A STREAMLINED ENCODER/DECODER ARCHITECTURE FOR MELODY EXTRACTION

Tsung-Han Hsieh[1,2], Li Su[3] and Yi-Hsuan Yang[2]

[1] Data Science Degree Program, National Taiwan University, Taiwan
[2] Research Center for IT Innovation, Academia Sinica, Taiwan
[3] Institute of Information Science, Academia Sinica, Taiwan
{bill317996,yang}@citi.sinica.edu.tw, lisu@iis.sinica.edu.tw

ABSTRACT

Melody extraction in polyphonic musical audio is important for music signal processing. In this paper, we propose a novel streamlined encoder/decoder network that is designed for the task. We make two technical contributions. First, drawing inspiration from a state-of-the-art model for semantic pixel-wise segmentation, we pass through the pooling indices between pooling and un-pooling layers to localize the melody in frequency. We can achieve result close to the state-of-the-art with much fewer convolutional layers and simpler convolution modules. Second, we propose a way to use the bottleneck layer of the network to estimate the existence of a melody line for each time frame, and make it possible to use a simple argmax function instead of ad-hoc thresholding to get the final estimation of the melody line. Our experiments on both vocal melody extraction and general melody extraction validate the effectiveness of the proposed model.

Index Terms— Melody extraction, encoder/decoder

1. INTRODUCTION

Melody extraction is the task that aims to estimate the fundamental frequency (F0) of the dominant melody. Automatic melody extraction has been an active topic of research in the literature, since it has many important downstream applications in music analysis and retrieval [1–4].

Lately, many deep neural network architectures have been proposed for melody extraction [5–9]. The basic idea of such neural network based methods is to use the neural nets to learn the mapping between a matrix that represents the input audio and another matrix that represents the melody line. For the input, it is usually a time-frequency representation such as the spectrogram, which can be viewed as an $F \times T$ real-valued matrix, where F and T denote the number of frequency bins and time frames, respectively. For the output, it is another $F \times T$ matrix but this time it is a binary matrix indicating the F0 of the melody line for each frame. We only consider music with a single melody line in the music, so at most one frequency bin would be active per frame. It is also possible that there is no melody for some frames. From the training

data, we have a number of such input and output pairs. We can use the difference between the target output and the predicted one to train the neural net in a supervised way.

Existing work has shown that using the neural nets to learn the nonlinear mapping between audio and melody leads to promising result. However, there are two issues that require further research. First, as it is easier for a neural net to deal with continuous values, the output of most existing models (if not all) is actually an $F \times T$ real-valued matrix, not a binary one. This is fine for the training stage, since we can still use cost functions such as cross entropy to measure the difference between a real-valued matrix (the estimated one) and a binary matrix (the groundtruth). However, for the testing stage, we still need to *binarize* the output of the neural net. This binarization cannot be easily achieved simply by picking the frequency bin with the maximal activation per frame, because this would lead to false positives for frames that do not have melody. Therefore, most existing methods have to use a threshold whose value is empirically determined in a rather ad-hoc way for binarization. [7, 9]

The second issue is that existing models that lead to state-of-the-art result in melody extraction benchmark datasets may be overly complicated. For example, the model presented by Lu and Su [9] uses in total 45 convolution or up-convolution layers, using residual blocks for the convolution modules and a sophisticated spatial pyramid pooling layer. The goal of this paper is to propose a streamlined network architecture that has much simpler structure, and that does not need additional post-processing to binarize the model output. With a simple structure, we can better interpret the function of each layer of the network in generating the final result. We hope that the network can have accuracy that is comparable with, if not superior to, the state-of-the-art models.

