TOWARDS IMPROVING HARMONIC SENSITIVITY AND PREDICTION STABILITY FOR SINGING MELODY EXTRACTION

Keren Shao* **Ke Chen*** **Taylor Berg-Kirkpatrick** **Shlomo Dubnov**
University of California San Diego
{k5shao, knutchen, tberg, sdubnov}@ucsd.edu

ABSTRACT

In deep learning research, many melody extraction models rely on redesigning neural network architectures to improve performance. In this paper, we propose an input feature modification and a training objective modification based on two assumptions. First, harmonics in the spectrograms of audio data decay rapidly along the frequency axis. To enhance the model's sensitivity on the trailing harmonics, we modify the Combined Frequency and Periodicity (CFP) representation using discrete z-transform. Second, the vocal and non-vocal segments with extremely short duration are uncommon. To ensure a more stable melody contour, we design a differentiable loss function that prevents the model from predicting such segments. We apply these modifications to several models, including MSNet, FTANet, and a newly introduced model, PianoNet, modified from a piano transcription network. Our experimental results demonstrate that the proposed modifications are empirically effective for singing melody extraction.

1. INTRODUCTION

Singing melody extraction is a challenging task that aims to detect and identify the fundamental frequency (F0) of singing voice in polyphonic music recordings. This task is more complicated than the monophonic pitch detection task due to the presence of various instrumental accompaniments and background noises, making it more difficult to accurately extract the singing melody. Singing melody extraction is not only crucial for music analysis by itself, but also has many downstream applications, such as cover song identification [1], singing evaluation [2], and music recommendation [3].

Deep neural networks have been widely adopted in the singing melody extraction task to produce promising performance in terms of extraction accuracy. Early models [4–6] simply leveraged deep neural networks (DNN) and convolutional neural networks (CNN) [7]. In more recent models, musical and structural priors were incorporated to improve performance. These include MSNet [8] with a vocal detection component at the encoder-decoder bottleneck, joint detection model [9] setting up an auxiliary network, and TONet [10] with tone-octave predictions. Additionally, models can capture frequency relationships better with multi-dilation [11], cross-attention networks [12], graph-based neural networks [13], or harmonic constant-Q transform (HCQT) [14].

One of our observations relates to the input representations of the models, which play an important role in affecting the extraction performance. Timbre, which is closely related to harmonics, is one of the key components that helps models distinguish the vocal from other instruments. When the CFP representation [15] is chosen as the input representation, its second feature, the generalized cepstrum, allows the model to learn the strength of harmonics of any given fundamental frequency in a localized manner. However, in music, the harmonics of a single sound usually decays rapidly along the frequency axis (detail in section 2.1), which can pose a challenge for the model to distinguish sounds that only differ significantly at the trailing harmonics.

The transformation from the spectrum to the generalized cepstrum in CFP is a Fourier transform, and hence mostly captures the first few peaks with large magnitudes. As a result, this representation is not helpful in sensing the trailing harmonics. This motivates us to apply a different transformation function that produces a generalized cepstrum with better harmonics sensitivity.

Another observation relates to the vocal detection component. Extremely short vocal segments surrounded by non-vocal regions, and vice versa, rarely occur since vocalists typically sing a melody for at least half a second or rest for at least a few hundred milliseconds. Threshold-based removal [16], mean or median filtering [17, 18] and Viterbi-based smoothing [19, 20] are frequently used to address the problem. When they are implemented alongside a network-based algorithm, however, the network remains unaware of our smoothing intention and configuration. To investigate whether such awareness can increase the prediction performance, we derive a differentiable loss component that specifically penalizes spurious short-term predictions of these kinds during training, thus potentially guiding the model to produce consistently stable predictions.

In this paper, we propose two techniques that attempt

*The first two authors have equal contribution.

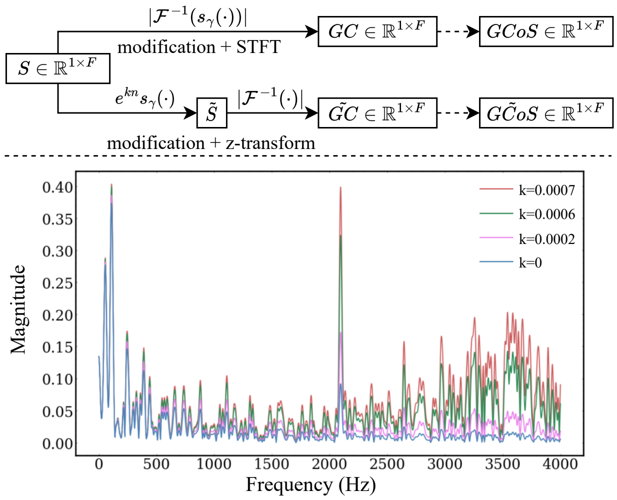

Figure 1. Top: the transformation pipeline of the original CFP representation, and our proposed z-CFP representation. Bottom: modified Spectrum $\tilde{S}$ with different growing rates k applied. Note that the original CFP corresponds to the case of $k = 0$.

to improve the two concerns mentioned above, namely the harmonic sensitivity and the prediction stability of singing melody extraction models. Our contributions are as follows:

- We propose to use exponentially growing sinusoids along the frequency axis to transform the spectrum into the generalized cepstrum of the CFP representation. This approach is equivalent to taking a z-transform instead of Fourier transform, which increases the harmonic sensitivity of the input.
- We design a differentiable loss function as part of the training objective to teach the network to avoid predicting unrealistically short sequences of vocal and non-vocal at the voice detection bin.
- We evaluate our techniques by applying them on several melody extraction models. Additionally, we adapt PianoNet [21], originally developed for piano transcription, into the melody extraction task. Experimental results demonstrate state-of-the-art performance of our improved models.

