which are non-autoregressive generation models. Our method can also be easily ported to other TTS and SVS models by doing so to make the system editable. We stack 4 feed-forward Transformer blocks in both the encoder and decoder of the acoustic model and set the hidden size to 256, and the same configuration is also used in the MVA and F0 predictor of FPIP. The V/UV predictor consists of a 4-layer 1D-convolutional network.

4.2 Results

In this subsection, we evaluate the performance of EditSinger on voice quality, prosody and similarity. In addition, we have also transplanted some speech editing methods for singing editing as a comparison, which can be seen in quantitative evaluation of this section and Section 4.3 in detail.

Qualitative evaluation. To evaluate the perceptual audio performance comprehensively, we conduct the audio quality (MOS-Q), prosody (MOS-P) and similarity (MOS-S) evaluation on the test set. Each singing sample is listened by at least ten qualified testers. We compare three indicators of the synthesized audio samples among the following systems: 1) *GT*, the ground truth singing audio; 2) *GT (Mel+PWG)*, where we first convert the ground truth singing audio to the ground truth mel-spectrograms, and then convert these mel-spectrograms back to audio using PWG vocoder; 3-5) *EditSinger (In./Re./De.)*, where we modify (i.e. insert, replace or delete) several words on the original lyrics and then synthesize the edited audio by EditSinger. The results are shown in Table 1. It can be seen that EditSinger synthesizes high-quality singing voices with only 0.15, 0.21 and 0.27 mean MOS gap of three operations to the *GT (Mel+PWG)* upper bound on the three subjective metrics respectively.

Method	MOS-Q	MOS-P	MOS-S
GT	4.32 ± 0.05	4.35 ± 0.05	-
GT (Mel+PWG)	4.14 ± 0.05	4.23 ± 0.05	4.17 ± 0.05
EditSinger (In.)	3.94 ± 0.06	3.97 ± 0.06	3.85 ± 0.08
EditSinger (Re.)	4.02 ± 0.06	4.15 ± 0.05	4.06 ± 0.07
EditSinger (De.)	4.01 ± 0.06	3.92 ± 0.06	3.79 ± 0.08

Table 1: MOS of three aspects with 95% confidence intervals. MOS-Q indicates the quality of the audio. MOS-P indicates the coherence and naturalness of prosody. MOS-S is specially designed to evaluate the correlation between the edited prosody and the original prosody without considering the influence of lyrics. These indicators are all scored on a scale of 1 to 5, with higher scores indicating higher quality of the referred explanation.

Method	*w/ Duration Ref*	*w/o Duration Ref*
MAE-D (Frame)	**9.78**	10.46
Method	*w/ Pitch Ref*	*w/o Pitch Ref*
MAE-P (Hz)	**7.83**	9.97

Table 2: MAE-D and MAE-P of different settings.

Quantitative evaluation. To explore prosody modeling effect of EditSinger objectively, we first extracted the phoneme-level duration and frame-level pitch sequence of the original audio as ground truth as introduced in Section 3.1. Then we randomly mask 15% words of the original lyrics and replace them with the phoneme "d ang". The prosody modules are required to reconstruct the duration and pitch of the mask region. We calculate the indicators *MAE-D* and *MAE-P* for evaluation, which represent the mean absolute error (MAE) of phoneme-level duration and frame-level pitch respectively. In terms of models, we evaluate our masked variance adaptor mentioned in Section 3.2, which is denoted as *w/ Duration/Pitch Ref*. In addition, we also calculate the simple linear correction method which is mentioned in EditSpeech [Tan *et al.*, 2021] as the baseline comparison, which is denoted as *w/o Duration/Pitch Ref*. Since only duration is dealt with in their work, we make a pitch version according to their principle, which first takes the edited phoneme to pitch predictor, and then calculates the ratio of the ground truth value to the predicted value in the non-edited region, and then this ratio is multiplied by the predicted pitch in the edited region to get final output. As shown in Table 2, obviously, our module achieves a smaller MAE both on duration and pitch prediction in the edited region, which denotes that

Method	CMOS-Q	CMOS-P
EditSinger (In./De.)	**0**	**0**
w/o MVA (In.)	-0.09	-0.33
w/o Pitch Ref	-0.06	-0.25
w/o Duration Ref	-0.05	-0.07
w/o ML-GAN (In.)	-0.31	-0.05
w/o MVA (De.)	-0.12	-0.31
w/o ML-GAN (De.)	-0.35	-0.08

Table 3: Ablation experiments on insertion and deletion operations. CMOS-P and CMOS-Q are with the scale of -2 to 2.

it is better than the previous work in terms of prosody modeling. In addition, it can also be seen that MVA demonstrates better performance in pitch modeling relatively.