We make two technical contributions to realize this. First, following Lu and Su [9], we use an encoder/decoder architecture to learn the audio-to-melody mapping. But, while they use the skip connections to pass the output of the convolution layers of the encoder to the up-convolution layers of the decoder, we propose to add links between the pooling layers of the encoder and the un-pooling layers of the decoder, and pass
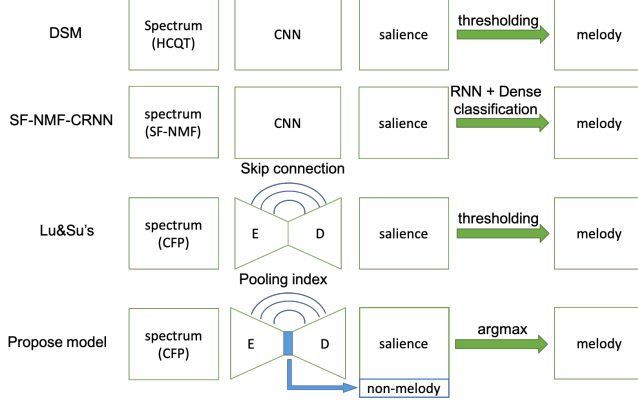
Fig. 1. Comparison of the network architecture of SF-NMF-CRNN [11], DSM [12], Lu & Su's model [9], and the proposed model. (Notation—'E': encoder, 'D': decoder.)

Fig. 2. Details of the proposed model. We detect non-melody activity as a sub-target at the bottleneck layer and concatenate it with the output of the decoder, the salience frequency map.

along the "pooling indices" [10]. While the skip connections they use will be short paths for gradient propagation, there is no trainable weights in pooling and un-pooling layers. We argue from a functional point of view that our method makes it easier for the model to localize the melody. Second, we propose to use the bottleneck layer of the network to estimate the existence of melody per frame, and design a way such that we can simply use argmax to binarize the output. The final model has in total only 7 convolution or up-convolution layers.
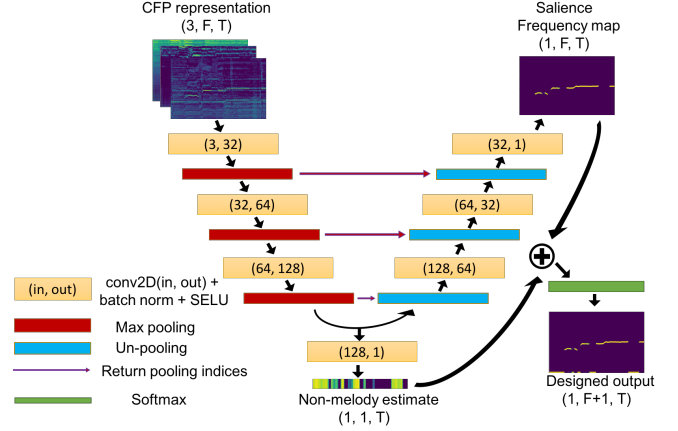
2. RELATED WORK

We show in Fig. 1 the network architectures of three previous methods that are proposed lately. The first one is the deep salience model (DSM) proposed by Bittner *et al.* [7]. It uses a convolutional neural network (CNN) that takes a time-frequency representation of music as the input, and generates a salience map as output for estimating the melody. Finally, they apply a threshold to the salience map to get the binary melody estimate. The second one is the SF-NMF-CRNN model proposed by Basaran *et al.* [11]. Instead of thresholding, it learns recurrent and dense layer to binarize the frequency map. Another model presented by Lu and Su [9], which is based on the DeepLabV3+ model [13], shows that better result for vocal melody extraction can be obtained by an encoder/decoder architecture with skip connections. This model also uses thresholding to binarize the result.

The thresholding operation can be found in many music-related tasks. It can be done with a fixed threshold, an adaptive threshold [9], or other advanced methods [14, 15].

3. PROPOSED MODEL

The system overview is given in Fig. 2. It has a simple encoder/decoder architecture. For the encoder, we use three

convolution layers and three max pooling layers. The output of the encoder is taken as the input by two separate branches of layers. The first branch is simply the decoder that uses three up-convolution layers and three un-pooling layers to estimate the salience frequency map. The second branch uses one convolution layer to estimate the existence of melody per frame, leading to the "non-melody" estimate in the bottom. Finally, the salience map and the non-melody estimate are then concatenated (along the frequency axis), after which we get a binary-valued estimate of the melody line with a simple softmax layer. We give more details of the network below.