2. METHODOLOGY

In this section, we introduce three main parts of our methodology. First, we propose a modified CFP representation, z-CFP, to enhance the harmonic sensitivity of the network input. Second, we introduce extraction models used for evaluating our techniques, namely MSNet, FTANet, and PianoNet. Third, we propose a new loss function as part of training objective to improve the prediction stability of models.

2.1 z-CFP Representation for Harmonic Sensitivity

Our input representation of audio data is a modified version of the CFP representation. A CFP representation $X \in \mathbb{R}^{3\times T\times F}$ contains three features, with T the length of time frames and F the number of frequency bins. **At each time slice**, it contains: (1) a power spectrum $S \in \mathbb{R}^{1\times F}$; (2) a generalized cepstrum $GC \in \mathbb{R}^{1\times F}$; and (3) a generalized cepstrum of spectrum $GCoS \in \mathbb{R}^{1\times F}$,

As illustrated in the upper part of Figure 1, the standard CFP generation process begins by computing the frame-wise spectrum of an input audio waveform using short-time Fourier transform (STFT). We then obtain the magnitude of each spectrum, which serves as the first feature of CFP, denoted as S. To derive the second feature, we compute the generalized cepstrum using the following equation:

$$GC = |\mathcal{F}^{-1}(s_\gamma(S))| = |\mathcal{F}(s_\gamma(S))| \tag{1}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denotes the Fourier transform and its inverse, $s_\gamma : \mathbb{R} \to \mathbb{R}$ is an element-wise applied, logarithm-like modification function as described in [15], and the absolute value sign represents an element-wise complex norm operation. The second equality comes directly from the fact that norm of a complex number equals to that of its conjugate.

As mentioned in the introduction, GC is not sensitive to the trailing harmonic dynamics, as it mostly captures the first few peaks with large magnitudes. Since the harmonics decay rapidly along with the frequency axis, we shall revert the decay to better preserve such dynamics. In other words, instead of applying complex sinusoids $\sum_n s_\gamma(S[n])e^{-iwn}$ as in Fourier transform (n is the entry of frequency bins in S), we apply growing complex sinusoids $\sum_n s_\gamma(S[n])e^{(k-iw)n}$, where $k \in \mathbb{R}$ and $k > 0$. This is equivalent to taking a discrete z-transform $\sum_n s_\gamma(S[n])z^{-n}$, where $z = e^{iw-k}$.

In the actual implementation, k is manually assigned and fixed across different w. Therefore, as illustrated in Figure 1, we can separate the computation of k part and w part as follows:

$$\tilde{S}[n] = e^{kn}s_\gamma(S[n]) \text{ for } \forall n \tag{2}$$

$$\tilde{GC} = |\mathcal{F}^{-1}(\tilde{S})| = |\mathcal{F}(\tilde{S})| \tag{3}$$

In the lower part of Figure 1, we present $\tilde{S}$ of an audio waveform with different values of k. We can observe that the harmonics of $\tilde{S}$ at the tail gets amplified so that the subsequent Fourier transform can better capture their dynamics. While we observe some amplifications of harmonics at frequencies other than the fundamental frequencies, their magnitudes are always smaller than those of nearby fundamental frequencies. Therefore, they pose no sufficient distraction for the extraction model, as long as the chosen k is not too large. In our experiments, we set $k = 0.0006$.

We then generate the generalized cepstrum of spectrum $\tilde{GCoS}$ from cepstrum $\tilde{GC}$ the same way as in the original CFP. Finally, **each time slice** of our modified CFP representation $\tilde{X} \in \mathbb{R}^{3\times T\times F}$ consists of $\{S, \tilde{GC}, \tilde{GCoS}\}$ with log-scaled frequency axis. For the rest of the paper, we denote it z-CFP.

