Music Transcription Using Deep Learning

Luoqi Li
EE, Stanford University
liluoqi@stanford.edu

Isabella Ni
SCPD CS, Stanford University
nizhongy@stanford.edu

Liang Yang
GS, Stanford University
lyang6@stanford.edu

***Abstract*—Music transcription, as an essential component in music signal processing, contributes to wide applications in musicology, accelerates the development of commercial music industry, facilitates the music education as well as benefits extensive music lovers. However, the work relies on a lot of manual work due to heavy requirements on knowledge and experience. This project mainly examines two deep learning methods, DNN and LSTM, to automatize music transcription. We transform the audio files into spectrograms using constant Q transform and extract features from the spectrograms. Deep learning methods have the advantage of learning complex features in music transcription. The promising results verify that deep learning methods are capable of learning specific musical properties, including notes and rhythms.**

Keywords—automatic music transcription; deep learning; deep neural network (DNN); long short-term memory networks (LSTM)

I. INTRODUCTION

Automatic music transcription (AMT) reverse-engineers the 'source code' of a musical audio signal, generating symbolic representations such as music sheet that capture meaningful music parameters for performing or synthesizing the piece of music [1]. AMT not only plays a significant role in professional scope, for example, music information retrieval, musicological analysis of improvised and ethnic music [1], creation of interactive music systems [2] and so on, but is beneficial for amateurs to play their favorite songs as well. Music transcription relies on expertise and musical ear training [1]. The difficulty of automating it with comparable accuracy and flexibility with human professionals mainly arises from two reasons. First, each note is a full spectrum of harmonics with varying intensity that is sensitive to music instruments. Second, overlap of harmonics introduced by polyphony (i.e. multiple notes present concurrently) causes source-separation problem [3]. In our approach, two neural network models, DNN (Deep Neural Network) and LSTM (Long Short-Time Memory), were built for realizing audio-to-score conversion. Acoustic signal recording polyphonic piano music in the format of .wav file is inputted into our algorithm, and the output is MIDI (Musical Instrument Digital Interface) file that can be easily converted into music score.

II. RELATED WORKS

AMT has been attempted since the 1970s and polyphonic music transcription dates to the 1990s [4]. Unsupervised learning approaches assume that prior knowledge is not required for music transcription. Non-negative matrix factorization (NMF), which is in some sense equivalent to Independent Components Analysis (ICA), was performed on the input magnitude spectrum to obtain the frequency spectra for each pitch and their corresponding activities in time domain [5] [6]. By learning enough examples, a basis set including spectra for all the pitches could be created. However, the learned dictionary matrix may not match perfectly with music notes, which causes interpretation problem at the output [7]. A model proposed in [8] used 87 Support Vector Machine (SVM) classifiers to perform frame-level classification with the advantage of simplicity, and then an Hidden Markov Model (HMM) post-processing was adopted to smooth the results temporally. On top of it, Deep Belief Network (DBN) was added to learn higher layer representation of features in [9]. Since none of the approaches has reached the same level accuracy as human experts, most music transcription work is completed by musicians. With the development of deep learning in recent years, applying neural networks to accomplish AMT has inspired research interests. A model based on CNN was proposed in [10]. More models adopted RNN or LSTM due to its capability of dealing with sequential data [4] [11] [12]. In [7], 5 models were compared and the ConvNet model was reported as resulting in the best performance. In our project, two neural networks, DNN and LSTM were established and compared based on the work of [4] and [7]. Our codes are adapted from the GitHub project [4].

III. DATASET AND FEATURES

A. Dataset

All of the data for our model comes from the MUS (piece of music) set of the MIDI Aligned Piano Sounds (MAPS) database [13]. The outstanding characteristic that makes MAPS an optimal data source is that each .wav file has a corresponding text document with ground-truth onset and offset time for all the pitches. The total number of pieces of music is 270 (around 11G), recorded in nine different conditions in terms of environment and instrument. The data is split into training, validation and test sets based on 60-20-20 percentage ratio.

