Multipitch tracking in music signals using Echo State Networks

Peter Steiner, Simon Stone, Peter Birkholz
Institute of Acoustics and Speech Communication
Technische Universität Dresden
Dresden, Germany
peter.steiner@tu-dresden.de

Azarakhsh Jalalvand
IDLab
Ghent University – imec
Ghent, Belgium

***Abstract*—Currently, convolutional neural networks (CNNs) define the state of the art for multipitch tracking in music signals. Echo State Networks (ESNs), a recently introduced recurrent neural network architecture, achieved similar results as CNNs for various tasks, such as phoneme or digit recognition. However, they have not yet received much attention in the community of Music Information Retrieval. The core of ESNs is a group of unordered, randomly connected neurons, i.e., the reservoir, by which the low-dimensional input space is non-linearly transformed into a high-dimensional feature space. Because only the weights of the connections between the reservoir and the output are trained using linear regression, ESNs are easier to train than deep neural networks. This paper presents a first exploration of ESNs for the challenging task of multipitch tracking in music signals. The best results presented in this paper were achieved with a bidirectional two-layer ESN with 20 000 neurons in each layer. Although the final F-score of 0.7198 still falls below the state of the art (0.7370), the proposed ESN-based approach serves as a baseline for further investigations of ESNs in audio signal processing in the future.**

***Index Terms*—Reservoir Computing, Echo State Network, Multipitch, RNN, MIR**

I. Introduction

The fundamental frequency f_0 is the smallest noticeable frequency in a quasiperiodic signal, and related to the perceived pitch p. Detecting the f_0 in a speech signal is nowadays a well-investigated task. Algorithms implemented in software packages, such as Praat [1] or RAPT [2], as well as YIN-based techniques [3], [4] are able to extract the f_0 of speech or monophonic musical instruments without a high computational complexity and without the need for training data.

In music, however, multiple sources can be active at the same time, and each source has its own f_0. Thus, the goal of algorithms for *multiple-*f_0 extraction is to extract all f_0 values present at any time. Such algorithms need to be able to consider unknown polyphony (number of active notes or sources during the same time), different and unknown instruments, and fast note transitions. A common way of simplifying this task is to map the f_0 values to a reduced set of center frequencies of the notes in the Western Music, i.e. the MIDI pitches. In this context, multiple-f_0 estimation can also be considered as a multilabel classification, where each MIDI pitch is a separate class.

MIREX[1] has a challenge for multiple-f_0 detection in musical signals, in which many different approaches have been proposed and evaluated in recent years. The model proposed achieved rank 1 and 2 during the MIREX challenge in 2019.

The very first approaches for multipitch tracking extracted single f_0 values iteratively without using training data. Methods as described in [5]–[7] detect f_0 values iteratively by cancelling identified values and their harmonics, or by smoothly modeling the spectrum using recognized f_0 values. To obtain a sequence of extracted f_0s, models for note transitions extend the capabilities of these algorithms. The f_0 values can be mapped to pitches for multipitch tracking.

More recent algorithms [8]–[12] employed non-negative matrix factorization (NMF). These algorithms are very efficient and decompose complex music signals into basic components, e.g. notes or MIDI pitches. Typically, these algorithms use dictionaries pretrained on audio recordings.

SONIC [13] was the first algorithm for multipitch tracking using neural networks. The features, extracted using auditory filterbanks, were fed into a network of adaptive oscillators. The output of the latter one was used as feature vector for neural networks. The current state of the art is based on Convolutional Neural Networks (CNNs). Thickstun et al. [14] utilized a bank of logarithmically-spaced frequency filters to obtain feature vectors for the CNN. The latter one consisted of two layers with one additional linear classifier for each note on top. Without data augmentation, this algorithm achieved an F-score of 0.7275 on the MusicNet dataset [14]. The recently published algorithm [15] also used a deep CNN and reached an F-score of 0.737 on the MusicNet dataset. They used a combination of two different filterbanks, namely, a non-linear power spectrum and a pitch representation.