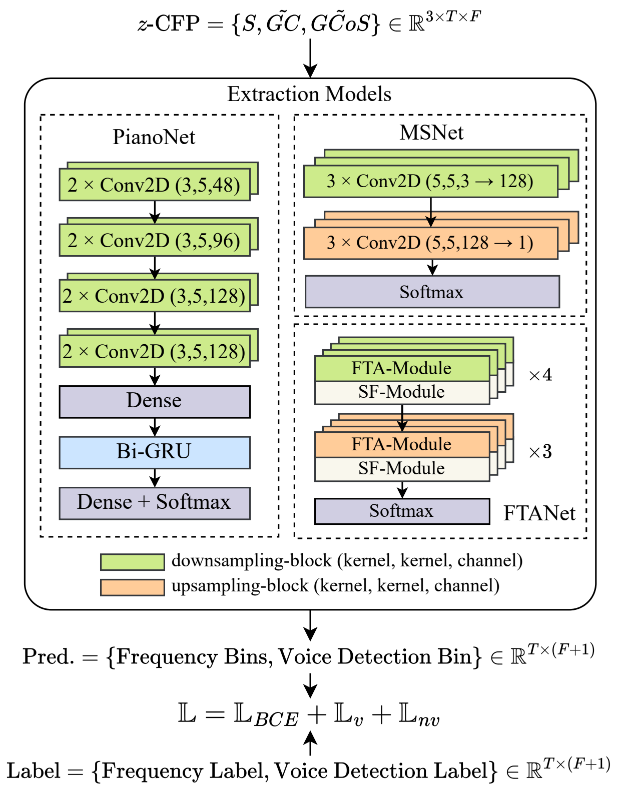
z-CFP $= \{S, \tilde{GC}, \tilde{GCoS}\} \in \mathbb{R}^{3\times T\times F}$
Extraction Models
PianoNet
2 × Conv2D (3,5,48)
2 × Conv2D (3,5,96)
2 × Conv2D (3,5,128)
2 × Conv2D (3,5,128)
Dense
Bi-GRU
Dense + Softmax
MSNet
3 × Conv2D (5,5,3 → 128)
3 × Conv2D (5,5,128 → 1)
Softmax
FTA-Module
SF-Module
×4
FTA-Module
SF-Module
×3
Softmax
FTANet
downsampling-block (kernel, kernel, channel)
upsampling-block (kernel, kernel, channel)
Pred. = {Frequency Bins, Voice Detection Bin} $\in \mathbb{R}^{T\times(F+1)}$
$\mathbb{L} = \mathbb{L}_{BCE} + \mathbb{L}_v + \mathbb{L}_{nv}$
Label = {Frequency Label, Voice Detection Label} $\in \mathbb{R}^{T\times(F+1)}$

Figure 2. The model architecture. Note that we choose only one of the three extraction models at a time.

2.2 Model Architecture

Our extraction models are referred from three state-of-the-art (SoTA) models, MSNet [8], FTANet [12], and PianoNet [21]. Different from MSNet and FTANet, PianoNet is the SoTA model of piano transcription. Given its superior performance on piano transcription, we incorporate a sub-network of PianoNet into singing melody extraction, as we hypothesize that it may also yield good results for melody extraction.

MSNet contains a 3-layer encoder, a 3-layer decoder, and a bottleneck module. The channel size is shifted as $3 \rightarrow 32 \rightarrow 64 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 1$. The bottleneck module maps the encoder output to a 1-channel featuremap for voice detection. All 2D-convolutional layers come with (5×5) kernel size. FTANet contains a 4-layer encoder, a 3-layer decoder, and a 4-layer bottleneck module. Both encoder and decoder contain FTA-modules and SF-modules to process the audio latent features. The channel size is shifted from 3 to 128, then back to 1. More specifications of MSNet and FTANet can be found in their papers [8, 12].

The PianoNet we use for this task is modified from a sub-network of [21]. It starts with four convolutional blocks, each block containing two 2D-convolutional layers with kernel sizes (3, 5) and (3, 3) respectively, a batch normalization layer and a ReLU activation. Then it is followed by bidirectional-GRU and softmax layers, with dense layers as transitions. The layer bias is turned off for all layers before the Bi-GRU.

Figure 2 illustrates a more detailed structure of the three extraction models. Following the pipeline, we first process the audio waveform into z-CFP representations. Then we feed them into the extraction model, which produces output feature maps $\tilde{Y} \in \mathbb{R}^{T\times(F+1)}$. The additional one feature along the frequency axis denotes the voice detection bin output. It is then compared against the ground truth label $Y \in \mathbb{R}^{T\times(F+1)}$, through the loss function introduced in the following section.

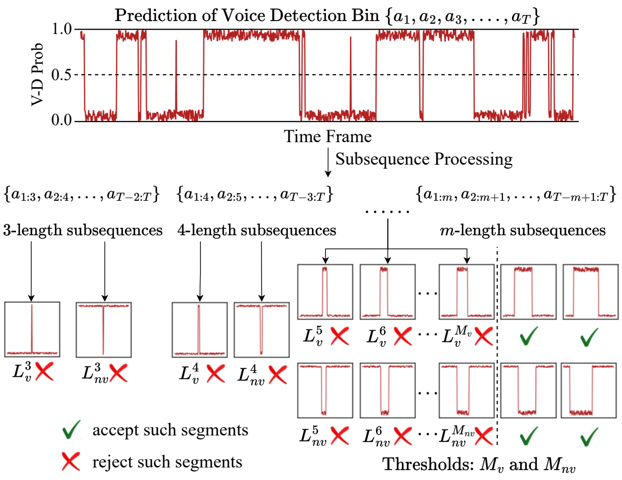
Prediction of Voice Detection Bin $\{a_1, a_2, a_3, \ldots, a_T\}$
V-D Prob
1.0
0.5
0.0
Time Frame
Subsequence Processing
$\{a_{1:3}, a_{2:4}, \ldots, a_{T-2:T}\}$
$\{a_{1:4}, a_{2:5}, \ldots, a_{T-3:T}\}$
......
$\{a_{1:m}, a_{2:m+1}, \ldots, a_{T-m+1:T}\}$
3-length subsequences
4-length subsequences
m-length subsequences
L_v^3 ✗
L_{nv}^3 ✗
L_v^4 ✗
L_{nv}^4 ✗
L_v^5 ✗ L_v^6 ✗ ··· $L_v^{M_v}$ ✗ ✓ ✓
L_{nv}^5 ✗ L_{nv}^6 ✗ ··· $L_{nv}^{M_{nv}}$ ✗ ✓ ✓
✓ accept such segments
✗ reject such segments
Thresholds: M_v and M_{nv}

