A STUDY ON LSTM NETWORKS FOR POLYPHONIC MUSIC SEQUENCE MODELLING

Adrien Ycart and Emmanouil Benetos
Centre for Digital Music, Queen Mary University of London, UK
{a.ycart, emmanouil.benetos}@qmul.ac.uk

ABSTRACT

Neural networks, and especially long short-term memory networks (LSTM), have become increasingly popular for sequence modelling, be it in text, speech, or music. In this paper, we investigate the predictive power of simple LSTM networks for polyphonic MIDI sequences, using an empirical approach. Such systems can then be used as a music language model which, combined with an acoustic model, can improve automatic music transcription (AMT) performance. As a first step, we experiment with synthetic MIDI data, and we compare the results obtained in various settings, throughout the training process. In particular, we compare the use of a fixed sample rate against a musically-relevant sample rate. We test this system both on synthetic and real MIDI data. Results are compared in terms of note prediction accuracy. We show that the higher the sample rate is, the better the prediction is, because self transitions are more frequent. We suggest that for AMT, a musically-relevant sample rate is crucial in order to model note transitions, beyond a simple smoothing effect.

1. INTRODUCTION

Recurrent neural networks (RNNs) have become increasingly popular for sequence modelling in various domains such as text, speech or video [7]. In particular, long short-term memory networks (LSTMs) [10] have helped make tremendous progress in natural language modelling [15]. Those so-called *language models* can, in turn, be combined with *acoustic models* to improve speech recognition systems. Many recent improvements in this field have stemmed from better language models [8].

Automatic music transcription (AMT) is to music what speech recognition is to natural language: it is defined as the problem of extracting a symbolic representation from music signals, usually in the form of a time-pitch representation called *piano-roll*, or in a MIDI-like representation. Despite being one of the most widely discussed topics in music information retrieval (MIR), it remains an open problem, in particular in the case of polyphonic music [2]. While there has been extensive research on acoustic models for music transcription, music language models (MLMs) have received little attention until quite recently.

In this paper, we propose a study on the use of LSTM neural networks for symbolic polyphonic music modelling, in the form of piano-roll representations. We evaluate the impact of various parameters on the predictive performance of our system. Instead of relying on ever more complex architectures, we choose to use an LSTM with only one layer, and see how each parameter influences the final result, namely, the number of hidden nodes, the learning rate, the sampling rate of the piano-roll, and doing data augmentation. We also compare the use of time frames of fixed length against the use of beat-quantised time frames of fixed musical length (such as a 16th note). We evaluate the predictive performance of our system in terms of precision, recall and F-measure, and we monitor the evolution of these metrics throughout the learning process. We also conduct proof-of-concept experiments on AMT by post-processing the output of an existing acoustic model with our predictive models. We show that time-based time steps yield better results in terms of prediction because self-transitions are more frequent. This results in a simple smoothing effect when used in the context of transcription. On the other hand, note-based time steps perform worse for prediction, but show interesting behaviour that might be crucial for transcription, in particular the ability to model note transitions and some basic rhythmic features. To the best of our knowledge, such a study has not yet been done in the context of polyphonic music prediction.

The remainder of the paper is organised as follows. In section 2, we review existing works on symbolic music sequence modelling. In section 3, we describe the neural network architecture chosen for the experiments. In section 4, we describe the two employed datasets, one made of synthetic MIDI data, the other of real-life MIDI data. In section 6, we describe the various experiments performed for prediction and their results. In section 7, we show preliminary results on the application of the language model in the context of AMT. Finally, in section 8, we discuss the results obtained and their implications for future work.

2. STATE OF THE ART

Although MLMs have been discussed for quite a long time [14], they have not been specifically used in audio analysis until quite recently. Temperley [18] was one of the

first to propose a joint probabilistic model for harmony, rhythm and stream separation, and suggested its use for AMT. Since then, several other audio analysis systems, such as [16], have used probabilistic models of high-level musical knowledge in order to improve their performance.

