# Joint Singing Voice Separation and F0 Estimation with Deep U-Net Architectures

Andreas Jansson
*City, University of London / Spotify Inc.*
New York, USA

Rachel M. Bittner
*Spotify Inc.*
New York, USA

Sebastian Ewert
*Spotify Inc.*
London, UK

Tillman Weyde
*City, University of London*
London, UK

***Abstract*—Vocal source separation and fundamental frequency estimation in music are tightly related tasks. The outputs of vocal source separation systems have previously been used as inputs to vocal fundamental frequency estimation systems; conversely, vocal fundamental frequency has been used as side information to improve vocal source separation. In this paper, we propose several different approaches for jointly separating vocals and estimating fundamental frequency. We show that joint learning is advantageous for these tasks, and that a stacked architecture which first performs vocal separation outperforms the other configurations considered. Furthermore, the best joint model achieves state-of-the-art results for vocal-$f_0$ estimation on the iKala dataset. Finally, we highlight the importance of performing *polyphonic*, rather than monophonic vocal-$f_0$ estimation for many real-world cases.**

***Index Terms*—music, voice, singing, fundamental frequency estimation, pitch, melody, source separation, multitask learning**

## I. Introduction

The singing voice plays a major role in most music cultures around the world. In music signal processing, two common tasks related to the singing voice are vocal separation (recovering a recording of the singing voice from a complete song) and vocal $f_0$ estimation (estimating the fundamental frequency of the singing voice over time). As demonstrated in [17], there are dependencies between the two tasks, which allowed for improving the performance on one task by integrating information obtained via a method designed for the other. These dependencies can be modeled in different ways. For example, the fundamental frequency can be estimated and employed as side information in the separation process [10], [17]. Alternatively, an estimate of the clean singing voice can be used as input to simplify the estimation of the fundamental frequency of the voice [4]. Given that both directions were successfully exploited in the past, it remains unclear how these dependencies should be modeled, especially given that prior work typically solves one task independently of the other and conditions the other on the resulting point estimate — eliminating the potential benefits of circular influence. Attempts have been made to learn both tasks iteratively, in an alternating fashion [7]. Learned joint pitch and separation models have been proposed for speech [19]. However, we are not aware of prior work in music that jointly performs vocal separation and vocal melody estimation.

Another important aspect of combining models for different tasks is the issue of a resulting mismatch between the data distributions at training and test time. Consider performing vocal $f_0$ estimation by first applying a source separation algorithm to a mixed signal and then running a standard pitch tracker. Pitch trackers are typically designed based on two assumptions: the signal being pitch tracked has little to no noise or interference, and is monophonic. Due to these assumptions, pitch tracker performance can suffer from artifacts introduced by source separation, which can include residual sounds that are pitched or other interference in the background. Figure 1 shows the performance of three different pitch tracking algorithms — Crepe [9], pYIN [11], and Deep Salience [1] on both clean and source-separated vocals in the iKala dataset [3]. The Overall Accuracy (`OA`) and Raw Pitch Accuracy (`RPA`) metrics (standard metrics in pitch tracking, see [15] for a description) considerably decrease for all three algorithms when applied to source separated instead of clean vocals. Additionally, for songs containing more than one singing voice, many current source separation systems tend to isolate *all* singing voices; the resulting polyphonic signal can severely reduce the accuracy of pitch trackers.

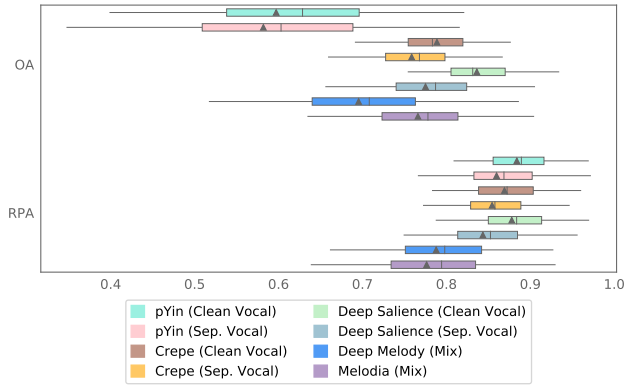


Fig. 1. Performance of pYIN [11], Crepe [9], and Deep Salience [2] on the iKala [3] dataset. pYIN and Crepe are run on clean iKala vocals and on vocals computed by running a source separation algorithm ("Source Only" in Figure 3) from iKala mixtures as input. Deep Salience and Melodia are run on iKala mixtures as input. Boxplots show the distribution of `OA` and `RPA` over each track in the dataset.