3.1. Model Input

While the model can take any audio representation as the input, we choose to use the Combined Frequency and Periodicity (CFP) representation [16]. It contains three parts: the power-scaled spectrogram, generalized cepstrum (GC) [17, 18] and generalized cepstrum of spectrum (GCoS) [19]. The latter two are *periodicity* representations that have been shown useful to multi-pitch estimation (MPE) [20]. Given $\mathbf{X}$, the magnitude of the short-time Fourier transform (STFT) of an input signal, GC and GCoS can be computed as:

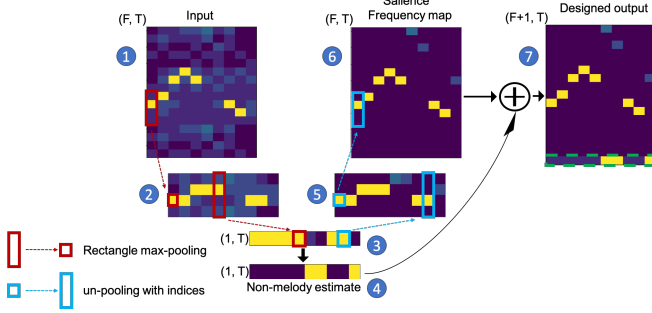$$\mathbf{Z}_{\text{S}} = \sigma_0 \left(\mathbf{W}_f \mathbf{X} \right) , \tag{1}$$

$$\mathbf{Z}_{\text{GC}} = \sigma_1 \left(\mathbf{W}_t \mathbf{F}^{-1} \mathbf{Z}_{\text{S}} \right) , \tag{2}$$

$$\mathbf{Z}_{\text{GCoS}} = \sigma_2 \left(\mathbf{W}_f \mathbf{F} \mathbf{Z}_{\text{GC}} \right) , \tag{3}$$

where $\mathbf{W}_f$ and $\mathbf{W}_t$ are high-pass filters for removing the DC terms, $\mathbf{F}$ an DFT matrix and σ_i activation functions [16, 19].

3.2. Encoder and Decoder

The design of the encoder and decoder represents the first technical contribution of this work. As depicted in Fig. 2,

Fig. 3. An illustration of our model in action. For time frames with melody notes, their values would be close to 1 (bright yellow) in the output of the last encoding layer (③), but close to 0 (dark blue) in the output of the non-melody detector (④). We "reverse the bits" here because we can then concatenate ④ with ⑥ so that a simple argmax on ⑦ can tell us whether there is a melody note and where it is.

we use simple convolution/up-convolution and pooling/un-pooling layers in our model. Moreover, we pass the pooling indices between the pooling and un-pooling layers.

The design is motivated by SegNet [10], a state-of-the-art model for semantic pixel-wise segmentation of images. We found that melody extraction is similar to image segmentation in that both tasks require learning the mapping between a real-valued, dense matrix and a binary-valued, relatively sparser matrix. For melody extraction, the target output is indeed sparse—we only have at most one active entry per column (i.e. per time frame). Therefore, we like to test the idea of SegNet and use pooling indices to inform the un-pooling layers the exact entries picked by the pooling layers in the encoding process. This makes it easier for the decoder to localize the melody in frequency. This is illustrated in Fig. 3.

In each convolution block, we use only one convolution layer with batch normalization and scaled exponential linear units (SELU) [21] as the activation function. The convolution kernel size is (5,5) with padding size (2,2) and stride size (1,1). For the max-pooling layer, we use kernel size (4,1) and pool only along the frequency dimension. The feature map at the bottleneck of the network is a $128 \times T$ matrix.

