Robust Singing Voice Transcription Serves Synthesis

Ruiqi Li, Yu Zhang, Yongqi Wang, Zhiqing Hong, Rongjie Huang, Zhou Zhao*
Zhejiang University
{ruiqili,yuzhang34,rongjiehuang,zhaozhou}@zju.edu.cn

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. Codes can be found at https://github.com/RickyL-2000/ROSVOT.

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; Huang 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.

*Corresponding author

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 is a popular method for data collection (Ren et al., 2020), but the quality varies. AST methods are vulnerable to noise as sound artifacts tend to disrupt boundary localization and pitch perception.

In this paper, we present ROSVOT, a ROBust Singing VOice Transcription 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, we demonstrate 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. VO-CANO (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. 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. Gu et al. (2023) attempts to achieve robust transcription through self-supervised pre-training and multimodal injection.

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 \( y_{bd} = [y_{bd}^{1}, y_{bd}^{2}, ..., y_{bd}^{T}] \), where \( y_{bd}^{t} = 1 \) if the state is boundary at timestep \( t \) and 0 is not. The silence note has a pitch value of 0. Notice that \( \sum_{t=1}^{T} y_{bd} = \text{len}(p) - 1 \), where \( p = [p^1, p^2, ..., p^L] \) 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 \( p \).

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 \( y_{bd} \) 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 \( y_{bd} \) and a Gaussian filter \( G[n] \):

\[
G[n] = \begin{cases} 
\frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{n^2}{2\sigma^2}}, & \text{if } |n| \leq \left\lfloor \frac{W_G}{2} \right\rfloor \\
0, & \text{otherwise} 
\end{cases} 
\]  

(1)

\[
\tilde{y}_{bd} = y_{bd} * \left( \frac{G[n]}{\max(G[n])} \right) 
\]  

(2)

where the filter \( G[\tau] \) is normalized before convolution, so the middle of each soft label remains 1. \( W_G \) 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 \( \eta \) 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:

\[
\tilde{y} = y + \eta \times \frac{\text{RMS}(y/10^{\text{SNR}/20})}{\text{RMS}(\eta)} 
\]  

(3)

\[
\tilde{X} = \mathcal{F}(\tilde{y}) 
\]  

(4)

where \( \mathcal{F}(\cdot) \) is Mel-spectrogram extraction operation, \( \text{RMS}(\cdot) \) is root-mean-square operation, and \( \tilde{X} \) 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 $y_{wbd}$ has the same form as note boundaries $y_{bd}$, involving silence or “NONE” words. The regulation is 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. Moreover, 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 $y_{bd}$ 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 $y_{bd}$, 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 $L_B$ is applied to train note segmentation. The details of loss functions are listed in Appendix D, similarly hereinafter.

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. Also, 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), $L_{FC}$, to focus more on hard samples.

3.6.2 Pitch Prediction

For pitch value prediction $D_p$, 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 $W_A \in \mathbb{R}^{C \times H}$, $S = \sigma(ZW_A)$, 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: $Z'_i = Z_i \otimes S_i$ 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 $s \in \mathbb{R}^{T}$ by averaging along the head dimension.

Subsequently, we use the note boundary sequence $y_{bd}$ to segment $Q$ along the time axis, resulting in a group sequence $G = \{G^1, G^2, ..., G^L\}$ with length of $L$, number of notes. Each group (or segment) $G^i$ contains $l_i$ consecutive 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 $y^j_{bd} = 1$. We also do the same for the averaged weights $s$: $G^i_s = \{s^{j+1}, s^{j+2}, ..., s^{j+l_i}\}$. For each group, we compute a weighted average $z^i$:

$$
z^i = \frac{\sum_{s_k=1}^{l_i} Q^{j+k}}{\sum_{s_k=1}^{l_i} s^{j+k}}$$  \hfill (5)

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

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 $L = L_B L_{FC} + \lambda_P L_p$ is controlled by balancing parameters $\lambda_B$, $\lambda_{FC}$, and $\lambda_P$.

In the inference stage, firstly we compute the boundary probability $\sigma(y_{bd})$ and use a threshold $\mu$ to decide the boundary state. That is, a note boundary exists at time step $t$ if $\sigma(y_{bd}) > \mu$, 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 $\mu$ can control the granularity of generated notes. In other words, a lower $\mu$ may result in more fine-grained and subdivided pitches, while a higher one ignores small fluctuations. This is because a lower $\mu$ 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 $S$. 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).