Given this interdependent nature of the two tasks, it is an open question how to design a joint estimator, and whether such a system would actually yield benefits. As a first contribution in this paper, we demonstrate that incorporating "oracle" (ground

TABLE I
DEFINITIONS OF RECURRING NOTATIONS

| Notation | Description |
|---|---|
| $y_m$, $y_v$ | mixture and vocal audio signals |
| $Y_m$, $Y_v$ | magnitude STFT of mixture and vocal audio |
| $S_v$ | vocal $f_0$ salience produced by Deep Salience on $Y_v$ [1] |
| $\hat{Y}_v$, $\hat{S}_v$ | model estimate of $Y_v$, $S_v$ |
| $\hat{M}_v$ | model estimated ratio mask |

truth) information for both pitch and separated vocals can indeed improve the learned results for the other task. As a second contribution, we then design, implement and evaluate different model architectures that estimate pitch and vocals individually or jointly in a variety of ways, each reflecting a different perspective on the interdependency of the two tasks. Finally, inspired by end-to-end unfolding techniques for representing iterative re-estimation processes of dependent components inside a network [18] as well as stacking networks [12], we propose an architecture which resolves the task dependencies within a sequential re-estimation model.

## II. Input and Output Representations

We make use of an internal dataset of roughly 2500 pairs of music audio signals $y_m$ and corresponding isolated vocal audio signal $y_v$ from a number of musical genres, including pop, rock and rap vocals. All singing voices present in the mixture $y_m$ are included in the isolated vocals signal $y_v$, meaning that $y_v$ may — and often does — contain more than one active voice at a time (e.g. a lead singer and background harmonies). $y_m$ and $y_v$ are converted to mono with a sample rate of 22050 Hz. Let $Y_m$ and $Y_v$ be the magnitude of the Short Time Fourier Transform (STFT) spectrogram of $y_m$ and $y_v$ respectively. STFTs are computed with a hop size of 256 and with 1024 points in the FFT, as shown in Figure 2 (left) and (middle) respectively.

Our dataset does not contain ground truth vocal $f_0$ annotations. As a proxy for ground truth $f_0$, we run the Deep Salience multiple-$f_0$ estimation model [1] on the isolated vocals $y_v$ in our training set. Deep Salience predicts a matrix $S_v$ of $f_0$ salience values — i.e. the likelihood of an $f_0$ value being present over a grid of time-frequency points (see Figure 2, right). Note that this algorithm does not assume the audio is monophonic — if multiple pitches are present at the same time, there can be multiple high-likelihood $f_0$ bins. $S_v$ is the target output for the vocal $f_0$ estimation (pitch) component of our models. Note that we are training a model to reproduce the output of another trained model (Deep Salience), similar to a teacher-student training paradigm [6]. One notable difference is we are training our model to produce $S_v$ given *mixtures* as input, while the pre-trained Deep Salience model is given *isolated vocals* as input. This also means that the performance of our model will likely be upper bounded by the performance of Deep Salience on isolated vocals.

The largest public datasets with mixtures, corresponding isolated vocals and annotated vocal $f_0$ are iKala [3] ($\approx$ 2 hours) and MedleyDB [2] ($\approx$ 3 hours, since only half of the tracks contain vocals). Because Deep Salience was trained using

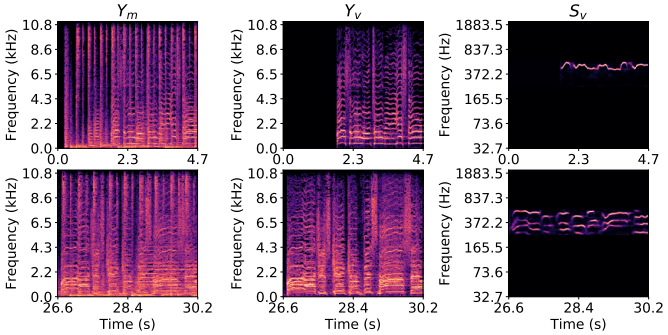


Fig. 2. An example of the input and output representations used for training. (Left) Input magnitude STFT of the mixture audio. (Middle) Target magnitude STFT of the isolated vocal audio. (Right) Target vocal salience produced by the Deep Salience algorithm [1]. The top row shows an example where there is one solo singer, while in the bottom row, three singers are singing in harmony.