Echo State Networks (ESNs), proposed by Herbert Jaeger [16], are a special kind of Recurrent Neural Networks (RNNs) and have achieved comparable results to CNNs in several recognition tasks, such as speech and image recognition [17], [18]. However, they are fairly new in the context of Music Information Retrieval (MIR). Encouraged by the positive

This research was financed by Europäischer Sozialfonds (ESF) and the Free State of Saxony (Application number: 100327771).

[1]https://www.music-ir.org/mirex/wiki/MIREX_HOME

performance in various research areas, this work explored the potential of ESNs for the challenging task of multiple-f_0 estimation. ESNs have some beneficial properties for this task:

- They are suitable for processing temporal information due to recurrent connections, and the training procedure is much easier than concurrent approaches due to less free parameters.
- A sequence of musical notes is strongly context dependent. ESNs offer a long-term memory that can keep information from the past for a long time and can thus model context information.
- They are quite robust against noise and unseen conditions [18].
- Since the input dimension has minimum (almost no) impact on the complexity of the model, they are interesting candidates for processing high dimensional data. In this paper, the feature vector has 512 dimensions.

II. Multipitch Tracking with Echo State Networks

Echo State Networks (ESNs) are a variant of recurrent neural networks (RNNs). In contrast to typical RNN architectures, which usually consist of sequential layers, the fundamental difference is that the input weights and the recurrent connection weights are fixed by random values, and only the output weights are trained using linear regression.

Because of the recurrent connections inside the reservoir, information from previous inputs is retained or "echoing" in the reservoir for a certain amount of time. Thus, depending on the choice of the hyper-parameter values, it can act as a fading long- or short-term memory. Because the neurons inside the reservoir are non-linear, the reservoir acts as a non-linear transformation of the low-dimensional input space into a high-dimensional feature space, where the desired output is a multi-linear function of the transformed features. Moreover it has been shown in [18] that if the reservoir is large enough, both input and recurrent connections can be very sparse with minimum (or no) loss in the performance. The main outline of our proposed ESN-based model for multiple-f_0 estimation is depicted in Figure 1.

A. Framing

The input signal $s[k]$ with the sample index k and the sampling frequency $f_s = 44.1\,\mathrm{kHz}$ was divided into overlapping frames with a frame rate of $100\,\mathrm{Hz}$, a frame length of 4096, and with the frame index n.

B. Feature extraction

For each frame, the same spectral feature vectors as in [19] were calculated, using a bank of 512 sine and cosine filters with logarithmically-spaced frequencies ranging from $50\,\mathrm{Hz}$ to $6000\,\mathrm{Hz}$. We also applied a cosine window to each filter. This is similar to a typical STFT filterbank. The main difference is the spacing of the center frequencies of the frequency bins, which is linear for the STFT and logarithmic here. The extracted feature vectors contained 512 spectral magnitudes for each frame. Before feeding the feature vectors into the ESN, each feature was normalized over time to have zero mean and unit variance to obtain the normalized feature vectors $\mathbf{u}[n]$. This normalization step was done separately for each audio file, from which the means and variances were computed.

C. Echo State Network (ESN)

The main outline of an ESN is depicted in the center of Figure 1. It consists of the input weights $\mathbf{W}^{\mathrm{in}}$, the reservoir weights $\mathbf{W}^{\mathrm{res}}$ and the output weights $\mathbf{W}^{\mathrm{out}}$.

The input weight matrix $\mathbf{W}^{\mathrm{in}}$ has the dimension of $N^{\mathrm{res}} \times N^{\mathrm{in}}$ where $N^{\mathrm{in}} = 512$ and N^{res} are the size of the input feature vector and the size of the reservoir, respectively. All values in this matrix were initialized from a uniform distribution between ± 1.0. Next, each node of the reservoir was only connected to $K^{\mathrm{in}} = 10$ randomly selected input entries. The other connections were set to zero, leading to a very sparse matrix $\mathbf{W}^{\mathrm{in}}$. The input weight matrix was then scaled using the input scaling factor α_{U}, which was a hyper-parameter to be tuned.