B. Features

Since it is inefficient to deal with audio files directly, the input .wav files are transformed into spectrograms to extract features. Methods widely used for preprocessing audio signals include STFT (Short-Time Fourier Transform), Mel Filterbank Cepstrum (MFCC) and constant Q transform (CQT). CQT has advantages over STFT in note identification since constant center frequency-to-resolution ratio results in constant pattern of sounds with harmonic components in logarithm-scaled frequency domain, which is easier for resolving notes that played simultaneously [14]. Besides, CQT outperforms MFCC based on the experiments in [5]. Therefore, CQT is chosen as the preprocessing method to extract features for our model. To enhance the efficiency of CQT, the audio signal is first down sampled from 44.1 kHz to 16 kHz, resulting in discretization of 32 ms per frame. Correspondingly, there are 512 samples per frame. The number of bins for each of the 7 octaves (C, D, E, F, G, A, and B) is set to 36. Consequently, there are 252 features for each frame. Therefore, a data matrix with the size of $252 \times number\ of\ features$ is obtained from the CQT transformation. The frames of all data are grouped into 40,000 frames per file for each set. The validation and test sets are normalized by subtracting the mean of the training set so that the features' magnitude will not exceed the limits of the training set. Figure 1 demonstrates an example from CQT transform with x-axis denoting time frame and y-axis denoting features.

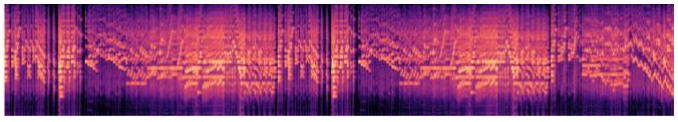


Figure 1. CQT spectrum example

C. Labels

The labels are created from the .txt files in the form of multilabel one hot representation. Figure 2 shows a small part of a text file. The ground-truth is also sampled at 31.25 frames per second. At each time frame, a vector of 88 elements corresponds to the 88 notes in a piano, with 1 representing that a note is played at the time frame and 0 representing that a note is not played. The vectors are also stacked and split into 40,000 frames per file to align with the input matrix.

OnsetTime	OffsetTime	MidiPitch
0.500004	2.45304	38
0.500004	2.45304	50
1.21885	2.45304	57
1.45206	2.45304	69
1.45206	2.45304	66
1.45206	2.45304	62
1.91848	2.45304	33
...		

Figure 2. Text file example. It contains onset/offset time and notes

IV. METHODS

A. Multi-label logistic regression (baseline model)

For evaluating the performance, we use the multi-label logistic regression as our baseline model. The multi-label logistic regression is identical to a neural network without hidden layers. We use the same input data matrix as DNN and the output units use sigmoid functions: $f(x) = \frac{1}{1+e^{-x}}$.

B. DNN

DNNs are powerful machine learning models that can be used for classification and regression tasks. DNNs are characterized by having one or more layers of non-linear transformations [7]. One layer of a DNN performs:

$$h_{l+1} = f(W_l h_l + b_l)$$

where W_l and b_l are weight matrix and bias for layer $0 \leq l \leq L$. For the input layer $l = 0$, $h_0 = x$, where x represents input features. For hidden layers, we use ReLU for f:

$$f(x) = x^+ = \max(x, 0)$$

For the output layer, we use sigmoid for f:

$$f(x) = \frac{1}{1+e^{-x}}.$$

The DNN architecture is as shown in Figure 3.

C. LSTM

Recurrent Neural Networks (RNN) has been widely applied into speech recognition and handwriting

recognition due to its capability of dealing with sequential data [11]. However, since only one past unit is included in the frame of RNN, vanishing gradients during back propagation becomes the major problem [15]. One possible solution is a special kind of RNNs, called Long Short-Time Memory Networks (LSTM), whose structure involving several past units enables it to learn long-term dependencies [4]. One layer of RNN performs:

$$h_{t+1}^{t} = f\left(W_l^{f} h_l^{f} + W_l^{r} h_l^{r} + b_l\right)$$

Where W_l^{f} is the weight matrix from the input layer to the hidden layer, and $W_l^{r} h_l^{t-1}$ is the weight matrix for the recurrent connection and b_l is the bias for layer $0 \leq l \leq L$. For the input layer $l = 0$, $h_0 = x$, where x represents input features. For hidden layers, we use tanh for f:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

For the output layer, we use sigmoid for f: $f(x) = \frac{1}{1+e^{-x}}$. LSTM is an extension of this, and uses the memory cell, in which 4 neural networks are included. The update step is done with these 4 neural networks. The memory cells of LSTM are as shown in Figure 4.

