A RELATED WORK DETAILS

A.1 Music Source Separation

Music source separation (MSS) is a crucial task in music information retrieval (MIR), involving the decomposition of music into its constitutive components, such as isolating vocals, bass, and drums [51]. When focusing specifically on clean vocals and accompaniment without further separating the accompaniment into individual instruments, it becomes a singing voice separation task [45], a universal and specialized form of MSS. Various deep learning methods address the MSS task, broadly categorized into three types: the traditional methods, the deep learning-based methods, and the side information informed methods.

A.1.1 Traditional Methods. Traditional methods are based on digital signal processing and mathematical statistics, include Independent Components Analysis (ICA) [55], Principal Component Analysis (PCA) [62], and Non-negative Matrix Factorization (NMF) [1]. For instance, ICA is initially employed to solve the blind source separation task [27], akin to the MSS task. The PCA-based method [24] utilizes robust principal component analysis to represent accompaniment through a separated low-rank matrix and singing voices through a separated sparse matrix. And the NMF-based methods [1] model each source with a dictionary, capturing source signals within the non-negative span of this dictionary. Although these traditional methods are interpretable and accomplish the music source separation task to some extent, they often lack the ability to distinguish between different instruments.

A.1.2 Deep Learning-Based Methods. Deep learning-based methods fall into three categories: methods in the frequency domain, methods in the time domain, and hybrid methods in both domains. For example, U-Net [29], Spleeter [21], CWS-PResUNet [40], and ResUNetDecouple+ [36] belong to methods in the frequency domain, which process mixture music in the frequency domain (e.g., spectrogram) to predict masks or spectrograms of target sources. Alternatively, Demucs [10], Wave-U-Net [57], and its follow-ups [41, 47] are the methods in the time domain. They directly process raw audio using one-dimensional convolutional networks to predict target sources in the time domain. Additionally, other methods such as KUIELAB-MDX-Net [33], Hybrid Demucs [9], and HT Demucs [53] are hybrid methods in both domains. They combine features from both domains for improved performance of music source separation. However, the above methods focus only on the features extracted from raw audio, ignoring the important role of auxiliary information such as melody, rhythm, lyrics, and so on in the music source separation task.

A.1.3 Side Information Informed Methods. Side information informed methods leverage additional information, such as lyrics, music scores (pitches), or spatial details, to improve the performance of music source separation. For instance, JOINT3 [54] utilizes phoneme-level lyrics alignment to improve the performance of music source separation. SPAIN-NET [48] and Soundprism [12, 14] use music scores and spatial information to enhance the performance of music source separation. However, these methods treat side information as auxiliary features and lack the capability to perform highly related tasks in music information retrieval simultaneously.

A.2 Pitch Estimation

Pitch estimation (PE) is a fundamental task in MIR, playing a crucial role in various downstream applications. Following previous studies [18, 32, 61], we default to using pitch estimation to refer to single pitch estimation (SPE), unless explicitly stated otherwise in this paper. Then we will introduce both single pitch estimation (SPE) and multi-pitch estimation (MPE) tasks in detail.

A.2.1 Single Pitch Estimation (SPE). The SPE task involves predicting no more than one pitch at any timestamp. It can be further categorized into SPE from clean music and SPE from mixture music.

For SPE from clean music, there are two primary methods: the heuristic-based methods and the data-driven methods. The heuristic-based methods, such as ACF [13], YIN [8], SWIPE [5] and pYIN [42], leverage candidate-generating functions to predict pitches. In contrast, the data-driven methods, such as CREPE [32], DeepF0 [56] and HARMOF0 [61], employ supervised training of models for SPE. While these methods achieve a good performance on clean music, they prove to be less effective in the mixture music due to their limited robustness to accompaniments.