3.3. Non-melody Detector and ArgMax Layer

The design of the non-melody detector represents the second technical contribution of this work. As depicted in Fig. 2, we learn one additional convolution layer that converts the $128 \times T$ matrix into a $1 \times T$ vector. This vector is then concatenated with the salience map to make an $(F + 1) \times T$ matrix, where the last row corresponds to this vector (see Fig. 3 for an illustration). We then use the argmax function to pick the entries with the maximal value per time frame and return the melody line with the following rule—*if the argmax is the $F + 1$ entry for a frame, we consider that there is no melody*

for that frame. In this way, the output of the model is an $F \times T$ binary matrix with only one or no active entry per frame.

In the model training process, the model output would be compared with the groundtruth output to calculate the loss and to update the network parameters. Therefore, according to our design, the convolution layer we just mentioned would be "forced" to learn whether there is a melody for each frame. Moreover, the frames without melody would tend to have high activation (close to '1'), whereas those with melody would have low activation (close to '0'), as shown in Fig. 3. This is why we call this branch the non-melody detector.

We can view the non-melody detector as a singing voice detector [22–24] when the task is to detect the vocal melody. But, our design is for general melody extraction, not only for vocal melody extraction.

The argmax layer is significant in that we do not need a separate, postprocessing step to discern melody/non-melody frames and to binarize the model output. The non-melody detection and binarization are built-in and trained together with the rest of the network to optimize the accuracy of melody extraction. To our best knowledge (also see Section 2), there is no such a model in the literature.

The argmax layer is not a general solution for any music-related tasks that require binarization. For example, in MPE [16, 20] there are usually multiple active entries per frame.

3.4. Model Update

While Lu and Su [9] use the focal loss [25] to deal with the sparsity of melody entries, we find our model works well with a simple loss function—the binary cross entropy between the estimated melody and the groundtruth one. Model update is done with mini-batch stochastic gradient descent (SGD) and the Adam optimizer. The model is implemented using PyTorch. For reproducibility, we share the source code at `https://github.com/bill317996/Melody-extraction-with-melodic-segnet`.

4. EXPERIMENT

4.1. Experimental Setup

We evaluate the proposed method on general melody extraction for one dataset, and on vocal melody extraction for three datasets. For **general melody extraction**, we use the MedleyDB dataset [26]. Specifically, we use the "melody2" annotation, which is the F0 contours of the melody line drawn from multiple sound sources. Following [11], among the 108 annotated songs in the dataset, we use 67 songs for training, 14 songs for validation and 27 songs for testing.

For **vocal melody extraction**, we use the MIR-1K dataset [1] and a subset of MedleyDB for training. The former contains 1,000 Chinese karaoke clips, whereas the latter contains 48

---

[1]https://sites.google.com/site/unvoicedsoundseparation/mir-1k

songs where the vocal track represents the melody. The testing data are from three datasets: 12 clips from ADC2004, 9 clips from MIREX05,[2] and 12 songs from MedleyDB. We set the training and testing splits of MedleyDB according to [9]. There is no overlap between the two splits.

We compare the performance of our model with the three state-of-the-art deep learning based methods [9, 11, 12] described in Section 2. Moreover, to validate the effectiveness of the non-melody detector branch, we implement an *ablated* version of our model that removes the non-melody detector. For binarization of this method, we run a grid search to find the optimal threshold value using the validation set.

Following the convention in the literature, we use the following metrics for performance evaluation: overall accuracy (OA), raw pitch accuracy (RPA), raw chroma accuracy (RCA), voicing recall (VR) and voicing false alarm (VFA). These metrics are computed by the `mir_eval` [27] library with the default setting—e.g., a pitch estimate is considered correct if it is within 50 cents of the groundtruth one. Among the metrics, OA is often considered more important.

To adapt to different pitch ranges required in vocal and general melody extraction, we use different hyperparameters in computing the CFP for our model. For vocal melody extraction, the number of frequency bins is set to 320, with 60 bins per octave, and the frequency range is from 31 Hz (`B0`) to 1250 Hz (`D#6`). For general melody extraction, the number of frequency bins is set to 400, with 60 bins per octave, and the frequency range is from 20 Hz (`E0`) to 2048 Hz (`C7`). Moreover, since we use more frequency bins for general melody extraction, we increase the filter size of the third pooling layer of the encoder from (4,1) to (5,1) for this task.