Figure 3. The illustration of how we perform the loss functions $\mathbb{L}_v$ and $\mathbb{L}_{nv}$ on the subsequences of the voice detection prediction. Each loss components L are used to give large penalties (i.e., ✗) to certain types of subsequences.

2.3 Loss Function for Prediction Stability

We add two differentiable training objectives, $\mathbb{L}_v$ and $\mathbb{L}_{nv}$, to the conventional binary cross entropy loss $\mathbb{L}_{BCE}$ to teach the extraction model to avoid unrealistically short vocal and non-vocal sequences at **the vocal detection bin**. Since the design for these two cases are symmetric, we first introduce the loss object $\mathbb{L}_v$, for the vocal case.

As shown on the top of Figure 3, the predictions at the vocal detection bin is a time series $\{a_1, a_2, ..., a_T\}$. First, since our training objectives are dealing with certain types of short burst segments of vocal and non-vocal, we extract all possible subsequences, with stride 1. For example, for 3-length subsequences we have $\{a_{1:3}, a_{2:4}, ..., a_{T-2:T}\}$, and similarly $\{a_{1:4}, a_{2:5}, ..., a_{T-3:T}\}$ for subsequences of length 4, etc.

Second, to simplify the problem a bit at the beginning, we assume that the voice detection output is binary valued $a \in \{0, 1\}$. Formally, we do not want "sharp-burst" sequences inside the following set:

$$B_v = \bigcup_{m=3}^{M_v} \{a_1...a_m | a_1 = a_m = 0, a_i = 1 \text{ for } \forall i \neq 1, m\} \quad (4)$$

where M_v is a hyperparameter threshold, above which the duration of vocal segments becomes reasonable. Figure 3 illustrates examples of "sharp-burst" sequences in B_v (and B_{nv}) as red segments inside black-border boxes.

Suppose $m = 3$, all possible binary sequences are $\{000, 001, 010, 011, 100, 101, 110, 111\}$ and $010 \in B_v$. To make the model avoid predicting the short burst vocal segment, i.e., 010, we construct a polynomial objective that can fulfill the goal by satisfying the following:

$$L_v^3(a_1a_2a_3) = \begin{cases} 1 & \text{where } a_1a_2a_3 = 010 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

A decent choice will then be

$$L_v^3(a_1a_2a_3) = (1-a_1)a_2(1-a_3) \tag{6}$$

which can be easily extended to sequences with longer length m.

$$L_v^4(a_1a_2a_3a_4) = (1-a_1)a_2a_3(1-a_4)$$

$$\vdots$$

$$L_v^m(a_1...a_m) = (1-a_1)(1-a_m)\prod_{i=2}^{m-1} a_i \tag{7}$$

However, there is a small caveat in this extension when we move back from binary values to probability values $a \in [0,1]$. For example, our loss component will be having trouble capturing sequences like $\{0.1, 0.4, 0.6, 0.1\}$ and $\{0.1, 0.6, 0.4, 0.1\}$ as both L_v^3 and L_v^4 result in relatively small values. However, we observe that polynomials $(1-a_1)(1-a_2)a_3(1-a_4)$ and $(1-a_1)a_2(1-a_3)(1-a_4)$ respectively works better than our original L_v^4, but still insufficient to work standalone.

Since none of the polynomials above gives high values to sequences outside of B_v in 4-length, a simple solution would be to redefine L_v^4 to be the sum of all such polynomials:

$$L_v^4 = (1-a_1)(1-a_4)(a_2a_3 + a_2(1-a_3) + (1-a_2)a_3)$$

$$\vdots$$

$$\begin{aligned} L_v^m &= (1-a_1)(1-a_m) \sum_{\substack{c_1...c_m \in \{0,1\}^m \\ \text{at least one } c_i \neq 0}} \prod_{i=2}^{m-1} a_i^{c_i}(1-a_i)^{1-c_i} \\ &= (1-a_1)(1-a_m)(1 - \prod_{i=2}^{m-1}(1-a_i)) \end{aligned} \tag{8}$$

This redefined loss L_v allows better recognition of the bad sequences mentioned above while not falsely flagging sequences outside of B_v. Furthermore, when dealing with longer sequences, for example $\{0.1, 0.9, ..., 0.9, 0.1\}$ with increasingly many 0.9s in the middle, the original L_v's output quickly diminishes while the redefined L_v does not.

This redefined objective does come with a small side effect, as it over-counts the shorter bad sequences. For example, $(0.1, 0.9, 0.1, 0.1)$ now gets a high loss value not only in L_v^3, but also in L_v^4. However, we believe this side effect does not have significant impact as it does not matter whether neural network decides to stop producing shorter bad sequences or longer bad sequences first.

