Robust Singing Voice Transcription Serves Synthesis

Anonymous ACL submission

Abstract

Note-level Automatic Singing Voice Transcription (AST) converts singing recordings into note sequences, facilitating the automatic annotation of singing datasets for Singing Voice Synthesis (SVS) applications. Current AST methods, however, struggle with accuracy and robustness when used for practical annotation. This paper presents ROSVOT, the first robust AST model that serves SVS, incorporating a multi-scale framework that effectively captures coarse-grained note information and ensures fine-grained frame-level segmentation, coupled with an attention-based pitch decoder for reliable pitch prediction. We also established a comprehensive annotation-and-training pipeline for SVS to test the model in real-world settings. Experimental findings reveal that ROSVOT achieves state-of-the-art transcription accuracy with either clean or noisy inputs. Moreover, when trained on enlarged, automatically annotated datasets, the SVS model outperforms its baseline, affirming the capability for practical application. Audio samples are available at https://rosvot.github.io.

1 Introduction

Note-level automatic singing voice transcription (AST) refers to converting a singing voice recording into a sequence of note events, including note pitches, onsets, and offsets (Mauch et al., 2015; Hsu et al., 2021; Wang et al., 2022a; Yong et al., 2023). As part of the music information retrieval (MIR) task, AST is widely used in professional music production and post-production tuning. With the recent advancements of singing voice synthesis (SVS) (Liu et al., 2022; Zhang et al., 2022b; He et al., 2023), there is a growing demand for annotated data, while AST methods just demonstrate the potential for automatic annotation.

Note transcription from singing voices is particularly difficult than from musical instruments, as the pitch component of human voices is highly dynamic. When singing, people articulate words, leading to unstable pitches and blurry note boundaries. For instance, if a word starts with a voiceless consonant, the pitch onset may be slightly delayed. Also, singing techniques like vibrato and appoggiatura further complicate boundary localization.

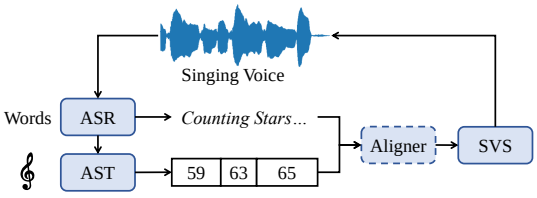

Figure 1: AST and ASR systems serve SVS.

An AST task is mainly decomposed into two steps: note segmentation and pitch estimation. The first step predicts boundaries, or onset and offset of each note, which is always implemented as classification (Hsu et al., 2021; Yong et al., 2023) or object detection (Wang et al., 2022a) tasks. For pitch estimation, previous works primarily adopt weighted median or average operations on F0 values.

Despite previous accomplishments, there is no AST model that, to our knowledge, achieves a complete annotation pipeline for training an SVS model. Applying AST approaches to automated annotation for SVS tasks still faces several challenges:

- **Insufficient accuracy.** Despite numerous efforts to improve accuracy, the performance is still insufficient for automatic annotation. Currently, AST results serve merely as a preliminary guide, necessitating additional manual refinement for actual application (Zhang et al., 2022a).
- **Asynchronization between notes and texts.** SVS models often require text-note synchronized annotation. Currently, transcribing singing voices without the supervision of word/phoneme boundaries requires additional post-processing for alignment, introducing accumulative errors.
- **Inadequate robustness.** Web crawling has become a popular method for data collection (Ren

et al., 2020), but the quality varies. Current AST methods are vulnerable to noise because sound artifacts tend to disrupt boundary localization and pitch perception.

In this paper, we present ROSVOT, a **RO**bust **S**inging **VO**ice **T**ranscription model that ultimately serves SVS. The note boundary prediction is formulated as one-dimensional semantic segmentation, and an attention-based decoder is employed for pitch prediction. To achieve both coarse-grained semantic modeling and fine-grained frame-level segmentation, we devise a multi-scale architecture by integrating Conformer (Gulati et al., 2020) and U-Net (Ronneberger et al., 2015). Moreover, the model incorporates word boundaries to guide the segmentation process. We randomly mix the input waveforms with MUSAN (Snyder et al., 2015) noise to simulate a noisy environment, forming a bottleneck and bolstering denoising capabilities.

To demonstrate the potential of ROSVOT in practical annotation applications, we conduct extensive experiments on a comprehensive annotation-and-training pipeline on an SVS task, simulating real-world scenarios. We choose and slightly modify RMSSinger (He et al., 2023), one of the state-of-the-art SVS models, to be the singing acoustic model. Experiments show that the SVS model trained with pure transcribed annotations achieves 91% of the pitch accuracy compared to manually annotated data, without loss of overall quality. We also explore the generalization performance on cross-lingual tasks, where we use ROSVOT trained with Mandarin corpora to annotate an English corpus, which is then used to train an SVS model. Our contributions are summarized as follows:

- We propose ROSVOT, the first robust AST model that serves SVS, which achieves state-of-the-art transcription accuracy under either clean or noisy environments.
- We construct a comprehensive annotation-and-training pipeline to investigate the effect of automatically transcribed annotations on SVS tasks.
- The proposed multi-scale model outperforms the previous best published method by 17% relative improvement on pitch transcription, and by 23% with noisy inputs.
- By incorporating automatically annotated large-scale datasets, the SVS model outperforms its baseline in terms of overall quality, demonstrating the ROSVOT's capability of practical application and the opportunity to alleviate data scarcity in SVS.
- We explore the cross-lingual generalization capabilities of ROSVOT.

2 Related Works

2.1 Automatic Singing Voice Transcription

AST is useful not only in automatic music transcription (AMT) (Bhattarai and Lee, 2023), but also a promising task for audio language models (Yang et al., 2023) and speech-singing interaction modeling (Li et al., 2023). TONY (Mauch et al., 2015) predicts note events by applying hidden Markov models (HMM) on extracted pitch contours. VOCANO (Fu and Su, 2019; Hsu et al., 2021) considers the note boundary prediction as a hierarchical classification task. and leverages a hand-crafted signal representation for feature engineering. MusicYOLO (Wang et al., 2022a) adopts object detection methods from image processing to localize the onset and offset positions. Despite their success, the overall accuracy is still insufficient for practical application. Considering the linguistic characteristic of singing voices, Yong et al. (2023) introduces extra phonetic posteriorgram (PPG) information to improve accuracy. However, a PPG extractor requires an extra training process and makes the AST model difficult to generalize across languages.

2.2 Singing Voice Synthesis

Recently, there has been notable progress in the field of SVS. HifiSinger (Chen et al., 2020) and WeSinger (Zhang et al., 2022c) employ GAN-based networks for high-quality synthesis. (Liu et al., 2022) introduces a shallow diffusion mechanism to address over-smoothness issues in the general Text-to-Speech (TTS) field. Taking inspiration from VITS (Kim et al., 2021), VISinger (Zhang et al., 2022b) constructs an end-to-end architecture. To achieve singer generalization, NaturalSpeech 2 (Shen et al., 2023) and StyleSinger (Zhang et al., 2023) utilize a reference voice clip for timbre and style extraction. To bridge the gap between realistic musical scores and MIDI annotations, RMSSinger (He et al., 2023) proposes word-level modeling with a diffusion-based pitch modeling approach. Open-source singing voice corpora also boost the development of SVS (Huang et al., 2021; Zhang et al., 2022a; Wang et al., 2022b). However, the quantity of annotated singing voice corpora is still small compared to speech, while note annotations of some corpora are even unavailable.

