RECURRENT NEURAL NETWORKS FOR DRUM TRANSCRIPTION

Richard Vogl, Matthias Dorfer, Peter Knees
Department of Computational Perception
Johannes Kepler University Linz, Austria
richard.vogl@jku.at

ABSTRACT

Music transcription is a core task in the field of music information retrieval. Transcribing the drum tracks of music pieces is a well-defined sub-task. The symbolic representation of a drum track contains much useful information about the piece, like meter, tempo, as well as various style and genre cues. This work introduces a novel approach for drum transcription using recurrent neural networks. We claim that recurrent neural networks can be trained to identify the onsets of percussive instruments based on general properties of their sound. Different architectures of recurrent neural networks are compared and evaluated using a well-known dataset. The outcomes are compared to results of a state-of-the-art approach on the same dataset. Furthermore, the ability of the networks to generalize is demonstrated using a second, independent dataset. The experiments yield promising results: while F-measures higher than state-of-the-art results are achieved, the networks are capable of generalizing reasonably well.

1. INTRODUCTION AND RELATED WORK

Automatic music transcription (AMT) methods aim at extracting a symbolic, note-like representation from the audio signal of music tracks. It comprises important tasks in the field of music information retrieval (MIR), as — with the knowledge of a symbolic representation — many MIR tasks can be address more efficiently. Additionally, a variety for direct applications of AMT systems exists, for example: sheet music extraction for music students, MIDI generation/re-synthesis, score following for performances, as well as visualizations of different forms.

Drum transcription is a sub-task of AMT which addresses creating a symbolic representation of all notes played by percussive instruments (drums, cymbals, bells, etc.). The source material is usually, as in AMT, a monaural audio source—either from polyphonic audio containing multiple instruments, or a solo drum track. The symbolic representation of notes played by the percussive instruments can be used to derive rhythmical meta-information like tempo, meter, and downbeat. The repetitive rhythmical structure of the drum track, as well as changes therein, can be used as features for high-level MIR tasks. They provide information about the overall structure of the song which can be utilized for song segmentation [18]. The drum rhythm patterns can also be utilized for genre classification [7]. Other applications for rhythmic patterns include query-by-tapping and query-by-beat-boxing [11,19].

A common approach to the task of drum transcription is to apply methods used for source separation like non-negative matrix factorization (NMF), independent component analysis (ICA), or sparse coding. In recent work, Dittmar and Gärtner [5] use an NMF approach to transcribe solo drum tracks into three drum sound classes representing bass drum, snare drum, and hi-hat. They achieve F-measure values of up to 95%. Their approach focuses on real-time transcription of solo drum tracks for which training instances of each individual instrument are present. This is a very specific use case and in many cases separate training instances for each instrument are not available. A more general and robust approach which is able to transcribe different sounding instruments is desirable. Smaragdis [28] introduces a convolutional NMF method. It uses two-dimensional matrices (instead of one-dimensional vectors used in NMF) as *temporal-spectral bases* which allow to consider temporal structures of the components. Smaragdis shows that this method can be applied to transcribe solo drum tracks. Lindsay-Smith and McDonald [21] extend this method and use convolutive NMF to build a system for solo drum track transcription. They report good results on a non-public, synthetic dataset.

Fitzgerald et al. [9] introduce prior subspace analysis, an ICA method using knowledge of the signals to be separated, and demonstrate the application for drum transcription. Spich et al. [29] extend this approach by incorporating a statistical music language model. These works focus on transcription of three and two instruments, respectively.

Scholler and Purwins [26] use a sparse coding approach to calculate a similarity measure for drum sound classification. They use eight basis vectors to represent the sounds for bass drum, snare drum, and hi-hat in the time domain. Yoshii et al. [33] present an automatic drum transcription system based on template matching and adaptation, similar to sparse coding approaches. They focus on transcription of snare and bass drum only, from polyphonic audio signals.

