# DEEP SALIENCE REPRESENTATIONS FOR $F_0$ ESTIMATION IN POLYPHONIC MUSIC

**Rachel M. Bittner[1*], Brian McFee[1,2], Justin Salamon[1], Peter Li[1], Juan P. Bello[1]**
[1]Music and Audio Research Laboratory, New York University, USA
[2]Center for Data Science, New York University, USA
[*]Please direct correspondence to: rachel.bittner@nyu.edu

## ABSTRACT

Estimating fundamental frequencies in polyphonic music remains a notoriously difficult task in Music Information Retrieval. While other tasks, such as beat tracking and chord recognition have seen improvement with the application of deep learning models, little work has been done to apply deep learning methods to fundamental frequency related tasks including multi-$f_0$ and melody tracking, primarily due to the scarce availability of labeled data. In this work, we describe a fully convolutional neural network for learning salience representations for estimating fundamental frequencies, trained using a large, semi-automatically generated $f_0$ dataset. We demonstrate the effectiveness of our model for learning salience representations for both multi-$f_0$ and melody tracking in polyphonic audio, and show that our models achieve state-of-the-art performance on several multi-$f_0$ and melody datasets. We conclude with directions for future research.

## 1. INTRODUCTION

Estimating fundamental frequencies in polyphonic music remains an unsolved problem in Music Information Retrieval (MIR). Specific cases of this problem include multi-$f_0$ tracking, melody extraction, bass tracking, and piano transcription among others. Percussion, overlapping harmonics, high degrees of polyphony, and masking make these tasks notoriously difficult. Furthermore, training and benchmarking is difficult due to the limited amount of human-labeled $f_0$ data available.

Historically, most algorithms for estimating fundamental frequencies in polyphonic music have been built on heuristics. In melody extraction, two algorithms that have retained the best performance are based on pitch contour tracking and characterization [8,27]. Algorithms for multi-$f_0$ tracking and transcription have been based on heuristics such as enforcing spectral smoothness and emphasizing harmonic content [17], comparing properties of co-occurring spectral peaks/non-peaks [11], and combining time and frequency-domain periodicities [29]. Other approaches to multi-$f_0$ tracking are data-driven and require labeled training data, e.g. methods based on supervised NMF [32], PLCA [3], and multi-label discriminative classification [23]. For melody extraction, machine learning has been used to predict the frequency bin of an STFT containing the melody [22], and to predict the likelihood an extracted frequency trajectory is part of the melody [4].

There are a handful of datasets with fully-annotated continuous-$f_0$ labels. The Bach10 dataset [11] contains ten 30-second recordings of a quartet performing Bach chorales. The Su dataset [30] contains piano roll annotations for 10 excerpts of real-world classical recordings, including examples of piano solos, piano quintets, and violin sonatas. For melody tracking, the MedleyDB dataset [5] contains melody annotations for 108 full length tracks that are varied in musical style.

More recently, deep learning approaches have been applied to melody and bass tracking in specific musical scenarios, including a BLSTM model for singing voice tracking [25] and fully connected networks for melody [2] and bass tracking [1] in jazz music. In multi-$f_0$ tracking, deep learning has also been applied to solo piano transcription [7,28], but nothing has been proposed that uses deep learning for multi-$f_0$ tracking in a more general musical context. In speech, deep learning has been applied to both pitch tracking [14] and multiple pitch tracking [18], however there is much more labeled data for spoken voice, and the space of pitch and spectrum variations is quite different than what is found in music.

The primary contribution of this work is a model for learning pitch salience representations using a fully convolutional neural network architecture, which is trained using a large, semi-automatically annotated dataset. Additionally, we present experiments that demonstrate the usefulness of the learned salience representations for both multi-$f_0$ and melody extraction, outperforming state-of-the-art approaches in both tasks. All code used in this paper, including trained models, is made publicly available. [1]

## 2. SALIENCE REPRESENTATIONS

Pitch salience representations are time-frequency representations that aim to measure the saliency (i.e. perceived am-

[1] github.com/rabitt/ismir2017-deepsalience

plitude/energy) of frequencies over time. They typically rely on the assumption that sounds humans perceive as having a pitch have some kind of harmonic structure. The ideal salience function is zero everywhere where there is no perceptible pitch, and a positive value that reflects the pitches' perceived loudness at the fundamental frequency. Salience representations are core components of a number of algorithms for melody [8, 12, 27] and multi-$f_0$ tracking [17,26]. Computations of salience representations usually perform two functions: (1) de-emphasize un-pitched or noise content (2) emphasize content that has harmonic structure.