MedleyDB, we evaluate the performance of our models on iKala. We compare vocal $f_0$ outputs with iKala's $f_0$ annotations using the `mir_eval` [13] implementation of standard melody metrics [15]. Vocal source separation outputs are evaluated using the Signal-to-Distortion Ratio (SDR) metric[1] from the `mir_eval` implementation of *BSS Eval* [16].

## III. Model Overview and Training

The models presented in the subsequent sections are composed of one or more U-Nets [14]. We use the same architecture as in [8] for estimating both vocal separation $\hat{Y}_v$ and vocal $f_0$ salience $\hat{S}_v$. The vocal separation network produces a soft ratio mask $\hat{M}_v$ from where we derive the vocal magnitude STFT $\hat{Y}_v = \hat{M}_v \odot Y_m$, and is optimized by minimizing the L1 loss $\left\|\hat{Y}_v - Y_v\right\|_1$. The isolated vocal signal $\hat{y}_v$ is synthesized by applying the phase of the original complex mixture spectrogram to the estimated magnitude spectrogram, and transforming to the time-domain by means of the Inverse Short Time Fourier Transform (ISTFT). The salience network outputs $\hat{S}_v$ directly and is optimized with L2 loss $\left\|\hat{S}_v - S_v\right\|_2$.

In the case of monophonic targets (e.g. as in iKala), $f_0$ time series are generated from $\hat{S}_v$ by returning the frequency with maximum likelihood at each time frame. The voicing (when the voice is active/inactive) is determined by a simple threshold on the maximum likelihood at each time frame; frames where the likelihood falls below the threshold are reported as "unvoiced". In the results from our models, we fix the voicing threshold to 0.4. For the comparison models (Crepe, Melodia, etc.), we compute performance for a full grid of possible thresholds, and report the performance for the threshold that maximizes performance in each case ("oracle" threshold).

## IV. Baselines and Oracle Experiments

As a baseline, we first train separate models, shown in Figure 3 (left): `Pitch only` which estimates $f_0$ salience given mixtures as inputs, and `Source only` which performs vocal

[1]Signal-to-Interference and Signal-to-Artifact Ratios followed the same trends as SDR and are therefore not included.

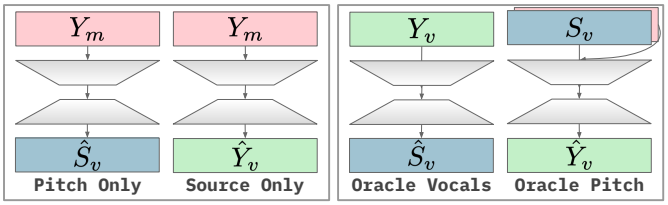


Fig. 3. (Left) Baseline models take the mixture magnitude STFT $Y_m$ as input, and output vocal $f_0$ salience $\hat{S}_v$ in "Pitch Only", and the vocal magnitude STFT $\hat{Y}_v$ "Source Only". (Right) Oracle models which are given "perfect" input information. "Oracle Source" is given isolated vocals magnitude STFTs $Y_v$ as input and trained to output $S_v$. "Oracle Pitch" is given $Y_m$ and oracle vocal $f_0$ salience $S_v$ as input and trained to output $Y_v$.

source separation. These models are completely independent and do not share weights. As an upper bound, we train models that receive *oracle* information — `Oracle Vocals` which estimates vocal salience $\hat{S}_v$ given ground truth vocals $Y_v$ as input, and `Oracle Pitch` which estimates the vocal spectrogram magnitudes $\hat{Y}_v$ given the mixture $Y_m$ and ground truth $f_0$ salience $S_v$ as inputs, as shown in Figure 3 (right). `Oracle Vocals` tells us how well we can estimate $f_0$ performance given perfect information, i.e. the performance reported provides an estimate for the upper bound achievable with this architecture and number of parameter. `Oracle Pitch` tells us how much it helps source separation performance to have $f_0$ salience as side information, and similarly gives us an upper bound on source separation performance.