More recently, some approaches have used neural networks to post-process the output of an acoustic model. Indeed, it seems that neural networks are more suitable to model polyphonic sequences compared to probabilistic models such as hidden Markov models (HMMs). In [4], a neural network architecture combining a RNN with a Restricted Boltzman Machine (RBM) was proposed to estimate at each time-step a pitch distribution, given the previous pitches. This architecture was later integrated in various systems. In [17], it was integrated in an end-to-end neural-network for multi-pitch detection in piano music, coupled with a variety of neural-network-based acoustic models. In all these models, the time-step is of the order of 10 ms, which is small compared to the typical duration of a music note. Moreover, this time-step is constant, and does not depend on the tempo of the input music signal.

Some systems have also modelled symbolic music sequences for other purposes. Pachet et al. [9] proposed an architecture consisting of four joint neural networks in order to generate Bach chorales. In [11], another architecture using reinforcement learning to enforce musical rules in a RNN was proposed for music generation.

All the above neural architectures rely on sophisticated combinations of neural networks, and have many parameters, which means that they need a lot of training data, and can be prone to over-fitting. In this study, we focus on a simple architecture, and try to build from that using an experimental method to assess the importance of various hyper-parameters. A study similar to the present has been conducted in [13] on chord sequence modelling (thus on modelling monophonic sequences instead of polyphonic ones). In this previous study, the advantage of RNNs over HMMs is questioned in the context of chord sequence modelling. In particular, it is argued that when the frame rate is high (order of 10 fps), the RNN only has a smoothing effect, and thus is no more efficient than simpler models such as an HMM. On the other hand, it is suggested that on the chord-level (ie. one symbol per chord, no matter how long), the RNN used is significantly better than an HMM. We aim at investigating similar questions in the context of polyphonic note sequence modelling in the current study.

3. MODEL

In this section, we describe the model we used in the experiments. This model is trained to predict the pitches present in the next time frame of a piano-roll, given the previous ones.

3.1 Data Representation

The input is a piano-roll representation, in the form of an $88 \times T$ matrix M, where T is the number of timesteps, and 88 corresponds to the number of keys on a piano, between MIDI notes A0 and C8. M is binary, such that $M[p, t] = 1$ if and only if the pitch p is active at the timestep t. In particular, held notes and repeated notes are not differentiated. The output is of the same form, except it only has $T-1$ timesteps (the first timestep cannot be predicted since there is no previous information). We use two different timesteps:

- a fixed time step of 10 milliseconds, that we refer to as *time-based*
- a variable time step with a fixed musical length of a sixteenth note, referred to as *note-based*.

3.2 Network Architecture

Our primary goal is to study the influence of various parameters, namely the number of hidden nodes, the learning rate, the use of data augmentation, and the time steps used. In order to assess the influence of those parameters as accurately as possible, without being influenced by other parameters, we deliberately choose to use the most simple LSTM architecture possible. In particular, we choose not to use multiple layers, nor to use dropout or any other regularisation method during training. These will be investigated in future work.

We thus use an LSTM with 88 inputs, one single hidden layer, and 88 outputs. The number of hidden nodes is chosen among: $64, 128, 256, 512$. The network is trained using the Adam optimiser [12], using the cross-entropy between the output of the network and the ground truth as cost function. The learning rate is kept constant, and is chosen among: $0.01, 0.001, 0.0001$.

The output of the network is then sent through a sigmoid, and thresholded to obtain a binary piano-roll. The threshold is determined by choosing the one that gives the best results on the validation dataset (see section 4).

4. DATASETS

We use two different datasets for training and testing our models. One is a synthetic dataset, the other is a dataset made of real music pieces. Both datasets were split into training, validation and test datasets with the following respective proportions: 70%-10%-20%.

4.1 Synthetic MIDI Dataset

The synthetic MIDI dataset used in this study consists of combinations of notes and chords in the C major key. It contains only notes on the C major scale, between C3 and C5. The chords are either a note and the note a third above (major or minor, such that the second note is also in C major), or a note, the note a third above, and the note a fifth above. All generated files are 3sec long, with a tempo of 60, so each file is 3-quarter-notes long. All notes have a duration multiple of a quarter note, so note changes can occur after 1 second, 2 seconds, both or neither. We take all possible combinations of 3 notes or chords and we allow repetition. When a note or a chord is repeated we create two distinct files, one corresponding to the note being held, one corresponding to the note being played again. Overall,

the dataset contains more than 36,000 files, representing 30 hours of data and will be referred to as *Synth dataset*.