For SPE from mixture music, there are mainly two approaches. The first approach is pipeline methods, which involves using MSS models (e.g., Spleeter [21] and U-Net [29]) to extract target sources from the mixture music, and then using PE models to estimate the pitch of target sources. These models are trained separately, resulting in a mismatch between the data distributions at training and testing times, which limits the performance of PE from mixture music. The second approach is end-to-end methods (e.g., DSM-HCQT [3], CNN-Raw [11] and JDC [37]), which are designed to directly estimate pitches from mixture music. However, these methods are also limited in performance due to the presence of other sources in the mixture music.

A.2.2 Multi Pitch Estimation (MPE). The MPE task entails predicting multiple pitches at any timestamp, also known as music transcription in the field of music information retrieval. Given its increased complexity compared to SPE, our focus in this paper is exclusively on the single pitch estimation task. Methods for music transcription are usually classified into two categories: frame-level transcription methods and note-level transcription methods.

The frame-level transcription methods, such as OAF [19], ADSRNet [30] and Non-Saturating GAN [31], employing CNN and LSTM to predict pitches for each frame. These frame-level pitches are then transformed into sequential notes, achieving the MPE task. On the contrary, the note-level transcription methods, such as seq-to-seq [20], MT3 [17] and EBPT [63] treat notes as events, predicting note-level pitch results and achieving the MPE task at the note level. However, the above MPE methods often focus on specific instruments and may perform inadequately on the SPE task, lacking robustness across different instruments.

A.3 Joint Learning For MSS and PE

With the development of joint learning, several research tasks in MIR have shown the potential for mutual improvement through multi-tasks joint learning. Among these tasks, music source separation and pitch estimation are closely related, leading to the exploration of methods that take advantage of their mutually beneficial relationship. Several existing methods have attempted to exploit the mutually beneficial relationship between music source separation and pitch estimation tasks.

For example, JDC [37] is a joint learning approach that addresses voice detection and pitch estimation. However, it mainly employs voice detection as an auxiliary task for accompaniment processing. Other methods, such as [25] and [6], utilize transcription as an auxiliary task, incorporating joint transcription and source separation training for a limited number of instruments. HS-W$_p$ [44] and Joint-UNet [28] represent joint learning methods specifically designed for music source separation and pitch estimation tasks. Furthermore, models like MSI-DIS [38] and JOINTIST [7] aim to use a unified model for both music source separation and transcription tasks. It is worth noting that these models often require training on datasets containing both pitch labels and target sources. While these existing joint learning approaches provide valuable insights, practical challenges arise when combining music source separation and pitch estimation tasks, especially in scenarios where training data with both pitch labels and target sources is limited.

B EXPERIMENTAL SETUP DETAILS

B.1 Datasets

To compare with previous MSS models and PE models fairly, we use three public datasets for our experiments: MIR-1K [22], MedleyDB [4], MIR_ST500 [58] and MUSDB18 [50].

MedleyDB [4] dataset consists of 122 songs, 108 of the original multi-tracks include melody annotations. Based on the melody definition "The f0 curve of the predominant melodic line drawn from a single source", the melody annotation is the pitch of the stem with the most predominant melodic source. Thus, the MedleyDB dataset has both the target sources and corresponding pitches, making it the fully-labeled dataset. This dataset has various musical instruments, including vocals, guitar, violin, dizi, and so on. The total duration of vocals music data is about 3.21 h, while the total duration of each other instruments is less than 0.69 h. We focus only on vocal music in this paper since the amount of data for each other musical instruments is limited.

MIR-1K [22] dataset is designed for singing voice separation and contains 1000 song clips extracted from 110 karaoke songs sung by researchers from the MIR lab. The dataset provides both the mixture track and the clean vocals track, as well as pitch labels for the vocal parts, making it a fully-labeled dataset. The total length of MIR-1K is 133 minutes, and each clip ranges from 4 to 13 seconds.

MIR_ST500 [58] is a singing voice transcription dataset containing 500 Chinese pop songs. It provides the YouTube URL of the mixture music and the note labels of vocal parts, which can be used for PE from mixture music. Thus, it is considered a single-labeled dataset. This dataset is divided into a train dataset (400 songs) and a test dataset (100 songs), with the total duration of about 32 hours.

