A LIGHTWEIGHT INSTRUMENT-AGNOSTIC MODEL FOR POLYPHONIC NOTE TRANSCRIPTION AND MULTIPITCH ESTIMATION

Rachel M. Bittner[♭], *Juan José Bosch*[♭], *David Rubinstein*[♭], *Gabriel Meseguer-Brocal*[♯], *Sebastian Ewert*[♭]

[♭]Spotify, [♯]IRCAM

ABSTRACT

Automatic Music Transcription (AMT) has been recognized as a key enabling technology with a wide range of applications. Given the task's complexity, best results have typically been reported for systems focusing on specific settings, e.g. instrument-specific systems tend to yield improved results over instrument-agnostic methods. Similarly, higher accuracy can be obtained when only estimating frame-wise f_0 values and neglecting the harder note event detection. Despite their high accuracy, such specialized systems often cannot be deployed in the real-world. Storage and network constraints prohibit the use of multiple specialized models, while memory and run-time constraints limit their complexity. In this paper, we propose a lightweight neural network for musical instrument transcription, which supports polyphonic outputs and generalizes to a wide variety of instruments (including vocals). Our model is trained to jointly predict frame-wise onsets, multipitch and note activations, and we experimentally show that this multi-output structure improves the resulting frame-level note accuracy. Despite its simplicity, benchmark results show our system's note estimation to be substantially better than a comparable baseline, and its frame-level accuracy to be only marginally below those of specialized state-of-the-art AMT systems. With this work we hope to encourage the community to further investigate low-resource, instrument-agnostic AMT systems.

Index Terms— automatic music transcription, note estimation, multi-pitch estimation, polyphonic, low-resource

1. INTRODUCTION

The automatic transcription of music has been studied for more than four decades [1]. During this time, systems have considerably improved, in particular since the rise of deep learning. Yet, the task remains unsolved, partially due to various intrinsic challenges [1] but also due to a lack of an objective ground truth on which humans consistently agree [2]. Because of the intrinsic difficulty of the task, AMT systems are usually designed with a limited scope, and focus on a sub-task. There are a number of common sub-tasks in AMT which branch along three dimensions: (1) the degree of output polyphony (monophonic, polyphonic) (2) the types of output to be estimated (notes, f_0), and (3) the type of input audio (pop songs, solo piano, solo guitar, jazz ensembles, etc.). For example, specializing for a specific instrument class allows models to exploit instrument-specific characteristics to increase the transcription accuracy, e.g. piano [3–5], guitar [6, 7] or singing voice [8, 9]. Similarly, models built to estimate a particular output type, or which are restricted to monophonic settings [10] can further increase accuracy in these scenarios. In many real-world applications, deploying a number of specialized systems becomes intractable, for example because of storage, network and maintenance constraints. Further, for many instruments it is challenging to create a dataset large enough to train modern methods. Applications can also add additional restrictions w.r.t. the size of the model, its (peak) memory consumption and run-time. Therefore, there is often a gap between the latest published state of the art and models that can practically be deployed in a range of settings.

In this work, we consider a broad scenario: an instrument-agnostic[1] polyphonic AMT model which estimates both notes and multipitch outputs. The proposed model is a lightweight neural network which runs efficiently on low-end devices, thanks to its low memory and processing time requirements. Unless otherwise noted, we deal with polyphonic recordings of a single instrument class (e.g. solo piano, an ensemble of violins, solo vocals, a choir, etc.), but do not restrict which classes we consider. It is jointly trained to predict frame-level onset, multipitch and note posteriorgrams. During inference, we post-process the frame-level posteriorgrams to obtain note events and multipitch information. We study the ability of the proposed model to transcribe a variety of instruments and vocals without retraining, and compare with a recent baseline model for instrument-agnostic polyphonic note estimation. Further, we evaluate the contribution of components of the proposed model with an ablation study. All code and trained models discussed in this paper are made publicly available[2]. Additionally, we only use public datasets for training and evaluation in order to foster reproducibility.