A further improvement is to pass the value of L_v^m into the S-curve function:

$$L_v^m \leftarrow \frac{(L_v^m)^r}{(L_v^m)^r + (1-L_v^m)^r} \tag{9}$$

where $r \in \mathbb{R}$ and $r > 1$. It will amplify those sequences that receive loss values closer to 1 and suppress those sequences with loss values closer to 0.

Finally, for each $m \in [3, M_v]$, we compute L_v^m across all m-length subsequences in the model's output. The aggregated loss function $\mathbb{L}_v$ is then computed by concatenating all these L_v^m arrays and taking the average.

Now analogously, assuming non-vocal sequences beyond length M_{nv} become reasonable, we can perform the same analysis on the following set of sequences:

$$B_{nv} = \bigcup_{m=3}^{M_{nv}} \{a_1...a_m | a_1 = a_m = 1, a_i = 0 \text{ for } \forall i \neq 1, m\} \tag{10}$$

and consequently obtain $\mathbb{L}_{nv}$. Practically, L_{nv}^m of any sequence $a_1...a_m$ can be computed as L_v^m of the flipped sequence $b_1...b_m$, where $b_i = 1 - a_i$ for all $i \in \{1..m\}$. Our final loss function will then be:

$$\mathbb{L} = \mathbb{L}_{BCE} + \mathbb{L}_v + \mathbb{L}_{nv} \tag{11}$$

3. EXPERIMENTS

3.1 Datasets and Experiment Setup

For the training data, we complied with the setting of [10, 12] and chose all 1000 Chinese pop songs from MIR-1K [1] and 35 vocal tracks from MedleyDB [22]. For the testing data, we chose 12 tracks in ADC2004 and 9 tracks in MIREX05 [2]. We also selected 12 tracks from MedleyDB that are disjoint from those already used for training.

For the signal processing part, we used 8000 Hz sampling rate to process audio tracks. We use a window size of 768, a hop size of 80 to compute the STFT of audio tracks. Note that the time resolution of our labels is 0.01 seconds, and this hop size was chosen to match that. Then, when creating z-CFP representations, we set the time dimension of the representation to be $T = 128$, or 1.28 seconds, and the number of frequency bins $F = 360$, or 60 bins per octave across 6 octaves. The start and stop frequencies are 32.5 Hz and 2050 Hz. Hence, the input shape becomes $X \in \mathbb{R}^{3\times128\times360}$ and the output/label shape becomes $Y \in \mathbb{R}^{128\times361}$.

Within the extra loss component, we set the duration threshold of vocal segments $M_v = 30$ (0.3 seconds), the duration threshold of non-vocal segments $M_{nv} = 7$ (0.07 seconds), and the S-curve exponent parameter $r = 5$.

For the training hyperparameters, we use a batch size of 10, the Adam optimizer [23] with a fixed learning rate of 1×10^{-4}. The maximum training epoch is 500. During the evaluation, we use the standard metrics of the singing melody extraction task, namely, voice recall (VR), voicing false alarm (VFA), raw pitch accuracy (RPA), raw chroma accuracy (RCA), and overall accuracy (OA) from the `mir_eval` library [24]. Following the convention of this task, overall accuracy (OA) is regarded as the most important metric. All models are trained and tested in NVIDIA RTX 2080Ti GPUs and implemented in PyTorch [3].

[1] http://mirlab.org/dataset/public/MIR-1K.zip

[2] https://labrosa.ee.columbia.edu/projects/melody/

[3] https://pytorch.org/

Dataset	ADC 2004					MIREX 05					MEDLEY DB				
Metrics	VR	VFA↓	RPA	RCA	OA	VR	VFA↓	RPA	RCA	OA	VR	VFA↓	RPA	RCA	OA
PianoNet	87.21	14.62	84.28	84.30	84.48	91.98	6.14	86.54	86.55	89.19	69.38	13.74	61.81	62.80	73.70
PianoNet + z-CFP	88.25	**7.58**	84.87	84.93	86.27	**93.44**	6.21	86.78	86.79	89.33	68.76	**11.91**	62.22	63.10	**74.80**
PianoNet + 3 point median	87.33	14.58	84.35	84.38	84.55	92.08	6.15	86.60	86.62	89.23	69.49	13.77	61.86	62.86	73.71
PianoNet + 7 point median	87.58	14.53	84.46	84.48	84.65	92.47	6.14	86.78	86.8	89.35	69.71	13.83	61.92	62.91	73.71
PianoNet + 15 point median	89.13	14.21	84.89	84.91	85.06	93.27	6.58	86.82	86.84	89.21	70.31	14.43	61.91	62.90	73.42
PianoNet + $\{\mathbb{L}_v, \mathbb{L}_{nv}\}$	**90.92**	13.58	**86.06**	**86.12**	86.13	91.87	**5.79**	87.50	87.50	**89.94**	**71.16**	15.77	**63.66**	**64.81**	73.66
PianoNet + z-CFP + $\{\mathbb{L}_v, \mathbb{L}_{nv}\}$	90.50	7.99	85.76	85.82	**86.92**	92.84	6.39	**87.57**	**87.59**	89.76	68.88	12.29	62.05	62.91	74.53
MSNet	89.78	23.12	80.83	81.60	80.10	84.85	**11.44**	77.76	78.09	81.68	53.49	**9.41**	46.90	48.24	68.15
MSNet + z-CFP + $\{\mathbb{L}_v, \mathbb{L}_{nv}\}$	**90.61**	**14.62**	**81.96**	**82.57**	**82.59**	**88.38**	14.85	**80.83**	**81.01**	**82.39**	**62.95**	14.60	**53.60**	**55.31**	**69.07**
FTANet	81.26	**2.70**	77.17	77.36	80.89	87.34	**5.11**	81.56	81.61	86.40	62.44	10.41	55.94	56.58	72.30
FTANet + z-CFP + $\{\mathbb{L}_v, \mathbb{L}_{nv}\}$	**90.29**	10.83	**85.06**	**85.19**	**85.82**	**90.50**	6.63	**83.94**	**83.99**	**87.36**	**63.71**	**9.35**	**56.32**	**57.29**	**73.02**