Algorithms based on source separation usually use the input signal to produce prototypes (or components) rep-

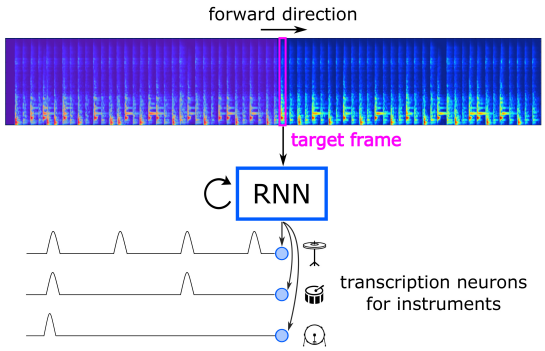

Figure 1. Overview of the proposed method. The extracted spectrogram is fed into the trained RNN which outputs activation functions for each instrument. A peak picking algorithms selects appropriate peaks as instrument onset candidates.

resenting individual instruments and so called activation curves which indicate the activity of them. A peak picking algorithm is needed to identify the instrument onsets in the activation curves. Additionally, the identified component prototypes have to be assigned to instruments. This is usually done using machine learning algorithms in combination with standard audio features [6, 16].

Another approach found in the literature is to first segment the audio stream using onset detection and classify the resulting fragments. Gillet and Richard [13] use a combination of a source separation technique and a support vector machine (SVM) classifier to transcribe drum sounds from polyphonic music. Miron et al. use a combination of frequency filters, onset detection and feature extraction in combination with a k-nearest-neighbor [23] and a k-means [22] classifier to detect drum sounds in a solo drum audio signal in real-time. Hidden Markov Models (HMMs) can be used to perform segmentation and classification in one step. Paulus and Klapuri [24] use HMMs to model the development of MFCCs over time. Decoding the most likely sequence yields activation curves for bass drum, snare drum, and hi-hat and can be applied for both solo drum tracks as well as polyphonic music.

Artificial neural networks consist of nodes (neurons) forming a directed graph, in which every connection has a certain weight. Since the discovery of gradient descent training methods which make training of complex architectures computationally feasible [17], artificial neural networks regained popularity in the machine learning community. They are being successfully applied in many different fields. Recurrent neural networks (RNNs) feature additional connections (recurrent connections) in each layer, providing the outputs of the same layer from the last time step as additional inputs. These connections can serve as memory for neural networks which is beneficial for tasks with sequential input data. RNNs have been shown to perform well, e.g., for speech recognition [25] and handwriting recognition [15]. Böck and Schedl use RNNs to improve beat tracking results [3] as well as for polyphonic

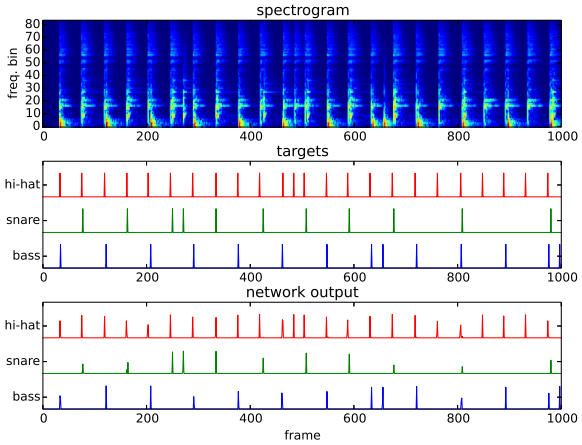

Figure 2. Spectrogram of a drum track and target functions for bass drum, snare drum, and hi-hat. The target function has a value of 1.0 at the frames at which annotations for the instruments exist and 0.0 otherwise. The frame rate of the target function is 100Hz, the same as for the spectrogram. The third graph shows the output of the trained RNN for the spectrogram in the first image.

piano transcription [4]. Sigtia et al. [27] use RNNs in the context of automatic music transcription as music language models to improve the results of a frame-level acoustic classifier. Although RNNs have been used in the past for transcription systems [4], we are not aware of any work using RNNs for transcription of drum tracks.