4.3 Method Analyses

In this section, we conduct additional experimental studies to analyze some specific designs in EditSinger for three operations. Since there are some similarities between singing insertion and deletion in our approach, we combine them with one section for analysis, while the replacement in another section. The introduction of the analyses are as follows:

Insertion/Deletion. In this section we analyze the role of several designs in insertion and deletion operations. Our design here includes masked variance adaptor and ML-GAN, which are mainly used to solve the challenges of comprehensive prosody modeling and sound quality in edited region respectively. Therefore, we conduct the comparative audio quality (CMOS-Q) and comparative prosody (CMOS-P) evaluation on the test set. We apply the setting *EditSinger (In./De.)* introduced in Section 4.2 as a benchmark and design the following experiments:

- Insertion: 1) *w/o MVA (In.)*, which removes the masked variance adaptor on both duration and pitch in the insertion operation. We replace it with the variance adaptor introduced in FastSpeech 2, and use the simple linear correction method introduced in EditSpeech [Tan *et al.*, 2021]. Besides, to explore the influence of the two reference inputs on the prosody respectively, we subdivide them into i) *w/o Pitch Ref*, which only removes the reference input of pitch and ii) *w/o Duration Ref*, which only removes the reference input of duration. 2) *w/o ML-GAN (In.)*, which removes ML-GAN in the insertion operation.
- Deletion: 1) *w/o MVA (De.)*, which removes the masked variance adaptor on pitch in the deletion operation, and utilizes the method mentioned in EditSpeech [Tan *et al.*, 2021] to splice the remaining audio clips directly after deletion. 2) *w/o ML-GAN (De.)*, which removes ML-GAN.

The results are shown in Table 3. Analyzing the results we could find that 1) the masked modeling of pitch and duration in singing voice editing are crucial to the overall prosody of the singing, therefore, removing any of them will make the CMOS-P drop; 2) Just a simple splicing operation will lead to the artificialness of remaining audio fragment, resulting in a decrease in the sense of hearing. While our MVA can solve this problem well when applied in deletion operation; 3) ML-GAN can improve the sound quality of the edited audio effectively.

Method	MOS-Q	MOS-P	MOS-S
Direct	3.27 ± 0.09	3.36 ± 0.09	3.71 ± 0.08
w/o FPIP	3.62 ± 0.08	2.61 ± 0.09	1.92 ± 0.09
w/o VQ-VAE	3.03 ± 0.09	3.29 ± 0.08	3.62 ± 0.08
w/o ML-GAN	3.32 ± 0.08	3.71 ± 0.08	3.73 ± 0.07
EditSinger (Re.)	$\mathbf{3.77 \pm 0.07}$	$\mathbf{3.78 \pm 0.08}$	$\mathbf{3.82 \pm 0.08}$

Table 4: Ablation experiments on replacement operation. The explanation of each indicator is the same as Table 1.

Replacement. To analyze our design in the replacement operation, we introduce "Dang Test", which means replacing the phoneme of each word in the original lyrics with "d ang" (also can be replaced with other phoneme) to test the effect of various singing replacement settings. The three indicators introduced in Section 4.2 are also leveraged here to comprehensively evaluate the performance of the replacement models in terms of sound quality, prosody and similarity. We mainly compare the following models: 1) *EditSinger (Re)*, the replacement operation of EditSinger; 2) *Direct*, which places the pitch of the original audio on the new lyrics directly; 3) *w/o FPIP*, which removes the fusion pitch predictor and uses the method introduced in EditSpeech [Tan *et al.*, 2021]; 4) *w/o VQ-VAE*, which removes the VQ-VAE in FPIP; 5) *w/o ML-GAN*, which removes ML-GAN.