Table 1. Ablation studies on ADC2004, MIREX05 and MedleyDB testsets. Baselines use CFP as the input representation and $\mathbb{L}_{BCE}$ as the loss function. $\{\mathbb{L}_v, \mathbb{L}_{nv}\}$ denotes the use of our proposed loss function in section 2.3. Among median filter sizes in the range $[3, 100] \subset \mathbb{Z}$, 3 point works best for MedleyDB, 7 point works best for MIREX 05, and 15 point works best for ADC 2004. But they neither significantly outperform our proposed loss component in any single dataset, nor uniformly outperform in all three datasets.

3.2 Ablation Study

We choose three extraction models, namely MSNet [8], FTANet [12], and PianoNet [21], to evaluate our z-transform and loss functions. We conducted ablation studies and presented the results in Table 1. We re-trained these models from scratch, and the results are largely consistent with the original reports of [8, 10, 12]. The option z-transform denotes the use of z-CFP representations. Note that $\{\mathbb{L}_v, \mathbb{L}_{nv}\}$ in the table denote the use of loss functions to address short burst segments of vocal and non-vocal. Due to the page limitation, we present a detailed ablation study on PianoNet while ablating MSNet and FTANet in an all-or-nothing fashion.

From Table 1 we can clearly observe decent performance of both z-CFP and $\{\mathbb{L}_v, \mathbb{L}_{nv}\}$ when added to the PianoNet, MSNet, and FTANet. Among these results, the addition of loss functions $\{\mathbb{L}_v, \mathbb{L}_{nv}\}$ increases the overall accuracy while improving the VR, RPA, and RCA. The median filter postprocessing [18] is used as a comparison. Since our loss component focuses on the vocal detection, we took the pitches predicted by median filters only when the original predictions are non-vocal. Further, to ensure fairness, we optimized the filter size against each single dataset within the range $[3, 100] \subset \mathbb{Z}$ and listed the evaluation results of those optimal ones. As we can see in Table 1, none of these median filters outperforms our loss component in a consistent manner, nor do they obtain considerable margins in any single dataset.

The z-CFP also increases several metrics, especially either VR or VFA, on each dataset. This indicates that by preserving more dynamics in the high frequency bins, the model can distinguish different sounds better and consequently improve the extraction performance. Also, note that unlike TONet [10] and JDC [9], which achieved this through model design or music inductive bias, this technique relies solely on the inherent characteristics of the data.

When we incorporate both techniques into the extraction models, we observe a promising increase in each metric compared to the original models. However, we notice that some models with both techniques carried do not yield better performance than the models carrying only one of the techniques. These models appear to be an averaging weighting or an ensemble of models improved with either technique, implying better generalization.

3.3 Comprehensive Performance Comparison

Table 2 presents the results as we compare our best model, i.e., PianoNet with z-transform and $\{\mathbb{L}_v, \mathbb{L}_{nv}\}$, with other SoTA models. Among these SoTAs, there are two models with "*", indicating that these are only partial comparisons. For SpecTNT [25], since there is no official open-source implementation, we report its results based on our own re-implementation. For H-GNN [13], we directly copied its reported performance from the original paper.

From Table 2, our improved PianoNet with z-transform and $\{\mathbb{L}_v, \mathbb{L}_{nv}\}$ yield the best OA performance over all datasets, the best RPA and RCA on ADC 2004 and MIREX 05 datasets. We do note, despite the use of the extra loss component, that our model's VFA is not necessarily the smallest. This is because the extra loss component only targets a particular type of false positive, and is not meant to minimize the false positive rate in general. For example, sometimes the network's vocal to non-vocal transition happens later than the reference labels. In this case, since the vocal sequence itself lasts long enough, the extra loss component will not mark this type of false positives. Addressing this type of errors is potentially a future work.