The results in Figure 4a clearly show that a vocal pitch estimator trained on clean vocals $Y_v$ (`Oracle Vocals`) performs better than one trained on mixtures $Y_m$ (`Pitch only`). This is consistent with Figure 1 where pitch trackers with clean vocals as input outperform pitch trackers that operate on mixtures. A similar effect can be seen for vocal separation in Figure 4b, where the inclusion of ground truth pitch salience $S_v$ improves the source separation metric.

## V. Joint Models

In the following section, we present a series of different architectures that perform both vocal source separation and $f_0$ salience estimation. Figure 5 gives an overview of the architectures we compare; they share information in various ways, either through weight sharing (treating the problem in a standard multitask setup) or by directly giving the outputs of one stage of the model as input to the next.

### A. Conventional Multitask Models

We first experiment with architectures that share weights for both tasks. The `Shared Encoder` model, Figure 5 (top left), shows the simplest such architecture, which has one encoder that is shared and separate decoders for each task. The results for this experiment show that the shared encoder model is inferior to disjoint models for both vocal $f_0$ estimation (Figure 4a) and vocal separation (Figure 4b). One potential explanation for this result is that the shared encoder reduces the number of parameters in the network by 25%.

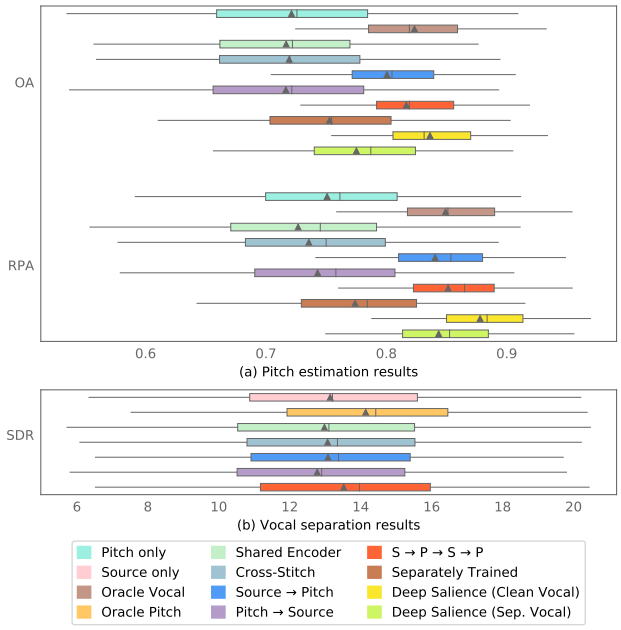


Fig. 4. Performance comparison of our experiments, oracle, and baseline models, evaluated on iKala. (Top) Single-$f_0$ metrics. (Bottom) Vocal source separation metrics.