MUSDB18 [50] is a dataset of music source separation consisting of 150 full-length music tracks of different European and American genres, such as pop, rap, and heavy metal. Each track is composed of isolated drums, bass, vocals, and other stems. Thus, it is considered a single-labeled dataset. The dataset is divided into a train folder with 100 songs and a test folder with 50 songs, with a total duration of about 10 hours.

B.2 Evaluation Metrics

The following evaluation metrics are used to evaluate the performance of the music source separation and pitch estimation tasks, as described in previous studies on music source separation [22, 36, 45] and pitch estimation [32, 56, 61], respectively. These metrics are computed using the mir_eval [49] library:

Raw Pitch Accuracy (RPA) [32] computes the proportion of melody frames in the reference for which the predicted pitch is within ±50 cents of the ground truth pitch.

Raw Chroma Accuracy (RCA) [32] computes the raw pitch accuracy after mapping the estimated and reference frequency sequences onto a single octave. It measures the raw pitch accuracy ignoring the octave errors.

Signal-to-Distortion Ratio (SDR) [36] measures the quality of the predicted sources with respect to the original target sources. It is defined as follows:

$$SDR(s, \hat{s}) = 10 \times \log_{10} \frac{||s||^2}{||\hat{s} - s||^2} \tag{16}$$

where $\hat{s}$ is the predicted sources and s is the target sources. A higher SDR indicates better separation results, and vice versa. A perfect separation would result in infinite SDR.

Global Normalized Signal-to-Distortion Ratio (GNSDR) [45] is calculated as follows:

$$GNSDR = \frac{\sum_{i=1}^{i=N} l_i NSDR(s, \hat{s}, x)}{\sum_{i=1}^{i=N} l_i} \tag{17}$$

where i is the index of a song, N is the total number of songs, l_i is the length of the ith song, and $NSDR(s, \hat{s}, x)$ is the normalized SDR. The $NSDR(s, \hat{s}, x)$ [45] is defined as follows:

$$NSDR(s, \hat{s}, x) = SDR(s, \hat{s}) - SDR(s, x) \tag{18}$$

where x is the mixture music. The NSDR is the improvement in SDR between the mixture and the predicted sources.

B.3 Implementation Details

The raw audio is sampled at 16kHz and then transformed into a spectrogram using the short-time Fourier transform (STFT) with a Hann window size of 2048 and a hop length of 320 (20ms). We use the librosa [43] and torchlibrosa [35] to perform the audio processing. During training of the model-agnostic joint learning (MAJL) framework, we use a batch size of 16 and the Adam optimizer [34]. The learning rate is initialized to 0.001 and is reduced by 0.98 of the previous learning rate every 10 epochs. In our framework, there are mainly three hyper-parameters, ω_{noise}, $upper\_bound$ and threshold (th). The hyper-parameters $upper\_bound$ and ω_{noise} only used for the naive DWHS in additional methods (Section C), where $upper\_bound$ ranges from 1 to 10 and ω_noise ranges from 0 to 1. While the hyper-parameter threshold (th) is used to filter pseudo labels, ranging from 0.5 to 1.

Each training audio is divided into segments of 2.56 seconds. For the MIR-1K [22] dataset, we randomly split the dataset into training (80%) and testing (20%) sets. We treat MIR_ST500 [58] and MUSDB18 [50] datasets as single-labeled datasets since they

Table 6: Analysis for different cases in DWHS and corresponding weights calculated by the naive DWHS. The $\hat{y_t}$ is defined as $max(y) \times max(\hat{y}) + (1 - max(y)) \times (1 - max(\hat{y}))$. The $upper\_bound$ and ω_{noise} are two hyper-parameters for the naive DWHS. The $Clamp(1/\hat{y_t}, 1, upper\_bound)$ represents restrict $1/\hat{y_t}$ to be between 1 and $upper\_bound$.