4.2 Piano-midi.de Dataset

We use the Piano-midi.de dataset [1] as real-world MIDI data. This dataset currently holds 307 pieces of classical piano music from various composers. It was made by manually editing the velocities and the tempo curve of quantised MIDI files in order to give them a natural interpretation and feeling. This mode of production is the main reason why we chose it: it sounds real, with an expressive timing, and at the same time, the rhythmic ground truth is readily available. We can thus easily compute note-based time steps, without having to rely on a beat and meter detection algorithm.

This dataset holds pieces of very different durations (from 20 seconds to 20 minutes). In order to avoid excessive zero-padding for neural network training and to be more computationally efficient, we only keep the first minute from each file (and we zero-pad the shorter files). The resulting dataset is 5 hours long, and will be referred to as the *Piano dataset*. We also augment our dataset by transposing every piece in all keys from 7 semitones below to 6 semitones above. This increases the size of the dataset 12-fold, up to 60 hours. That way, all tonalities are equally represented, without shifting the range of notes too much.

5. EVALUATION METRICS

To evaluate the performance of our system, we compute several metrics following the MIREX Multiple-F0 Estimation task [1], namely the precision, recall and F-measure. Those metrics are computed for each file, and then averaged over a dataset. The progress throughout learning is computed on the validation dataset, and the performance of the trained model is computed on the test dataset.

6. PREDICTION

In this section, we compare the results obtained in various configurations, both quantitatively and qualitatively.

6.1 Number of Hidden Nodes and Learning Rate

We trained on the Synth dataset a series of models, with various numbers of hidden nodes in the hidden layer (*n_hidden*), and various learning rates (*learning_rate*). We tried all possible combinations with n_hidden $\in \{64, 128, 256, 512\}$ and learning_rate $\in \{0.0001, 0.001, 0.01\}$. We trained each model for 50 epochs, with a batch size of 50. All the implementations were made in Python, using Tensorflow [6]. The code necessary for the experiments can be found at: `http://www.eecs.qmul.ac.uk/~ay304/code/ismir17`.

In each case, the model converges to the same state: at epoch 50, all the models get the same precision, recall and F-measure with 10^{-2} precision. The only difference among all the models is the convergence speed. Similar observations were made for the other numbers of hidden nodes.

Quite expectedly, the parameter that has the most influence on convergence speed is the learning rate. Generally speaking, the larger the number of nodes is, the quicker the model converges (we could not compare when the learning rate is 0.01 since all the models converge in one epoch). Those empirical observations are consistently observed in all the other experiments as well (on the Piano dataset, with or without note-based time steps, with or without data augmentation).

When inspecting the output of the network before going through the sigmoid, we can notice some interesting features. All the notes that are outside the scale of C (that never appear in the training set) have a very low output, showing that the network is able to learn which notes might appear. This can be double-edged: notes outside the key are not mistakenly detected, but if they appear (which happens), they will not be detected either. In Section 7 we attempt to run this model on a real-life input file, and those findings are confirmed: the prediction clearly masks out every note that was not in the set of notes seen during training.

Considering the results of this preliminary experiment, we decide to keep for the rest of the experiments only n_hidden $\in [128, 256]$ and learning_rate $\in [0.001, 0.01]$. Indeed, 512 hidden nodes is too heavy computationally (around 20% longer training time compared to all the other configurations) without any clear improvement over 256 nodes. Similarly, a learning rate of 0.0001 converges too slowly compared to the others, with no noticeable advantage in the end result. We nevertheless choose to keep several models, not only the best one, because the best model on this dataset will not necessarily be the best one on another.

6.2 Data Augmentation

On the Piano dataset, we compare the performance of the model trained on the original 5-hour dataset, and on the augmented 60-hour dataset. The evolution of the F-measure on the validation dataset with and without data augmentation can be found in Figure 1. Results show that data augmentation improves greatly the results. All models trained on augmented data converge more quickly in terms of number of epochs, which is understandable since 12 times more data is processed at each epoch. However, in both cases, the results obtained after 50 epochs are approximately the same in terms of metrics.

6.3 Time Step

We compare the behaviour of the network when using time-based and note-based time steps, on both datasets. A comparison of the evolution of the F-measure on the Piano validation dataset with time-based or note-based time steps can be found in Figure 1.