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 G.

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 $D_1$. $D_1$ 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. Two 1% subsets of MUSAN noise corpus are also isolated. The details of data collection are listed in Appendix C.

Implementation and Training The sample rate of waveforms is 24 kHz. 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 $\mu$ is set to 0.8. More details are listed in Appendix B.

Evaluation We utilize the mir_eval library (Raffel et al., 2014) for performance evaluation. Specifically, we apply the metrics proposed in (Molina et al., 2014): COn (correct onset), COff (correct offset), ConP (correct onset and pitch), ConPOff (correct onset, pitch, and offset). An average overlap ratio (AOR) is also calculated for correctly transcribed note duration, and a raw pitch accuracy (RPA) for overall perception performance. Given GT and corresponding predicted notes, their overlap ratio (OR) is defined as the ratio between the duration of the time segment in which the two notes overlap and the time segment spanned by the two notes combined. The AOR is given by the mean OR computed over all matching GT and predicted notes. For RPA, we transform the GT and predicted note events into frame-level sequences and compute matching scores.

We remove silence notes and designate the boundaries that enclose each note as onset and offset for the evaluation of ROSVOT. 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), an 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 $E_W$.

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 major scores are listed here, and the complete scores are listed in Appendix F.

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 validated 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 $E_W$, the performance does not decline significantly.
Table 1: Evaluation results of AST systems.

<table>
<thead>
<tr>
<th>Method</th>
<th>COn (F) ↑ clean</th>
<th>COff (F) ↑ noisy</th>
<th>COnPOff (F) ↑ clean</th>
<th>Pitch (AOR) ↑ clean</th>
<th>Melody (RPA) ↑ clean</th>
</tr>
</thead>
<tbody>
<tr>
<td>TONY</td>
<td>67.5</td>
<td>49.2</td>
<td>57.8</td>
<td>47.0</td>
<td>43.9</td>
</tr>
<tr>
<td>VOCANO</td>
<td>75.8</td>
<td>64.7</td>
<td>71.2</td>
<td>66.1</td>
<td>50.2</td>
</tr>
<tr>
<td>MusicYOLO (Yong et al., 2023)</td>
<td>82.2</td>
<td>79.7</td>
<td>81.7</td>
<td>76.5</td>
<td>58.9</td>
</tr>
<tr>
<td>$\mathcal{M}$ (conformer)</td>
<td>92.0</td>
<td>88.5</td>
<td>91.4</td>
<td>89.7</td>
<td>65.8</td>
</tr>
<tr>
<td>$\mathcal{M}$ (conv)</td>
<td>91.6</td>
<td>91.5</td>
<td>92.6</td>
<td>92.6</td>
<td>70.9</td>
</tr>
<tr>
<td>$\mathcal{M}$ (w/o wbd)</td>
<td>91.3</td>
<td>91.1</td>
<td>91.8</td>
<td>91.2</td>
<td>70.2</td>
</tr>
<tr>
<td>$\mathcal{M}$ (w/o noise)</td>
<td>93.8</td>
<td>90.9</td>
<td>94.2</td>
<td>91.5</td>
<td>76.4</td>
</tr>
<tr>
<td>$\mathcal{M}$ (w/ $E_W$)</td>
<td>93.3</td>
<td>93.5</td>
<td>93.2</td>
<td>92.9</td>
<td>77.1</td>
</tr>
<tr>
<td>$\mathcal{M}$ (ours)</td>
<td><strong>94.0</strong></td>
<td><strong>93.8</strong></td>
<td><strong>94.5</strong></td>
<td><strong>94.4</strong></td>
<td><strong>77.4</strong></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Rate</th>
<th>Step (ms)</th>
<th>COn</th>
<th>COff</th>
<th>COnPOff</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10.7</td>
<td>92.7</td>
<td>92.4</td>
<td>70.7</td>
</tr>
<tr>
<td>4</td>
<td>21.3</td>
<td>93.9</td>
<td>93.6</td>
<td>73.6</td>
</tr>
<tr>
<td>8</td>
<td>42.7</td>
<td><strong>94.4</strong></td>
<td>94.1</td>
<td>76.8</td>
</tr>
<tr>
<td>16</td>
<td>85.3</td>
<td>94.0</td>
<td><strong>94.5</strong></td>
<td><strong>77.4</strong></td>
</tr>
<tr>
<td>32</td>
<td>170.7</td>
<td>94.3</td>
<td>94.1</td>
<td>77.2</td>
</tr>
</tbody>
</table>

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 $D_P$ 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 $D_2$, 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 C.