The de-emphasis stage can be performed in a variety of ways, including harmonic-percussive source separation (HPSS), re-weighting frequency bands (e.g. using an equal loudness filter or a high pass filter), peak picking, or suppressing low amplitude or noise content [8, 12, 17, 26, 27]. In practice most salience functions also end up emphasizing harmonics and subharmonics because they are difficult to untangle from the fundamental, especially in complex polyphonies. The many parameters of these filtering and smoothing steps are typically set manually.

Harmonic content is most commonly emphasized via harmonic summation, which re-weights the input representation across frequency, where frequency bins in the salience representation are a weighted sum of harmonically related bins in the input representation [17, 27]. The weights in this summation vary from method to method, and are usually chosen heuristically based on assumptions about the data. In another variant, the input representation is modeled using non-negative least squares to a manually constructed set of ideal harmonic templates [19]. The Fan Chirp transform [9] uses harmonic information in the transform itself, thus directly performing the harmonic "weighting".

In melody extraction, the salience representation has been found to be a bottleneck in algorithmic performance [4], often because large portions of the melody are not emphasized. In particular, the salience representation used in Melodia [27] was found to emphasize vocal content well, but often miss instrumental content.

The combination of HPSS, equalization, and harmonic summation to emphasize pitched content and suppress the rest can be naturally extended in the context of deep learning architectures. For example, a simple version of HPSS performs median filtering with one kernel in time and frequency, and assigns bins to the harmonic or percussive component by a max filtering operation [13]. The harmonic and percussive decompositions can be cascaded to compute, for example, the harmonic component of the percussive signal as in [10, 25] to recover content that is not strongly activated by vertical or horizontal median filters such as singing voice. This cascade of median filtering can be naturally extended to a convolutional neural network setting, where instead of using only two manually set kernels, any number of kernels can be learned and their outputs combined in order to generalize to many types of musical sounds. Similarly, the parameters of harmonic summation can be implicitly learned by using an input representation that aligns harmonically related content—namely we introduce the *harmonic* CQT which we describe in Section 3.1. Furthermore, with a convolutional architecture, the parameters of the de-noising stage and the harmonic emphasis stage can be learned jointly.

## 3. METHOD

We frame our approach as a de-noising problem as depicted in Figure 1: given a time-frequency representation (e.g. a CQT), learn a series of convolutional filters that will produce a salience representation with the same shape in time and frequency. We constrain the target salience representation to have values between 0 and 1, where large values should occur in time-frequency bins where fundamental frequencies are present.

### 3.1 Input Representation

In order to better capture harmonic relationships, we use a *harmonic* constant-Q transform (HCQT) as our input representation. The HCQT is a 3-dimensional array indexed by harmonic, frequency, and time: $\mathcal{H}[h, t, f]$, measures the $h$th harmonic of frequency $f$ at time $t$. The harmonic $h = 1$ refers to the fundamental, and we introduce the notation $\mathcal{H}[h]$ to denote harmonic $h$ of the "base" CQT $\mathcal{H}[1]$. For any harmonic $h > 0$, $\mathcal{H}[h]$ is computed as a standard CQT where the minimum frequency is scaled by the harmonic: $h \cdot f_{\text{min}}$, and the same frequency resolution and number of octaves is shared across all harmonics. The resulting representation $\mathcal{H}$ is similar to a color image, where the $h$ dimension is the *depth*.

In a standard CQT representation, the $k$th frequency bin measures frequency $f_k = f_{\text{min}} \cdot 2^{k/B}$ for $B$ bins per octave. As a result, harmonics $h \cdot f_k$ can only be directly measured for $h = 2^n$ (for integer $n$), making it difficult to capture odd harmonics. The HCQT representation, however, conveniently aligns harmonics across the first dimension, so that the $k$th bin of $\mathcal{H}[h]$ has frequency $f_k = h \cdot f_{\text{min}} \cdot 2^{k/B}$, which is exactly the $h$th harmonic of the $k$th bin of $\mathcal{H}[1]$. By aligning harmonics in this way, the HCQT is amenable to modeling with two-dimensional convolutional neural networks, which can now efficiently exploit locality in time, frequency, and harmonic.