The reservoir weight matrix $\mathbf{W}^{\mathrm{res}}$ is a square matrix of the size $N^{\mathrm{res}} \times N^{\mathrm{res}}$, which was initialized from a standard normal distribution. Each reservoir node received values from only $K^{\mathrm{rec}} = 10$ randomly selected other nodes. The other connections were set to zero. The reservoir matrix $\mathbf{W}^{\mathrm{res}}$ was normalized by its largest absolute eigenvalue to achieve a spectral radius $\rho = 1.0$, because it was shown in [16] that the echo state property holds as long as $\rho \leq 1.0$.

It has been shown in [20] that K^{in} and K^{rec} can be fixed regardless the reservoir size and the feature vector size. By tuning α_{U} and ρ, it is possible to balance, how strongly the network memorizes past inputs compared to the present input.

If $\mathbf{r}[n]$ represents the reservoir state, the basic equations to describe the ESN can be written in the following way:

$$\begin{aligned}\mathbf{r}[n] =& (1-\lambda)\mathbf{r}[n-1] + \\ & \lambda f_{\mathrm{res}}(\mathbf{W}^{\mathrm{in}}\mathbf{u}[n] + \mathbf{W}^{\mathrm{res}}\mathbf{r}[n-1] + \mathbf{W}^{\mathrm{bi}}) \end{aligned} \quad (1)$$

$$\mathbf{y}[n] = \mathbf{W}^{\mathrm{out}}\mathbf{r}[n] \quad (2)$$

Equation (1) is a leaky integration of the reservoir neurons. Depending on the leakage $\lambda \in (0, 1]$, the reservoir can act as a long-term or a short-term memory. The reservoir activation function $f_{\mathrm{res}}(\cdot)$ controls the non-linearity of the system. Here, the $\tanh$-function was used, because its lower and upper boundaries of ± 1 ensure stable reservoir states. The bias vector $\mathbf{W}^{\mathrm{bi}}$ with N^{res} entries is an additional bias term, which consists of fixed random values from a uniform distribution between ± 1.0. It was scaled using the bias scaling factor α_{B}, which was a hyper-parameter to be tuned.

Equation (2) shows how to compute the N^{out}-dimensional output vector $\mathbf{y}[n]$ from a given reservoir state $\mathbf{r}[n]$, which was expanded by one interception term. The output is obtained by a linear combination of the reservoir state and the output weight matrix $\mathbf{W}^{\mathrm{out}}$. For training, all reservoir states were collected in the reservoir state collection matrix $\mathbf{R}$, and expanded by

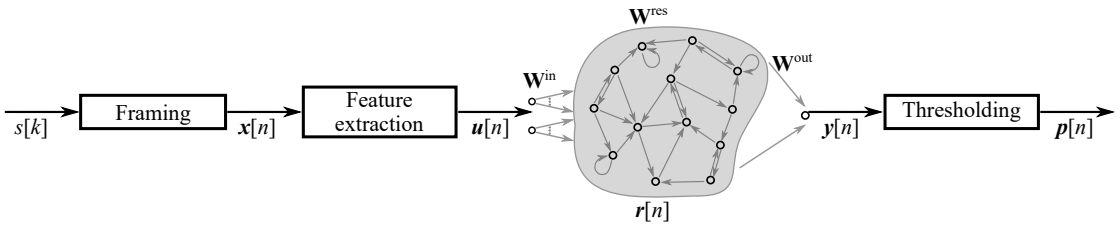

Fig. 1. Outline of the ESN-based proposed model: The input signal $s[k]$ with the sample index k was divided into overlapping frames, from which normalized feature vectors $\mathbf{u}[n]$ were extracted and fed into the reservoir using the input weight matrix $\mathbf{W}^{\text{in}}$. The reservoir consists of unordered and via the reservoir matrix $\mathbf{W}^{\text{res}}$ sparsely connected neurons. The output vector $\mathbf{y}[n]$ with $N^{\text{out}} = 128$ dimensions is a linear combination of the reservoir states $\mathbf{r}[n]$ and the output weight matrix $\mathbf{W}^{\text{out}}$, which was trained using linear regression. Each component of the output vector $\mathbf{y}[n]$ corresponded to one musical note. Using a threshold, the output was converted to a binary sequence of notes $\mathbf{p}[n]$.