Another thing we found is that the PianoNet, as one of SoTAs in the piano transcription task and ported by us to the melody extraction task in this paper, has already yields very high performance on MIREX 05 dataset. This indicates that there may exist more powerful network architectures for this task yet to be explored. Additionally, it is

Dataset	ADC 2004				
Metrics	VR	VFA↓	RPA	RCA	OA
MCDNN [4]	65.0	10.5	61.6	63.1	66.4
DSM [14]	89.2	51.3	75.4	77.6	69.8
MSNet [8]	89.8	23.1	80.8	81.6	80.1
FTANet [12]	81.3	**2.7**	77.2	77.4	80.9
TONet [10]	**91.8**	17.1	82.6	82.9	82.6
SpecTNT* [25]	85.4	8.2	83.5	83.6	85.0
H-GNN* [13]	89.2	21.3	84.8	86.1	83.9
Ours	90.5	8.0	**85.7**	**85.8**	**86.9**
Dataset	MIREX 05				
Metrics	VR	VFA↓	RPA	RCA	OA
MCDNN [4]	66.5	**4.6**	64.1	64.4	75.4
DSM [14]	91.4	45.3	75.7	77.0	68.4
MSNet [8]	84.8	11.4	77.8	78.1	81.7
FTANet [12]	87.3	5.1	81.6	81.6	86.4
TONet [10]	91.6	8.5	83.8	84.0	86.6
SpecTNT* [25]	82.2	8.7	77.4	77.5	82.5
H-GNN* [13]	**93.2**	21.7	85.2	86.4	81.3
Ours	92.8	6.4	**87.6**	**87.6**	**89.8**
Dataset	MEDLEY DB				
Metrics	VR	VFA↓	RPA	RCA	OA
MCDNN [4]	37.4	**5.3**	34.2	35.3	62.3
DSM [14]	**86.6**	44.3	**70.2**	**72.4**	64.8
MSNet [8]	53.5	9.4	46.9	48.2	68.1
FTANet [12]	62.4	10.4	55.9	56.6	72.3
TONet [10]	64.2	12.5	56.6	58.0	71.6
SpecTNT* [25]	62.7	18.8	54.7	56.4	63.9
H-GNN* [13]	71.7	21.6	61.2	65.8	67.9
Ours	68.9	12.3	62.1	62.9	**74.5**

Table 2. The comprehensive performance comparison among our improved models and current baselines.

noteworthy that our proposed PianoNet architecture has a small number of parameters (5.5 million), which is comparable with MCDNN (5.6 million), FTANet (3.4 million) and far less than TONet (152 million). This demonstrates its potential in practical applications where computational resources are limited. Again, as demonstrated in Table 1, our techniques could help models other than PianoNet achieve higher performance than their original versions.

3.4 Loss Value and Extraction Visualization

To empirically verify if applying the polynomial loss functions $\mathbb{L}_v$ and $\mathbb{L}_{nv}$ could reduce the voice detection errors, i.e., short burst segments of vocal and non-vocal, we visualize two types of plots in Figure 4. The top three plots demonstrate the loss values of L_v^{30} between the original extraction models and the improved models with $\mathbb{L}_v$ and $\mathbb{L}_{nv}$, across the entire MIREX05 dataset (i.e., we concatenate all tracks in the dataset). We see that cases in which the improved models' prediction receive loss values close to 1 diminishes comparing to those of the original models. This phenomenon implies that after applying $\mathbb{L}_v$ and $\mathbb{L}_{nv}$, the chance of models to predict short burst segments significantly reduces.

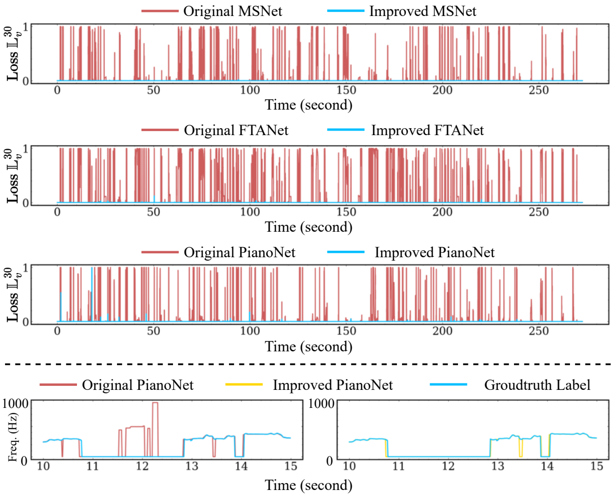

Figure 4. The effect of applying the loss $\mathbb{L}_v$ and $\mathbb{L}_{nv}$. The top three plots are values of L_v^{30} across the entire MIREX05 dataset. The bottom two plots are one 5-sec MIREX05 predictions.

The pair of plots in the last row compares the prediction performance of PianoNets, trained without and with the extra loss components, on a zoomed-in section of MIREX05. Note that the original PianoNet has a short burst non-vocal segment in between the 10th second and 11th second. Further, it has a considerable number of short burst vocal segments around the 12th second. Once trained with the extra loss components, these issues are resolved. Also note that both the original version and the improved version make a mistake in between the 13th and the 14th second. This is because the length of that non-vocal transition is greater than our threshold M_{nv}, which ends up not triggering $\mathbb{L}_{nv}$. All these observations further verify the effectiveness of our proposed loss components.