In this work, we compute HCQTs with $h \in \{0.5, 1, 2, 3, 4, 5\}$: one subharmonic below the fundamental (0.5), the fundamental (1), and up to 4 harmonics above the fundamental. Our hop size is $\approx$11 ms in time, and we compute 6 octaves in frequency at 60 bins per octave (20 cents per bin) with minimum frequency at $h = 1$ of $f_{\text{min}} = 32.7$ Hz (i.e. C1). We include a subharmonic in addition to harmonics to help disambiguate between the fundamental frequency and the first harmonic, whose patterns of upper harmonics are often similar – for the fundamental, the first subharmonic should have low energy, where for the first harmonic, a subharmonic below it will have energy. Our implementation is based on the CQT implementation in `librosa` [21].

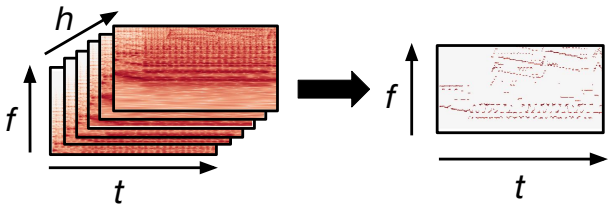


**Figure 1**. Input HCQT (left) and target salience function (right).

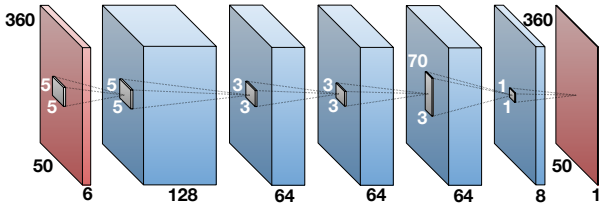


**Figure 2**. CNN architecture. The input to each layer is batch-normalized. The output of each layer is passed through a rectified linear unit activation function except the last layer which is passed through a sigmoid.

### 3.2 Output Representation

The target outputs we use to train the model are time-frequency representations with the same shape as $\mathcal{H}[1]$. Ground truth fundamental frequency values are quantized to the nearest time-frequency bin, and given magnitude $= 1$ in the target representation. The targets are Gaussian blurred in frequency such that the energy surrounding a ground truth frequency decays to zero within a quarter-tone, in order to soften the penalty for near-correct predictions during training. Additionally, since the data is human labeled it may not be accurate to 20 cents, so we do not necessarily want to label nearby frequencies as "wrong". Similar training label "blurring" techniques have been shown to help the performance of models for beat/downbeat tracking [6] and structural boundary detection [31].

xunlian biaoqian mohu jishu

### 3.3 Model

Our model uses a fully convolutional architecture, with 5 convolutional layers of varying dimensionality, as illustrated in Figure 2. The first two layers have 128 and 64 (5 x 5) filters respectively, which cover approximately 1 semitone in frequency and 50 ms in time. The following two layers each have 64 (3 x 3) filters, and the final layer has 8 (70 x 3) filters, covering 14 semitones in frequency to capture relationships between frequency content within an octave. At each layer, the convolutions are zero padded such that the input shape is equal to the output shape in the time-frequency dimension. The input to each layer is batch normalized [15], and the outputs are passed through rectified linear units. The final layer uses logistic activation, mapping each bin's output to the range $[0, 1]$. The predicted saliency map can be interpreted as a likelihood score of each time-frequency bin belonging to an $f_0$ contour. Note that we do not include pooling layers, since we do *not* want to be invariant to small shifts in time frequency.

The model is trained to minimize cross entropy:

$$L(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}) \qquad (1)$$

where both $y$ and $\hat{y}$ are continuous values between 0 and 1. We fit our model using the Adam [16] optimizer.

## 4. MULTIPLE-$F_0$ TRACKING EXPERIMENTS

We first explore the usefulness of our model when trained to produce a multi-$f_0$ salience representation.

### 4.1 Data Generation

Because there is no large human-labeled dataset to use for training, we generate a dataset from a combination of human and machine generated $f_0$ annotations by leveraging multitrack data. Our total dataset contains 240 tracks from a combination of the 108 MedleyDB multitrack dataset [5] and a set of 132 pop music multitracks. The pop multitrack set consists of western popular music from the 1980s through today, and were obtained from a variety of sources and are not available for redistribution—because of this we only use these examples during training. The tracks are split into train, validate, and test groups using an artist-conditional randomized split (i.e. tracks belonging to the same artist must all belong to the same group). The test set is constrained to contain only tracks from MedleyDB, and contains 28 full-length tracks. The training and validation sets contain 184 and 28 full-length tracks respectively, totaling to about 10 hours of training data and 2 hours of validation data.