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. For a more intuitive demonstration, we record Comparative Mean Opinion Scores (CMOS) and discuss the results in Appendix F.

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 annotations with different ratios are mixed into $D_1$ 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: 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.

<table>
<thead>
<tr>
<th>Model</th>
<th>RPA</th>
<th>MOS-P</th>
<th>MOS-Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>GT</td>
<td>96.1</td>
<td>4.02±0.05</td>
<td>4.04±0.06</td>
</tr>
<tr>
<td>S(large)</td>
<td>45.2</td>
<td>3.36±0.12</td>
<td>3.45±0.09</td>
</tr>
</tbody>
</table>

Table 4: Results of cross-lingual SVS generalization.

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 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, denoted as $S$(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 slightly improves, as the model has been exposed to a sufficient variety of pronunciation styles and singing patterns. This indicates that ROSVOT provides an opportunity for SVS models to scale up, which could be beneficial for large-scale singing voice pre-training or zero-shot SVS.

4.4.3 Cross-lingual Generalization

We further investigate the zero-shot cross-lingual capability of ROSVOT, and explore the feasibility of generalizing a Mandarin SVS model to English. We use the same ROSVOT model combined with $E_W$ trained with M4Singer and test it on TONAS (Gómez and Bonada, 2013), a flamenco a cappella singing corpus. The quantitative results are listed in Table 6 in Appendix F. Performance is degraded, and we believe it may be due to the flamenco singing style (which has rich techniques like appoggiatura) and the unseen language, since the model is trained with Mandarin pop songs.

For SVS, we finetune the pre-trained model $S$(large) on the English corpus $\mathcal{D}_2$, a 6-hour dataset transcribed automatically using MFA and the AST model above. Note that since the word duration information is obtained using MFA, we drop $E_W$ and directly generate word-note synchronized transcriptions. We finetune both stages 1 and 2 of RMSSinger for 100k steps. The evaluation results are listed in Table 4.

Although the performance is not superior, the results still demonstrate certain cross-lingual capabilities. There are vast differences in pronunciation styles.
ation rules and phonetic characteristics between English and Mandarin, since they belong to two distinct language families. If a model possesses a certain capability to transfer from Mandarin to English, it essentially demonstrates a degree of general cross-linguistic ability. We will leave elaborate validations and evaluations of general cross-lingual capabilities for future work.

5 Conclusion
In this paper, we introduced ROSVOT, the first robust AST model that ultimately serves SVS. We leveraged a multi-scale architecture to achieve a balance between coarse-grained note modeling and fine-grained segmentation. An attention-based decoder with dynamic weight was devised for pitch regression. Additionally, we established a comprehensive pipeline for SVS training. Experimental results revealed that our model achieved the best performance under either clean or noisy environments. Annotating and incorporating larger datasets improved the SVS model’s performance, indicating the capability of practical annotation of ROSVOT.

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.

Acknowledgement
This work was supported in part by the National Natural Science Foundation of China under Grant No.62222211.

References


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 $E_W$ 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 $E_W$, followed by fine-tuning it with M4Singer and $D_1$.

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 $W_G$. The temperature parameters $T_1$ and $T_2$ are set to 0.2 and 0.01. To balance the various objectives, we set $\lambda_B$, $\lambda_{FC}$, and $\lambda_P$ to 1.0, 3.0, and 1.0. For inference, the boundary threshold $\mu$ is set to 0.8. The hyperparameters $\alpha$ and $\gamma$ 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.

We train the AST 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.