We use 44,100 Hz sampling rate, 2,048-sample window size, and 256-sample hop size for computing the STFT. Moreover, to facilitate training the model with mini-batches, we divide the training clips into fixed-length segments of $T = 256$ frames, which is nearly 1.5 seconds. According to our implementation, the model training can converge within 20 minutes with a single GTX1080ti GPU.

4.2. Result

Table 1 first lists the performance of vocal melody extraction for three datasets. We see that the proposed model compares favorably with DSM [12] and Lu & Su's model [9], leading to the highest OA for the ADC 2004 and MedleyDB datasets. In particular, the proposed model outperforms the two prior arts greatly for MedleyDB, the most challenging dataset among the three. We also see that the proposed method outperforms DSM in VFA consistently across the three datasets, meaning that our model leads to fewer false alarms. This may be attributed to the built-in non-vocal detector.

The bottom of Table 1 shows the result of general melody extraction. The proposed method outperforms DSM [12] and

---

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

ADC2004 (vocal melody)					
Method	VR↑	VFA↓	RPA↑	RCA↑	OA↑
DSM [12]	**92.9**	50.5	77.1	78.8	70.8
Lu & Su's [9]	73.8	**3.0**	71.7	74.8	74.9
ours	91.1	19.2	**84.7**	**86.2**	**83.7**
ours (ablated)	74.3	6.1	72.0	75.6	75.1
MIREX05 (vocal melody)					
DSM [12]	**93.6**	42.8	76.3	77.3	69.6
Lu & Su's [9]	87.3	**7.9**	**82.2**	**82.9**	**85.8**
ours	84.9	13.3	75.4	76.6	79.5
ours (ablated)	71.9	12.6	66.3	67.8	73.8
MedleyDB (vocal melody)					
DSM [12]	**88.4**	48.7	**72.0**	**74.8**	66.2
Lu & Su's [9]	77.9	22.4	68.3	70.0	70.0
ours	73.7	**13.3**	65.5	68.9	**79.7**
ours (ablated)	62.1	14.1	53.1	58.8	68.4
MedleyDB (general melody)					
DSM [12]	60.9	**24.3**	**75.1**	69.2	61.7
CRNN [11]	69.8	31.0	71.4	**76.5**	**64.3**
ours	**70.9**	26.2	57.2	62.5	**64.3**
ours (ablated)	66.5	27.1	53.3	58.6	59.8

Table 1. Experiment results on several datasets. The ablated version of our model does not use the non-melody detector. The arrow next to each of the five performance metrics indicates whether the result is the higher or the lower the better. Please visit our github repo for the standard deviation values.

compares favorably with CRNN [11]. In general, this suggests that our simple model is effective for both vocal melody and general melody extraction.

A closer examination of the results reveals that, compared to existing methods, our model is relatively weaker in the two pitch-related metrics, RPA and RCA, especially for MedleyDB. For example, our model suffers from high frequency noises and make the wrong prediction sporadically. Detailed error analysis of our model can be found in our GitHub repo.

Table 1 also shows that our model outperforms its ablated version almost consistently across the five metrics and the four datasets, validating the effectiveness of the non-melody detector. Although not shown in the table, we have implemented another ablated version of our model that replaces CFP with the constant-Q transform (CQT). This would decrease the OA by about 10% for vocal melody extraction.

5. CONCLUSION

We have introduced a streamlined encoder/decoder architecture that is designed for melody extraction. It employs only 7 convolution or up-convolution layers. Due to the use of a built-in non-melody detector, we do not need further post-processing of the result. The code is public and we hope it contributes to other music tasks.

6. REFERENCES

[1] J. Salamon, E. Gómez, D. P. W. Ellis, and G. Richard, "Melody extraction from polyphonic music signals: Approaches, applications, and challenges," *IEEE Signal Processing Magazine*, vol. 31, no. 2, pp. 118–134, 2014.