Each multitrack in the dataset contains mixes and isolated stems, and a subset of these stems contain human-labeled $f_0$ annotations. To have a mix where all pitched content is annotated, we re-create partial mixes by combining any stems with human annotations, all stems with monophonic instruments (e.g. electric bass), and all percussive stems, effectively creating mixes that are similar to the originals, but with all "unknown" pitch content removed. The stems are linearly mixed with weights estimated from the original mixes using a least squares fit. The human-labeled $f_0$ annotations are directly added to the ground truth labels. Annotations for monophonic instrument stems without human labels are created by running `pYIN` [20] and using the output as a proxy for ground truth.

### 4.2 Results

To generate multi-$f_0$ output, we need to explicitly select a set of fundamental frequency values for each time frame from our salience representation. A natural way to do this

would be to threshold the representation at 0.5, however since the model is trained to reproduce Gaussian-blurred frequencies, the values surrounding a high energy bin are usually above 0.5 as well, creating multiple estimates very close to one another. Instead, we perform peak picking on the learned representation and select a minimum amplitude threshold by choosing the threshold that maximizes the multi-$f_0$ accuracy on the validation set.

We evaluate the model on three datasets: the Bach10 and Su datasets, and the test split of the MedleyDB data described in Section 4.1, and compare to well-performing baseline multi-$f_0$ algorithms by Benetos [3] and Duan [11].

Figure 3 shows the results for each algorithm on the three datasets. We see that our CNN model under-performs on Bach10 compared to Benetos' and Duan's models by about 10 percentage points, but outperforms both algorithms on the Su and MedleyDB datasets. We attribute the difference in performance across these datasets to the way each model was trained. Both Benetos' and Duan's methods were in some sense developed with the Bach10 dataset in mind simply because it has been one of the few available test sets when the algorithms were developed. On the other hand, our model was trained on data most similar to the MedleyDB test set, so it is unsurprising that it performs better on this set. The Bach10 dataset is homogeneous (as can be seen by the small variance in performance across all methods), and while our model performs obtains higher scores on the Bach10 dataset than the other two used for evaluation, this dataset only measures how well an algorithm performs on simple 4-part harmony classical recordings. Indeed, we found that on the MedleyDB test set, both Benetos' and Duan's models perform best (50% and 48% accuracy respectively) on the example that is most similar to the Bach10 data (a string quartet), and our approach performs similarly on that track to the overall performance on the Bach10 set with 59% accuracy.

To get a better sense of the track level performance, Figure 4 displays the difference between the CNN accuracy and the best accuracy of Benetos and Duan's model per track. In addition to having a better score on average for MedleyDB (from Figure 3), we see that the CNN model outperforms the other two models on every track on MedleyDB by quite a large margin. We see a similar result for the Su dataset, though on one track (Beethoven's Moonlight sonata) we have a lower score than Benetos. A qualitative analysis of this track showed that our algorithm retrieves the melody and the bass line, but fails to emphasize several notes that are part of the harmony line. Unsurprisingly, on the Bach10 dataset, the other two algorithms outperform our approach for every track.

To further explain this negative result, we explore how our model will perform in an oracle scenario by constraining the maximum polyphony to 4 (the maximum for the Bach10 dataset) and look at the accuracy when we vary the detection threshold. Figure 5 shows the CNN's average accuracy on the Bach10 dataset as a function of the detection thresholds. The solid dotted line shows the threshold automatically estimated from the validation set. For the Bach10 dataset, the optimal threshold is much lower (0.05 vs. 0.3), and the best performance (63% accuracy) gets closer to that of the other two datasets (68% for Duan and 76% for Benetos). Even in this ideal scenario, the difference in performance is due to recall – similarly to the Su example, our algorithm is good at retrieving the melody and bass lines in the Bach10 dataset, but often misses notes that occur in between. This is likely a result of the characteristics of the artificial mixes in our training set: the majority of automatically annotated (monophonic) stems are either bass or vocals, and there are few examples with simultaneous harmonically related pitch content.

Overall, our model has good precision, even on the Bach10 dataset (where the scores are hurt by recall), which suggests that the learned salience function does a good job of de-emphasizing non-pitched content. However, the low recall on the Bach10 and Su datasets suggests that there is still room for the model to improve on emphasizing harmonic content. Compared to the other two algorithms, the CNN makes fewer octave mistakes (3% of mistakes on MedleyDB compared with 5% and 7% of mistakes for Benetos and Duan respectively), reflected in the difference between the accuracy and chroma accuracy.