one bias term. The desired output vectors $\mathbf{d}[n]$ were collected into the desired output collection matrix $\mathbf{D}$. Afterwards, $\mathbf{W}^{\text{out}}$ was obtained using regularized linear regression (3), i.e. ridge regression to prevent overfitting to the training data. The regularization parameter $\epsilon = 0.0001$ penalized large values in $\mathbf{W}^{\text{out}}$, and $\mathbf{I}$ is the identity matrix.

$$\mathbf{W}^{\text{out}} = \left(\mathbf{R}\mathbf{R}^{\text{T}} + \epsilon\mathbf{I}\right)^{-1}\left(\mathbf{D}\mathbf{R}^{\text{T}}\right) \tag{3}$$

The size of the output weight matrix $N^{\text{out}} \times (N^{\text{res}} + 1)$ determines the total number of free parameters to be trained in ESNs. Each component of the output vector $\mathbf{y}[n]$ corresponded to one musical note.

D. Bidirectional and stacked reservoirs

In the case of bidirectional reservoirs, the input was first fed through the ESN and the reservoir states collected as described before. Next, the input was fed through the same reservoir again, but backwards in time. Afterwards, the reservoir states were again reversed in time. The final reservoir state collection matrix $\mathbf{R}$ was finally built by concatenating the states from the forward and backward pass. This doubled the number of free parameters for the linear regression. For example, the number of features for a reservoir with 500 neurons is 500 in the unidirectional and 1000 in the bidirectional case. The final training remained the same as before.

In the case of stacked reservoirs, the layers were trained sequentially using the same desired outputs in every layer. After fixing the hyper-parameters for one layer, the output of that served as the input for the next layer. By stacking reservoirs, the temporal modeling capacity of a single layer model is extended. In [17], [18], it was shown that stacking reservoirs improved the results for phoneme or digit recognition.

E. Thresholding

After the linear regression, the output $\mathbf{y}[n]$ indicated the presence of every note with continuous values. Ideally, it would be zero for an absent and one for a present note. However, due to the linear regression, the presence is neither binary nor bounded between zero and one. To map this sequence of continuous note presence values into a binary value for the actual notes, an absolute threshold δ was applied to each element of the output vector $\mathbf{y}[n]$, as defined in Equation (4).

$$\mathbf{p}[n] = \begin{cases} 0 & \text{if } \mathbf{y}[n] < \delta \\ 1 & \text{if } \mathbf{y}[n] \geq \delta\ , \end{cases} \tag{4}$$

where $<$ and $\geq$ was the element-wise comparison of the output vector $\mathbf{y}[n]$ with the scalar threshold δ.

III. Experimental setup

A. Dataset

To evaluate the capabilities of our model to transcribe music, the recently introduced MusicNet database [14] was used, which is the largest freely available database. It contains in total 330 classical music recordings, using 11 different instruments. All recordings are mono audio files and sampled with $f_s = 44\,100\,\text{Hz}$. In total, there are more than $30\,\text{h}$ of music with sample-based annotations for instruments, notes and more.

The MusicNet is by default split into a 320 training and 10 test files. The hyper-parameters were tuned solely on the 320 files from the training set, and the test files were just used one time in the end to report measurements. The two state of the art algorithms [15], [19] were evaluated on the same dataset.

B. Measurements

We followed the reference publication [15] and report several metrics using the `mir_eval`-library [21] with standard settings.