Case	predicted_source2Pitch	target_source2Pitch	Analysis			Naive DWHS	
			MSS Module	PE Module	Data	ω_{mss}	ω_{pe}
1	Correct	Correct	✓	✓	✓	1	1
2	Correct	Incorrect	✓	✓	×	1	$0 \le \omega_{noise} < 1$
3	Incorrect	Correct	×	✓	✓	$Clamp(1/\hat{y_t}, 1, upper\_bound)$	1
4	Incorrect	Incorrect	✓	×	✓	1	$Clamp(1/\hat{y_t}, 1, upper\_bound)$

lack the target sources and pitch labels, respectively. The splitting way for MIR_ST500 and MUSDB18 datasets is introduced in [58] and [50], respectively. During experiments, we only consider the target source of vocals due to the lack of fully-labeled data from other sources such as bass and drums.

B.4 Comparison Systems

We compare MAJL with several existing methods, including end-to-end, pipeline, and joint learning methods. For end-to-end methods, we chose CNN-Raw [11] and JDC [37], as they achieved the best performance in the pitch estimation task among all end-to-end methods. For pipeline methods, we consider U-Net [29] and ResUNetDecouple+ [36] as music source separation models because U-Net is the base model for many music source separation models, and ResUNetDecouple+ is the state-of-the-art model for the music source separation task. For the pitch estimation task, we use CREPE [32] and HARMOF0 [61], as they are the most common and state-of-the-art models. Finally, for joint learning methods, we select HS-W$_p$[44] and S→P→S→P[28] as the baselines since they are the latest joint learning methods for music source separation and pitch estimation tasks.

C ADDITION METHODS

As illustrated in Section 4.2.2, the most direct method involves setting different weights for different cases as outlined in Table 6. We refer to this method as naive DWHS, which utilizes two hyper-parameters (ω_{noise} and $upper_bound$) to assign weights to hard samples and noisy samples, as specified in Table 6.

Starting with default weights set at 1, for *Case 1*, where there are no issues with the MSS Module, PE Module, or the music data, the default weights remain unchanged. In *Case 2*, involving music data with noisy pitch labels, we decrease the weight of such samples by adjusting the weight (ω_{noise}) within the range of 0 to 1. This adjustment aims to mitigate the impact of noisy labels. In *Case 3*, where the music data is hard for the MSS task, we increase the weight of such samples in the MSS task by setting the weight to $1/\hat{y_t}$. The $\hat{y_t}$ is defined as $max(y) \times max(\hat{y}) + (1 - max(y)) \times (1 - max(\hat{y}))$, where y is the ground truth of pitch results and $\hat{y}$ is the predicted value. For *Case 4*, where the music data is hard for the PE task, we increase the weight of such samples in the PE task by setting the weight to $1/\hat{y_t}$. Besides, we constrain $1/\hat{y_t}$ to fall within the range of 1 to $upper\_bound$ to avoid invalid weights that are too large. The weights (ω_{mss} and ω_{pe}) for different cases are shown in Table 1. With the naive DWHS, the loss function of stage I is written as:

$$\mathcal{L}_{total} = \omega_{mss} \times \mathcal{L}_{mss} + \omega_{pe} \times \mathcal{L}_{pe} \quad (19)$$

And the loss function of stage II is written as:

$$\mathcal{L}_{total} = \text{confi}_{mss} \times \omega_{mss} \times \mathcal{L}_{mss} + \text{confi}_{pe} \times \omega_{pe} \times \mathcal{L}_{pe} \quad (20)$$

where confi$_{mss}$ and confi$_{pe}$ are the same as those in Eq. 8.

D ADDITIONAL EXPERIMENTAL RESULTS

D.1 Comparison of naive DWHS and DWHS

To compare the effectiveness of naive DWHS and DWHS, we conduct an ablation study on the MIR-1K dataset using ResUNetDecouple+ as the MSS Module and CREPE as the PE Module, consistent with the previous experiment in Section 6.1.