The results are shown in Table 4. The results indicate that 1) directly migrating the pitch of the original audio to the new lyrics will cause serious V/UV errors, resulting in a great drop in the sound quality and prosody of the synthesized singing; 2) the fusion pitch predictor which combines linguistic information of the edited lyrics and the pitch information of the reference audio to generate the pitch of the new lyrics has a great performance in the three indicators of sound quality, prosody and similarity; 3) VQ-VAE is of great help to the prosody stability of the generated singing. Removing VQ-VAE could cause a serious drop in MOS-Q and MOS-P; 4) ML-GAN is helpful for the sound quality of the replacement.

5 Conclusions

In this paper, we proposed EditSinger, a zero-shot text-based singing editing system that supports various text-based singing editing operations. We dive into the challenges of singing editing and design diverse modules mainly for prosody modeling. A general masked variance adaptor is introduced for comprehensive prosody modeling to ensure the naturalness and coherence of the generated audio in zero-shot singing editing. And we also design a fusion pitch predictor so that the prosody of the original audio can be preserved and maintain a natural pronunciation of the generated audio. Experiments conducted on OpenSinger demonstrate that EditSinger can nearly match the upper bound on three indicators of quality, prosody and similarity. For future work, we will continually explore more diverse and effective prosody modeling methods on singing editing.

Acknowledgement

This work was supported in part by the National Key RD Program of China under Grant No.2020YFC0832505, National Natural Science Foundation of China under Grant No.61836002, No.62072397, Zhejiang Natural Science Foundation under Grant LR19F020006.

References

[Chen *et al.*, 2020] Jiawei Chen, Xu Tan, Jian Luan, Tao Qin, and Tie-Yan Liu. Hifisinger: Towards high-fidelity neural singing voice synthesis. *arXiv preprint arXiv:2009.01776*, 2020.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Gu *et al.*, 2021] 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 *2021 12th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pages 1–5. IEEE, 2021.

[Huang *et al.*, 2021] 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 *Proceedings of the 29th ACM International Conference on Multimedia*, pages 3945–3954, 2021.

[Jin *et al.*, 2016] Zeyu Jin, Adam Finkelstein, Stephen DiVerdi, Jingwan Lu, and Gautham J Mysore. Cute: A concatenative method for voice conversion using exemplar-based unit selection. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5660–5664. IEEE, 2016.

[Jin *et al.*, 2017] Zeyu Jin, Gautham J Mysore, Stephen Diverdi, Jingwan Lu, and Adam Finkelstein. Voco: Text-based insertion and replacement in audio narration. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017.

[Li *et al.*, 2019] Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu. Neural speech synthesis with transformer network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6706–6713, 2019.

[Lu *et al.*, 2020] Peiling Lu, Jie Wu, Jian Luan, Xu Tan, and Li Zhou. Xiaoicesing: A high-quality and integrated singing voice synthesis system. *arXiv preprint arXiv:2006.06261*, 2020.

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

[Morrison *et al.*, 2021] Max Morrison, Lucas Rencker, Zeyu Jin, Nicholas J Bryan, Juan-Pablo Caceres, and Bryan Pardo. Context-aware prosody correction for text-based speech editing. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7038–7042. IEEE, 2021.

[Ren *et al.*, 2019] 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 *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 3171–3180, 2019.

[Ren *et al.*, 2020a] 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.

[Ren *et al.*, 2020b] 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 *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1979–1989, 2020.

[Rubin *et al.*, 2013] Steve Rubin, Floraine Berthouzoz, Gautham J Mysore, Wilmot Li, and Maneesh Agrawala. Content-based tools for editing audio stories. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 113–122, 2013.

[Tan *et al.*, 2021] Daxin Tan, Liqun Deng, Yu Ting Yeung, Xin Jiang, Xiao Chen, and Tan Lee. Editspeech: A text based speech editing system using partial inference and bidirectional fusion. *arXiv preprint arXiv:2107.01554*, 2021.

[Tang *et al.*, 2021] 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.

[van den Oord *et al.*, 2017] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6309–6318, 2017.

[Wang *et al.*, 2004] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[Wang *et al.*, 2017] 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*, 2017.

[Yamamoto *et al.*, 2020] 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)*, pages 6199–6203. IEEE, 2020.