3 Method

3.1 Problem Formulation

In the note segmentation step, the model predicts onset/offset states at each timestep t, where $t \in [1, T]$ and T is the temporal length of the spectrogram. Without loss of generality, we introduce silence notes to connect each note in the entire sequence end-to-end, replacing the onset/offset tuples by a single note boundary notation sequence $\boldsymbol{y}_{\text{bd}} = [y^1_{\text{bd}}, y^2_{\text{bd}}, ..., y^T_{\text{bd}}]$, where $y^t_{\text{bd}} = 1$ if the state is boundary at timestep t and 0 is not. The silence note has a pitch value of 0. Notice that $\sum \boldsymbol{y}_{\text{bd}} = \text{len}(\boldsymbol{p}) - 1$, where $\boldsymbol{p} = [p^1, p^2, ..., p^L]$ is the pitch value sequence, L is the total number of notes, and $\text{len}(\cdot)$ computes lengths of sequences. Therefore, the first step can be treated as semantic segmentation, predicting a binary-label sequence. The second step is to predict the pitch sequence $\boldsymbol{p}$.

3.2 Overview

As shown in Figure 1, a common data collection pipeline for SVS consists of two stages: a) phoneme/word annotation and b) note annotation, where the former can be achieved by utilizing automatic speech recognition (ASR) approaches and forced alignment tools, such as MFA (McAuliffe et al., 2017). The second stage, however, is far from reaching a fully automatic level. Arduous manual annotation hinders large-scale data collection. A high-precision and robust annotator is required.

Note segmentation is a multi-scale classification task, in that the note events are coarse-grained while the predicted boundary sequence $\boldsymbol{y}_{\text{bd}}$ is fine-grained. Therefore, we construct a multi-scale model, combining a U-Net backbone and a downsampled Conformer, as illustrated in Figure 2. The model takes Mel-spectrograms, F0 contours, and word boundaries as inputs. To improve robustness, we train the model under noisy environments and apply various data augmentation operations. For pitch prediction, we adopt an attention-based method to obtain dynamic temporal weights and perform weighted averages. The note segmentation part and the pitch prediction part are trained jointly to acquire optimal results.

3.3 Data Augmentation

3.3.1 Label Smoothing

The exact temporal positions of note onset and offset are difficult to demarcate on a microscopic scale, because transitions between notes are continuous and smooth. Therefore, label smoothing is a popular strategy in AST tasks (Hsu et al., 2021; Yong et al., 2023). Also, soft labels carry more information than hard labels, such as the desired confidence for the model. Specifically, we apply temporal convolution operation between the label sequence $\boldsymbol{y}_{\text{bd}}$ and a Gaussian filter $\mathcal{G}[n]$:

$$\mathcal{G}[n] = \begin{cases} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\tau^2}{2\sigma^2}}, & \text{if } |n| \leq \lfloor \frac{W_{\mathcal{G}}}{2} \rfloor \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

$$\widetilde{\boldsymbol{y}}_{\text{bd}} = \boldsymbol{y}_{\text{bd}} * \left(\frac{\mathcal{G}[n]}{\max(\mathcal{G}[n])} \right) \quad (2)$$

where the filter $\mathcal{G}[\tau]$ is normalized before convolution, so the middle of each soft label remains 1. $W_{\mathcal{G}}$ indicates the window length of the filter.

3.3.2 Noise

We mix realistic noise signals with waveforms before extracting spectrograms. MUSAN noise corpus is utilized to randomly incorporate the interference. MUSAN corpus consists of a variety of noises, such as babble, music, noise, and speech. The intensity of incorporated noise is randomly adjusted according to a signal-to-noise ratio (SNR) interval of $[6, 20]$. The noise signal is repeated or chunked to meet the length of each training sample. In the training stage, we conduct noise mixing followed by on-the-fly extraction of Mel-spectrograms:

$$\widetilde{\boldsymbol{y}} = \boldsymbol{y} + \boldsymbol{y}_{\text{noise}} \times \frac{\text{RMS}(\boldsymbol{y}/10^{(\text{SNR}/20)})}{\text{RMS}(\boldsymbol{y}_{\text{noise}})} \quad (3)$$

$$\widetilde{X} = \mathcal{F}(\widetilde{\boldsymbol{y}}) \quad (4)$$

where $\mathcal{F}(\cdot)$ is Mel-spectrogram extraction operation, $\text{RMS}(\cdot)$ is root-mean-square operation, and $\widetilde{X}$ is the resulting spectrogram.

In addition to spectrograms, we also add noise to F0 contours and label sequences. Since the model takes F0 contours as input, a clean F0 contour can leak information. We simply add Gaussian noise to logarithmic F0 contours and soft labels to improve robustness.

3.4 Word Boundary Condition

To regulate segmentation results and better suit practical annotation, we incorporate word boundary conditions. The word boundary sequence $\boldsymbol{y}_{\text{wbd}}$ has the same form as note boundaries $\boldsymbol{y}_{\text{bd}}$, involving silence or "NONE" words. The regulation is

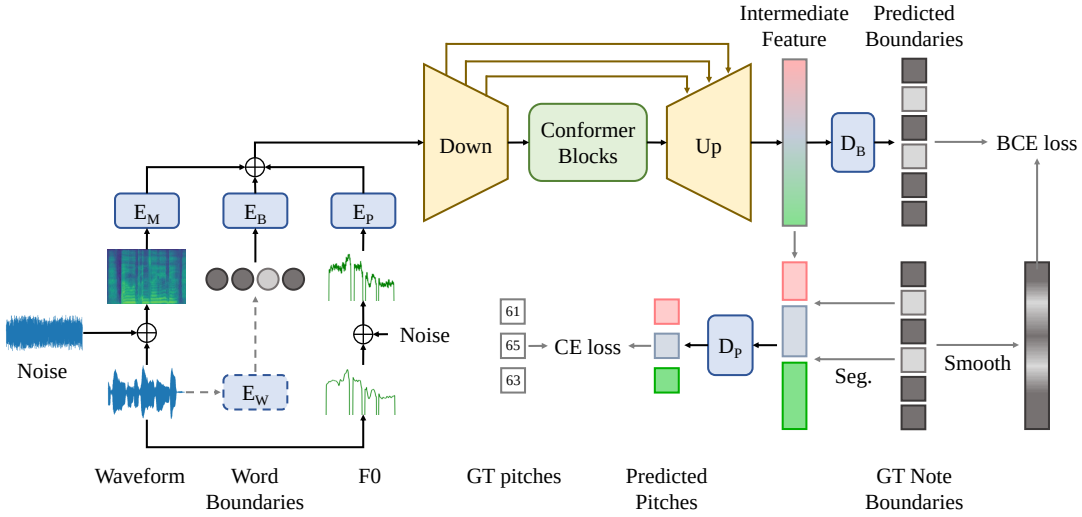

Figure 2: The overall architecture. E_M, E_B, and E_P represent encoders of Mel-spectrogram, word boundaries, and F0 contour input. D_B and D_P stand for decoders of note boundaries and pitches. The "Down" and "Up" parts denote the encoder and decoder of the U-Net backbone. The "Seg." and "Smooth" notations indicate temporal segmentation and label smoothing operations. E_W indicates an optional extractor used to provide word boundaries.