2. TASK AND MOTIVATION

In this work, we introduce a new method for automatic transcription of solo drum tracks using RNNs. While it is a first step towards drum transcription from polyphonic music, there also exist multiple applications for the transcription of solo drum tracks. In electronic music production, it can be used to transcribe drum loops if a re-synthesis using different sounds is desired. The transcription of recorded solo drum tracks can be used in the context of recording and production of rock songs. Nowadays it is not unusual to use sampled drums, e.g., in low-budget productions or in modern heavy metal genres. One the one hand, this is due to the complexity and costs of recording drums. On the other hand, with sampled drums it is easier to achieve the high precision and even robotic sounding style desired in some genres. Instead of manually programming the drum track, automatic transcription of a simple low-quality drum recording can be used as basis for the production of a song. As in other works, we focus on the transcription of bass drum, snare drum, and hi-hat. These instruments usually define the main rhythmic patterns [24], depending on genre and play style. They also cover most (>80% in the case of the ENST-Drums dataset, see Section 4.1) of the played notes in full drum kit recordings. Since simple RNN architectures already provide good transcription results (cf. Section 5), it is worthwhile exploring their application in this task further.

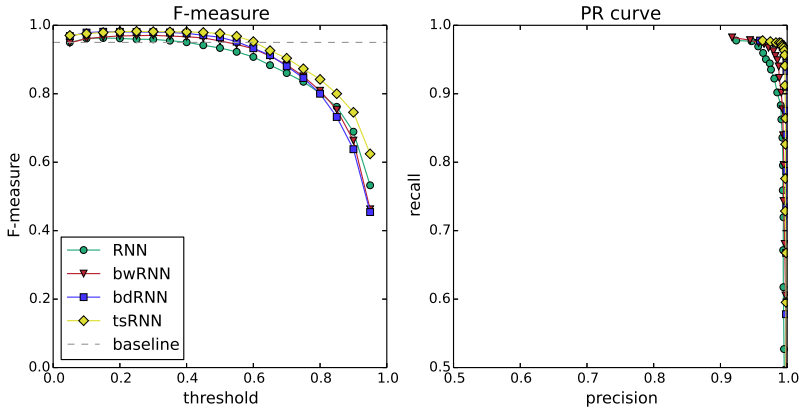

Figure 3. Result visualization for the evaluation on the IDMT-SMT-Drums dataset. The left plot shows the F-measure curve, the right plot the precision-recall curve for different threshold levels for peak picking.

3. METHOD

To extract the played notes from the audio signal, first a spectrogram of the audio signal is calculated. This is frame-wise fed into an RNN with three output neurons. The outputs of the RNN provide activation signals for the three drum instruments. A peak picking algorithm then identifies the onsets for each instrument's activation function, which yields the finished transcript (cf. Figure 1).

In this work, we compare four RNN architectures designed for transcribing solo drum tracks. These are: *i.* a simple RNN, *ii.* a backward RNN (bwRNN), *iii.* a bidirectional RNN (bdRNN), and *iv.* an RNN with time shift (tsRNN). The next section will cover the preprocessing of the audio signal, which is used for all four RNNs. After that, the individual architectures are presented in detail.

3.1 Signal Preprocessing

All four RNN architectures use the same features extracted from the audio signal. As input, mono audio files with 16 bit resolution at 44.1 kHz sampling rate are used. The audio is normalized and padded with 0.25 seconds of silence, to avoid onsets occurring immediately at the beginning of the audio file. First a logarithmic power spectrogram is calculated using a 2048 samples window size and a resulting frame rate of 100Hz. The frequency axis is then transformed to a logarithmic scale using twelve triangular filters per octave for a frequency range from 20 to 20,000 Hz. This results in a total number of 84 frequency bins.

3.2 Network Architectures

In this work, four different architectures of RNNs are compared. The four architectures comprise a plain RNN and three variations which are described in detail in the following.

3.2.1 Recurrent Neural Network

The plain RNN features a 84-node input layer which is needed to handle the input data vectors of the same size. The recurrent layer consists of 200 recurrently connected rectified linear units (ReLUs [14]). Although RNNs with ReLU activations can be difficult to train [20], good results without special initialization or treatment were achieved in this work. The connections between the input and the recurrent layer, the recurrent connections, and the connections between the recurrent layer and the output layer are all realized densely (every node is connected to all other nodes). The output layer consists of three nodes with sigmoid transfer functions, which provide the activation functions for the three instrument classes defined earlier. The sigmoid transfer function was chosen because binary cross-entropy was used as loss function for training, which turned out to be easier to train in the experiments.