While the algorithm improves on the state of the art on two datasets, the overall performance still has a lot of room to improve, with the highest score on the Su dataset reaching only 41% accuracy on average. To explore this further, in Figure 6 we plot the outputs on excerpts of tracks from each of the three datasets. In each of the excerpts, the outputs look reasonably accurate. The top row shows an excerpt from Bach10, and while our model sometimes misses portions of notes, the salient content (e.g. melody and bass) is emphasized. Overall, we observe that the CNN model is good at identifying bass and melody patterns even when higher polyphonies are present, while the other two models try to identify chords, even when only melody and bass are present.

### 4.3 Model Analysis

The output of the CNN for an unseen track from the Su dataset is shown in Figure 7. $\mathcal{H}[1]$ is plotted in the left plot, and we can see that it contains a complex polyphonic mixture with many overlapping harmonics. Qualitatively, we see that the CNN was able to de-noise the input representation and successfully emphasize harmonic content.

To better understand what the model learned, we plot the 8 feature maps from the penultimate layer in Figure 8. The red-colored activations have positive weights and the blue-colored have negative weights in the output filter. Activations (a) and (b) seem to emphasize harmonic content, including some upper harmonics. Interestingly, activation (e) deemphasizes the octave mistake from activation (a), as does activation (d). Similarly, activations (f) and (g) act as a "cut out" for activations (a) and (b), deemphasizing the broadband noise component. Activation (h) appears to deemphasize low-frequency noise.

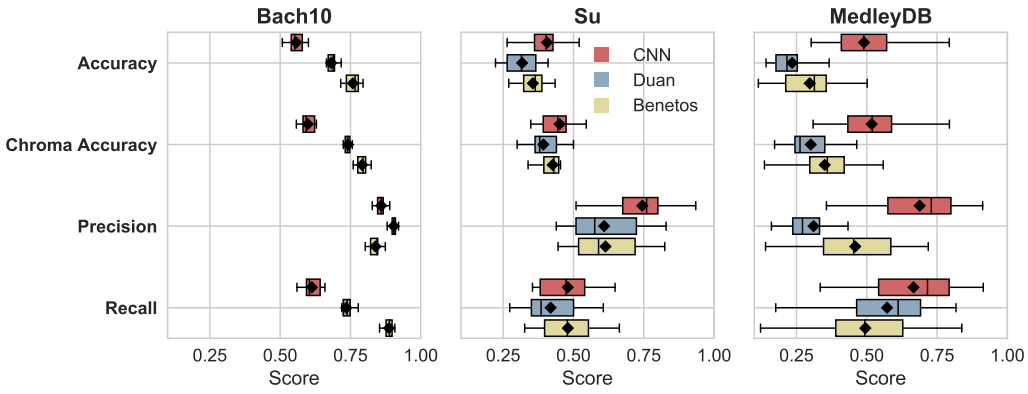


**Figure 3**. A subset of the standard multiple-f0 metrics on the Bach10, Su, and MedleyDB test sets for the proposed CNN-based method, Duan [11], and Benetos [3].

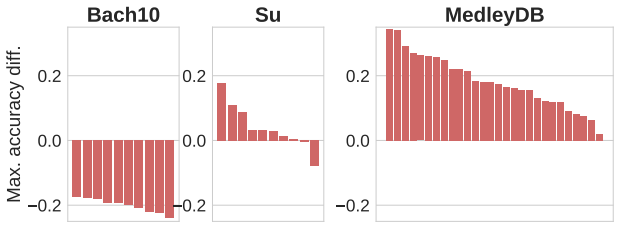


**Figure 4**. The per-track difference in accuracy between the CNN model and the maximum score achieved by Duan or Benetos' algorithm on each dataset. Each bar corresponds to CNN - $\max$(Duan, Benetos) on a single track.

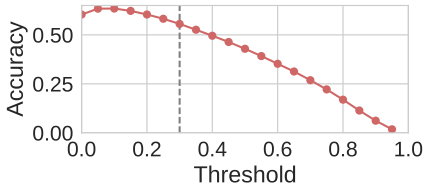


**Figure 5**. CNN accuracy on the Bach10 dataset as a function of the detection threshold, and when constraining the maximum polyphony to 4. The vertical dotted line shows the value of the threshold chosen on the validation set.