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 upsam-
pled 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.](image)

### C Data

We recruit 12 professional singers (8 female, 4 male) to record $D_1$ and 8 singers (5 female, 3 male) for $D_2$. Each singer was compensated at an hourly rate of $600. Singers were informed that the recordings were for scientific research use. For $D_1$, we have hired music experts to manually annotate the note and word information. Each annotator was compensated at an hourly rate of $20. Participants were informed that the data would be used for scientific research. For $D_2$, we automatically annotate the words and notes through an ASR model (Radford et al., 2023), MFA, and the proposed AST model. The length of $D_1$ is 20.9 hours and $D_2$ is 6 hours. We use all the datasets under license CC BY-NC-SA 4.0.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mel Encoder Kernel</td>
<td>3</td>
</tr>
<tr>
<td>Mel Encoder Layers</td>
<td>2</td>
</tr>
<tr>
<td>Mel Encoder Hidden</td>
<td>256</td>
</tr>
<tr>
<td>Condition Encoder Pitch Embedding</td>
<td>300</td>
</tr>
<tr>
<td>Condition Encoder UV Embedding</td>
<td>3</td>
</tr>
<tr>
<td>Condition Encoder WBD Embedding</td>
<td>3</td>
</tr>
<tr>
<td>Condition Encoder Encoder Kernel</td>
<td>3</td>
</tr>
<tr>
<td>Condition Encoder Encoder Layers</td>
<td>1</td>
</tr>
<tr>
<td>Condition Encoder Encoder Hidden</td>
<td>256</td>
</tr>
<tr>
<td>U-Net Kernel</td>
<td>3</td>
</tr>
<tr>
<td>U-Net Enc &amp; Dec Layers</td>
<td>4</td>
</tr>
<tr>
<td>U-Net Downsampling Rate</td>
<td>16</td>
</tr>
<tr>
<td>U-Net Enc &amp; Dec Hidden</td>
<td>256</td>
</tr>
<tr>
<td>Conformer Kernel</td>
<td>9</td>
</tr>
<tr>
<td>Conformer Heads</td>
<td>4</td>
</tr>
<tr>
<td>Conformer Layers</td>
<td>2</td>
</tr>
<tr>
<td>Conformer Attention Hidden</td>
<td>256</td>
</tr>
<tr>
<td>Conformer FFN Hidden</td>
<td>1024</td>
</tr>
<tr>
<td>Total Number of Parameters</td>
<td>12M</td>
</tr>
</tbody>
</table>

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

### D Objectives

The binary cross-entropy (BCE) loss applied to train the note segmentation stage:

$$L_B = \frac{1}{T} \sum y_{bd} \sigma(\hat{y}_{bd}) - \frac{1}{T} \sum (1 - y_{bd}) \ln(1 - \sigma(\hat{y}_{bd} / T_1)) + (1 - \hat{y}_{bd}) \ln(1 - \sigma(\hat{y}_{bd} / T_1))$$  \hspace{1cm} (6)

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

To tackle the imbalance problem, we employ a focal loss (Lin et al., 2017), $L_{FC}$, to focus more on hard samples:

$$p_t = y_{bd} \sigma(\hat{y}_{bd}) + (1 - y_{bd}) (1 - \sigma(\hat{y}_{bd}))$$  \hspace{1cm} (7)

$$\alpha_t = \alpha y_{bd} + (1 - \alpha) (1 - y_{bd})$$  \hspace{1cm} (8)

$$L_{FC} = \frac{1}{T} \sum \alpha_t (1 - p_t)^\gamma \BCE(y_{bd}, \hat{y}_{bd})$$  \hspace{1cm} (9)

where $\alpha$ is a hyperparameter controlling weight of positive samples, and $\gamma$ controls balance between easy and hard samples.
The cross-entropy (CE) loss used to train the pitch prediction stage:

\[
\mathcal{L}_P = -\frac{1}{L} \sum_{i=1}^{L} \sum_{c=1}^{P} \tilde{p}_c^i \ln \left( \frac{\exp(\tilde{p}_c^i / T_2)}{\sum_{k=1}^{C} \exp(\tilde{p}_k^i / T_2)} \right)
\]  

(10)

where \( T_2 \) is the temperature hyperparameter.

E  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.

F  Extensional Experiments

F.1  Additional AST Results

The additional experimental results are listed in Table 9 and Table 10, where the former is under a clean environment and the latter is noisy.

F.2  Additional Comparisons

We perform additional out-of-domain (OOD) tests on ROSVOT and compare with (Gu et al., 2023), which focuses more on multimodality, but still has noticeable audio-only AST capability. We directly run OOD tests on datasets MIR-ST500 (Wang and Jang, 2021) and TONAS (Gómez and Bonada, 2013), where the former is used by (Gu et al., 2023) as an in-domain (ID) set and the latter also an OOD set. The ROSVOT model in this test is still trained with only M4Singer. The results are listed in Table 6 (the results of (Gu et al., 2023) are copied from their original paper).