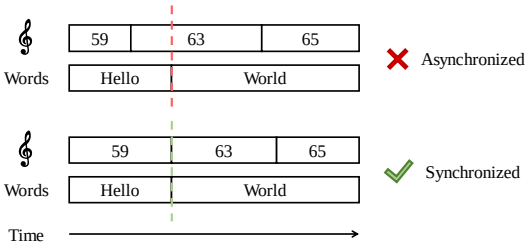

Figure 3: Word-note synchronization.

necessary because, in practical annotation, word sequence and note sequence need to be temporally synchronized, as shown in Figure 3. In other words, the presence of a word boundary at timestep t implies the existence of a note boundary at t, but the reverse may not hold true. This is because melisma is a commonly used singing technique. Without regulation, additional post-processing is required to synchronize words and note sequences.

Since in practice, the note annotation stage follows the phoneme annotation stage, word boundaries should already be obtained through forced alignment tools like MFA. We directly encode word boundaries as an additional condition to ensure word-note synchronization. However, to provide note-only support, we train an extra word boundary extractor E_W to deal with scenarios like vocal tuning in music industries, where word alignment is unavailable. More details are listed in Appendix A.

3.5 Multi-scale Architecture

The semantic information of note events is coarse-grained and high-level, while the segmentation result $\boldsymbol{y}_{\text{bd}}$ is fine-grained and frame-level. To tackle this problem, we design a multi-scale model, incorporating multiple feature encoders and a pitch decoder, illustrated in Figure 2.

For precise segmentation, high-resolution results are essential to prevent rounding errors. Hence, we employ a U-Net architecture for its ability to downsample representations while ensuring detailed reconstruction. To capture the high-level features associated with note events, we utilize a Conformer network, one of the most popular ASR models. The U-Net architecture envelops the Conformer, directing its focus towards the downsampled features and easing the computational load of processing long sequences. Through the integration of skip connections, our model achieves refined frame-level accuracy by fusing features across multiple scales.

The U-Net backbone's encoder and decoder each comprise K downsampling and upsampling layers, respectively. The downsampling rate is set to 2, and the channel dimension remains the same as input to alleviate overfitting. The intermediate part of the backbone is replaced by a 2-layer Conformer block with relative position encoding (Dai et al., 2019). The detailed architecture is listed in Appendix B.

3.6 Decoders and Objectives

3.6.1 Note Segmentation

We adopt a note boundary decoder, denoted as D_B, to transform the output feature Z from the U-Net backbone into logits $\hat{\boldsymbol{y}}_{\text{bd}}$, where $Z \in \mathbb{R}^{T \times C}$ and C is the channel dimension. D_B is implemented by a single matrix $W_B \in \mathbb{R}^{C \times 1}$. A binary cross-entropy

(BCE) loss is applied to train note segmentation:

$$\begin{aligned}\mathcal{L}_{\mathrm{B}} =& \frac{1}{T}\sum \mathrm{BCE}(\boldsymbol{y}_{\mathrm{bd}}, \hat{\boldsymbol{y}}_{\mathrm{bd}}) \\ =& -\frac{1}{T}\sum_{t=1}^{T}(y_{\mathrm{bd}}^{t}\ln(\sigma(\hat{y}_{\mathrm{bd}}^{t}/T_1)) \\ &+ (1-y_{\mathrm{bd}}^{t})\ln(1-\sigma(\hat{y}_{\mathrm{bd}}^{t}/T_1))) \end{aligned} \quad (5)$$

where T_1 is the temperature hyperparameter, and $\sigma(\cdot)$ stands for sigmoid function.

It is worth mentioning that in the note segmentation task, there is a significant imbalance between positive and negative samples, with a ratio of approximately 1:500[1]. Furthermore, the inclusion of word boundary conditions results in varying classification difficulties, with some boundaries being inherently easier to classify than others. To tackle this imbalance problem, we employ a focal loss (Lin et al., 2017) to focus more on hard samples:

$$p_{\mathrm{t}} = \boldsymbol{y}_{\mathrm{bd}}\sigma(\hat{\boldsymbol{y}}_{\mathrm{bd}}) + (1-\boldsymbol{y}_{\mathrm{bd}})(1-\sigma(\hat{\boldsymbol{y}}_{\mathrm{bd}})) \quad (6)$$

$$\alpha_{\mathrm{t}} = \alpha\boldsymbol{y}_{\mathrm{bd}} + (1-\alpha)(1-\boldsymbol{y}_{\mathrm{bd}}) \quad (7)$$

$$\mathcal{L}_{\mathrm{FC}} = \frac{1}{T}\sum \alpha_{\mathrm{t}}(1-p_{\mathrm{t}})^{\gamma}\mathrm{BCE}(\boldsymbol{y}_{\mathrm{bd}}, \hat{\boldsymbol{y}}_{\mathrm{bd}}) \quad (8)$$

where α is a hyperparameter controlling weight of positive samples, and γ controls balance between easy and hard samples.

3.6.2 Pitch Prediction

For pitch value prediction $\mathrm{D_P}$, we leverage an attention-based weighted average operation to aggregate the fine-grained features, instead of simply applying a weighted median or average. Given the output feature $Z \in \mathbb{R}^{T\times C}$, we obtain an attention weight matrix S through a projection matrix $W_{\mathrm{A}} \in \mathbb{R}^{C\times H}$: $S = \sigma(ZW_{\mathrm{A}})$, where $S \in \mathbb{R}^{T\times H}$ and H denotes the number of attention heads. Then we perform an outer product operation between each vector of Z and S along the time dimension to obtain a pre-weighted representation: $Z_1^t = Z^t \otimes S^t$ and $Z_1 \in \mathbb{R}^{T\times C\times H}$, which is further averaged along the head dimension to acquire the weighted representation $Q \in \mathbb{R}^{T\times C}$. In addition, we compute the averaged weights $\boldsymbol{s} \in \mathbb{R}^{T}$ by averaging along the head dimension.

Subsequently, we use the note boundary sequence $\boldsymbol{y}_{\mathrm{bd}}$ to segment Q along the time axis, resulting in a group sequence $\boldsymbol{G} = [G^1, G^2, ..., G^L]$ with length of L, number of notes. Each group G^i contains l_i vectors: $G^i = [Q^{j+1}, Q^{j+2}, ..., Q^{j+l_i}]$, where $\sum_{i=1}^{L} l_i = T$, $i \in [1, L]$, $j \in [1, T]$, and $y_{\mathrm{bd}}^{j} = 1$. We also do the same for the averaged weights $\boldsymbol{s}$: $G_s^i = [s^{j+1}, s^{j+2}, ..., s^{j+l_i}]$. For each group, we compute a weighted average $\boldsymbol{z}^i$:

$$\boldsymbol{z}^i = \frac{\sum G^i}{\sum G_s^i} = \frac{\sum_{k=1}^{l_i} Q^{j+k}}{\sum_{k=1}^{l_i} s^{j+k}} \quad (9)$$

Finally, we multiply $\boldsymbol{z}$ with a matrix W_O to compute the logits: $\hat{\boldsymbol{p}} = \boldsymbol{z}W_{\mathrm{O}}$, where $\boldsymbol{z} \in \mathbb{R}^{L\times C}$ and $W_{\mathrm{O}} \in \mathbb{R}^{C\times P}$. P is the number of pitch categories. A cross-entropy (CE) loss is utilized:

$$\mathcal{L}_{\mathrm{P}} = -\frac{1}{L}\sum_{i=1}^{L}\sum_{c=1}^{P} p_c^i \ln\left(\frac{\exp(\hat{p}_c^i/T_2)}{\sum_{k=1}^{C}\exp(\hat{p}_c^k/T_2)}\right) \quad (10)$$

where T_2 is the temperature hyperparameter.