[2] N. Kroher and E. Gómez, "Automatic transcription of flamenco singing from polyphonic music recordings," *IEEE/ACM Trans. Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 901–913, 2016.

[3] R. M. Bittner et al., "Pitch contours as a mid-level representation for music informatics," in *AES Int. Conf. Semantic Audio*, 2017.

[4] S. Beveridge and D. Knox, "Popular music and the role of vocal melody in perceived emotion," *Psychology of Music*, vol. 46, no. 3, pp. 411–423, 2018.

[5] F. Rigaud and M. Radenen, "Singing voice melody transcription using deep neural networks.," in *ISMIR*, 2016, pp. 737–743.

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

[7] R. M. Bittner et al., "Deep salience representations for $f0$ estimation in polyphonic music," in *Proc. ISMIR*, 2017, pp. 63–70.

[8] L. Su, "Vocal melody extraction using patch-based CNN," in *Proc. ICASSP*, 2018.

[9] W.-T. Lu and L. Su, "Vocal melody extraction with semantic segmentation and audio-symbolic domain transfer learning," in *Proc. ISMIR*, 2018, pp. 521–528, [Online] https://github.com/s603122001/Vocal-Melody-Extraction.

[10] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[11] D. Basaran, S. Essid, and G. Peeters, "Main melody extraction with source-filter NMF and CRNN," in *Proc. ISMIR*, 2018.

[12] R. M. Bittner et al., "Deep salience representations for f_0 estimation in polyphonic music," in *Proc. ISMIR*, 2017, [Online] https://github.com/rabitt/ismir2017-deepsalience.

[13] L.-C. Chen, Y. Zhu, P. George, S. Florian, and A. Hartwig, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *eprint arXiv:1802.02611*, 2018.

[14] H.-W. Dong and Y.-H. Yang, "Convolutional generative adversarial networks with binary neurons for polyphonic music generation," in *Proc. ISMIR*, 2018.

[15] C. Southall, R. Stables, and J. Hockman, "Improving peak-picking using multiple time-step loss," in *Proc. ISMIR*, 2018, pp. 313–320.

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

[17] T. Kobayashi and S. Imai, "Spectral analysis using generalized cepstrum," *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. 32, no. 5, pp. 1087–1089, 1984.

[18] K. Tokuda, T. Kobayashi, T. Masuko, and S. Imai, "Mel-generalized cepstral analysis: a unified approach to speech spectral estimation.," in *Proc. Int. Conf. Spoken Language Processing*, 1994.

[19] L. Su, "Between homomorphic signal processing and deep neural networks: Constructing deep algorithms for polyphonic music transcription," in *Proc. APSIPA ASC*, 2017.

[20] G. Peeters, "Music pitch representation by periodicity measures based on combined temporal and spectral representations," in *Proc. IEEE ICASSP*, 2006.

[21] G. Klambauer et al., "Self-normalizing neural networks," *arXiv preprint arXiv:1706.02515*, 2017.

[22] B. Lehner, G. Widmer, and R. Sonnleitner, "On the reduction of false positives in singing voice detection," in *Proc. ICASSP*, 2014, pp. 7480–7484.

[23] J. Schlüter and T. Grill, "Exploring data augmentation for improved singing voice detection with neural networks," in *Proc. ISMIR*, 2015.

[24] D. Stoller, S. Ewert, and S. Dixon, "Jointly detecting and separating singing voice: A multi-task approach," in *Proc. Latent Variable Analysis and Signal Separation*, 2018, pp. 329–339.

[25] T.-Y. Lin et al., "Focal loss for dense object detection," *eprint arXiv:1708.02002*, 2017.

[26] R. Bittner et al., "MedleyDB: A multitrack dataset for annotation-intensive MIR research," in *Proc. ISMIR*, 2014, [Online] http://medleydb.weebly.com/.

[27] C. Raffel et al., "mir_eval: a transparent implementation of common mir metrics," in *Proc. ISMIR*, 2014, [Online] https://github.com/craffel/mir_eval.