With time-based time steps, we find that all the models seem to achieve similar results: with data augmentation, all

[1] `http://piano-midi.de/`

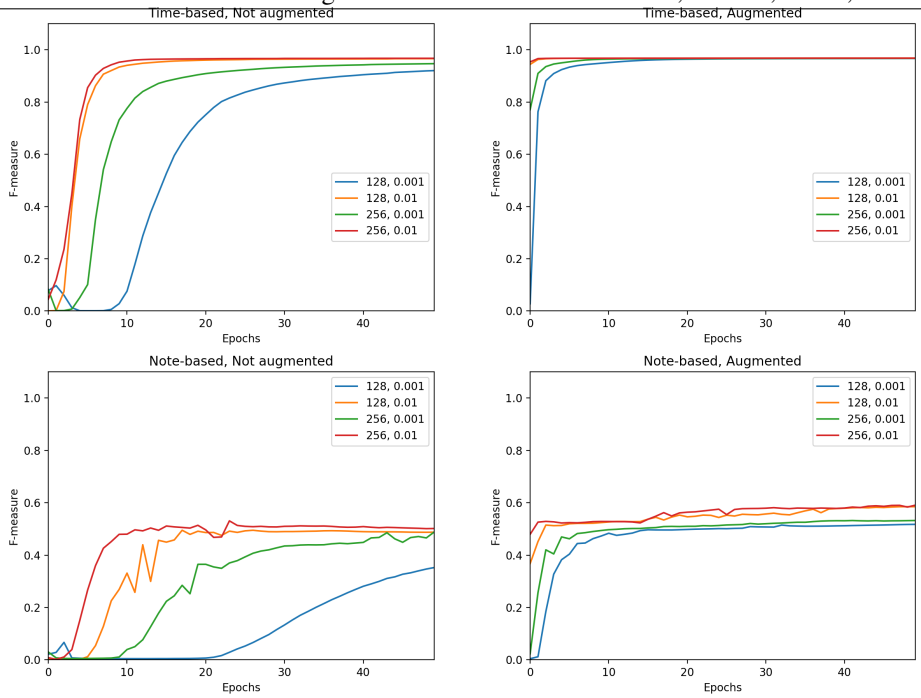

Figure 1. Comparison of the evolution of the F-measure across epochs, on the Piano validation dataset, with time-based or note-based timesteps, with or without data augmentation. A threshold of 0.5 is used.

	F-Measure	Precision	Recall
Without augmentation			
128, 0.001	0.451	0.409	0.509
256, 0.001	0.513	0.484	0.549
128, 0.01	0.548	0.536	0.560
256, 0.01	**0.553**	**0.544**	**0.562**
With augmentation			
128, 0.001	0.558	0.557	0.560
256, 0.001	0.571	0.552	0.592
128, 0.01	0.597	0.61	0.588
256, 0.01	**0.601**	**0.615**	**0.592**

Table 1. Prediction performance computed with note-based time steps on the Piano test dataset.

the models achieve a F-measure score of 0.966. Without data augmentation, the models trained with a learning rate of 0.01 achieve the same performance, with a learning rate of 0.001, the 128-hidden-node model achieves 0.917, and the 256-hidden-node model achieves 0.944. This might be due to the fact that they haven't fully converged after 50 epochs. All those scores were computed with a threshold of 0.5.

We also compute the precision, recall and F-measure on the Piano test dataset with note-based time steps. These results can be found in Table 1. We observe that this time, higher learning rate, higher number of nodes and data augmentation not only lead to quicker convergence, but also to better results.

For both datasets, the predictive accuracy is better in time-based configurations, since the frame rate is much higher, and thus there are more self-transitions (ie. notes are prolongated from the previous time steps). It seems indeed that in both cases, the system is uncertain when there are note changes, but learns to repeat the ongoing notes. The difference in performance can thus be attributed to the fact that note changes are much more frequent in the note-based case (once every 4 time steps versus once every 100 timesteps for the Synth dataset).