2. BACKGROUND AND RELATED WORK

There is a huge body of work on AMT. Due to space constraints we refer to [1, 11] for a more comprehensive overview. As previously mentioned, AMT systems have three dimensions: (1) the degree of output polyphony considered, (2) the type of output estimated and (3) the type of input audio. In this work we consider the polyphonic setting, where more than one note/pitch may be present in the output at a time; note that monophonic AMT is a strict subset of polyphonic AMT, and thus we also support monophonic sources. AMT outputs are typically either **frame-level multipitch estimation (MPE)** or **note-level estimation**, which transcribe polyphonic music at different levels of granularity [1]. Both are useful depending on the application: MPE provides lower-level expressive performance information (such as vibrato, glissando), whereas note-level estimation gives information closer to the musical score. MPE methods predict the fundamental frequencies (f_0s) which are active at a given time-frame (note that even if not strictly equivalent, we use pitch and f_0 interchangeably following the literature in the field [1]). They commonly first estimate a pitch posteriorgram [12, 13], where each time-frequency bin is assigned an estimate of the likelihood of that fundamental frequency being active at a given time. Such matrices

[1]By "instrument agnostic" we mean "not specific to an instrument class".

[2]`https://github.com/spotify/basic-pitch`

typically contain multiple bins per semitone, which allows estimation of small ("continuous") variations of pitch. Various methods aim at estimating and subsequently grouping MPE outputs from polyphonic recordings into note events [14–18], or attempt to group pitches into contours [11, 12, 19]. Note estimation (or note tracking) methods aim at estimating notes events (defined as: pitch, onset time, offset time). Notes cannot be trivially estimated from the output of an MPE system, because MPE information does not encode onsets/offsets, and preserves fluctuations in pitch which should not always be quantized to the nearest semitone. In particular, note estimation is difficult for singing voice, which may have a high degree of fluctuation around a center pitch [9] compared to instruments such as the piano. Multiple methods have been proposed for estimating notes from pitch posteriorgrams e.g. using median filtering [11], Hidden Markov Models [16] or neural networks [20, 21]. While most approaches consider each semitone independently, some approaches attempt to model the interactions between notes, using spectral likelihood models [1, 18], or music language models [3, 17]. Transformers have recently been applied to AMT, directly predicting MIDI-like note events from spectrograms in piano music [5]. A few AMT models perform both note and pitch estimation [14, 18, 22], and most work with monophonic data. Regarding input audio characteristics, traditional AMT methods based on signal-processing have been more generalizable to multiple instruments than more recent approaches, as well as being simpler and faster [1, 12]. However, the best performing systems often come at the expense of higher computational requirements and a focus on instrument-specific systems [4].

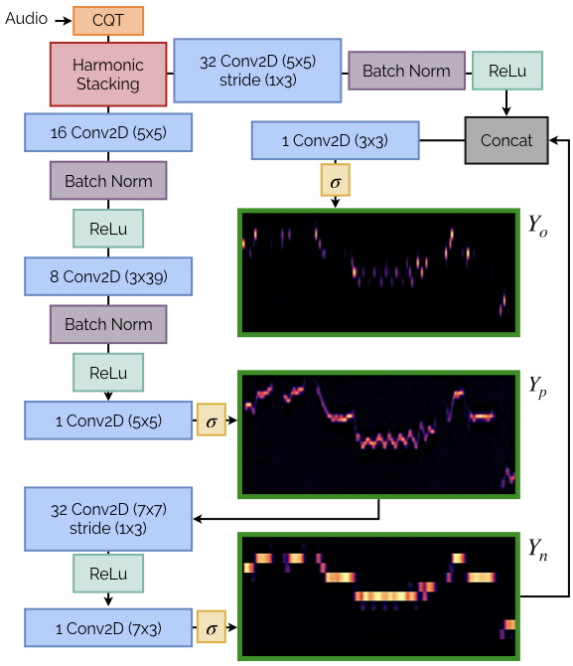

Fig. 1. The NMP architecture. The matrix posteriorgram outputs Y_o, Y_p, and Y_n are outlined in green. σ indicates a sigmoid activation.

3. MODEL

Our goal is to create an AMT model that generalizes across a set of polyphonic (or monophonic) instruments without retraining, while being lightweight enough to run in low-resource settings. We consider both the speed and the peak memory usage when running inference, and purposely limit ourselves to a shallow architecture to keep the memory needs low. Note that the number of parameters of a model does not necessarily correlate with its memory usage; e.g. a convolution layer requires few parameters, but can still have high memory usage due to the feature map sizes.