3.2.2 Backward RNN

This RNN is very similar to the basic RNN with the only difference being that the recurrent connections are backward instead of forward in time. This was done in order to evaluate if the short sustain phase of percussive instruments provides additional information for the classification. The plain RNN has to identify the instruments at exactly the time frame of the onset annotation, thus the sustain phase of the notes can not be considered by it. This architecture is not real-time-capable since the audio to be transcribed is analyzed in reverse. Moreover, it might be more hard for this architecture to find the exact position of the onsets since the steep slope of the onset is only seen in forward direction.

3.2.3 Bidirectional RNN

The architecture of the bidirectional RNN used in this work consists of 100 nodes in a forward layer and 100 nodes in a backward layer. Both the forward and backward layers are directly connected to the input layer. Bidirectional RNNs often produce better results than unidirectional RNNs because they can also use the context of future frames for classification. In this work, they are meant to combine both the strengths of the forward and backward RNN. Unfortunately, this system has the same limitations as the backward RNN, making it not usable for real-time applications.

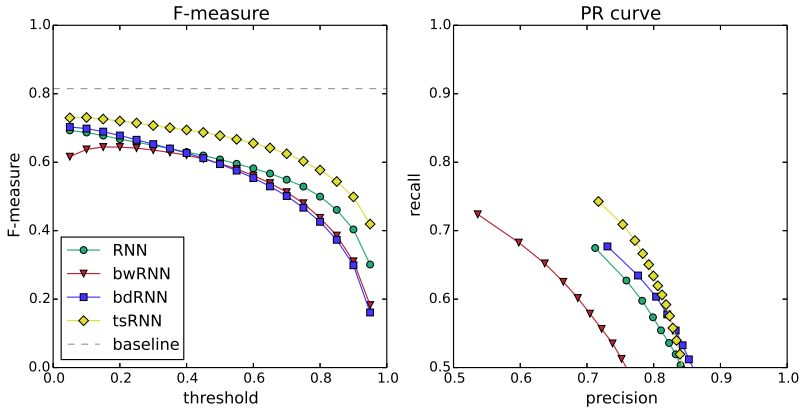

Figure 4. Result visualization for the evaluation on the ENST-Drums dataset. The left plot shows the F-measure curve, the right plot the precision-recall curve for different threshold levels for peak picking.

3.2.4 RNN with Time Shift

This approach is architecturally the same as the simple forward RNN, with the addition that this network can see more frames of the spectrogram to identify the instruments active at an onset. For training, the annotations are shifted into the future by 25ms and after transcription the detected onsets are shifted back by the same time. Doing this, the RNN can take a small portion of the sustain phase of the onset's spectrogram also into account. This is meant to improve the performance of the classification in the same way the backward connections do, without losing the real-time capabilities. The system is to a limited degree still real-time capable—depending on the length of the time shift. The used delay of 25ms in this work might still be sufficiently small for certain applications like score following and other visualizations and it can be tuned to meet the demands of certain applications.

3.3 Peak Picking

The output neurons of the RNNs provide activation functions for every instrument. To identify the instrument onsets, a simple and robust peak picking method designed for onset detection is used [2]. Peaks are selected at a frame n of the activation function $F(n)$ if the following three conditions are met:

1. $F(n) = max(F(n - pre\_max : n + post\_max))$,
2. $F(n) \geq mean(F(n - pre\_avg : n + post\_avg)) + \delta$,
3. $n - n_{lastpeak} > combination\_width$,

where δ is a threshold varied for evaluation. Simply put, a peak has to be the maximum of a certain window, and higher than the mean plus some threshold of another window. Additionally there has to be a distance of at least $combination\_width$ to the last peak. Parameters for the windows were chosen to achieve good results on a development data set while considering that 10 ms is the threshold of hearing two distinct events (values are converted from frames to ms): $pre\_max = 20ms$, $post\_max = 0ms$, $pre\_avg = 20ms$, $post\_avg = 0ms$, and $combination\_width = 20ms$. Setting $post\_max$ and $post\_avg$ to zero allows the application in online scenarios.