However, the note-based model shows very interesting behaviour at the uncertain time steps (ie. at each beat). On the Synth dataset, when the note changes, it gives a small probability to every note of the scale (the notes that might appear in the next frame), and rules out all the out-of-scale notes. Moreover, even when the note is kept for more than one beat, the model still shows the same "uncertain" behaviour, which does not happen with the time-based model. This is an error (which partly explains the worse scores), because the note should be held, but it has some very interesting consequences in terms of music modelling. This shows that the note-based model has learned that periodically, notes have a strong probability to change. This can be related to the rhythmic structure of music, as note changes are more frequent on strong metric positions. An example prediction output before thresholding is shown in Figure 2. We can see those "uncertain" time-steps in position 3 and 7, which correspond to time-steps 4 and 8 in the input (ie note changes).

We also find this behaviour with the Piano dataset, however only appearing with data augmentation. It is not clear if this is specific to data augmentation, or if it is simply because models without data augmentation were under-trained. In this case, the "uncertain" behaviour occurs at every eighth note, and with stronger probabilities at each quarter note. This suggests that the model behaves this way at the smallest metrical level, and not only at strong metrical positions. This might be a problem in the future, since it might encourage transitions too frequently. However, a small probability is only given to notes of the cor-

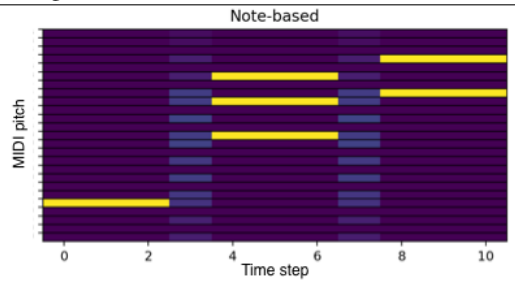

Figure 2. The prediction system output (n_hidden = 256, learning_rate=0.01) with note-based time steps after going through the sigmoid, before thresholding. The ground truth is: E3, C4E4G4, F4A4. At each note change, a small probability is given to all notes in C major scale.

rect scale, which shows that the model is able to get the tonal context of a piece to some extent. An example output before thresholding is shown in Figure 3.

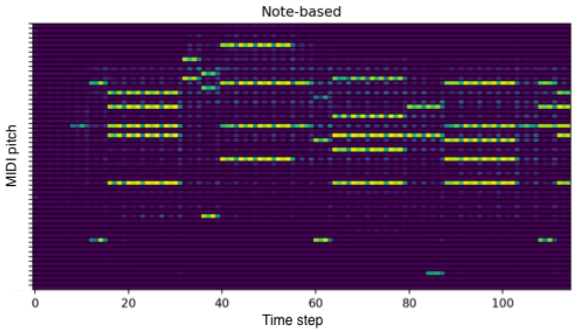

Figure 3. The output of the prediction system (n_hidden = 256, learning_rate=0.01) with note-based time steps after going through the sigmoid, before thresholding. The ground truth is Chopin's Prelude, No. 7, Op. 28 in A Major. At each eighth note, a small probability is given to some notes in A major, the tonality of the piece.

7. AUDIO TRANSCRIPTION

A preliminary experiment on assessing the potential of prediction models in the context of AMT is carried out. For this experiment, we use a single piece taken from the Piano dataset: Chopin's Prelude No. 7, Op. 28 in A Major.

7.1 Acoustic Model

For the experiment on integrating AMT with polyphonic music prediction, we use the system of [3], which is based on Probabilistic Latent Component Analysis. The system decomposes an input spectrogram into several probability distributions for pitch activations, instrument contributions and tuning deviations, using a dictionary of pre-extracted spectral templates. For this experiment, a piano-specific system is used, trained using isolated notes from the MAPS database [5]. The output of the acoustic model is a posteriogram of pitch activations over time.

7.2 Method

We synthesise the MIDI file with GarageBand, using the default Steinway Grand Piano soundbank. We analyse 3 different audio files:

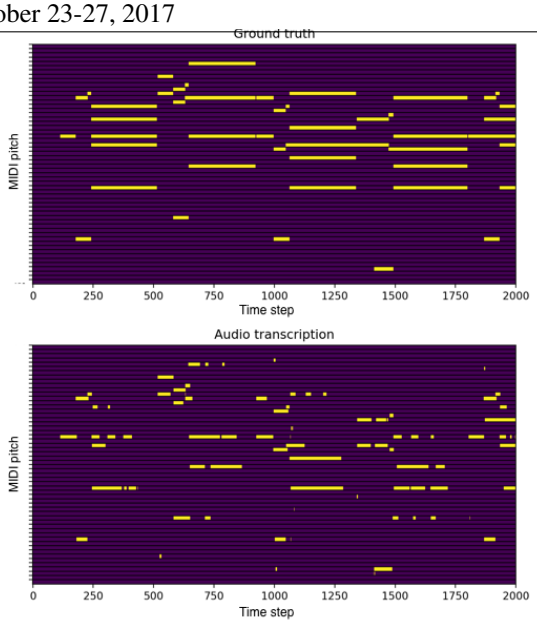

Figure 4. The first 20 seconds of the thresholded output of the baseline AMT system, compared with the ground truth.