The Precision P is defined as the ratio of true positives for each frame ($\text{TP}[n]$) and the summation of $\text{TP}[n]$ and the false positives for each frame ($\text{FP}[n]$). If it is low, many additional notes have been wrongly recognized.

$$P = \frac{\sum_{n=0}^{N-1} \text{TP}[n]}{\sum_{n=0}^{N-1} \left(\text{TP}[n] + \text{FP}[n]\right)} \tag{5}$$

The Recall R is defined as the ratio of $\text{TP}[n]$ and the summation of $\text{TP}[n]$ and the false negatives for each frame ($\text{FN}[n]$). If it is low, many notes have been missed.

$$R = \frac{\sum_{n=0}^{N-1} \text{TP}[n]}{\sum_{n=0}^{N-1} \left(\text{TP}[n] + \text{FN}[n]\right)} \tag{6}$$

The F-score F is the harmonic mean of P and R and determines the overall classification result.

$$F = \frac{2 \cdot P \cdot R}{P + R} \tag{7}$$

In this paper, F served as the objective function to determine the detection threshold δ.

C. Implementation and optimization strategy

The algorithm was developed in Python 3, and was based on [18]. Table I shows the hyper-parameters to be optimized, and the result of optimizing uni- and bidirectional models with one and two layer architectures. The optimization process was conducted using a sequence of grid and line searches.

TABLE I
OVERVIEW OVER ALL HYPER-PARAMETERS TO BE TUNED. THE VALUES SHOW THE SEARCH RANGE AND THE STEP SIZE, IN WHICH THE EXHAUSTIVE GRID SEARCH TOOK PLACE. THE FINAL VALUES OF EVERY MODEL WERE FIXED FOR THE EVALUATION. THE OPTIMIZATION LED TO EQUAL VALUES FOR UNIDIRECTIONAL (U) AND BIDIRECTIONAL (B) RESERVOIRS FOR ALL HYPER-PARAMETERS BUT FOR THE BIAS SCALING α_B IN LAYER 1.

hyper-parameter	Range	Step	Final values	
			Layer 1	Layer 2
Input scaling α_U	$[0.0, 2.0]$	0.1	0.1	1.9
Spectral radius ρ	$[0.0, 1.0]$	0.05	0.8	0.1
Bias scaling α_B	$[0.0, 2.0]$	0.05	1.8 (u) 0.85 (b)	1.35
Leakage λ	$(0.0, 1.0]$	0.1	0.1	0.2
Threshold δ	$[0.0, 1.0]$	0.05	0.35	

The optimization workflow to fix the ESNs hyper-parameters consisted of three steps:

1) The starting point was a grid search across α_U and ρ. These two hyper-parameters needed to be optimized together to determine a trade-off between forward and recurrent connections. Therefore, α_B and λ were fixed to their default values 0.0 and 1.0.
2) Next, a line search was conducted to optimize α_B, while the default value for λ was kept constant. This parameter changes the default operating point of the non-linear neurons in the reservoir, which led to additional non-linearity for the system. Thus, for more non-linear tasks, we expect α_B to increase.
3) Finally, λ was optimized using a line search. This parameter determines the different temporal evolutions of the input compared to the output.

This relatively simple way of independently optimizing the hyper-parameters was based on [18] and made it possible to design a reservoir for the task of multipitch tracking. For every parameter combination, the mean squared error (MSE) between the target and the computed output was reported on the whole training set. After each step, the hyper-parameters leading to the smallest error were fixed and used for the optimization of the next parameter.

To minimize the computational complexity, the reservoir size was fixed at 2000 neurons during the hyper-parameter tuning process. Since the reservoir size tends to be independent from all other hyper-parameters [17], [18], the later evaluation was performed with an increased reservoir size of $20\,000$ neurons, which improved the expressive power of the model.

Considering the multilayer ESN, the hyper-parameters were optimized layerwise. At first, the hyper-parameters for one layer were fixed. Next, the output weight matrix of this layer was trained using linear regression to finalize the layer. Afterwards, its output served as the input for the next layer, which was optimized as before. Due to this cascaded training paradigm, the free parameters and hyper-parameters for an n-layer ESN are the same as for the first n layers in an m-layer ESN (with $n < m$).