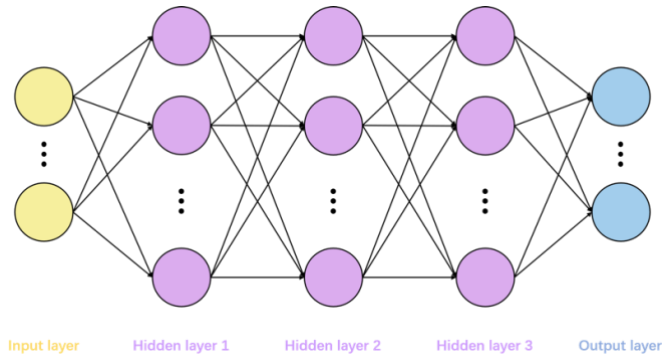


Figure 3. DNN architecture

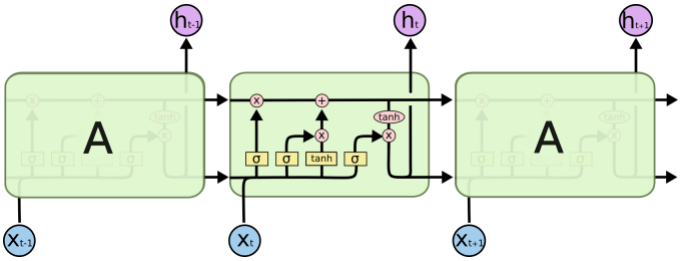


Figure 4. Memory cells of a LSTM. There are 4 neural networks inside each memory cell [4]

V. EXPERIMENTS AND RESULTS

A. Model configurations and parameters

Our project tests two neural network models for automatic music transcription, which are DNN and LSTM, as well as a baseline model, which is multi-label logistic regression. All of our models are implemented using Keras with a Tensorflow backend. All of them use binary cross entropy (bce) as the loss function:

$$L_{bce}^{(i)}(y_{(i)}, \hat{y}_{(i)}) = -\left(y_{(i)} \cdot log\left(\hat{y}_{(i)}\right) + \left(1 - y_{(i)}\right) \cdot log\left(1 - \hat{y}_{(i)}\right)\right)$$

We choose the Adam optimizer [17] as our optimization algorithm. Adam optimizer estimates 1st-order moment (the gradient mean) and 2nd-order moment (element-wise squared gradient) of the gradient using exponential moving average, and corrects its bias. The final weight update is proportional to learning rate times 1st-order moment divided by the square root of 2nd-order moment. Compared to SGD, Adam optimizer converges faster. Our initial learning rate is 0.1. The mini-batch size is 400 for a balanced trade-off between accuracy and efficiency.

For both DNN and LSTM, we test the performance using 2, 3 and 4 hidden layers with 256 units in each hidden layer. The input layer has 252 units because we have 252 features. The output layer has 88 units for 88 notes. In addition to that, for LSTM, 100 is chosen to be the number of recursive connections. Therefore, the input data matrix after preprocessing is reshaped into:

$$number\ of\ features/100 \times 100 \times number\ of\ frames$$

In order to avoid over-fitting, we use an early stop strategy and dropouts. We stopped training if the validation error did not decrease after 20 iterations over the entire training set. We compare the performance with 0%, 10%, 15%, 20%, 25% and 30% dropout rates.

B. Postprocessing

To improve accuracy, two types of post-processing was performed. First, the notes with duration lower than the threshold duration were eliminated. Second, the gaps with duration smaller than the threshold were filled.

C. Results and discussion

We use 3 measures: F-measure, Recall and Accuracy to compare all models. They are defined as follows [4]:

$$Precision(P) = \sum_{t=0}^{T} \frac{TruePositives(t)}{TruePositives(t) + FalsePositives(t)}$$

$$Recal(R) = \sum_{t=0}^{T} \frac{TruePositives(t)}{TruePositives(t) + FalseNegatives(t)}$$

$$Accuracy(A) = \sum_{t=0}^{T} \frac{TruePositives(t)}{TruePositives(t) + FalsePositives(t) + FalseNegatives(t)}$$