- The full audio file, with expressive timing.
- The right-hand of the piece, transposed in C. In this case, predictive models trained both on the Synth and Piano dataset are evaluated.
- The full audio file, with quantised timing. The output of the transcription system is then downsampled to obtain a time step of a 16th note.

We take the posteriogram output by the previously described acoustic model, and feed it to various polyphonic prediction models, in various conditions:

- The raw posteriogram, with positive values theoretically unbounded (*raw_post*)
- The raw posteriogram thresholded in order to have a binary piano-roll (*raw_piano*)

The output of our predictive model is then thresholded using the value determined on the validation dataset in the experiments described in Section 6.3. The resulting piano-roll is compared to the ground truth using the accuracy metrics described in Section 5. An example of output of the baseline transcription system is shown in Figure 4.

7.3 Results

Results in terms of multi-pitch detection are shown in Table 2. Although we tested every configuration with all our models, we only report the results of the most meaningful experiments.

In the vast majority of cases, the results with the predictive model are below those of the acoustic model only, without post-processing. The only cases where the post-processing step improves the results is when the prediction is made on the whole piece, with time-based time steps, in *raw_piano* configuration. Otherwise, the results are at best equivalent to those of the baseline system. In the case where the results are improved, we inspect what improvements are made by the predictive model. It seems that the only improvements were few isolated frames that are temporally smoothed. We do not notice any missing notes be-

	F-Measure	Precision	Recall
Full audio, raw_piano			
Baseline	0.455	0.960	0.299
128, 0.001	0.458	0.938	0.303
256, 0.001	0.458	0.941	0.303
128, 0.01	0.460	0.959	0.303
256, 0.01	**0.460**	**0.961**	**0.303**
Right hand in C, raw_post, Synth			
Baseline	**0.670**	0.898	**0.535**
128, 0.001	0.556	0.955	0.393
256, 0.001	0.607	**0.966**	0.442
128, 0.01	0.522	0.834	0.380
256, 0.01	0.527	0.877	0.377
Full note-based, raw_piano			
Baseline	**0.526**	**0.963**	**0.361**
128, 0.001	0.434	0.624	0.332
256, 0.001	0.440	0.651	0.332
128, 0.01	0.478	0.852	0.332
256, 0.01	0.481	0.875	0.332

Table 2. Some results on transcription from audio, compared to the output of the baseline acoustic model.

ing added, and very few spurious notes are removed.

When using the Synth-dataset-trained models on the right hand transposed in C, the results are mixed. The precision measure is improved, due to the fact that many spurious notes are removed. On the other hand, some notes went completely missing because they were not in the C major scale, which leads to a lower recall score. Overall, the F-measure is lower than that of the acoustic model .

When using frame-based time steps, in every configuration, the results are worse. We have identified two reasons for that. The first is that sometimes, the system would add evenly distributed noise at the beginning of the prediction. This is probably due to the fact that the network takes a few frames to build a memory good enough to make correct predictions. More training removes that problem (the problem does not appear with models trained with a learning rate of 0.01). The second reason is that the system has some latency: a note is only activated one frame after it is seen in the input, and it is only deactivated one frame after it disappears of the input. When comparing the output of the system shifted one frame back with the output of the baseline system, the results were very close, and in some cases, identical (though never better).

8. DISCUSSION

In this study, we compare the influence of various parameters of an LSTM network on modelling polyphonic music sequences with respect to the training process and prediction performance. Those parameters are: the learning rate, the number of hidden nodes, the use of data augmentation by transposition, and the use of time-based time steps against note-based time steps. We found that with a given time step and a given dataset, the learning rate is the most important factor for learning speed, and that the more hidden nodes there are, the quicker the convergence is. We also found that data augmentation greatly improves both the convergence speed and the final performance.