3.7 Training and Inference Pipeline

In the training stage, we use ground-truth (GT) note boundaries to segment the intermediate features and optimize the pitch decoder. The overall loss $\mathcal{L} = \lambda_{\mathrm{B}}\mathcal{L}_{\mathrm{B}} + \lambda_{\mathrm{FC}}\mathcal{L}_{\mathrm{FC}} + \lambda_{\mathrm{P}}\mathcal{L}_{\mathrm{P}}$ is controlled by balancing parameters λ_{B}, λ_{FC}, and λ_{P}.

In the inference stage, firstly we compute the boundary probability $\sigma(\hat{\boldsymbol{y}}_{\mathrm{bd}})$ and use a threshold μ to decide the boundary state. That is, a note boundary exists at time step t if $\sigma(\hat{\boldsymbol{y}}_{\mathrm{bd}}) > \mu$, otherwise, it does not. The predicted results will undergo post-processing to clean up boundaries with excessively small spacing between them. Finally, we segment the intermediate feature Z and decode pitches.

It is worth mentioning that μ can control the granularity of generated notes. In other words, a lower μ may result in more fine-grained and subdivided pitches, while a higher one ignores small fluctuations. This is because a lower μ allows more boundaries.

3.8 Singing Voice Synthesis System

Once we complete the inference and automatically annotate a dataset, the new datasets are used to train an SVS system to further investigate the practical performance. We choose RMSSinger as the singing acoustic model and a pre-trained HiFi-GAN (Kong et al., 2020) model as the vocoder. RMSSinger is originally proposed for word-level realistic music score inputs, denoted as $\mathcal{S}$. To suit our settings, we drop the word-level attention module and directly use the fine-grained MIDI input. The alignment between MIDI notes and phonemes and other settings are reproduced according to He et al. (2023).

[1]Statistically, there are approximately 2.42 note boundaries per second in our datasets.

4 Experiments

In this section, we begin by showcasing experiments on AST tasks, followed by simulations and comparisons of a comprehensive annotation-and-training pipeline for an SVS task. We also investigated the model's performance in low-resource scenarios; however, due to space limitations, this part is included in Appendix E.

4.1 Experimental Setup

Data We utilize two Mandarin datasets. The first is M4Singer (Zhang et al., 2022a), a multi-singer and multi-style singing voice corpus, which is approximately 26.5 hours after pre-processing. Secondly, we collect and annotate a high-quality song corpus, denoted as $\mathcal{D}_1$. $\mathcal{D}_1$ is composed of songs sung by 12 professional singers, with a total length of 20.9 hours. For training AST models, these two datasets are used jointly, with two 3% subsets used as the validation and the testing sets. The details of data collection are listed in Appendix C.

Implementation and Training We sample waveforms with a sample rate of 24k Hz. F0 contours are extracted through a pre-trained RMVPE (Wei et al., 2023) estimator, where each F0 value is quantized into 256 categories. The length of softened boundaries is set to 80 ms. The U-Net backbone is constructed with 4 down- and up-sampling layers, with $16\times$ downsampling rate. For inference, the boundary threshold μ is set to 0.8. We train the model for 60k steps using 2 NVIDIA 2080Ti GPUs with a batch size of 60k max frames. An AdamW optimizer is used with $\beta_1 = 0.9$, $\beta_2 = 0.98, \epsilon = 10^{-8}$. The learning rate is set to 10^{-5} with a decay rate of 0.998 and a decay step of 500 steps. More details are listed in Appendix B.

Evaluation We utilize the `mir_eval` library (Raffel et al., 2014) for performance evaluation. Specifically, we compute F1, precision, and recall scores of onset, offset, and pitch value. An average overlap ratio (AOR) is also calculated for correctly transcribed notes. In addition, we compute the melody measures to reflect the overall perception performance, by transforming the GT and predicted note events into frame-level and computing raw pitch accuracy (RPA). For ROSVOT, we remove silence notes and designate the boundaries that enclose each note as onset and offset. This step is unnecessary for other baselines. The onset tolerance is set to 50 ms, and the offset tolerance is the larger value between 50 ms and 20% of note duration. The pitch tolerance is set to 50 cents. All numbers demonstrated are multiplied by 100.

Baselines We compare ROSVOT, denoted as $\mathcal{M}$, with multiple baselines: 1) *TONY* (Mauch et al., 2015), a automatic software with visualization; 2) *VOCANO* (Hsu et al., 2021), retrained on the joint datasets; 3) *MusicYOLO*, retrained; 4) *(Yong et al., 2023)*, reproduced and retrained. We also compare the results of several variants of $\mathcal{M}$: 1) $\mathcal{M}$ *(conformer)*, where the U-Net is dropped and the backbone is the Conformer alone; 2) $\mathcal{M}$ *(conv)*, where the middle Conformer blocks are replaced by 8-layer convolution blocks; 3) $\mathcal{M}$ *(w/o wbd)*, canceling word boundary condition; 4) $\mathcal{M}$ *(w/o noise)*, which is identical to $\mathcal{M}$ but trained without noisy environment; 5) $\mathcal{M}$ *(w/* E_W*)*, meaning that the GT word boundaries are not available and need to be extracted from the extractor $\mathrm{E_W}$.

4.2 Main Results

We run two sets of experiments under clean and noisy environments, respectively. The noisy environment is produced by mixing MUSAN noises with a probability of 0.8 and an SNR range of $[6, 20]$. The main results are listed in Table 1. For the sake of brevity, only F1 scores, averaged overlap ratios, and raw pitch accuracies are listed here, and the complete scores are listed in Appendix D.

From the results, we can see that 1) the proposed multi-scale model achieves better performances for both boundary detection and pitch prediction by a large margin, even without noises; 2) Training under a noisy environment significantly improves the robustness, while the performances of baselines are severely degraded when facing noisy inputs; 3) The involvement of noises in training stage also improves the inference performance facing clean waveforms, this may because the noise mixing operation forms a bottleneck to force the model to focus on note-related information.