3.4 RNN Training

The task which has to be solved by the RNNs in this work is a three-way binary classification problem. When provided with the input spectrogram, the RNN has to identify the onsets of the three instrument classes by predicting the activation functions at the output neurons. The training algorithm has to adapt the weights and biases of the network in a way to achieve this functionality. In this work, the *rmsprop* method proposed by Hinton and Tieleman [31] is used as training algorithm. Additionally, dropout [30] between the recurrent and the output layer of the RNNs is used for training. When using dropout, randomly chosen connections are disabled for a single training iteration. The amount of disabled connections is determined by the dropout rate.

The goal of the training algorithm is to minimize a loss function. The loss function measures how much error the networks makes while reproducing the target functions. As loss function for training, the mean of the binary cross-entropy of the values of the three output neurons and the target functions is used (see Figure 2). The training with *rmsprop* is based on mini batches. In this work, mini batches with a size of eight instances were used. The training instances consist of 100-frame-segments of the extracted spectrogram. These are generated by extracting the spectrogram as described in Section 3.1 from the training files and cutting it into 100-frame-segments with 90 frames overlap (i.e. 10 frames hop-size). The order of the segments for training is randomized.

During one epoch the training data is used to adapt the weights and biases of the network. At the end of an epoch, the validation data is used to estimate the quality of the trained network. The training of the RNNs is aborted as soon as the resulting loss for the validation set has not decreased for 10 epochs. As learning rate decay strategy, the following method is applied: after every seven epochs the

Results for IDMT-SMT-Drums

algorithm	best F-measure[%]	at threshold
RNN	96.3	0.15
bwRNN	97.1	0.30
bdRNN	98.1	0.15
tsRNN	**98.2**	0.25
NMF [5]	**95.0**	-

Table 1. Evaluation results on the IDMT-SMT-Drums dataset. The NMF approach serves as state-of-the-art baseline.

Results for ENST-Drums

algorithm	best F-measure[%]	at threshold
RNN	69.3	0.05
bwRNN	64.4	0.15
bdRNN	70.3	0.05
tsRNN	**73.1**	0.10
HMM [24]	**81.5**	-

Table 2. Evaluation results on the ENST-Drums dataset. The HMM approach serves as state-of-the-art baseline.

learning rate is halved. For the simple RNN, backward RNN, and time shifted RNN the following parameter settings are used: initial learning rate $r_l = 0.001$ and dropout rate $r_d = 0.2$. In case of the bidirectional RNN the following parameter settings are used: initial learning rate $r_l = 0.0005$ and the dropout rate $r_d = 0.3$. The network is initialized with weights randomly sampled from a uniform distribution in the range ± 0.01, and zero-value biases.

All hyperparameters like network architecture, dropout rate, and learning rate were chosen according to empirical experimentation on a development data set, experience, and best practice examples.

3.5 Implementation

The implementation was done in Python using Lasagne [8] and Theano for RNN training and evaluation. The madmom [1] framework was used for signal processing and feature extraction, as well as for peak picking and evaluation metric calculation (precision, recall, and F-measure).

4. EVALUATION

To evaluate the presented system, the audio files of the test subset are preprocessed as explained in Section 3.1. Subsequently the spectrogram of the audio file is fed into the input layer of the RNN. The three neurons of the output layer provide the activation functions for the three instruments for which the peak picking algorithm then identifies the relevant peaks. These peaks are interpreted as instrument onsets. The true positive and false positive onsets are then identified by using a 20 ms tolerance window. It should be noted that in the state-of-the-art methods for the ENST-Drums dataset [24] as well as for the IDMT-SMT-Drums dataset [5], less strict tolerance windows of 30 ms and 50 ms, respectively, are used. Using these values, precision, recall, and F-measure for the onsets are calculated.

4.1 Datasets

For training and evaluation the IDMT-SMT-Drums [5] dataset was used. Some missing annotations have been added and additionally annotations for the *#train* tracks have been created. The *#train* tracks are tracks containing separated strokes of the individual instruments. These are only used as additional training examples and not used in the test set, to maintain a fair comparison with the results in [5]. The dataset was split into train, validation, and test subsets using 70%, 15%, and 15% of the files, respectively.