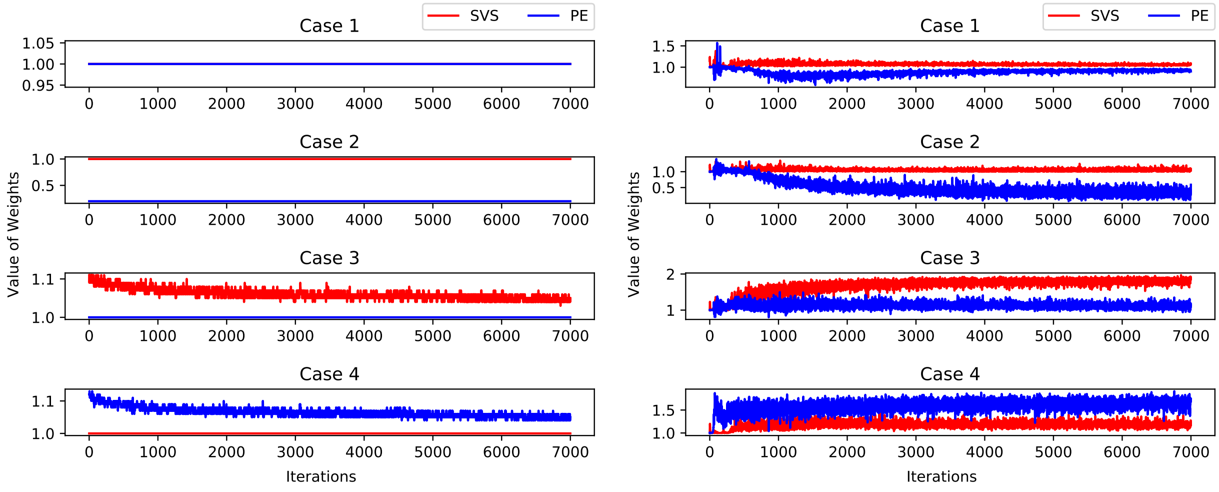
Table 7: Comparison results of the naive DWHS and the DWHS on MIR-1K dataset.

Methods	$upper\_bound$	ω_{noise}	MSS		PE (%)	
			SDR	GNSDR	RPA	RCA
Pipeline			12.06	9.13	91.40	92.07
Naive Joint Learning			11.91	8.92	91.88	92.15
MAJL-Stage I with naive DWHS	4	0	11.63	8.68	91.67	92.14
	5	0	12.09	9.16	92.46	92.74
	6	0	12.02	9.04	91.92	92.34
	5	0.2	12.13	9.18	92.62	93.03
	5	0.4	11.97	9.03	91.98	92.49
MAJL-Stage I with DWHS			**12.33**	**9.36**	**93.17**	**93.65**

The results in Table 7 demonstrate that the DWHS significantly enhances the joint learning of MSS and PE tasks, with MAJL-Stage I with DWHS achieving the best performance among all methods for both tasks. Specifically, the naive joint learning method leads to a decrease of 0.15 in SDR for the MSS task when compared to the pipeline method, due to the problem of misalignment between different objectives. In contrast, MAJL-Stage I with DWHS outperforms both the pipeline and naive joint learning methods, concurrently enhancing both tasks. For instance, it achieves a SDR improvement of 0.42 and a RPA improvement of 1.29% compared to the naive joint learning method. This is because the DWHS implemented in MAJL can effectively handle hard samples for both tasks and assign appropriate weights to different samples. Moreover, MAJL-Stage I with naive DWHS outperforms the naive joint learning method by 0.22 in SDR and 0.74% in RPA when hyper-parameters $upper\_bound$ is set to 5 and ω_{noise} to 0.2. Besides, the DWHS outperforms the naive DWHS and achieves better performance on both tasks without these hyper-parameters. These results indicate that the DWHS not only offers improved performance but also has a lower training cost, making it effective for the joint learning of both tasks.