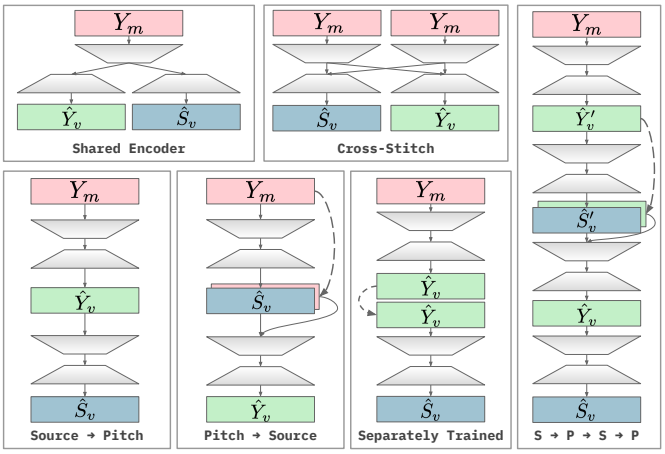


Fig. 5. Joint U-Net models. Each model takes the magnitude spectrogram of the mixture $Y_m$ as input and outputs estimates of the vocal magnitude spectrogram $\hat{Y}_v$ and the vocal $f_0$ salience $\hat{S}_v$. In `Pitch→Source` model, $Y_m$ is given as additional input to the vocal source separation portion of the model (indicated by a dotted line), and similarly in the `S→P→S→P` model, the first vocal estimate $\hat{Y}'_v$ is given as additional input to the second vocal source separation model. In `Separately Trained`, each network is trained separately, first optimizing $\hat{Y}_v$, and then using the optimized $\hat{Y}_v$ as input to a second model that outputs $\hat{S}_v$.

In the `Cross-Stitch` model, shown in Figure 5 (top right), each output has separate encoder-decoders, but the encoders are concatenated before being passed to the individual decoders. However, the skip connections are task-specific. This model has a capacity equivalent to that of the two baseline models, and should not suffer from the same potential capacity issue of the Shared Encoder model. `Cross-Stitch` performs slightly better than `Shared Encoder`, but is still worse than the baseline

models on both tasks.

### B. Stacked Models

In Figure 5, (bottom left and middle), the tasks are learned in a cascaded manner. In the `Source→Pitch` model, a first U-Net computes $\hat{Y}_v$ given $Y_m$, and a second U-Net computes $\hat{S}_v$ given $\hat{Y}_v$ as input. `Pitch→Source` is similar but in the reverse order, and with the addition of concatenating $Y_m$ and $\hat{S}_v$ as input to the second U-Net model. This concatenation was added because there is not enough information in $\hat{S}_v$ alone to compute $\hat{Y}_v$ — the mixture information is needed as well.

As shown in Figure 4a, `Source→Pitch` outperforms all of `Pitch only`, `Shared Encoder` and `Cross-Stitch` for pitch estimation. This is perhaps unsurprising, since the baseline model trained on clean vocals (`Oracle Vocals`) performs better than the baseline trained on mixtures (`Pitch only`). However, `Pitch→Source` does not result in a similar improvement for vocal separation, even though `Oracle Pitch` saw significant improvements. We hypothesize that the lack of accuracy of pitch estimates from mixture (see Figure 4a) prevents improvements of the vocal separation on the level of `Oracle Pitch`.

### C. Stacked Refinement

In our final experiment, we attempt to further refine the results of `Source→Pitch`. *Stacked Refinement* [5] is an architectural pattern in which a network module is repeated, feeding the output of a module as input to an identically designed module, with different weights. We adapt this pattern to our domain by extending `Source→Pitch` to a "Source → Pitch → Source → Pitch" (S→P→S→P) network 5 (right). We notate the output of the first source network $\hat{Y}_v'$ and the output of the first pitch network $\hat{S}_v'$. The second source network is fed a concatenation of $\hat{Y}_v'$ and $\hat{S}_v'$, and it outputs $\hat{Y}_v$, which it then feeds into the second pitch network to output $\hat{S}_v$. This allows higher level modules to learn a refinement function from initial predictions to cleaner predictions. The scores for the resulting model are plotted in Figure 4a and Figure 4b. The stacked refinement model surpasses the performance of all of our other models for both vocal source separation and vocal melody estimation.

The second source separation network is presented with an adequate pitch estimation, which provides additional guidance to refine the vocal source estimate. It is a somewhat surprising result that the difference for melody estimation is higher than the difference for vocal separation, since that implies that the inputs to the first and second melody estimation networks have little difference. We hypothesize that the doubled network capacity might be an important additional factor in explaining this result.

## VI. Joint vs. Separate Training

We saw in the previous section that `Source→Pitch` performs better than the other configurations of the same network capacity. In order to test if the joint training is necessary, we take the output $\hat{Y}_v$ of `Source only` as input to a model which outputs $\hat{S}_v$, as shown in Figure 5 (`Separately Trained` bottom, 2nd from right). The results, plotted as `Separately Trained` in Figure 4a, show that joint optimization is indeed beneficial for vocal melody estimation. This is possibly because our pitch model is "borrowing" capacity from the separation model, or alternatively because the separation results become more tailored towards pitch estimation.