After fixing all hyper-parameters in a single unidirectional reservoir, the detection threshold was found by a line search. Now, the F-score on all files from the training set served as the objective function to be maximized. The determined threshold was used for all models.

IV. RESULTS

Table II presents the recognition results of the proposed algorithm with a reservoir size of $20\,000$, the baseline ridge regression model [14], and the current state of the art techniques [15], [19]. The results show that both additional reservoirs and bidirectional structures improved the recognition results. Additional layers can be used to correct errors from previous layers. We can see that this improved the unidirectional results from an F-score of 69.53 in a one-layer-system to 71.13 in a two-layer-system. The precision decreased in case of the two-layer-system, because we used the detection threshold from the first layer for the second reservoir. Using a bidirectional instead of a unidirectional reservoir also increased the F-score, because this increased the temporal context to be considered. Because of the large dataset, this did not lead to overfitting. The system using two layers with $20\,000$ reservoir neurons each in the bidirectional configuration was the best performing system.

TABLE II
PRECISION P, RECALL R AND F-SCORE F FOR DIFFERENT ALGORITHMS EVALUATED ON THE TEST SET OF THE MUSICNET. THE REFERENCE ALGORITHM [19] WAS RETRAINED USING A HOP SIZE OF $10\,\mathrm{ms}$ ACCORDING TO THE MIREX STANDARD. ALL OTHER SETTINGS WERE KEPT THE SAME AS IN THE ORIGINAL CODE.

Method	P	R	F
Ridge Regression [14]	49.0	40.55	44.35
1 layer uni	69.05	70.02	69.53
1 layer bi	69.13	72.20	70.63
2 layer uni	66.43	76.55	71.13
2 layer bi	66.86	77.93	71.98
[19]	69.59	76.19	72.74
[15]	69.34	79.29	73.70

Compared to the model [19], which used exactly the same feature vectors together with a CNN, the ESN still fell short

by $0.76\,\%$ F-score. The two reference approaches presented in [15] could not be retrained with a hop size of $10\,\mathrm{ms}$. Compared with the reported performance (which is unlikely to degrade much with a smaller hop size), the ESN's F-score was $1.72\,\%$ points lower. However, compared to the ridge regression model [14], all ESN-based approaches led to significantly improved results.

One big advantage of ESNs is that the number of free parameters is significantly lower than in CNNs. For example, the model used in [19] had $26\,132\,480$ free parameters, which need to be trained concurrently. Increasing the number of layers leads to more free parameters. The bidirectional ESN with two layers has $2 \times 5\,120\,128 = 10\,240\,256$ parameters in total, which is already significantly less compared to the CNN. Moreover, each layer in the ESN is trained separately. Thus, in each training block, only $5\,160\,129$ parameters are trained using the entire available dataset. This makes the training much more efficient, and decreases the danger of overfitting, even for a relatively large reservoir.

Figure 2 visualizes an excerpt from the file "2106.wav", which is a typical example from the test set of the MusicNet database. It was obtained with the best performing model. Although there were many FP and FN recognitions, most of the notes were recognized correctly.

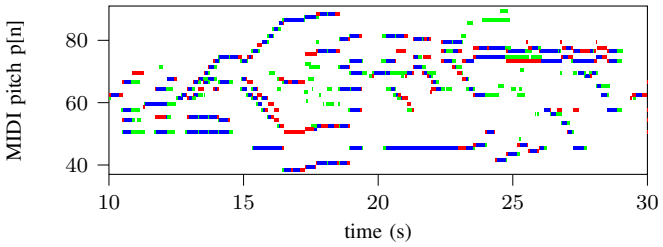

Fig. 2. Transcription result for a part of 2106.wav from the test set. It achieved an F-score of 74.94. Although the red (FN) and green (FP) colors indicate several errors, the blue color (TP) indicates that most of the notes were correctly recognized.

V. Conclusions and Outlook

We presented a new approach for multiple-f_0 estimation in musical signals based on ESNs. The results showed that this very first attempt achieved similar results as a CNN trained on the same amount of data and with the same feature set. In the future, the behaviour of the reservoir with other non-linearities, such as the ReLU-activation, could be carefully studied. The optimization strategy for the hyper-parameters can also be improved by Bayesian Optimization.