## 5. MELODY ESTIMATION EXPERIMENTS

To further explore the usefulness of the proposed model for melody extraction, we train a CNN with identical an architecture on melody data.

### 5.1 Data Generation

Instead of training on HCQTs computed from partial mixes and semi-automatic targets (as described in Section 4.1), we use HCQTs from the original full mixes from MedleyDB, as well as targets generated from the human-labeled melody annotations. The ground truth salience functions contain only melody labels, using the "Melody 2" definition from MedleyDB (i.e. one melody pitch per unit time coming from multiple instrumental sources). We estimate the melody line from the learned salience representation by choosing the frequency with the maximum salience at every time frame. The voicing decision is determined by a fixed threshold chosen on the validation set. In this work we did not explore more sophisticated decoding methods.

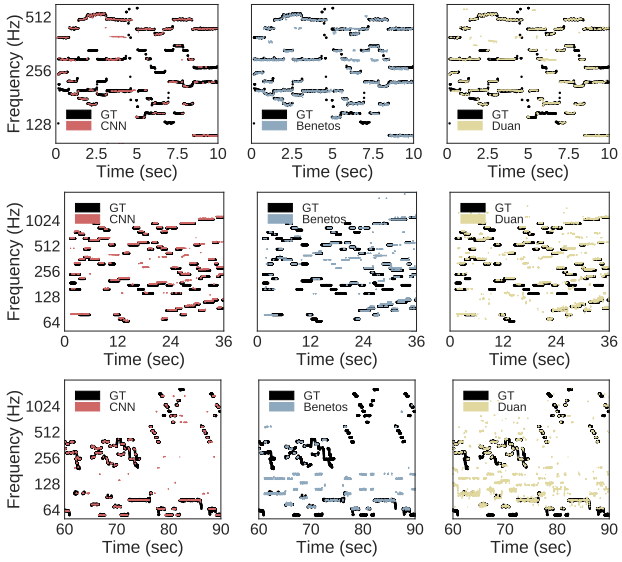


**Figure 6**. Multi-f0 output for each of the 3 algorithms for an example track from the Bach10 dataset (top), the Su dataset (middle), and the MedleyDB test set (bottom)

### 5.2 Results

We compare the output of our CNN-based melody tracking system with two strong, salience-based baseline algorithms: "Salamon" [27] and "Bosch" [8]. The former is a heuristic algorithm that long held the state of the art in melody extraction. The latter recently reached state-of-the-art performance by combining a source-filter based salience function and heuristic rules for contour selection—this model is the current best performing baseline. Figure 9 shows the results of the three methods on the MedleyDB test split described in Section 4.1.

On average, the CNN-based melody extraction outperforms both Bosch and Salamon in terms of Overall (+ 5 and

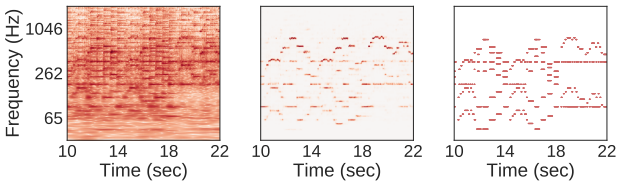


**Figure 7**. (left) Input $\mathcal{H}[1]$, (middle) predicted output, (right) ground truth annotation for an unseen track in the Su dataset.

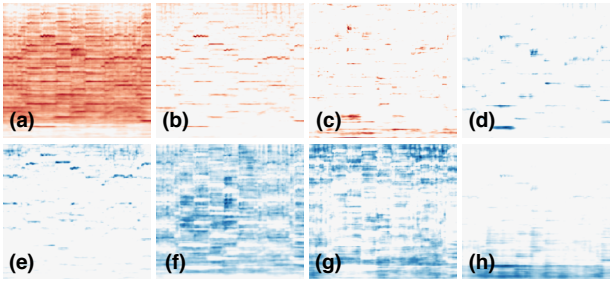


**Figure 8**. Activations from the final convolutional layer with octave height filters for the example given in Figure 7. Activations (a)–(c) have positive coefficients in the output layer, while the others have negative coefficients.