Harmonic Stacking. Given the input audio, the model first computes a Constant-Q Transform (CQT) with 3 bins per semitone and a hop size of ≈ 11 ms. Rather than using, e.g. a mel spectrogram and ultimately learning the projection onto the output log-spaced frequency scale using a Dense or LSTM layer (which requires the model to have a full-frequency receptive field) [4], we start with a representation with the desired frequency scale. The Harmonic CQT (HCQT) [13] is a transformation of the CQT which aligns harmonically-related frequencies along a third dimension, allowing small convolutional kernels to capture harmonically-related information. As an efficient approximation of the HCQT, following [23], we copy the CQT and shift it vertically by the number of frequency bins corresponding to each harmonic. In this work we use 7 harmonics and 1 sub-harmonic.

Architecture. The architecture illustrated in Fig. 1 is a fully convolutional model taking audio as input and produces three posteriorgram outputs, with a total of only 16,782 parameters. The model's three output posteriorgrams are time-frequency matrices encoding if (1) the onset of a note is taking place (Y_o) (2) a note is active (Y_n) and (3) a pitch is active (Y_p). All outputs have the same number of time frames as the input CQT, and in frequency, both Y_o and Y_n have a resolution of 1 bin per semitone while Y_p has a resolution of 3 bins per semitone. Besides having different frequency resolutions, Y_n and Y_p are trained to capture different concepts: Y_n captures frame-level note event information "musically quantized" in time and frequency, while Y_p encodes frame level multipitch information, capturing details such as vibrato. During training, the target for each of these outputs are binary matrices generated from note and pitch annotations.

The architecture is structured in order to exploit the differing properties of the three outputs. We assume that Y_p is the output which is "closest" to the input audio. The architecture estimating Y_p is similar to that of [13], but with fewer convolutional layers to reduce memory usage. Notably, we employ the same octave plus one semitone-sized kernel in frequency, which we found to be helpful for avoiding octave mistakes. This stack of convolutions performs a form of "denoising", in order to emphasize the multipitch posterior outputs and de-emphasize transients, harmonics and other un-pitched content. An added benefit to using a limited receptive field in frequency is that it removes the need for pitch shifting data augmentations. Y_p followed by two small convolutional layers are used to estimate Y_n. These convolutions can be seen as "musical quantization" layers, learning how to perform the non trivial grouping of multipitch posteriorgrams into note event posteriorgrams. Finally, as in [24], Y_o is estimated using as input both Y_n and convolutional features computed from the audio, which are necessary to identify transients.

Training. Binary cross entropy is used as the loss function for each output, and the total loss is the sum of the three losses. However, for Y_o, there is a heavy class imbalance that drives models to output $Y_o = 0$ everywhere. As a countermeasure, we use a class-balanced cross entropy loss, where the weight for the negative class is 0.05 and the positive is 0.95 (set empirically by observing the properties of the resulting Y_o) which helps the model capture the onsets while remaining sparse. During training, the model input is 2 seconds of audio at a sample rate of 22050 Hz. We train the model with a batch size of 16 and use the Adam optimizer with a learning rate of 0.001. During training, random label-preserving augmentations are applied to the audio, including adding noise, equalization filters, and reverb.

Posteriorgram post-processing. Similar to many note or contour creation post-processing methods, we create note events, defined by a start time t^0, end time t^1 and a pitch f by running a post-processing step using Y_o and Y_n as inputs [1], following a process similar to that described in Onsets and Frames [4]. A set of onset candidates $\{(t_i^0, f_i)\}$ are populated by peak picking Y_o across time, and discarding peaks with likelihood < 0.5. Note events are created for each i in descending order of t_i^0, by tracking forward in time through Y_n until the likelihood falls below a threshold τ_n for longer than an allowed tolerance (11 frames), then ending the note. When notes are created, the likelihood of all corresponding frames of Y_n are updated to 0. After all onsets have been used, additional note events are created by iterating through bins of Y_n that have likelihood $> \tau_n$ in descending order, following the same note creation procedure but instead tracing both forward and backward in time. Finally, note events which are shorter than ≈ 120 ms are removed. Multi-pitch estimates are created by simply peak picking Y_p across frequency and retaining all peaks greater than τ_n.