Acknowledgment

The parameter optimizations were performed on a Bull Cluster at the Center for Information Services and High Performance Computing (ZIH) at TU Dresden.

References

[1] P. Boersma, "Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound," in *IFA Proceedings 17*, 1993, pp. 97–110.

[2] D. Talkin, *A robust algorithm for pitch tracking (RAPT)*. New York, NY, USA: Elsevier Science Inc., 1995, ch. 14, pp. 495 – 518.

[3] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917 – 1930, 2002.

[4] S. Stone, P. Steiner, and P. Birkholz, "A time-warping pitch tracking algorithm considering fast f0 changes," in *Proc. Interspeech 2017*, 2017, pp. 419–423. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2017-382

[5] A. Klapuri, "Multiple fundamental frequency estimation based on harmonicity and spectral smoothness," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 6, pp. 804 – 816, November 2003.

[6] A. Pertusa and J. M. Inesta, "Multiple fundamental frequency estimation using gaussian smoothness," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2008, pp. 105 – 108.

[7] C. Yeh, A. Roebel, and X. Rodet, "Multiple fundamental frequency estimation and polyphony inference of polyphonic music signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1116 – 1126, August 2010.

[8] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No.03TH8684)*, October 2003, pp. 177 – 180.

[9] P. Smaragdis, B. Raj, and M. Shashanka, "A probabilistic latent variable model for acoustic modeling," in *In Workshop on Advances in Models for Acoustic Processing at NIPS*, 2006.

[10] G. Grindlay and D. P. W. Ellis, "Transcribing multi-instrument polyphonic music with hierarchical eigeninstruments," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1159 – 1169, October 2011.

[11] E. Benetos and S. Dixon, "A shift-invariant latent variable model for automatic music transcription," *Computer Music Journal*, vol. 36, no. 4, pp. 81 – 94, December 2012.

[12] B. Fuentes, R. Badeau, and G. Richard, "Harmonic adaptive latent component analysis of audio and application to music transcription," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 9, pp. 1854 – 1866, September 2013.

[13] M. Marolt, "A connectionist approach to automatic transcription of polyphonic piano music," *IEEE Transactions on Multimedia*, vol. 6, no. 3, pp. 439 – 449, June 2004.

[14] J. Thickstun, Z. Harchaoui, and S. M. Kakade, "Learning features of music from scratch," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. [Online]. Available: https://openreview.net/forum?id=rkFBJv9gg

[15] Y.-T. Wu, B. Chen, and L. Su, "Polyphonic music transcription with semantic segmentation," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 166 – 170.

[16] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," German National Research Center for Information Technology, Tech. Rep. GMD Report 148, 2001. [Online]. Available: http://www.faculty.iu-bremen.de/hjaeger/pubs/EchoStatesTechRep.pdf

[17] F. Triefenbach, K. Demuynck, and J.-P. Martens, "Large vocabulary continuous speech recognition with reservoir-based acoustic models," *IEEE Signal Processing Letters*, vol. 21, no. 3, pp. 311 – 315, March 2014.

[18] A. Jalalvand, K. Demuynck, W. D. Neve, and J.-P. Martens, "On the application of reservoir computing networks for noisy image recognition," *Neurocomputing*, vol. 277, pp. 237 – 248, 2018, hierarchical Extreme Learning Machines.

[19] J. Thickstun, Z. Harchaoui, D. P. Foster, and S. M. Kakade, "Invariances and data augmentation for supervised music transcription," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 2241 – 2245.

[20] A. Jalalvand, F. Triefenbach, K. Demuynck, and J.-P. Martens, "Robust continuous digit recognition using reservoir computing," *Computer Speech & Language*, vol. 30, no. 1, pp. 135 – 158, 2015.

[21] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "MIR_EVAL: A transparent implementation of common MIR metrics," in *ISMIR*, 2014.