TABLE II
Vocal Melody Estimation Results on the iKala Dataset

| Experiment | OA | RPA |
|---|---|---|
| Melodia | 0.766 | 0.776 |
| Deep Salience | 0.695 | 0.788 |
| Pitch only | 0.721 | 0.751 |
| Shared Encoder | 0.717 | 0.727 |
| Cross-Stitch | 0.719 | 0.736 |
| Source → Pitch | 0.800 | 0.841 |
| Pitch → Source | 0.716 | 0.743 |
| S → P → S → P | **0.817** | **0.851** |

## VII. Discussion

### A. Vocal Melody Estimation from Mixtures

The fusion of vocal source separation and vocal salience estimation results in system that is able to estimate vocal melody from musical mixtures. As we saw in Section I, vocal pitch trackers that take isolated vocals as input have higher accuracy than systems that estimate vocal melody directly from a mixture. We now pose the question of whether our jointly trained vocal separator and multi-$f_0$ estimator can predict single-$f_0$ vocal melody as well as pitch trackers that were explicitly trained to predict single-$f_0$ from mixtures. Table II shows the standard melody metric averages on the iKala dataset for our models and two strong baseline models, Melodia [15] and Deep Salience [1]. We see that our best model (S→P→S→P) does indeed perform better than both other models on vocal melody estimating from polyphonic mixtures.

### B. Qualitative Analysis

To build an intuition of the characteristics of the model's pitch estimates, we zoom in on a few bars of the 1965 pop song "Turn! Turn! Turn!" by The Byrds. Figure 6 shows the final pitch estimation using S→P→S→P, with a voicing threshold set to $0.4$. Since we do not have vocal melody ground truth for this track, we only show the model outputs. While there are a few scattered false positives, the majority of the pitch estimates appear to belong to vocal notes. The likelihoods for sustained notes sometimes drop below the voicing threshold, leaving only activations at the initial note onset transient.

It is also instructive to visualize the raw estimated pitch salience, before applying the voicing threshold. Figure 7 show vocal salience matrices, estimated on the first phrase of the example above, using our model (left) and Crepe (right). The most striking difference is our model's ability to predict multiple simultaneous vocal parts, while Crepe oscillates between the two notes. Despite the fact that our model is presented with a full mixture, and Crepe with vocals

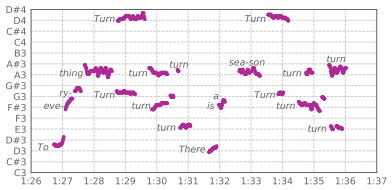


Fig. 6. Vocal $f_0$ estimation from S→P→S→P on an excerpt from the pop song "Turn! Turn! Turn!" by The Byrds.

isolated by the source separation output of our model, our model produces a cleaner salience matrix, with fewer artifacts. It appears that during the joint training procedure, the vocal melody estimation module learns to ignore source separation artifacts, despite the fact that artifacts are present in this excerpt. It should also be noted that our model is less precise than Crepe, lacking some of the fine-grained sharpness of the vibratos that Crepe more accurately captures.

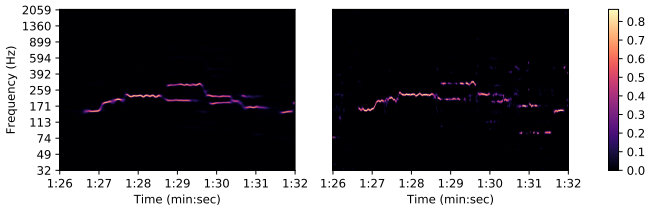


Fig. 7. Estimated salience matrix $\hat{S}_v$. Left: our S→P→S→P model, predicted from mixture $Y_m$. Right: Crepe, predicted from estimated vocal source $\hat{Y}_v$. The excerpts begins with one voice, and a second voice enters at 1:29.