4. EXPERIMENTS

In this section we examine the performance of the proposed method, "Notes and Multipitch" (NMP), focusing on the note estimation task, but also briefly commenting on the MPE task. AMT methods have commonly been evaluated using a set of metrics proposed for MIREX[3] evaluation tasks. In this work we report the note-level F-measure (`F`), where notes are considered correct if the pitch is within a quarter tone, the onset is within 50 ms, and the offset is within 20% of the note's duration, the note-level F-measure-no-offset (`Fno`) with the same criterion as F-measure, but ignoring offsets, and the frame-level note accuracy (`Acc`), which is computed for frames with a hop size of 10 ms. We use `Fno` as the main measure of overall note estimation accuracy since the definition of offsets is less objective than onsets (e.g. due to reverberation, sustain pedal, annotation procedure) [25]. We compute these metrics using `mir_eval` [26]. For NMP and each of the ablation studies, we fine-tune the note creation parameter τ_n on the validation dataset such that it maximizes `Fno`.

In order to assess how well NMP and the baseline perform across different instrument classes, we use a wide variety of training and test data spanning multiple instrument types, summarized in Table 1 (see the cited papers for more specific details), using the `mirdata` [27] library. A random 5% of tracks from the training set are used for validation. We note a few additional details for some the datasets: We use the de-duplicated "redux" version of Slakh, and test on an instrument-balanced subset of 120 of the non-percussive test-set stems with the least silence; the note annotations in MedleyDB and iKala are automatically generated using pyin-notes [22]; the audio files for MedleyDB are taken from the pitch tracking subset[4], and for iKala, we use the isolated vocals; for Phenicx, we use the 42 instrumental section-grouped stems (e.g. violins, bassoons) and annotations.

4.1. Note Transcription Baseline Comparison

We compare our model with a recent, strong baseline model, **MI-AMT** [34] which is a polyphonic, instrument-agnostic note estimation method. It uses a U-Net architecture with an attention mechanism and outputs a note-activation posteriorgram with a total of over 20M parameters, trained on MAESTRO and MusicNet. The note posteriorgram is post-processed in order to create note events.

[3] `http://www.music-ir.org/mirex/`
[4] `https://zenodo.org/record/2620624`

Dataset	*Polyphony*	*Instrument*	*Labels*	*Train*	*Test*
Molina [28]	Mono	Vocals	N	-	38
GuitarSet [29]	Mono / Poly	Ac. Guitar	N + P	648	72
MAESTRO [4]	Poly	Piano	N	1154	128
Slakh [30]	Poly	Synthesizers	N	1590	120
Phenicx [31]	Poly	Orchestral	N	-	42
iKala [32]	Mono	Vocals	N + P	252	-
MedleyDB [33]	Mono	Multiple	N + P	103	-

Table 1. Summary of the datasets used. The *Train* and *Test* columns indicate the number of tracks. The *Labels* column indicates which kind of annotations are available: (N) Notes, (P) Multi-pitch.

Results for our proposed method and MI-AMT are presented in Table 2. We first remark that NMP considerably outperforms the baseline MI-AMT on all test datasets and metrics, with the exception of the comparable `Acc` on MAESTRO (piano) and Slakh (synthesizers). NMP performs strongly for datasets with polyphonic instruments (MAESTRO, Slakh, Phenicx, ½ of GuitarSet) as well as monophonic (Molina and ½ of GuitarSet) despite not imposing a monophonic constraint on the output note estimates. Additionally, we see consistent performance across datasets with varying instrument types, validating that NMP performs well without needing to be instrument-specific.

	Molina			**GuitarSet**			**Maestro**			**Slakh**			**Phenicx**		
	Acc	Fno	F	Acc	Fno	F	Acc	Fno	F	Acc	Fno	F	Acc	Fno	F
MI-AMT	.48	.31	.11	.43	.59	.27	.39	.30	.07	.40	.23	.07	.13	.12	.05
NMP	**.63**	.52	.35	**.70**	**.79**	**.56**	.38	.71	.11	**.44**	.42	.21	**.53**	.49	.35
NMP - P	.60	.55	.38	.67	.78	.55	.36	.65	.12	.40	**.43**	**.23**	.50	**.51**	**.36**
NMP - H	.45	.36	.20	.50	.65	.40	.27	.48	.10	.33	.36	.17	.37	.39	.23

Table 2. Average note event metrics on all test datasets for the baseline algorithm, proposed method, and ablation experiments. The best score for each column is in bold. The shade of green indicates how far a score is from the best score, with the worst scores in white. All non-underlined results are statistically significantly different with $p < 0.05$ compared with NMP (per metric/dataset) via a paired t-test.