When it comes to the choice of the time steps, it appears that time-based time steps yield a better prediction performance, because self transitions are more frequent. On the other hand, note-based time steps seem to show better musical properties. When trained on synthetic data containing only notes of the scale, it seems rather evident that notes that are have never been obeserved are very unlikely. More interestingly, when trained with real data in all tonalities, the system can still detect the scale of the piece : we can see with the example in Figure 3 that only notes of the correct tonality are given a some probability. We can also see that the system has learned the places where note changes might occur, and that the note changes are more frequent at beat positions than at half-beat positions.

We also use our system to post-process the output of an acoustic model for multi-pitch detection, in a proof-of-concept experiment. The first thing that we can notice from this experiment is that a good prediction performance does not necessarily translate to a good audio transcription performance. However, the order of performance for prediction seem to be kept for transcription: models with more nodes and higher learning rate tend to perform better.

The poor performance of our the predictive model for improving AMT performance is understandable: the input presented to the system in the transcription process is very different from those the models were trained on. Future work will include training predictive models not with piano-rolls, but with acoustic model posteriograms.

From all the above experiments, we can conclude that with time-based time steps, what the LSTM does is a simple smoothing, albeit a good one, since it improves transcription performance in some cases. The very fact that post-processing the output of the acoustic model with our system can improve the transcription performance, even though our language model was trained on a completely different kind of data, shows that it has in fact not learned much from the data it was fed, except temporal smoothing similar to e.g. a median filter. Since we aim at modelling the complex vertical and horizontal dependencies that exist within music, this behaviour is not sufficient.

On the other hand, we found some very interesting features in the output of the note-based models: they are able to learn when note changes might occur and what note might be activated which is very promising in terms of polyphonic music modelling. The downside of such models is that they would rely on meter detection algorithms when applied to audio, which might lead to error propagation. Future work will focus on investigating the possibilities of those models in the context of AMT, assuming that the meter and tempo are given as a first step.

Finally, we will extend this study in future work by gradually increasing the complexity of our model, and studying how the performance varies. In particular, we will study the result of adding more hidden nodes, and using more sophisticated regularisation techniques. We will also further investigate the results by visualising the learned parameters, as well as the activations of the hidden nodes.

9. ACKNOWLEDGEMENTS

A. Ycart is supported by a QMUL EECS Research Studentship. E. Benetos is supported by a RAEng Research Fellowship (RF/128).

10. REFERENCES

[1] Mert Bay, Andreas F. Ehmann, and J. Stephen Downie. Evaluation of Multiple-F0 Estimation and Tracking Systems. In *10th International Society for Music Information Retrieval Conference (ISMIR)*, pages 315–320, 2009.

[2] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[3] E. Benetos and T. Weyde. An efficient temporally-constrained probabilistic model for multiple-instrument music transcription. In *16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 701–707, 2015.

[4] N. Boulanger-Lewandowski, P. Vincent, and Y. Bengio. Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. *29th International Conference on Machine Learning*, pages 1159–1166, 2012.

[5] V. Emiya, R. Badeau, and B. David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech and Language Processing*, 18(6):1643–1654, August 2010.

[6] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[7] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT Press, 2016.

[8] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649. IEEE, 2013.

[9] G. Hadjeres and F. Pachet. DeepBach: a steerable model for Bach chorales generation. *arXiv preprint arXiv:1612.01010*, 2016.

[10] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] N. Jaques, S. Gu, R. E. Turner, and D. Eck. Tuning Recurrent Neural Networks with Reinforcement Learning. *5th International Conference on Learning Representations (ICLR)*, pages 1722–1728, 2017.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015.

[13] F. Korzeniowski and G. Widmer. On the Futility of Learning Complex Frame-Level Language Models for Chord Recognition. In *AES International Conference on Semantic Audio*, 2017.

[14] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, 1983.

[15] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[16] S. A. Raczyński, E. Vincent, and S. Sagayama. Dynamic Bayesian networks for symbolic polyhonic pitch modeling. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(9):1830 – 1840, 2013.

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

[18] D. Temperley. A Unified Probabilistic Model for Polyphonic Music Analysis. *Journal of New Music Research*, 38(1):3–18, 2009.