## VIII. Conclusions and Future Work

In this work, one of the biggest challenges was in assessing *why* certain models performed better or worse than others, and in this paper we suggested hypotheses with possible reasons. Future work includes testing these specific hypotheses to obtain better insights about the specific advantages and disadvantages of these architectures. Beyond this, we would like to explore applying joint $f_0$ and source separation models to other types of pitched sources beyond the singing voice, such as the piano. We think the same ideas could also be applied to drum separation and drum transcription. Overall, we explored a number of different architectures for jointly separating vocals and estimating fundamental frequency in a single data driven system based on deep U-Net neural network architectures. We saw that including oracle information improves performance, in particular for vocal-$f_0$ information. A joint stacked model that first performs vocal source separation followed by vocal $f_0$ estimation approaches the performance of the oracle models, and outperformed conventional multitask architectures, which underscores the value of incorporating domain knowledge when designing models. Additionally, we showed that the model achieves state-of-the-art results for vocal-$f_0$ estimation on the iKala dataset. Finally, we highlighted the importance of performing *polyphonic*, rather than monophonic vocal-$f_0$ estimation for many real-world cases.

## References


[1] Rachel M. Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan P. Bello. Deep salience representations for F0 estimation in polyphonic music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, October 2017.

[2] Rachel M. Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. MedleyDB: A multitrack dataset for annotation-intensive MIR research. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[3] Tak-Shing Chan, Tzu-Chun Yeh, Zhe-Cheng Fan, Hung-Wei Chen, Li Su, Yi-Hsuan Yang, and Roger Jang. Vocal activity informed singing voice separation with the ikala dataset. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 718–722. IEEE, 2015.

[4] Jean-Louis Durrieu, Gaël Richard, Bertrand David, and Cédric Févotte. Source/filter model for unsupervised main melody extraction from polyphonic audio signals. *IEEE Trans. Audio, Speech & Language Processing*, 18(3):564–575, 2010.

[5] Sheng He and Lambert Schomaker. Deepotsu: Document enhancement and binarization using iterative deep learning. *Pattern Recognition*, Jan 2019.

[6] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[7] Chao-Ling Hsu, DeLiang Wang, Jyh-Shing Roger Jang, and Ke Hu. A tandem algorithm for singing pitch extraction and voice separation from music accompaniment. *IEEE Trans. Audio, Speech & Language Processing*, 20(5):1482–1491, 2012.

[8] Andreas Jansson, Eric Humphrey, Nicola Montecchio, Rachel M. Bittner, Aparna Kumar, and Tillman Weyde. Singing voice separation with deep u-net convolutional networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[9] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello. Crepe: A convolutional representation for pitch estimation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 161–165. IEEE, 2018.

[10] Yipeng Li and DeLiang Wang. Separation of singing voice from music accompaniment for monaural recordings. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(4):1475–1487, 2007.

[11] Matthias Mauch and Simon Dixon. pyin: A fundamental frequency estimator using probabilistic threshold distributions. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 659–663. IEEE, 2014.

[12] Sungheon Park, Taehoon Kim, Kyogu Lee, and Nojun Kwak. Music source separation using stacked hourglass networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[13] Colin Raffel, Brian McFee, Eric Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel PW Ellis. mir_eval: A transparent implementation of common mir metrics. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[15] Justin Salamon, Emilia Gómez, Daniel PW Ellis, and Gaël Richard. Melody extraction from polyphonic music signals: Approaches, applications, and challenges. *IEEE Signal Processing Magazine*, 31(2):118–134, 2014.

[16] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *IEEE transactions on audio, speech, and language processing*, 14(4):1462–1469, 2006.

[17] Tuomas Virtanen, Annamaria Mesaros, and Matti Ryynänen. Combining pitch-based inference and non-negative spectrogram factorization in separating vocals from polyphonic music. In *INTERSPEECH*, 2008.

[18] Scott Wisdom, John Hershey, Jonathan Le Roux, and Shinji Watanabe. Deep unfolding for multichannel source separation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 121–125, 2016.

[19] Xueliang Zhang, Hui Zhang, Shuai Nie, Guanglai Gao, and Wenju Liu. A pairwise algorithm using the deep stacking network for speech separation and pitch estimation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(6):1066–1078, 2016.