4.2. Ablation Experiments

Harmonic Stacking. To examine the use of Harmonic Stacking as an input representation, we trained a model which omits the harmonic stacking layer but is equivalent otherwise, denoted as `NMP - H` in Table 2. Unsurprisingly, given the small receptive field, the omission of harmonic stacking substantially reduces performance across all metrics and datasets, in accordance with the results of similar experiments performed in [13, 23]. This indicates that Harmonic Stacking effectively allows the model to use smaller convolutional kernels while still capturing relevant information. One limitation of this comparison is that the number of channels is reduced when omitting harmonic stacking, which in turn reduces the model's capacity.

The Effect of Y_p. We measure the impact that the supervised bottleneck layer Y_p on Y_n has on note estimation by training an equivalent model, where Y_p is not supervised, and where Y_n is the output of the stack of convolutions preceding it, with the *Batch Norm* $\rightarrow$ *ReLu* $\rightarrow$ `1 Conv2D (5x5)` layers in Fig. 1 omitted. The results for this condition are denoted as `NMP - P` in Table 2. We first see that the constraint introduced by Y_p consistently improves `Acc` across all datasets, however the effect on `Fno` and `F` is mixed; there is no significant difference for GuitarSet, Slakh and Phenicx, Y_p im-

proves performance slightly for MAESTRO, and degrades slightly for Molina. This suggests that even if the additional supervision is neutral for onset/offset detection, it is helpful for identifying note pitches, and we get the benefit of an additional output which contains some information about ornamentation and expressivity.

4.3. Comparison with instrument-specific approaches

	Molina (Vocano)			GS-solo (TENT)			Maestro (OF)		
	Acc	Fno	F	Acc	Fno	F	Acc	Fno	F
Baseline	61.6	**64.2**	**51.3**	63.2	76.3	54.6	**43.8**	**95.2**	**36.4**
NMP	**62.6**	52.3	34.6	**71.7**	**84.0**	**65.0**	37.5	70.9	10.5

Table 3. Average note event metrics on NMP vs. instrument-specific models for vocals, guitar and piano. The best score for each column is in bold. The compared instrument-specific model name is indicated in the column headers in parenthesis. All non-underlined results are statistically significantly different with $p < 0.05$ compared with NMP via a paired t-test.

We've seen that the proposed model outperforms a comparable instrument-agnostic baseline on a variety of datasets. To further understand the upper limits of our model, we provide a comparison with recent, open-source instrument-specific models. **Onsets and frames (OF)** [4] is a polyphonic piano transcription method trained on the MAESTRO dataset which jointly predicts onset and note posteriorgrams using a CNN and RNN consisting of approximately 18M parameters, followed by a note-creation post-processing phase. **Vocano** [9] is a monophonic vocal transcription method which first performs vocal source separation, then applies a pre-trained pitch extractor followed by a note segmentation neural network, trained on solo vocal data. **TENT** [6] is a monophonic solo guitar transcription method, which first performs melody contour extraction followed by playing technique detection of common guitar elements such as string bend, slide and vibrato using a CNN architecture, and a post-processing phase which obtains the final notes given the melody contour and the identification of the different playing techniques at each time frame. We therefore only report results on the solo, monophonic half of GuitarSet.

For guitar, NMP outperforms TENT for all metrics, and more importantly, these are state of the art results on GuitarSet to the best of our knowledge. For vocals (*Molina*), Vocano outperforms NMP in `Fno` and `F`, but the frame-level pitch accuracy (`Acc`) is comparable to NMP, suggesting that `Fno` could increase with improved onset detection. The largest difference in performance between NMP and an instrument-specific method is in the MAESTRO dataset compared with OF, which was specifically trained for piano transcription, and achieves 95.2% `Fno`, in comparison to 70.9% of our method (which is notably still a reasonably high score for this task). The main reason for the difference in performance seems to be due to the onset detection accuracy which is higher in OF, since `Acc` is more similar for both methods (42.8% for OF vs. 37.5% for NMP). It is interesting to note that NMP would perform competitively in comparison with Melodyne[5] on piano data according to the results obtained in [24], even if a direct comparison would not be possible since they reported results on another similar piano dataset.