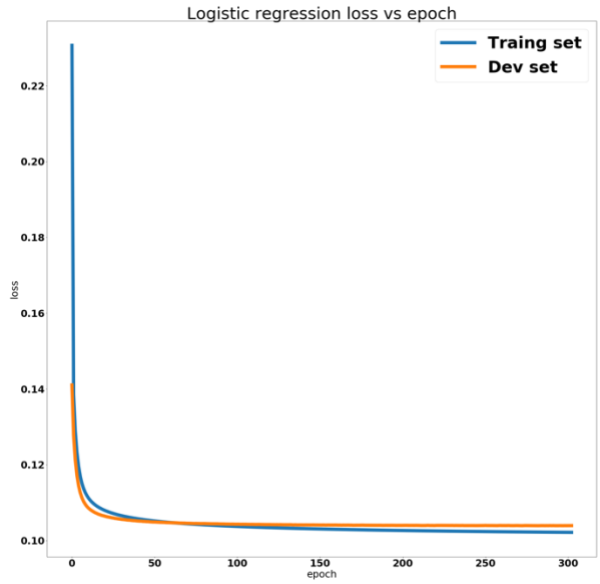
$$F - meature(F) = \sum_{t=0}^{T} \frac{2PR}{P + R}$$

The performance of our baseline model is as follows (Table 1):

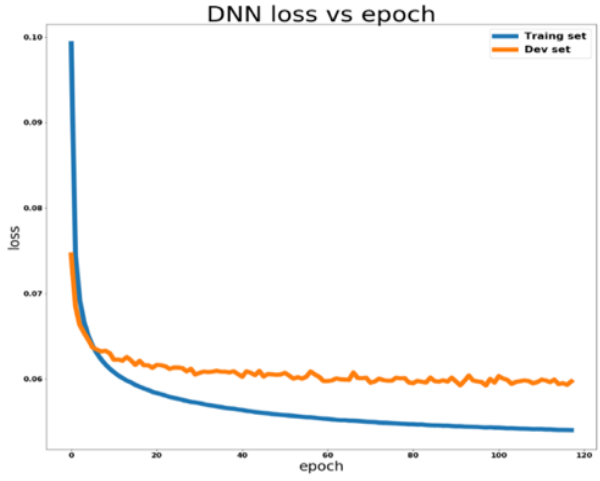
Table 1. Baseline model performance

F-measure	Recall	Accuracy
0.559	0.422	0.388

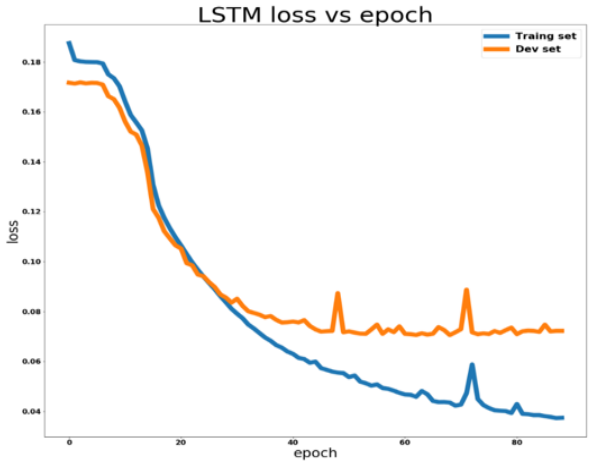
Figure 5 shows the training and validation losses for baseline model, DNN and LSTM. The baseline model has the largest training and validation losses. Compared to DNN, LSTM has a smaller training loss and a larger validation loss.



(a)



(b)



(c)

Figure 5. Cross entropy losses for training and validation set. (a) Baseline model losses; (b) DNN losses; (c) LSTM losses.

Table 2 shows the performance of DNN and LSTM with different number of hidden layers using a dropout rate of 20. It is obvious that all the neural network models experimented in our project outperform the baseline logistic regression model. Besides, we find that both DNN and LSTM achieve the best performance when using 3 hidden layers.

Table 2. DNN and LSTM performance with different number of hidden layers

DNN	*2 layers*	*3 layers*	*4 layers*
F-measure	0.767	**0.775**	0.772
Recall	0.701	0.706	**0.716**
Accuracy	0.621	**0.633**	0.629
LSTM	*2 layers*	*3 layers*	*4 layers*
F-measure	0.739	**0.741**	0.721
Recall	0.684	**0.688**	0.672
Accuracy	0.586	**0.588**	0.579