4.3 Ablation Study

To demonstrate the effectiveness of several designs in the proposed method, we conduct ablation studies and compare the results of different hyperparameters. From Table 1 we can see that dropping the U-Net backbone or replacing the Conformer with convolution blocks decreases the performance. In particular, the performance of $\mathcal{M}$ *(conformer)* significantly deteriorates when dealing with noisy inputs, suggesting that the downsampling layers contribute to a denoising effect. This is also vali-

Method	Onset (F) ↑		Offset (F) ↑		Pitch (F) ↑		Pitch (AOR) ↑		Melody (RPA) ↑	
	clean	noisy	clean	noisy	clean	noisy	clean	noisy	clean	noisy
TONY	67.5	49.2	57.8	47.0	43.9	28.4	73.8	46.6	73.9	45.2
VOCANO	75.8	64.7	71.2	66.1	50.2	43.4	81.4	71.9	76.6	59.8
MusicYOLO	82.2	79.7	81.7	76.5	58.9	51.5	85.4	78.6	81.6	78.9
(Yong et al., 2023)	92.0	88.5	91.4	89.7	65.8	62.1	91.6	86.4	83.1	80.6
$\mathcal{M}$ *(conformer)*	92.1	90.6	91.8	90.8	70.3	69.8	95.9	95.3	83.9	83.1
$\mathcal{M}$ *(conv)*	91.6	91.5	92.6	92.6	70.9	70.8	96.8	96.8	84.1	84.1
$\mathcal{M}$ *(w/o wbd)*	91.3	91.1	91.8	91.2	70.2	69.9	95.5	95.1	83.8	83.4
$\mathcal{M}$ *(w/o noise)*	93.8	90.9	94.2	91.5	76.4	70.1	**97.1**	95.2	87.1	83.1
$\mathcal{M}$ *(w/* E_W*)*	93.3	93.5	93.2	92.9	77.1	77.0	96.5	96.2	87.5	87.2
$\mathcal{M}$ *(ours)*	**94.0**	**93.8**	**94.5**	**94.4**	**77.4**	**77.0**	97.0	**97.1**	**87.6**	**87.4**

Table 1: Evaluation results of AST systems.

dated in the results of $\mathcal{M}$ *(w/o noise)*, indicating that even though it is trained with clean samples, it still exhibits a certain level of robustness. For a fair comparison, we test $\mathcal{M}$ *(w/* E_W*)* to demonstrate the performance in note-only scenarios. The results indicate that despite the accumulated errors introduced by the word boundary extractor $\mathrm{E_W}$, the performance does not decline significantly.

Rate	Step (ms)	Onset	Offset	Pitch
2	10.7	92.7	92.4	70.7
4	21.3	93.9	93.6	73.6
8	42.7	**94.4**	94.1	76.8
16	85.3	94.0	**94.5**	**77.4**
32	170.7	94.3	94.1	77.2

Table 2: Comparisons of different downsampling rates. "Step" denotes the downsampled step size in the Conformer, measured in milliseconds.

We record the comparison results of different overall downsampling rates of the U-Net backbone in Table 2, where only F1 scores are listed. The results align remarkably well with the length of the soft labels, which are both about 80 ms. We choose the rate of 16 in the final architecture $\mathcal{M}$ as it achieves better overall performance.

For pitch prediction, we compare the results between the proposed attention-based method and the weighted median method used in previous works. We drop the pitch decoder $\mathrm{D_P}$ and apply a weighted median algorithm on the F0 contours according to (Yong et al., 2023). The F1 and AOR scores of this algorithm with clean inputs are 70.5 and 92.6, while the scores with noisy inputs are 63.2 and 86.6. The results indicate that a simple weighted median is insufficient in dealing with fluctuated pitches in singing voices, which are full of expressive techniques like portamentos. Also, its performance is largely dependent on the F0 extractor.

4.4 Towards Automatic Annotation

The experimental results indicate that ROSVOT achieves superior performance, but what practical significance does it hold? In this section, we establish a comprehensive SVS pipeline, using ROSVOT as the automatic annotator.

4.4.1 Implementation and Pipeline

Data. We re-align and re-annotate the OpenSinger corpus (Huang et al., 2021), which consists of 84.8 hours of singing voices recorded by 93 singers. We also perform cross-lingual generalization by annotating an English corpus $\mathcal{D}_2$, which has a length of 6 hours. For future reference, we use the term pseudo-annotations for the automatically generated transcriptions. Details are listed in Appendix D.

Evaluation. For objective evaluation, we also apply the RPA score to measure the reconstructed F0 contours. The RPA scores for *GT Mel* are computed between F0s from GT vocoder generations and GT waveforms, while the others are between GT and generations. For subjective evaluation, we conducted crowdsourced mean opinion score (MOS) listening tests. Specifically, we score MOS-P and MOS-Q corresponding to pitch reconstruction and overall quality. The metrics are rated from 1 to 5 and reported with 95% confidence intervals.

4.4.2 SVS Results

Firstly, we investigate the effect of training with pseudo-annotations at different ratios. We only utilize M4Singer to train ROSVOT $\mathcal{M}$, which is used to generate the pseudo annotations. Pseudo

Real	R. Size	Pseudo	P. Size	RPA		MOS-P		MOS-Q	
				R	P	R	P	R	P
-	-	-	-	95.5		4.16±0.11		4.08±0.08	
$\mathcal{D}_1$	20.9	-	0.0	67.6	61.1	**3.69±0.09**	3.54±0.07	**3.81±0.04**	3.74±0.05
$\mathcal{D}_1 \times 50\%$	10.5	$\mathcal{D}_1 \times 50\%$	10.5	66.0	61.0	3.62±0.03	3.56±0.08	3.76±0.07	3.72±0.03
$\mathcal{D}_1 \times 10\%$	2.1	$\mathcal{D}_1 \times 90\%$	18.8	65.9	61.3	3.65±0.05	3.55±0.05	3.71±0.08	3.73±0.04
$\mathcal{D}_1 \times 5\%$	1.0	$\mathcal{D}_1 \times 95\%$	19.9	63.0	63.3	3.61±0.05	3.57±0.04	3.73±0.05	**3.78±0.03**
$\mathcal{D}_1 \times 1\%$	0.2	$\mathcal{D}_1 \times 99\%$	20.7	63.5	64.7	3.60±0.04	**3.59±0.04**	3.74±0.06	3.76±0.04
-	0.0	$\mathcal{D}_1$	20.9	61.8	64.6	3.60±0.06	3.58±0.03	3.73±0.04	3.73±0.08
M4	26.5	-	0.0	68.1	67.7	3.63±0.05	3.60±0.04	3.79±0.05	3.77±0.07
M4 + $\mathcal{D}_1$	47.4	-	0.0	67.4	66.5	**3.67±0.07**	3.59±0.08	3.81±0.04	3.80±0.06
M4	26.5	$\mathcal{D}_1$	20.9	66.6	64.9	3.64±0.08	**3.61±0.04**	3.80±0.07	3.80±0.04
M4	26.5	$\mathcal{D}_1$ + OP	105.7	66.1	64.1	3.63±0.10	3.60±0.07	**3.83±0.09**	**3.81±0.08**

Table 3: Evaluation results of SVS pipelines. The first row is for *GT Mel*, where we generate waveforms using the vocoder from GT Mel-spectrograms. "M4" denotes M4Singer, and "OP" denotes OpenSinger. "R" and "P" denote inference results using real or pseudo annotations, respectively. The sizes are measured in hours.