[5] version 4.1.1.011, `http://www.celemony.com/en/melodyne`

4.4. MPE Baseline

Here we briefly validate how NMP performs at MPE, comparing NMP's MPE outputs with the output of the deep salience model [13]. We report results on the Bach 10 [12] and Su [2] datasets, each of which contain 10 recordings of polyphonic western classical chamber music ensembles. The MPE outputs for NMP outperform deep salience for the Bach10 dataset with a frame-level accuracy of 72.5 ± 3.8 versus 55.7 ± 2.9 for deep salience. However, deep salience achieves better results on Su 43.6 ± 7.9 where NMP gets 37.7 ± 15.4. While this is a small-scale validation, these results indicate that the information captured by Y_p is meaningful and potentially competitive with strong baseline models. While the 3-bin-per-semitone resolution posteriorgrams may seem relatively low-resolution for this task, they can be used to estimate continuous multi pitch estimates, by using the amplitude values of the estimated f_0 bin, and those of its neighboring bins in frequency. Note that despite not being trained on multi-instrument mixtures, it seems to achieve compelling results.

4.5. Efficiency

To illustrate the computational efficiency of NMP, we compare peak memory usage and total run time against MI-AMT. Benchmarks were conducted on a 2017 Macbook Pro with a 3.1GHz Quad Core Intel Core i7 CPU and 16GB 2133MHz LPDDR3 Memory. All benchmarks were measured using first a "short" (.35 second) file of white noise to approximate the system's overhead, and a "long" (7 minute 45 second) file from the Slakh dataset in order to show a more realistic input for each method. Audio files were resampled to the expected sampling rate of the method before measurement. We find that both methods are comparable in estimated overhead, with NMP using 490 MB peak memory and taking 7 s and MI-AMT using 561 MB and taking 10 s; however on the long file, NMP substantially outperforms MI-AMT, using only 951 MB peak memory and taking 24 s, while MI-AMT used 3.3 GB and took 96 s. It's interesting to note that the peak memory of the instrument-specific models is even higher, with OF using 5.4 GB and Vocano using 8.5 GB.

5. CONCLUSIONS

We demonstrate that the proposed low-resource neural network-based model (NMP) can be successfully applied to instrument-agnostic polyphonic note transcription and MPE. NMP outperforms a recent strong baseline note estimation model across five different datasets, and performs similarly to deep salience for MPE. Further, we see that the use of harmonic stacking allows our model to remain low-resource while maintaining its performance. When compared with instrument-specific models, we see that NMP achieves state-of-the-art results on GuitarSet. It however did not outperform the instrument-specific models for piano and vocals. Nevertheless, NMP has the benefit of being a "one-size-fits-all" solution, and has much lower computational requirements. We hope to encourage further research into low-resource, multi-purpose AMT systems and believe that the proposed solution can be a valuable baseline.

Future work could explore low-resource transcription of audio mixtures containing many instruments, and the use of offset predictions in this low-resource setting. The note event creation method proposed is based on heuristics, and more carefully designed models similar to [16, 17] would likely result in note-event creation improvements. While this work aimed to create a lightweight model from the start, we did not explore classic model pruning or compression techniques, which would further improve the efficiency. Finally, the

interaction between the note and multipitch outputs could be explored, for example, to estimate note-level pitch bends.

6. REFERENCES

[1] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Process. Magazine*, vol. 36, no. 1, pp. 20–30, 2019.

[2] L. Su and Y.-H. Yang, "Escaping from the abyss of manual annotation: New methodology of building polyphonic datasets for automatic music transcription," in *Proc. CMMR*, 2015.

[3] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 24, no. 5, pp. 927–939, 2016.

[4] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *Proc. ICLR*, 2019.

[5] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. Engel, "Sequence-to-sequence piano transcription with transformers," *arXiv preprint arXiv:2107.09142*, 2021.

[6] T.-W. Su, Y.-P. Chen, L. Su, and Y.-H. Yang, "TENT: Technique-embedded note tracking for real-world guitar solo recordings," *TISMIR*, vol. 2, no. 1, pp. 15–28, 2019.

[7] A. Wiggins and Y. Kim, "Guitar tablature estimation with a convolutional neural network," in *Proc. ISMIR*, 2019, pp. 284–291.

[8] A. McLeod, R. Schramm, M. Steedman, and E. Benetos, "Automatic transcription of polyphonic vocal music," *Applied Sciences*, vol. 7, no. 12, p. 1285, 2017.