Additionally, the audio portion of the ENST-Drums [12] dataset was used as a second independent dataset to evaluate the generalization capabilities of the RNNs. From this dataset, the wet mixes of the drum-only tracks of all three drummers were used. Since all models were trained on the IDMT-SMT-Drums dataset, no splitting of this dataset was necessary.

For both datasets the three instruments' target functions are created by calculating the correct active frames (for a target frame rate of 100 Hz) using the annotations for each instrument. The target functions are one at the frames in which an annotation is present and zero otherwise. See Figure 2 for a visualization of the target functions in the context of the input spectrogram.

4.2 Experiments

For all four architectures, two different experiments were performed. First, the model was trained using the training and validation subsets of the IDMT-SMT-Drums dataset. Then, using the trained model, the tracks of the test split of the dataset were transcribed and the resulting precision, recall, and F-measure were calculated. Second, the trained model was evaluated by transcribing the ENST-Drums dataset and calculating the validation metrics. This was done to evaluate how well the trained models are able to generalize and if the models are over-fitted to the training dataset. Since the ENST-Drums dataset contains more than just the three instruments with which the model was trained, only the snare, bass, and hi-hat annotations were used. This makes it on the one hand easier to identify all annotated notes, on the other hand, there are some percussive onsets in the audio, which should not be transcribed and which are counted as false positives if the network falsely interprets them as snare, bass, or hi-hat hits. The percentage of snare, bass, and hi-hat annotations is 81.2% (i.e., 18.8% are other instruments which are ignored and potential false positives). The ENST-Drums dataset contains more expressive and faster drumming styles than the IDMT-SMT-Drums dataset, making it a more difficult dataset to transcribe. This fact is reflected in the transcription performances of both the state-of-the-art algorithms as well as the proposed methods. This behavior can also be observed in the work of Wu and Lerch [32] who apply their method to both datasets.

5. RESULTS AND DISCUSSION

Table 1 summarizes the results of all methods on the IDMT-SMT-Drums dataset. It can be seen that the F-measure values for all RNNs are higher than the state-of-the-art. It should be noted at this point, that the approach presented in [5] was introduced for real-time transcription. Nevertheless, the used NMF approach uses audio samples of the exact same instruments which should be transcribed as prototypes, which is the best-case scenario for NMF transcription. In contrast, our approach is trained on a split of the full dataset which contains many different instrument sounds, and thus is a more general model than the one used in the state-of-the-art approach used as baseline. It can be observed that the backward RNN performs slightly better than the plain RNN, which indicates that indeed the short sustain phases of the drum instruments contain information which is useful for classification. The bidirectional RNN again performs slightly better than the backward RNN, which comes as no surprise since it combines the properties of the plain forward and backward RNNs. The results of the forward RNN with time shift are not significantly different from the results of the bidirectional RNN. This indicates that the short additional time frame provided by the time shift provides sufficient additional information to achieve similar classification results as with a bidirectional RNN. Figure 3 shows a F-measure curve as well as a precision-recall curve for different threshold levels for peak picking.

The results of the evaluation of the models trained on the IDMT-SMT-Drums dataset used to transcribe the ENST-Drums dataset are shown in Table 2. The achieved F-measure values are not as high as the state-of-the-art in this case but this was expected. In contrast to [24], the model used in this work is not trained on splits of the ENST-Drums dataset and thus not optimized for it. Nevertheless, reasonable high F-measure values are achieved with respect to the fact that the model was trained on completely different and more simple data. This can be interpreted as an indication that the model in fact learns, to some degree, general properties of the three different drum instruments. Figure 4 shows an F-measure curve as well as a precision-recall curve for different threshold levels for peak picking.

In Figures 3 and 4, it can be seen that the highest F-measure values are found for low values for the threshold of the peak picking algorithm. This suggests that the RNNs are quite selective and the predicted activation functions do not contain much noise—which can in fact be observed (see Figure 2). This further implies that choices for peak picking window sizes are not critical, which was also observed in empiric experiments.

6. FUTURE WORK

Next steps for using RNNs for drum transcription will involve adapting the method to work on polyphonic audio tracks. It can be imagined to combine the presented method with a harmonic/percussive separation stage, using, e.g., the method introduced by Fitzgerald et al. [10], which would yield a drum track transcript from a full polyphonic audio track. As we show in this work, the transcription methods using RNNs are quite selective and therefore expected to be robust regarding artifacts resulting from source separation. Training directly on full audio tracks may also be a viable option to work on full audio tracks.