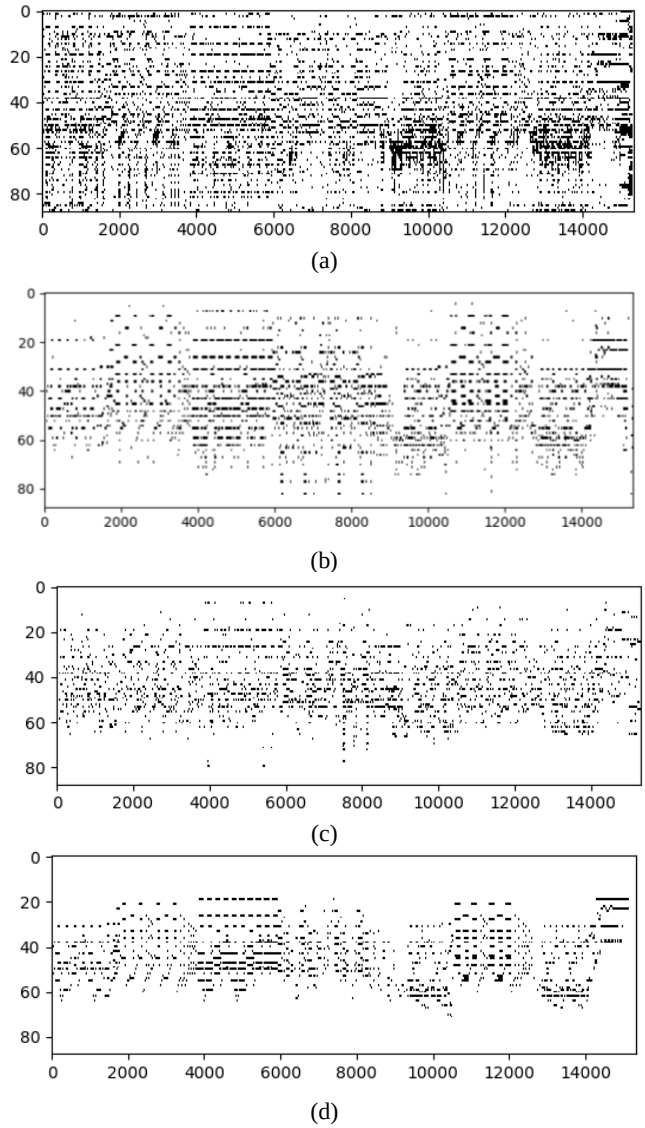
Table 3. DNN and LSTM performance with different dropout rates

DNN	*0%*	*10%*	*15%*	*20%*	*25%*	*30%*
F-measure	0.742	**0.776**	0.775	0.775	0.773	0.768
Recall	0.686	**0.716**	0.712	0.706	0.707	0.701
Accuracy	0.590	**0.634**	0.633	0.633	0.630	0.624
LSTM	*0%*	*10%*	*15%*	*20%*	*25%*	*30%*
F-measure	0.656	0.684	0.695	0.741	**0.756**	0.744
Recall	0.589	0.627	0.642	0.688	**0.711**	0.691
Accuracy	0.489	0.521	0.532	0.588	**0.608**	0.593

Table 3 shows the performance of DNN and LSTM with different dropout rates using 3 hidden layers. DNN achieves the best performance using a 10% dropout and

LSTM achieves the best performance using a 25% dropout.

Figure 6 shows the predictions of Baseline model, DNN and LSTM compared with the ground truth. From Figure 6, we can see that the large patterns are predicted correctly with DNN and LSTM, but the LSTM prediction is noisier. However, LSTM was supposed to perform better than DNN, because LSTM takes temporal dependency into consideration, but we obtained the opposite results. Compared to DNN, LSTM has a smaller training loss and a larger validation loss (Figure 5). Also, LSTM's performance improved when increasing dropout rates to 25% (Table 1). We think the reason that LSTM performs worse is overfitting.



(a)

(b)

(c)

(d)

Figure 6. (a) Baseline model prediction; (b) DNN prediction; (c) LSTM prediction; (d) Ground truth. X-axis is the time frame and Y-axis is the note. Dots indicate the presence of notes at those time frames.

We transform the predictions into music scores (Figure 7), and the predicted music scores are quite playable.

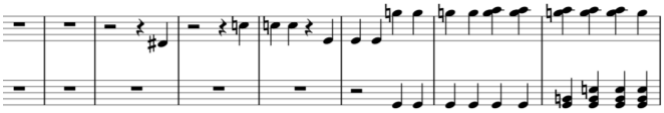


Figure 7. Output music scores

VI. CONCLUSIONS AND FUTURE WORK

In this project, we apply 2 deep learning methods: DNN and LSTM for music transcription. In general, the prediction accuracy is promising, given that our data set is small and the neural network is not very deep. By comparing their performance, we find DNN performs better than LSTM in general and LSTM has an overfitting issue in our tested case. By tweaking with the dropout rates, we can ameliorate the overfitting issue. To further increase the accuracy of our music transcription, the neural networks could be trained on a larger dataset, including isolated notes, chords and monophonic pieces of music other than polyphonic pieces of music.