Model	RPA	MOS-P	MOS-Q
$\mathcal{S}$*(large)*	45.2	3.36 ± 0.12	3.45 ± 0.09

Table 4: Results of cross-lingual generalization.

annotations with different ratios are mixed into $\mathcal{D}_1$ to form the training set. For inference, we reserve two 1% segments from each real and pseudo group for validation and testing. The results are listed in Table 3, rows 2-7. From the results, we can see that the pitch accuracy of real annotation inputs decreases when mixing more pseudo annotations, but the accuracy of pseudo inputs increases. This suggests a minor discrepancy in the distributions of real and pseudo annotations. However, the performance degradation is not significant: 99% of the pseudo mixing contributes only a 6% drop in performance. The MOS-Q scores share a similar pattern, but they involve a comprehensive evaluation with considerations of audio quality and more. A decrease in pitch accuracy does not necessarily lead to an overall decline in quality.

We further investigate the performance as data size increases. While the AST model remains the same, we train $\mathcal{S}$ only using M4Singer as the baseline. Next, we gradually mix $\mathcal{D}_1$ and OpenSinger to expand the data size. To consume the largest datasets in the last row, we construct a large version of RMSSinger with 320-dimensional channels and a 6-layer decoder[2], denoted as $\mathcal{S}$*(large)*. The results are listed in Table 3, rows 8-11. A slight reduction in pitch accuracy can be observed when integrating diverse datasets, which may result from the inherent differences in dataset characteristics and annotation styles. However, the overall quality improves, as the model has been exposed to a sufficient variety of pronunciation styles and singing patterns, and the modeling becomes more stable and robust. This indicates that ROSVOT provides an opportunity for SVS models to scale up.

[2]The dictionary of the text encoder is also merged with English phonemes for the following cross-lingual experiments

4.4.3 Cross-lingual Generalization

For cross-lingual experiments, we finetune the pretrained model $\mathcal{S}$*(large)* on the English corpus $\mathcal{D}_2$. Note that $\mathcal{D}_2$ only has 6 hours, with all the annotations generated automatically. We finetune both stages 1 and 2 for 100k steps. The results are listed in Table 4, which shows that the proposed method has the capability of cross-lingual generalization.

5 Conclusion

In this paper, we introduce ROSVOT, the first robust AST model that ultimately serves SVS. We leverage a multi-scale architecture to achieve a balance between coarse-grained note modeling and fine-grained segmentation. An attention-based decoder with dynamic weight is devised for pitch regression. Additionally, we establish a comprehensive pipeline for SVS training. Experimental results reveal that our model achieves the best performance under either clean or noisy environments. Annotating and incorporating larger datasets improves the SVS model's performance, indicating the capability of practical annotation of ROSVOT.

6 Limitations and Potential Risks

The proposed method acknowledges two primary limitations. First, the cross-lingual capability is only tested on a small-scale English dataset, necessitating extensional experiments for a comprehensive evaluation of generalization performance. Second, due to space constraints, only one SVS model is examined as the baseline. Additional verifications involving different SVS models are required to fully demonstrate practical performance. Future work will involve testing automatically annotated transcriptions on a more diverse set of SVS models.

The misuse of the proposed model for singing voice synthesis could potentially lead to copyright-related issues. To address this concern, appropriate constraints will be implemented to mitigate any illegal or unauthorized usage.

References

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460.

Bhuwan Bhattarai and Joonwhoan Lee. 2023. A comprehensive review on music transcription. *Applied Sciences*, 13(21):11882.

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

Alexis Conneau, Alexei Baevski, Ronan Collobert, Abdelrahman Mohamed, and Michael Auli. 2020. Unsupervised cross-lingual representation learning for speech recognition. *arXiv preprint arXiv:2006.13979*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Zih-Sing Fu and Li Su. 2019. Hierarchical classification networks for singing voice segmentation and transcription. In *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR 2019)*, pages 900–907.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. 2020. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*.

Jinzheng He, Jinglin Liu, Zhenhui Ye, Rongjie Huang, Chenye Cui, Huadai Liu, and Zhou Zhao. 2023. Rmssinger: Realistic-music-score based singing voice synthesis. *arXiv preprint arXiv:2305.10686*.

Jui-Yang Hsu, Li Su, et al. 2021. Vocano: A note transcription framework for singing voice in polyphonic music.

Rongjie Huang, Feiyang Chen, Yi Ren, Jinglin Liu, Chenye Cui, and Zhou Zhao. 2021. 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.

Jaehyeon Kim, Jungil Kong, and Juhee Son. 2021. Vits: Conditional variational autoencoder with adversarial learning for end-to-end text-tospeech. In *Proc. ICML*, pages 5530–5540.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. 2020. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033.

Ruiqi Li, Rongjie Huang, Lichao Zhang, Jinglin Liu, and Zhou Zhao. 2023. Alignsts: Speech-to-singing conversion via cross-modal alignment. *arXiv preprint arXiv:2305.04476*.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Jinglin Liu, Chengxi Li, Yi Ren, Feiyang Chen, and Zhou Zhao. 2022. Diffsinger: Singing voice synthesis via shallow diffusion mechanism. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):11020–11028.

Matthias Mauch, Chris Cannam, Rachel Bittner, George Fazekas, Justin Salamon, Jiajie Dai, Juan Bello, and Simon Dixon. 2015. Computer-aided melody note transcription using the tony software: Accuracy and efficiency.

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

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR.

Colin Raffel, Brian McFee, Eric J Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel PW Ellis, and C Colin Raffel. 2014. mir_eval: A transparent implementation of common mir metrics. In *In*

Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR. Citeseer.

Yi Ren, Xu Tan, Tao Qin, Jian Luan, Zhou Zhao, and Tie-Yan Liu. 2020. 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.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer.

Kai Shen, Zeqian Ju, Xu Tan, Yanqing Liu, Yichong Leng, Lei He, Tao Qin, Sheng Zhao, and Jiang Bian. 2023. Naturalspeech 2: Latent diffusion models are natural and zero-shot speech and singing synthesizers. *arXiv preprint arXiv:2304.09116*.

Yaman Kumar Singla, Jui Shah, Changyou Chen, and Rajiv Ratn Shah. 2022. What do audio transformers hear? probing their representations for language delivery & structure. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 910–925. IEEE.

David Snyder, Guoguo Chen, and Daniel Povey. 2015. MUSAN: A Music, Speech, and Noise Corpus. ArXiv:1510.08484v1.

Xianke Wang, Wei Xu, Weiming Yang, and Wenqing Cheng. 2022a. Musicyolo: A sight-singing onset/offset detection framework based on object detection instead of spectrum frames. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 396–400. IEEE.

Yu Wang, Xinsheng Wang, Pengcheng Zhu, Jie Wu, Hanzhao Li, Heyang Xue, Yongmao Zhang, Lei Xie, and Mengxiao Bi. 2022b. Opencpop: A high-quality open source chinese popular song corpus for singing voice synthesis. *arXiv preprint arXiv:2201.07429*.

Haojie Wei, Xueke Cao, Tangpeng Dan, and Yueguo Chen. 2023. Rmvpe: A robust model for vocal pitch estimation in polyphonic music. *arXiv preprint arXiv:2306.15412*.

Dongchao Yang, Jinchuan Tian, Xu Tan, Rongjie Huang, Songxiang Liu, Xuankai Chang, Jiatong Shi, Sheng Zhao, Jiang Bian, Xixin Wu, et al. 2023. Uniaudio: An audio foundation model toward universal audio generation. *arXiv preprint arXiv:2310.00704*.