4. CONCLUSION

In this paper, we propose two techniques to respectively utilize the two assumptions we made for improving the performance of singing melody extraction models. First, comparing to Fourier transform, the use of z-transform in generating cepstrum allows the network to better recognize the strength of harmonics of any fundamental frequencies. Empirically, while the trailing harmonics of those frequencies that do not actually appear in the audio also get elevated, the benefit of the technique is greater than its setback. Second, our extra loss components make the network less prone to predict vocal and non-vocal sequences are unreasonably short, while not affecting the network's overall accuracy due to its differentiability. Along with different extraction models, we achieve better performance when compared to their original version and other state-of-the-art models. We regard these two techniques as decent improvements on singing melody extraction models.

5. ACKNOWLEDGMENTS

We would like to thank the Institute for Research and Coordination in Acoustics and Music (IRCAM) and Project REACH: Raising Co-creativity in Cyber-Human Musicianship for supporting this project. This project has received funding from the European Research Council (ERC REACH) under the European Union's Horizon 2020 research and innovation programme (Grant Agreement #883313).

6. REFERENCES

[1] X. Du, K. Chen, Z. Wang, B. Zhu, and Z. Ma, "Bytecover2: Towards dimensionality reduction of latent embedding for efficient cover song identification," in *Proc. ICASSP*, 2022, pp. 616–620.

[2] N. Zhang, T. Jiang, F. Deng, and Y. Li, "Automatic singing evaluation without reference melody using bidense neural network," in *Proc. ICASSP*, 2019, pp. 466–470.

[3] K. Chen, B. Liang, X. Ma, and M. Gu, "Learning audio embeddings with user listening data for content-based music recommendation," in *Proc. ICASSP*, 2021, pp. 3015–3019.

[4] S. Kum, C. Oh, and J. Nam, "Melody extraction on vocal segments using multi-column deep neural networks," in *Proc. ISMIR*, 2016, pp. 819–825.

[5] S. Li, "Vocal melody extraction using patch-based cnn," in *Proc. ICASSP*, 2018, pp. 371–375.

[6] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," in *Proc. ICASSP*. IEEE, 2018, pp. 161–165.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, 1998.

[8] T.-H. Hsieh, L. Su, and Y.-H. Yang, "A streamlined encoder/decoder architecture for melody extraction," in *Proc. ICASSP*, 2019, pp. 156–160.

[9] S. Kum and J. Nam, "Joint detection and classification of singing voice melody using convolutional recurrent neural networks," *Applied Sciences*, 2019.

[10] K. Chen, S. Yu, C. Wang, W. Li, T. Berg-Kirkpatrick, and S. Dubnov, "Tonet: Tone-octave network for singing melody extraction from polyphonic music," in *Proc. ICASSP*, 2022, pp. 626–630.

[11] P. Gao, C. You, and T. Chi, "A multi-dilation and multi-resolution fully convolutional network for singing melody extraction," in *Proc. ICASSP*, 2020, pp. 551–555.

[12] S. Yu, X. Sun, Y. Yu, and W. Li, "Frequency-temporal attention network for singing melody extraction," in *Proc. ICASSP*, 2021, pp. 251–255.

[13] S. Yu, X. Chen, and W. Li, "Hierarchical graph-based neural network for singing melody extraction," in *Proc. ICASSP*, 2022, pp. 626–630.

[14] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, "Deep salience representations for f0 estimation in polyphonic music." in *Proc. ISMIR*, 2017, pp. 63–70.

[15] L. Su and Y.-H. Yang, "Combining spectral and temporal representations for multipitch estimation of polyphonic music," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 23, no. 10, pp. 1600–1612, 2015.

[16] R. M. Bittner, J. Salamon, J. J. Bosch, and J. P. Bello, "Pitch contours as a mid-level representation for music informatics," in *Audio engineering society conference: 2017 AES international conference on semantic audio*, 2017.

[17] J. Salamon and E. Gómez, "Melody extraction from polyphonic music signals using pitch contour characteristics," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 6, pp. 1759–1770, 2012.

[18] S. Rosenzweig, F. Scherbaum, and M. Müller, "Detecting stable regions in frequency trajectories for tonal analysis of traditional georgian vocal music." in *Proc. ISMIR*, 2019, pp. 352–359.

[19] M. Mauch and S. Dixon, "pyin: A fundamental frequency estimator using probabilistic threshold distributions," in *Proc. ICASSP*. IEEE, 2014, pp. 659–663.

[20] J. J. Bosch and E. Gómez Gutiérrez, "Melody extraction based on a source-filter model using pitch contour selection," in *Proceedings SMC 2016. 13th Sound and Music Computing Conference*, 2016.

[21] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, "High-resolution piano transcription with pedals by regressing onset and offset times," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 29, pp. 3707–3717, 2021.

[22] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, "Medleydb: A multitrack dataset for annotation-intensive mir research." in *Proc. ISMIR*, 2014, pp. 155–160.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proc. ICLR*, 2014.

[24] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, L. Dawen, D. P. Ellis, and C. C. Raffel, "mir_eval: A transparent implementation of common mir metrics," in *Proc. ISMIR*, 2014, pp. 367–372.

[25] W. T. Lu, J. Wang, M. Won, K. Choi, and X. Song, "Spectnt: a time-frequency transformer for music audio," in *Proc. ISMIR*, 2021, pp. 396–403.