Another option is to use more instrument classes than the three instruments used in this and many other works. Theoretically, RNNs are not as vulnerable as source separation approaches when it comes to the number of instruments to transcribe. It has been shown that RNNs can perform well when using a much greater number of output neurons, for example 88 neurons in the case of piano transcription [4]. Although, for this, a dataset which has a balanced amount of notes played by different instruments has to be created first.

7. CONCLUSION

In this work, four architectures for drum transcription methods of solo drum tracks using RNNs were introduced. Their transcription performances are better than the results of the state-of-the-art approach which uses an NMF method—even with the NMF approach having the advantage of being trained on exactly the same instruments used in the drum tracks. The RNN approaches seem to be able to generalize quite well, since reasonable high transcription results are yielded on another, independent, and more difficult dataset. The precision-recall curves show that the best results are obtained when using a low threshold for peak picking. This implies that the used transcription methods are quite selective, which is an indication that they are robust and not bound to be influenced by noise or artifacts when using additional preprocessing steps.

8. ACKNOWLEDGMENTS

This work is supported by the European Union's Seventh Framework Programme FP7 / 2007-2013 for research, technological development and demonstration under grant agreement no. 610591 (GiantSteps). This work is supported by the Austrian ministries BMVIT and BMWFW, and the province of Upper Austria via the COMET Center SCCH. We gratefully acknowledge the support of NVIDIA Corporation with the donation of one Titan Black and one Tesla K40 GPU used for this research.

9. REFERENCES

[1] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: a new python audio and music signal processing library. `https://arxiv.org/abs/1605.07008`, 2016.

[2] Sebastian Böck, Florian Krebs, and Markus Schedl. Evaluating the online capabilities of onset detection methods. In *Proc 13th Intl Soc for Music Information Retrieval Conf*, pages 49–54, Porto, Portugal, October 2012.

[3] Sebastian Böck and Markus Schedl. Enhanced beat tracking with context-aware neural networks. In *Proc 14th Conf on Digital Audio Effects*, Paris, France, September 2011.

[4] Sebastian Böck and Markus Schedl. Polyphonic piano note transcription with recurrent neural networks. In *Proc IEEE Intl Conf on Acoustics, Speech and Signal Processing*, pages 121–124, Kyoto, Japan, March 2012.

[5] Christian Dittmar and Daniel Gärtner. Real-time transcription and separation of drum recordings based on nmf decomposition. In *Proc 17th Intl Conf on Digital Audio Effects*, Erlangen, Germany, September 2014.

[6] Christian Dittmar and Christian Uhle. Further steps towards drum transcription of polyphonic music. In *Proc 116th Audio Engineering Soc Conv*, Berlin, Germany, May 2004.

[7] Simon Dixon, Fabien Gouyon, Gerhard Widmer, et al. Towards characterisation of music via rhythmic patterns. In *Proc 5th Intl Conf on Music Information Retrieval*, Barcelona, Spain, October 2004.

[8] Sander Dieleman et al. Lasagne: First release. `http://dx.doi.org/10.5281/zenodo.27878`, 2015.

[9] Derry FitzGerald, Robert Lawlor, and Eugene Coyle. Prior subspace analysis for drum transcription. In *Proc 114th Audio Engineering Soc Conf*, Amsterdam, Netherlands, March 2003.

[10] Derry FitzGerald, Antoine Liukus, Zafar Rafii, Bryan Pardo, and Laurent Daudet. Harmonic/percussive separation using kernel additive modelling. In *Irish Signals & Systems Conf and China-Ireland Intl Conf on Information and Communications Technologies*, pages 35–40, Limerick, Ireland, June 2014.

[11] Olivier Gillet and Gaël Richard. Drum loops retrieval from spoken queries. *Journal of Intelligent Information Systems*, 24(2-3):159–177, 2005.

[12] Olivier Gillet and Gaël Richard. Enst-drums: an extensive audio-visual database for drum signals processing. In *Proc 7th Intl Conf on Music Information Retrieval*, pages 156–159, Victoria, BC, Canada, October 2006.