Sangeon Yong, Li Su, and Juhan Nam. 2023. A phoneme-informed neural network model for note-level singing transcription. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Lichao Zhang, Ruiqi Li, Shoutong Wang, Liqun Deng, Jinglin Liu, Yi Ren, Jinzheng He, Rongjie Huang, Jieming Zhu, Xiao Chen, et al. 2022a. M4singer: A multi-style, multi-singer and musical score provided mandarin singing corpus. *Advances in Neural Information Processing Systems*, 35:6914–6926.

Yongmao Zhang, Jian Cong, Heyang Xue, Lei Xie, Pengcheng Zhu, and Mengxiao Bi. 2022b. Visinger: Variational inference with adversarial learning for end-to-end singing voice synthesis. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7237–7241.

Yu Zhang, Rongjie Huang, Ruiqi Li, JinZheng He, Yan Xia, Feiyang Chen, Xinyu Duan, Baoxing Huai, and Zhou Zhao. 2023. Stylesinger: Style transfer for out-of-domain singing voice synthesis. *arXiv preprint arXiv:2312.10741*.

Zewang Zhang, Yibin Zheng, Xinhui Li, and Li Lu. 2022c. Wesinger: Data-augmented singing voice synthesis with auxiliary losses. *arXiv preprint arXiv:2203.10750*.

A Word Boundary Condition

Word boundary conditions are introduced to regulate segmentation results. It seems similar to Yong et al. (2023), but a word boundary sequence forms a much narrower information bottleneck without introducing unnecessary information. In practice, we embed the word boundary sequences to inform the model of boundary conditions. Also, we use the word boundary sequence as a reference to regulate the predicted note boundaries. Specifically, we remove the note boundaries that are too close to the reference word boundaries, where the threshold is 40 ms.

This regulation is only for automatic annotation. For a note-only application, word-note synchronization is not necessary. In this scenario, we build a word boundary extractor $\mathrm{E_W}$ to provide weak linguistic supervision. The extractor shares the same architecture as the note segmentation part of ROSVOT. The multi-scale architecture also functions well in localizing frame-level word boundaries. Specifically, we use an MFA-aligned AISHELL-3 Mandarin corpus to pre-train $\mathrm{E_W}$, followed by fine-tuning it with M4Singer and $\mathcal{D}_1$.

B Architecture and Implementation Details

B.1 Hyperparameters

For hyperparameters, we sample waveforms with a sample rate of 24000 Hz. Mel-spectrograms are

computed with a window size of 512, and a hop size of 128. The number of Mel bins is set to 80. To form a bottleneck and alleviate overfitting, we only use the first 30 bins (low-frequency part) as input. MUSAN noises are added to the waveforms with a probability of 0.8 and an SNR range of $[6, 20]$. Gaussian noise is added to the logarithmic F0 contours with a random standard deviation range of $[0, 0.04]$, and is added to the softened boundary labels with $[0, 0.002]$. F0 contours are extracted through a pre-trained RMVPE (Wei et al., 2023) estimator, where each F0 value is quantized into 256 categories. We set P, the number of pitch categories, to 120, where each pitch number is the exact MIDI number. The length of softened boundaries is set to 80 ms, indicating a 15-frame window $W_{\mathcal{G}}$. The temperature parameters T_1 and T_2 are set to 0.2 and 0.01. To balance the various objectives, we set λ_{B}, λ_{FC}, andλ_{P} to 1.0, 3.0, and 1.0. For inference, the boundary threshold μ is set to 0.8. The hyperparameters α and γ in the boundary decoder are set to $1/(2.42 \times 128/24000)$ and 5.0, where the 2.42 in the former indicates the number of note boundaries in one second, and 128 and 24000 indicate the hop size and the audio sample rate.

B.2 Architecture

For model architecture, we apply three encoders E_{M}, E_{B}, E_{P} to encode Mel-spectrograms, word boundaries, and F0 contours. The encoders consist of a linear projection or an embedding layer, followed by residual convolution blocks.

The U-Net backbone's encoder and decoder each comprise K downsampling and upsampling layers, respectively, where $K = 4$ in our case, with $16\times$ downsampling rate. A downsampling layer consists of a residual convolution block and an average pooling layer with a downsampling rate of 2, resulting in an overall downsampling rate of 2^K. For an upsampling layer, the input feature is firstly upsampled through a transposed convolution layer, and is then concatenated with the corresponding skipped feature before a final convolution block. The downsampling rate is set to 2, and the channel dimension remains the same as input to alleviate overfitting. The intermediate part of the backbone is replaced by a 2-layer Conformer block with relative position encoding. The Conformer network is 2-layer with a kernel size of 9 and a head size of 4. The head dimension in the pitch decoder is 4. The overall channel dimension is 256. The overall architecture is listed in Table 5.

As for the N-layer residual convolution blocks mentioned many times in the main text, the configuration is illustrated in Figure 4.

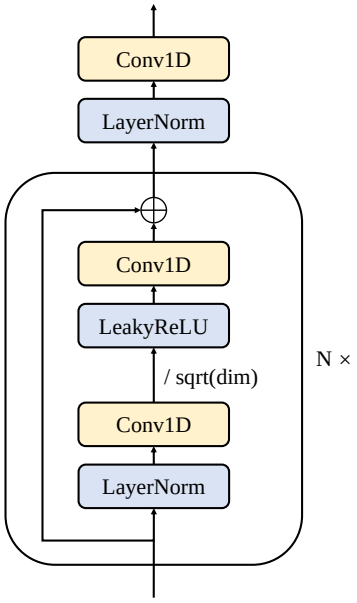

Figure 4: N-layer Residual convolution blocks.

C Data

We recruit 12 professional singers (8 female, 4 male) to record $\mathcal{D}_1$ and 8 singers (5 female, 3 male) for $\mathcal{D}_2$. Each singer was compensated at an hourly rate of $600. Singers were informed that the recordings were for scientific research use. Then we automatically annotate the phonemes and notes through an ASR model (Radford et al., 2023), MFA, and the proposed AST model. The length of $\mathcal{D}_1$ is 20.9 hours and $\mathcal{D}_2$ is 6 hours. We use all the datasets under license CC BY-NC-SA 4.0.

D Additional Experimental Results

The additional experimental results are listed in Table 7 and Table 8, where the former is under clean environment and the latter is noisy.

E Low-resource Scenarios

Considering the scarcity of annotated singing voice datasets, we investigate the performance of the proposed method under low-resource scenarios. We use M4Singer as the training set and test the model on $\mathcal{D}_1$. Firstly, we gradually decrease the amount of training data to see the performance degradation. After that, we incorporate features extracted from a pre-trained self-supervised learning (SSL) framework to enhance the performance.

Hyperparameter		Model
Mel Encoder	Encoder Kernel	3
	Encoder Layers	2
	Encoder Hidden	256
Condition Encoder	Pitch Embedding	300
	UV Embedding	3
	WBD Embedding	3
	Encoder Kernel	3
	Encoder Layers	1
	Encoder Hidden	256
U-Net	Kernel	3
	Enc & Dec Layers	4
	Downsampling Rate	16
	Enc & Dec Hidden	256
Conformer	Kernel	9
	Heads	4
	Layers	2
	Attention Hidden	256
	FFN Hidden	1024
Total Number of Parameters		12M

Table 5: Hyperparameters of the proposed modules. "WBD" represents word boundary.