From the results, we can see that ROSVOT outperforms (Gu et al., 2023) in an OOD setting (i.e., on TONAS) in terms of ConPOff and ConP, but falls short in Con. We believe this is because (Gu et al., 2023) incorporated English data in training, which has a relatively closer pattern to flamenco singing than M4Singer. ROSVOT also underperforms facing MIR-ST500. There are three possible reasons:

- Since each song in MIR-ST500 spans several minutes and needs to be source-separated first, it contains non-negligible un-voiced sections. We do not know whether (Gu et al., 2023) segments the voiced parts according to some rules. Hence, we transcribe the whole song and compute the scores at once, resulting in a smaller proportion of positive samples of on/offsets due to longer silences, hereby influencing the performance. Another noteworthy point is that ROSVOT is able to load a 5-minute song into one single 2080ti GPU and transcribe it at once, demonstrating considerable efficiency.
- The source-separation result influences the performance. The separator recognizes harmonies as vocals, thus resulting in polyphonic singing voices. We focus on solo vocal note transcription and harmonies are not considered noise in our settings. ROSVOT tends to transcribe vocal harmonies when the main vocal is silent, degrading the accuracy.
- MIR-ST500 is an ID test set to the model of (Gu et al., 2023), so it should have an advantage.

In conclusion, we believe that ROSVOT demonstrates comparable or superior performance. Considering the total number of parameters, ROSVOT shows considerable efficiency and the capability to process long sequences.

F.3  Additional SVS Results

In Table 3, the quality of test samples could be very similar; without a specific evaluation objective, evaluators may struggle to determine differences in quality. Therefore, we conduct comparative evaluations and record CMOS scores of two typical data combinations compared to 100% real \( D_1 \). We believe CMOS scores, rating from -2 to 2, allow the disregard of unrelated factors. The results are listed in Table 7.

The results demonstrate a trend that, when the ratio of pseudo annotations in the training set increases, the quality of samples generated using real annotation inputs decreases, while that using pseudo annotations also increases. This may indicate that there are certain biases in the annotations within datasets, and the AST model could learn...
Method & MIR-ST500 & TONAS & M (ours) \\ 
(Gu et al., 2023) & 78.1 & 70.0 & 52.8 & 64.4 & 36.9 & 24.1 \\ 
$\mathcal{M}$ (ours) & 72.1 & 65.9 & 47.4 & 55.7 & 49.4 & 30.0 \\

Table 6: Additional comparison with (Gu et al., 2023) on OOD tests.

<table>
<thead>
<tr>
<th>Real</th>
<th>R. Size</th>
<th>Pseudo</th>
<th>P. Size</th>
<th>CMOS-P</th>
<th>CMOS-Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1 \times 10%$</td>
<td>2.1</td>
<td>$D_1 \times 90%$</td>
<td>18.8</td>
<td>-0.26</td>
<td>+0.08</td>
</tr>
<tr>
<td>-</td>
<td>0.0</td>
<td>$D_1$</td>
<td>20.9</td>
<td>-0.43</td>
<td>+0.21</td>
</tr>
</tbody>
</table>

Table 7: Results of comparative evaluation. Evaluators are requested to provide relative scores on each sample compared with reference samples which are generated using the SVS model trained with 100% real $D_1$.

its own bias, which creates a domain gap of the annotation styles among different datasets.

### G Low-resource Scenarios

![Figure 5: Injection of self-supervised features.](image)

We use M4Singer as the training set and test the model on $D_1$. 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.

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 pretrained 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 $M_{(ssl)}$.