[13] Olivier Gillet and Gaël Richard. Transcription and separation of drum signals from polyphonic music. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(3):529–540, 2008.

[14] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proc 14th Intl Conf on Artificial Intelligence and Statistics*, pages 315–323, Ft. Lauderdale, FL, USA, April 2011.

[15] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for improved unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5), 2009.

[16] Marko Helen and Tuomas Virtanen. Separation of drums from polyphonic music using non-negative matrix factorization and support vector machine. In *Proc 13th European Signal Processing Conf*, Antalya, Turkey, September 2005.

[17] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, July 2006.

[18] Kristoffer Jensen. Multiple scale music segmentation using rhythm, timbre, and harmony. *EURASIP Journal on Applied Signal Processing*, 2007(1):159–159, 2007.

[19] Ajay Kapur, Manj Benning, and George Tzanetakis. Query-by-beat-boxing: Music retrieval for the dj. In *Proc 5th Intl Conf on Music Information Retrieval*, pages 170–177, Barcelona, Spain, October 2004.

[20] David Krueger and Roland Memisevic. Regularizing rnns by stabilizing activations. In *Proc 4th Intl Conf on Learning Representations*, San Juan, Puerto Rico, May 2015.

[21] Henry Lindsay-Smith, Skot McDonald, and Mark Sandler. Drumkit transcription via convolutive nmf. In *Intl Conf on Digital Audio Effects*, York, UK, September 2012.

[22] Marius Miron, Matthew EP Davies, and Fabien Gouyon. Improving the real-time performance of a causal audio drum transcription system. In *Proc Sound and Music Computing Conf*, pages 402–407, Stockholm, Sweden, July 2013.

[23] Marius Miron, Matthew EP Davies, and Fabien Gouyon. An open-source drum transcription system for pure data and max msp. In *Proc IEEE Intl Conf on Acoustics, Speech and Signal Processing*, pages 221–225, Vancouver, BC, Canada, May 2013.

[24] Jouni Paulus and Anssi Klapuri. Drum sound detection in polyphonic music with hidden markov models. *EURASIP Journal on Audio, Speech, and Music Processing*, 2009.

[25] Haşim Sak, Andrew W. Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proc 15th Annual Conf of the Intl Speech Communication Association*, pages 338–342, Singapore, September 2014.

[26] Simon Scholler and Hendrik Purwins. Sparse approximations for drum sound classification. *IEEE Journal of Selected Topics in Signal Processing*, 5(5):933–940, 2011.

[27] Siddharth Sigtia, Emmanouil Benetos, Nicolas Boulanger-Lewandowski, Tillman Weyde, Artur S d'Avila Garcez, and Simon Dixon. A hybrid recurrent neural network for music transcription. In *Proc IEEE Intl Conf on Acoustics, Speech and Signal Processing*, pages 2061–2065, Brisbane, Australia, April 2015.

[28] Paris Smaragdis. Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs. In *Independent Component Analysis and Blind Signal Separation*, volume 3195 of *Lecture Notes in Computer Science*, pages 494–499. Springer, Granada, Spain, September 2004.

[29] Andrio Spich, Massimiliano Zanoni, Augusto Sarti, and Stefano Tubaro. Drum music transcription using prior subspace analysis and pattern recognition. In *Proc 13th Intl Conf of Digital Audio Effects*, Graz, Austria, September 2010.

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, June 2014.

[31] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5rmsprop: Divide the gradient by a running average of its recent magnitude. In *COURSERA: Neural Networks for Machine Learning*, October 2012.

[32] Chih-Wei Wu and Alexander Lerch. Drum transcription using partially fixed non-negative matrix factorization with template adaptation. In *Proc 16th Intl Soc for Music Information Retrieval Conf*, pages 257–263, Málaga, Spain, October 2015.

[33] Kazuyoshi Yoshii, Masataka Goto, and Hiroshi G. Okuno. Automatic drum sound description for real-world music using template adaptation and matching methods. In *Proc 5th Intl Conf on Music Information Retrieval*, pages 184–191, Barcelona, Spain, October 2004.