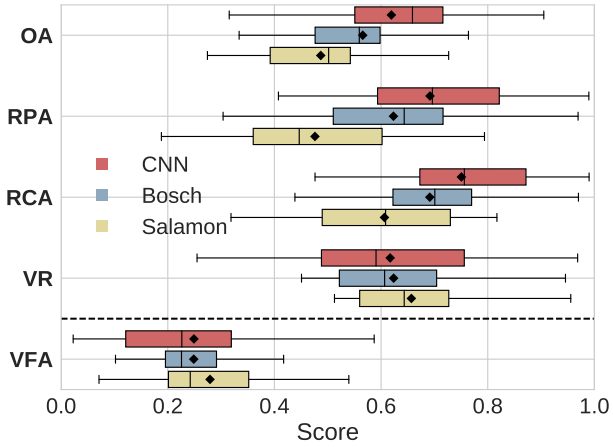


**Figure 9**. Melody metrics – Overall Accuracy (OA), Raw Pitch Accuracy (RPA), Raw Chroma Accuracy (RCA), Voicing Recall (VR) and Voicing False Alarm (VFA) – on the MedleyDB test set for the proposed CNN-based method, Salamon [27], and Bosch [8].

13 percentage points), Raw Pitch (+15 and 22 percentage points), and Raw Chroma Accuracy (+6 and 14 percentage points). The CNN approach is also considerably more varied in performance than the other two algorithms, with a wide range in performance across tracks.

Because we choose the frequency with maximum amplitude in our approach, the Raw Pitch Accuracy measures effectiveness of the salience representation: in an ideal salience representation for melody, the melody should have the highest amplitude in the salience function over time. In our learned salience function, $\approx 62\%$ of the time the melody has the largest amplitude. A qualitative analysis

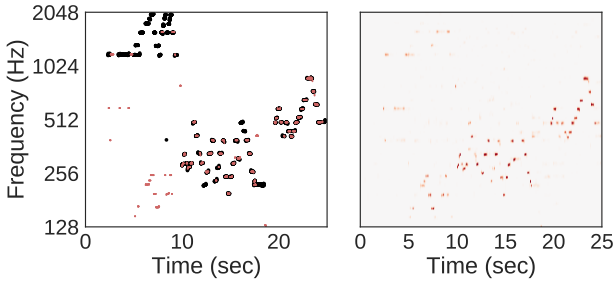


**Figure 10**. CNN output on a track beginning with a piano melody (0 - 10 seconds) and continuing with a clarinet melody (10 - 25 seconds). (left) CNN model melody output in red against the ground truth in back. (right) CNN melody salience output.

of the mistakes made by the CNN method revealed that the vast majority incorrect melody estimates occurred for melodies played by under-represented melody instrument classes in the training set, such as piano and guitar. For example, Figure 10 shows the output of the CNN model for an excerpt beginning with a piano melody and continuing with a clarinet melody. Clarinet is well represented in our training set and the model is able to retrieve most of the clarinet melody, while virtually none of the piano melody is retrieved. Looking at the salience output (Figure 10 right), there is very little energy in the early region where the piano melody is active. This could be a result of the model not being exposed to enough examples of the piano timbre to activate in those regions. Alternatively, in melody salience scenario, the model is trained to suppress "accompaniment" and emphasize melody. Piano is often playing accompaniment in the training set, and the model may not have enough information to untangle when a piano timbre should be emphasized as part of the melody and when it should be suppressed as accompaniment. We note that while in this qualitative example the errors could be attributed to the pitch height, we observed that this was not a consistent factor in other examples.

## 6. CONCLUSIONS

In this paper we presented a model for learning a salience representation for multi-$f_0$ tracking and melody extraction using a fully convolutional neural network. We demonstrated that simple decoding of both of these salience representations yields state-of-the art results for multi-$f_0$ tracking and melody extraction. Given a sufficient amount of training data, this architecture would also be useful for related tasks including bass, piano, and guitar transcription.

In order to further improve the performance of our system, data augmentation can be used to both diversify our training set and to balance the class distribution (e.g. include more piano and guitar). The training set could further be augmented by training on a large set of weakly-labeled data such as the Lakh-midi dataset [24]. In addition to augmentation, there is a wide space of model architectures that can be explored to add more temporal information, such as recurrent neural networks.

## 7. REFERENCES


[1] Jakob Abeßer, Stefan Balke, Klaus Frieler, Martin Pfleiderer, and Meinard Müller. Deep learning for jazz walking bass transcription. In *AES International Conference on Semantic Audio*, 2017.

[2] Stefan Balke, Christian Dittmar, Jakob Abeßer, and Meinard Müller. Data-driven solo voice enhancement for jazz music retrieval. In *ICASSP*, Mar. 2017.