The results are listed in Table 8. 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 $M$ exhibits a decline in performance, while $M_{(ssl)}$ experiences a comparatively smaller decrease.
<table>
<thead>
<tr>
<th>Method</th>
<th>COOn</th>
<th>COOff</th>
<th>COOnP</th>
<th>COOnPOff</th>
<th>PRF</th>
<th>PRF</th>
<th>PRF</th>
<th>PRF</th>
<th>AOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>TONY</td>
<td>65.0</td>
<td>70.2</td>
<td>67.5</td>
<td></td>
<td>46.1</td>
<td>45.4</td>
<td>45.7</td>
<td>46.1</td>
<td>42.1</td>
</tr>
<tr>
<td>VOCANO</td>
<td>73.7</td>
<td>78.1</td>
<td>75.8</td>
<td></td>
<td>56.3</td>
<td>59.9</td>
<td>58.0</td>
<td>48.4</td>
<td>52.1</td>
</tr>
<tr>
<td>MusicYOLO (Yong et al., 2023)</td>
<td>84.1</td>
<td>80.4</td>
<td>82.2</td>
<td></td>
<td>68.4</td>
<td>66.1</td>
<td>67.2</td>
<td>61.3</td>
<td>56.8</td>
</tr>
<tr>
<td>M (conformer)</td>
<td>93.7</td>
<td>90.0</td>
<td>92.0</td>
<td></td>
<td>74.4</td>
<td>71.2</td>
<td>72.7</td>
<td>65.3</td>
<td>66.3</td>
</tr>
<tr>
<td>M (conv)</td>
<td>94.5</td>
<td>93.2</td>
<td>93.8</td>
<td></td>
<td>79.9</td>
<td>79.3</td>
<td>79.6</td>
<td>76.0</td>
<td>76.8</td>
</tr>
<tr>
<td>M (w/o noise)</td>
<td>92.1</td>
<td>94.5</td>
<td>93.3</td>
<td></td>
<td>79.2</td>
<td>80.9</td>
<td>80.1</td>
<td>75.9</td>
<td>78.4</td>
</tr>
<tr>
<td>M (ours)</td>
<td>92.4</td>
<td>96.2</td>
<td>94.0</td>
<td></td>
<td>79.5</td>
<td>81.2</td>
<td>80.3</td>
<td>76.4</td>
<td>78.9</td>
</tr>
</tbody>
</table>

Table 9: Additional results of AST systems. These results are from clean inputs.

<table>
<thead>
<tr>
<th>Method</th>
<th>COOn</th>
<th>COOff</th>
<th>COOnP</th>
<th>COOnPOff</th>
<th>PRF</th>
<th>PRF</th>
<th>PRF</th>
<th>PRF</th>
<th>AOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>TONY</td>
<td>48.6</td>
<td>49.8</td>
<td>49.2</td>
<td></td>
<td>32.3</td>
<td>36.4</td>
<td>34.2</td>
<td>25.8</td>
<td>31.7</td>
</tr>
<tr>
<td>VOCANO</td>
<td>63.7</td>
<td>65.8</td>
<td>64.7</td>
<td></td>
<td>48.1</td>
<td>50.8</td>
<td>49.4</td>
<td>42.9</td>
<td>44.0</td>
</tr>
<tr>
<td>MusicYOLO (Yong et al., 2023)</td>
<td>81.3</td>
<td>78.2</td>
<td>79.7</td>
<td></td>
<td>62.1</td>
<td>59.2</td>
<td>60.6</td>
<td>53.9</td>
<td>49.9</td>
</tr>
<tr>
<td>M (conformer)</td>
<td>89.7</td>
<td>87.2</td>
<td>88.5</td>
<td></td>
<td>69.2</td>
<td>66.7</td>
<td>67.9</td>
<td>63.4</td>
<td>60.8</td>
</tr>
<tr>
<td>M (conv)</td>
<td>92.4</td>
<td>90.6</td>
<td>91.5</td>
<td></td>
<td>78.0</td>
<td>80.3</td>
<td>79.1</td>
<td>73.7</td>
<td>75.9</td>
</tr>
<tr>
<td>M (w/o noise)</td>
<td>91.9</td>
<td>89.9</td>
<td>90.9</td>
<td></td>
<td>78.8</td>
<td>77.4</td>
<td>78.1</td>
<td>70.2</td>
<td>70.0</td>
</tr>
<tr>
<td>M (w/o wbd)</td>
<td>92.4</td>
<td>94.6</td>
<td>93.5</td>
<td></td>
<td>83.2</td>
<td>83.8</td>
<td>83.5</td>
<td>76.3</td>
<td>77.7</td>
</tr>
<tr>
<td>M (ours)</td>
<td>92.9</td>
<td>94.7</td>
<td>93.8</td>
<td></td>
<td>83.2</td>
<td>84.4</td>
<td>83.8</td>
<td>76.7</td>
<td>77.3</td>
</tr>
</tbody>
</table>

Table 10: Additional results of AST systems. These results are from noisy inputs.