We have also started training on another model which is ResNet CNN [15]. A major advantage of CNN is that it exploits spatial structure, so it can be directly applied to several frames of inputs to learn features along both time and the frequency axes. Additionally, when using an input representation like the CQT, CNN can learn pitch-invariant features, since inter-harmonic spacings in music signals are constant across log-frequency. ResNet is a well-known CNN architecture which won the ILSVRC challenge in 2015. Instead of directly constructing a mapping from input to output, ResNet trains the weights on the residuals between input and output to reduce the effect of gradient vanishing. We have built a ResNet with context window size 7, 2 residual blocks, and each convolutional layer has 64 filters. Due to time limit, the training process is not complete so we did not finalize the result. In the future, we will finish the ResNet model, and continue doing parameters tuning and getting results for different set of parameters.

CONTRIBUTIONS

We contribute equal amount of work to this project in terms of data collection, preprocessing, debugging codes, evaluating algorithms, discussing and writing the report. Although we work on testing 3 different neural network architectures, we maintain great communication among teammates.

REFERENCES

1. A. Klapuri and M. Davy. *Signal processing methods for music transcription*. NY: Springer Science & Business Media LLC, 2006.
2. 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, 2012.
3. T. Berg-Kirkpatrick, J. Andreas and D. Klein, "Unsupervised transcription of piano music," *Advances in Neural Information Processing Systems*, pp. 1538-1546, 2014.
4. D. G. Morin, "Deep neural networks for piano music transcription," Jun. 2017. Available: https://github.com/diegomorin8/Deep-Neural-Networks-for-Piano-Music-Transcription.
5. P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp.177-180, 2003.
6. S. A. Abdallah and M. D. Plumbley, "Polyphonic music transcription by non-negative sparse coding of power spectra," *Proceedings of the 5th International Society for Muisc Information Retrieval Conference*, pp. 318-325, 2004.
7. 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.*, 24, 927–939, 2016. Available: https://arxiv.org/abs/1508.01774.
8. G. E. Poliner and D. P. Ellis, "A discriminative model for polyphonic piano transcription," *EURASIP Journal on Applied Signal Processing*, vol. 2007, no. 1, pp. 154, 2007.
9. J. Nam, J. Ngiam, H. Lee and M. Slaney, "A classification-based polyphonic piano transcription approach using learned feature representations," *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pp. 16-180, 2011.
10. K. Ullrich and E. van der Wel, "Music transcription with convolutional sequence-to-sequence models," *International Society for Music Information Retrieval*, 2017.
11. B. L. Sturm, J. F. Santos, O. Ben-Tal and I. Korsunova. "Music transcription modelling and composition using deep learning,". arXiv preprint arXiv:1604.08723, 2016.
12. S. Sigtia, E. Benetos, N. Boulanger-Lewandowski, T. Weyde, A. S. d'Avila Garcez and S. Dixon, "A hybrid recurrent neural network for music transcription," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 2061-2065, 2015.
13. V. Emiya, N. Bertin, B. David and R. Badeau, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Transactions on Audio, Speech and Language Processing*, (to be published). Available: http://www.tsi.telecom-paristech.fr/aao/en/2010/07/08/maps-database-a-piano-database-for-multipitch-estimation-and-automatic-transcription-of-music
14. J. C. Brown, "Calculation of a Constant Q Spectral Transform," *J. Acoust. Soc. Am.*, vol. 89, pp. 425-434, 1991.
15. R. Pascanu, T. Mikolov and Y. Bengio, "On the difficulty of training recurrent neural networks," *arXiv: 1211.5063v2*, 16 Feb. 2013. Available: https://arxiv.org/abs/1211.5063.
16. K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition journal," arXiv preprint arXiv:1512.03385, 2015.
17. P. Diederik and Ba. Jimmy Lei, "Adam: A method for stochastic optimization," *3rd International Conference for Learning Representations*, 2015.
18. R. Kelz, M. Dorfer, F. Korzeniowski, S. Böck, A. Arzt and G. Widmer, "On the potential of simple framewise approaches to piano transcription," arXiv:1612.05153, 2016.