[9] J.-Y. Hsu and L. Su, "VOCANO: A note transcription framework for singing voice in polyphonic music," in *Proc. ISMIR*, 2021.

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

[11] A. Klapuri and M. Davy, *Signal Processing Methods for Music Transcription*. Springer US2007.

[12] Z. Duan, B. Pardo, and C. Zhang, "Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions," *IEEE Trans. Audio Speech Lang. Process.*, vol. 18, no. 8, pp. 2121–2133, 2010.

[13] 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.

[14] M. P. Ryynanen and A. Klapuri, "Polyphonic music transcription using note event modeling," in *Proc. WASPAA*, 2005.

[15] Z. Duan, J. Han, and B. Pardo, "Multi-pitch streaming of harmonic sound mixtures," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 22, no. 1, pp. 138–150, 2013.

[16] E. Benetos, "Polyphonic note and instrument tracking using linear dynamical systems," in *Proc. AES Int. Conf. Semantic Audio*, 2017.

[17] A. Ycart and E. Benetos, "Polyphonic music sequence transduction with meter-constrained LSTM networks," in *Proc. ICASSP*, 2018, pp. 386–390.

[18] R. Nishikimi, E. Nakamura, M. Goto, K. Itoyama, and K. Yoshii, "Bayesian singing transcription based on a hierarchical generative model of keys, musical notes, and F0 trajectories," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 28, pp. 1678–1691, 2020.

[19] R. M. Bittner, J. Salamon, J. J. Bosch, and J. P. Bello, "Pitch contours as a mid-level representation for music informatics," in *Proc. AES Int. Conf. Semantic Audio*, 2017.

[20] S. Ewert and M. B. Sandler, "An augmented Lagrangian method for piano transcription using equal loudness thresholding and LSTM-based decoding," in *Proc. WASPAA*, 2017, pp. 146–150.

[21] R. Nishikimi, E. Nakamura, M. Goto, and K. Yoshii, "Audio-to-score singing transcription based on a CRNN-HSMM hybrid model," *APSIPA Trans. Signal Inf. Process.*, vol. 10, 2021.

[22] M. Mauch, C. Cannam, R. Bittner, G. Fazekas, J. Salamon, J. Dai, J. Bello, and S. Dixon, "Computer-aided melody note transcription using the Tony software: Accuracy and efficiency," in *Proc. Int. Conf. Tech. Music Notation Representation*, 2015.

[23] J. Balhar, "Melody extraction using a harmonic convolutional neural network," in *Proc. ISMIR*, 2018, p. 4.

[24] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and Frames: Dual-objective piano transcription," in *Proc. ISMIR*, 2018, pp. 50–57.

[25] C.-Y. Liang, L. Su, Y.-H. Yang, and H.-M. Lin, "Musical offset detection of pitched instruments: The case of violin," in *Proc. ISMIR*, 2015, pp. 281–287.

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

[27] M. Fuentes, R. Bittner, M. Miron, G. Plaja, P. Ramoneda, V. Lostanlen, D. Rubinstein, A. Jansson, T. Kell, K. Choi, and et al., "mirdata v.0.3.0," Jan 2021.

[28] E. Molina, A. M. Barbancho-Perez, L. J. Tardon-Garcia, I. Barbancho-Perez *et al.*, "Evaluation framework for automatic singing transcription," in *Proc. ISMIR*, 2014, pp. 567–572.

[29] Q. Xi, R. M. Bittner, J. Pauwels, X. Ye, and J. P. Bello, "Guitarset: A dataset for guitar transcription." in *Proc. ISMIR*, 2018, pp. 453–460.

[30] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, "Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity," in *Proc. WASPAA*, 2019, pp. 45–49.

[31] M. Miron, J. J. Carabias-Orti, J. J. Bosch, E. Gómez, and J. Janer, "Score-informed source separation for multichannel orchestral recordings," *Jour. of Electrical Computer Engineering*, vol. 2016, 2016.

[32] T.-S. Chan, T.-C. Yeh, Z.-C. Fan, H.-W. Chen, L. Su, Y.-H. Yang, and R. Jang, "Vocal activity informed singing voice separation with the ikala dataset," in *Proc. ICASSP*, 2015, pp. 718–722.

[33] 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.

[34] Y.-T. Wu, B. Chen, and L. Su, "Multi-instrument automatic music transcription with self-attention-based instance segmentation," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 28, pp. 2796–2809, 2020.