[3] Emmanouil Benetos and Tillman Weyde. An efficient temporally-constrained probabilistic model for multiple-instrument music transcription. In *ISMIR*, pages 701–707, 2015.

[4] Rachel M Bittner, Justin Salamon, Slim Essid, and Juan P Bello. Melody extraction by contour classification. In *ISMIR*, October 2015.

[5] Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan P. Bello. MedleyDB: A multitrack dataset for annotation-intensive MIR research. In *ISMIR*, October 2014.

[6] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *Proc. of the 17th Int. Society for Music Information Retrieval Conf.(ISMIR)*, 2016.

[7] Sebastian Böck and Markus Schedl. Polyphonic piano note transcription with recurrent neural networks. In *Acoustics, speech and signal processing (ICASSP), 2012 ieee international conference on*, pages 121–124. IEEE, 2012.

[8] Juan José Bosch, Rachel M Bittner, Justin Salamon, and Emilia Gómez. A comparison of melody extraction methods based on source-filter modeling. In *ISMIR*, pages 571–577, New York, August 2016.

[9] Pablo Cancela, Ernesto López, and Martín Rocamora. Fan chirp transform for music representation. In *DAFx*, 2010.

[10] Jonathan Driedger and Meinard Müller. Extracting singing voice from music recordings by cascading audio decomposition techniques. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 126–130. IEEE, 2015.

[11] Zhiyao Duan, Bryan Pardo, and Changshui Zhang. Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions. *IEEE TASLP*, 18(8):2121–2133, 2010.

[12] Jean-Louis Durrieu, Bertran David, and Gael Richard. A musically motivated mid-level representation for pitch estimation and musical audio source separation. *IEEE J. on Selected Topics on Signal Processing*, 5(6):1180–1191, Oct. 2011.

[13] Derry Fitzgerald. Harmonic/percussive separation using median filtering. 2010.

[14] Kun Han and DeLiang Wang. Neural network based pitch tracking in very noisy speech. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(12):2158–2168, 2014.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[16] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Anssi Klapuri. Multiple fundamental frequency estimation based on harmonicity and spectral smoothness. *IEEE TASLP*, 11(6):804–816, Nov. 2003.

[18] Yuzhou Liu and DeLiang Wang. Speaker-dependent multipitch tracking using deep neural networks. *The Journal of the Acoustical Society of America*, 141(2):710–721, 2017.

[19] Matthias Mauch and Simon Dixon. Approximate note transcription for the improved identification of difficult chords. In *ISMIR*, pages 135–140, 2010.

[20] Matthias Mauch and Simon Dixon. PYIN: a Fundamental Frequency Estimator Using Probabilistic Threshold Distributions. In *ICASSP*, pages 659–663. IEEE, 2014.

[21] Brian McFee, Matt McVicar, Oriol Nieto, Stefan Balke, Carl Thome, Dawen Liang, Eric Battenberg, Josh Moore, Rachel Bittner, Ryuichi Yamamoto, and et al. librosa 0.5.0, Feb 2017.

[22] Graham E. Poliner and Dan PW Ellis. A classification approach to melody transcription. In *ISMIR*, pages 161–166, London, Sep. 2005.

[23] Graham E Poliner and Daniel PW Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Applied Signal Processing*, 2007(1):154–154, 2007.

[24] Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, COLUMBIA UNIVERSITY, 2016.

[25] François Rigaud and Mathieu Radenen. Singing voice melody transcription using deep neural networks. In *ISMIR*, pages 737–743, 2016.

[26] Matti Ryynänen and Anssi Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music J.*, 32(3):72–86, 2008.

[27] Justin Salamon and Emilia Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE TASLP*, 20(6):1759–1770, Aug. 2012.

[28] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(5):927–939, 2016.

[29] Li Su and Yi-Hsuan Yang. Combining spectral and temporal representations for multipitch estimation of polyphonic music. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(10):1600–1612, 2015.

[30] Li Su and Yi-Hsuan Yang. Escaping from the abyss of manual annotation: New methodology of building polyphonic datasets for automatic music transcription. In *International Symposium on Computer Music Multidisciplinary Research*, pages 309–321. Springer, 2015.

[31] Karen Ullrich, Jan Schlüter, and Thomas Grill. Boundary detection in music structure analysis using convolutional neural networks. In *ISMIR*, pages 417–422, 2014.

[32] Emmanuel Vincent, Nancy Bertin, and Roland Badeau. Adaptive harmonic spectral decomposition for multiple pitch estimation. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):528–537, 2010.