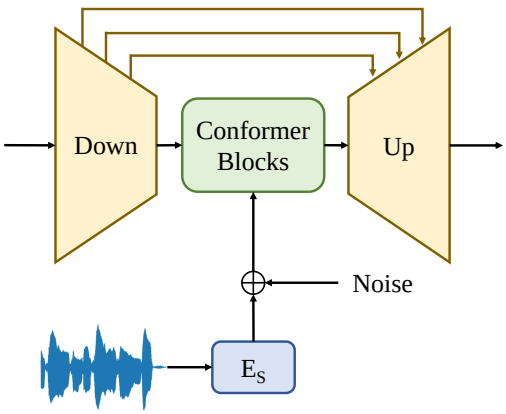

Figure 5: Injection of self-supervised features.

Model	Ratio	Onset (F)	Offset (F)	Pitch (F)
$\mathcal{M}$	100%	93.7	94.3	77.1
$\mathcal{M}$	50%	93.6	93.9	79.6
$\mathcal{M}$	10%	93.0	92.6	71.6
$\mathcal{M}$	1%	92.0	91.7	68.4
$\mathcal{M}(ssl)$	100%	94.3	94.0	76.2
$\mathcal{M}(ssl)$	50%	94.0	93.7	74.9
$\mathcal{M}(ssl)$	10%	93.8	93.9	73.5
$\mathcal{M}(ssl)$	1%	93.7	93.8	73.6

Table 6: Results under low-resource scenarios. "Ratio" indicates the proportion of the training set that is utilized.

Specifically, we modify the model architecture by introducing a latent feature encoder E_S, transforming the additional SSL representations into 256-dimensional features, and performing a fusion by element-wise addition. This fusion can be illustrated as Figure 5. E_S comprises two convolution layers and a convolution block, where the former reduces the dimension of the input features to the model channel dimension. The output of E_S is directly added to the output of the U-Net's encoder to perform the fusion.

We choose XLSR-53 (Conneau et al., 2020), a wav2vec 2.0 (Baevski et al., 2020) model pre-trained on 56k hours of speech in 53 languages, to be the SSL feature extractor. We believe that the knowledge of a pre-trained self-supervised model alleviates data scarcity. To simulate the low-resource environment, we actually can get access to singing voice corpora, only without annotations. Therefore, we use all the training data mentioned before to fine-tune the XLSR-53 model with a batch size of 1200k tokens for 20k steps. In this case, we incorporate self-supervised learning to cope with the low-resource problem.

According to Singla et al. (2022), features from the second layer of a 12-layer wav2vec 2.0 model are the most related to audio features like pitch and unvoiced ratio, we extract features from the 4th layer of the 24-layer XLSR-53 to be the input feature, which has a dimension of 1024. Before feeding the features to the model, we add Gaussian noises with a standard deviation of 0.05 to perform the data augmentation. The SSL-augmented model is denoted as $\mathcal{M}(ssl)$.

The results are listed in Table 6. From the results, we can see that there is no significant improvement after involving SSL features, if enough training

data is utilized. However, when decreasing the training data, the original model $\mathcal{M}$ exhibits a decline in performance, while $\mathcal{M}$*(ssl)* experiences a comparatively smaller decrease.

F Details of Evaluation

For each SVS experiment task, 20 samples are randomly selected from our test set for subjective evaluation. Professional listeners, totaling 20 individuals, are engaged to assess the performance. In MOS-Q evaluations, the focus is on overall synthesis quality, encompassing clarity and naturalness. For MOS-P, listeners are exposed to GT samples and instructed to concentrate on pitch reconstruction, disregarding audio quality. In both MOS-Q and MOS-P evaluations, participants rate various singing voice samples on a Likert scale from 1 to 5. It is crucial to highlight that all participants were remunerated for their time and effort, compensated at a rate of $10 per hour, resulting in a total expenditure of approximately $300 on participant compensation. Participants were duly informed that the data were for scientific research use.

Method	Onset			Offset			Pitch				Melody
	P	R	F	P	R	F	P	R	F	AOR	RPA
TONY	65.0	70.2	67.5	59.6	56.1	57.8	46.1	42.1	43.9	73.8	73.9
VOCANO	73.7	78.1	75.8	69.1	73.5	71.2	48.4	52.1	50.2	81.4	76.6
MusicYOLO	84.1	80.4	82.2	83.7	79.8	81.7	61.3	56.8	58.9	85.4	81.6
(Yong et al., 2023)	93.7	90.0	92.0	92.3	90.4	91.4	65.3	66.3	65.8	91.6	83.1
$\mathcal{M}$ *(conformer)*	90.2	93.0	93.7	92.3	96.0	94.1	75.3	77.4	76.3	97.0	86.9
$\mathcal{M}$ *(conv)*	92.2	91.0	91.6	92.8	92.3	92.6	71.6	70.2	70.9	96.8	84.1
$\mathcal{M}$ *(w/o noise)*	**94.5**	93.2	93.8	**94.7**	93.7	94.2	76.0	76.8	76.4	**97.1**	87.1
$\mathcal{M}$ *(w/o wbd)*	92.1	94.5	93.3	91.6	94.8	93.2	75.9	78.4	77.1	96.5	87.5
$\mathcal{M}$ *(ours)*	92.4	**96.2**	**94.0**	93.0	**96.8**	**94.5**	**76.4**	**78.9**	**77.4**	97.0	**87.6**

Table 7: Complete results of AST systems. These results are from clean inputs.

Method	Onset			Offset			Pitch				Melody
	P	R	F	P	R	F	P	R	F	AOR	RPA
TONY	48.6	49.8	49.2	45.9	48.2	47.0	25.8	31.7	28.4	46.6	45.2
VOCANO	63.7	65.8	64.7	64.7	67.5	66.1	42.9	44.0	43.4	71.9	59.8
MusicYOLO	81.3	78.2	79.7	79.6	73.7	76.5	53.9	49.9	51.5	78.6	78.9
(Yong et al., 2023)	89.7	87.2	88.5	90.1	89.3	89.7	63.4	60.8	62.1	86.4	80.6
$\mathcal{M}$ *(conformer)*	89.8	92.6	91.2	91.2	93.0	92.1	73.7	75.9	74.8	96.9	86.3
$\mathcal{M}$ *(conv)*	92.4	90.6	91.5	93.3	91.9	92.6	71.1	70.6	70.8	96.8	84.1
$\mathcal{M}$ *(w/o noise)*	91.9	89.9	90.9	92.3	90.7	91.5	70.2	70.0	70.1	95.2	83.1
$\mathcal{M}$ *(w/o wbd)*	92.4	94.6	93.5	91.6	94.2	92.9	76.3	**77.7**	77.0	96.2	87.2
$\mathcal{M}$ *(ours)*	**92.9**	**94.7**	**93.8**	**93.5**	**95.3**	**94.4**	**76.7**	77.3	**77.0**	**97.1**	**87.4**

Table 8: Complete results of AST systems. These results are from noisy inputs.