D.2 Visualization and Analysis of Dynamic Weights

To provide an intuitive representation of the weights set by both the naive DWHS and the DWHS, we visualize the changes in these weights over iterations. In this experiment, we employ ResUNetDecouple+ as the MSS Module and CREPE as the PE Module, consistent



(a)

(b)

Figure 7: (a) Dynamic weights extracted by the naive DWHS. (b) Dynamic weights extracted by the DWHS.

with previous experiment detailed in Section 6.1. In addition, the dynamic weights extracted by Dynamic Weights on Hard Samples (DWHS) are obtained from our framework, specifically MAJL-Stage I, to exclusively investigate the weight results of DWHS and eliminate potential interference, such as single-labeled music data.

As shown in Figure 7, Figure 7(a) illustrates the dynamic weights set by the naive DWHS, while Figure 7(b) displays the dynamic weights extracted by the learned DWHS. For the naive DWHS method, the weight assigned to the pitch estimation task for noisy music data is set to 0.2, thereby mitigating the negative impact on the pitch estimation task. In contrast, for Case 3 and Case 4, the weights assigned to music source separation and pitch estimation exceed 1, emphasizing the importance of hard samples. Regarding the learned DWHS method, the weights for music source separation and pitch estimation tasks correspond to those of the naive DWHS method, as shown in Figure 7(b). When considering the results presented in Table 7, the DWHS method outperforms the naive DWHS method. These results indicate that the DWHS method can adaptively determine appropriate weights for both noisy and hard samples, resulting in enhanced performance for both music source separation and pitch estimation tasks.

D.3 Significance Test

In this section, we validate the significance of the improvements achieved by our framework through statistical significance tests. Thus, we perform multiple experimental runs (6 times) from training to testing and calculate p-values for the SDR and the RPA. The summarized results of these experiments are provided in Table 8.

In this experiment, the pipeline method, ResUNetDecouple+ with CREPE is used as the baseline, since it has displayed the best performance among all baselines, as shown in Table 2. While our framework use both single-labeled datasets (MUSDB18 and MIR_ST500) in the Stage II, and the DWHS is used in this experiment. It is important to note that due to the stochastic nature of gradient-based optimization techniques like the Adam optimizer [34] employed in this paper, there may be slight variations in results even under identical experimental conditions. This variability arises from random factors during training within our framework: the random initialization of model parameters and the random ordering of training samples in each epoch. For each experiment in Table 8, these random initialization use the default initialization method in PyTorch [46]. The results reported in Table 2 are based on the best performance achieved from these repeated experiments. Based on the results presented in Table 8, we calculate the p-values for both the SDR and the RPA. The obtained p-value for SDR is 1.80e-4, and for RPA it is 1.78e-6. These results indicate that our proposed framework achieves a significant improvement when compared to the best-performing baseline.

Table 8: Performance results with significance test on the MIR-1K dataset. Baseline is the pipeline method using ResUNetDecouple+ [36] with CREPE [32]. MAJL here uses the DWHS method, and both MIR_ST500 and MUSDB18 datasets are used as the single-labeled music data.

Methods	Index	MSS		PE(%)	
		SDR	GNSDR	RPA	RCA
Baseline	0	12.06	9.13	91.40	92.07
	1	11.91	8.92	90.87	91.56
	2	11.86	8.92	90.79	91.40
	3	11.97	9.03	91.03	91.63
	4	12.04	9.11	91.21	91.85
	5	12.06	9.12	91.33	91.77
MAJL	0	12.98	9.99	94.11	94.38
	1	12.31	9.36	92.31	92.70
	2	12.41	9.47	92.70	93.15
	3	12.55	9.59	92.88	93.27
	4	12.74	9.77	93.16	93.63
	5	12.90	9.94	93.38	93.96