FINDING TEMPORAL STRUCTURE IN MUSIC: BLUES IMPROVISATION WITH LSTM RECURRENT NETWORKS

Douglas Eck and Jürgen Schmidhuber
IDSIA *Istituto Dalle Molle di Studi sull'Intelligenza Artificiale*
Galleria 2, 6928 Manno, Switzerland

Abstract. Few types of signal streams are as ubiquitous as music. Here we consider the problem of extracting essential ingredients of music signals, such as well-defined global temporal structure in the form of nested periodicities (or *meter*). Can we construct an adaptive signal processing device that learns by example how to generate new instances of a given musical style? Because recurrent neural networks can in principle learn the temporal structure of a signal, they are good candidates for such a task. Unfortunately, music composed by standard recurrent neural networks (RNNs) often lacks global coherence. The reason for this failure seems to be that RNNs cannot keep track of temporally distant events that indicate global music structure. Long Short-Term Memory (LSTM) has succeeded in similar domains where other RNNs have failed, such as timing & counting and learning of context sensitive languages. In the current study we show that LSTM is also a good mechanism for learning to compose music. We present experimental results showing that LSTM successfully learns a form of blues music and is able to compose novel (and we believe pleasing) melodies in that style. Remarkably, once the network has found the relevant structure it does not drift from it: LSTM is able to play the blues with good timing and proper structure as long as one is willing to listen.

INTRODUCTION

Music is among the most widely consumed types of signal streams. For this reason alone, signal processing techniques for finding and extracting and reproducing musical structure are of considerable interest. In particular, machine learning techniques for composing (good) music might have not only academic but also commercial potential.

Most music has well-defined global temporal structure in the form of nested periodicities or *meter*. A waltz, for example, has a $\frac{3}{4}$ meter, meaning

0-7803-7616-1/02/$17.00 (c) 2002 IEEE

that important melodic events occur every three quarter notes (or every first note in a *bar*). Chord changes occur more slowly but are in general aligned with the bars, with chords defining much of stylistic structure. For this reason one can say of music that some notes are more important than others: in general a learning mechanism should spend more resources on metrically-important notes than others. In doing so it can learn to recreate not only "surface level" musical structure but the deeper structure that defines a style.

Because recurrent neural networks (RNNs) can in principle learn such temporal structure, they are good candidates for such a task. The most straight-forward way to compose[1] music with an RNN is to use the network as single-step predictor. The network learns to predict notes at time $t + 1$ using earlier notes at times $\leq t$ as inputs. After learning has been stopped the network can be seeded with initial input values—perhaps from training data–and can then generate novel compositions by using its own outputs to generate subsequent inputs. This note-by-note approach was first examined by Todd et al. [19, 1] and later used by others [18, 13].

A feed-forward network would have no chance of composing music in this fashion. Lacking the ability to store any information about the past, such a network would be unable to keep track of where it is in a song. In principle an RNN does not suffer from this limitation. With recurrent connections it can use hidden layer activations as memory and thus is capable of exhibiting (seemingly arbitrary) temporal dynamics. In practice, however, RNNs do not perform very well at this task. As Mozer [13] aptly wrote about his attempts to compose music with RNNs, "While the local contours made sense, the pieces were not musically coherent, lacking thematic structure and having minimal phrase structure and rhythmic organization".

The reason for this failure is likely linked to the problem of *vanishing gradients* [10] in RNNs. In gradient methods such as Back-Propagation Through Time (BPTT) [20] and Real-Time Recurrent Learning (RTRL) [16] error flow either vanishes quickly or explodes exponentially, making it impossible for the networks to deal correctly with long-term dependencies. In the case of music, long-term dependencies are at the heart of what defines a particular style, with events spanning several notes or even many bars contributing to the formation of metrical and phrasal structure [2]. The clearest example of these dependencies are chord changes. In a musical form like early rock-and-roll music for example, the same chord can be held for four bars or more. Even if melodies are constrained to contain notes no shorter than an eighth note, a network must regularly and reliably bridge time spans of 32 events or more.

The most relevant previous research is that of Mozer [13], who did note-by-note composition of single-voice melodies accompanied by chords. In the "CONCERT" model, Mozer used sophisticated RNN procedures including BPTT, log-likelihood objective functions and probabilistic interpretation of

[1] In this paper we use the terms *composition* and *improvisation* loosely. It is probably more accurate to describe the behavior of the network as improvisation because it is inventing new melodies on top of a set form; however, the end goal is the creation of new melodies and new forms, thus the use of the term *composition*.

the output values. In addition to these neural network methods, Mozer employed a psychologically-realistic distributed input encoding [17] that gave the network an inductive bias towards chromatically and harmonically related notes. He used a second encoding method [12] to generate distributed representations of chords.

As was mentioned above, a BPTT-trained RNN does a poor job of learning long-term dependencies. To offset this, Mozer used a distributed encoding of duration that allowed him to process a note of any duration in a single network timestep. By representing in a single timestep a *note* rather than a slice of *time*, the number of time steps to be bridged by the network in learning global structure is greatly reduced. For example, to allow sixteenth notes in a network which encodes slices of time directly requires that a whole note span at minimum 16 time steps. One CONCERT composition (Figure 8 in the paper) contained only 76 notes but would require 172 time steps presuming the sixteenth notes found in the piece are to be allowed.

Even with sophisticated RNN techniques and psychologically-realistic distributed representation of melody, chords and duration the CONCERT architecture failed to capture global musical structure[2]. Though networks regularly outperformed third-order transition table approaches, they failed in all cases to find global structure. In analyzing this performance Mozer suggests that, for the note-by-note method to work it is necessary that the network can induce structure at multiple levels. We agree and offer the following architecture as one possible solution.

AN LSTM MUSIC COMPOSER

LSTM Architecture: Due to space constraints it is impossible to describe LSTM in depth. See [7, 8] for details. In summary, LSTM is designed to obtain constant error flow through time and to protect this error flow from undesired perturbations. LSTM uses linear units called *Constant Error Carousels* (CECs) to overcome error decay problems plaguing previous RNNs [9, 10]. Each CEC has a fixed self-connection and is surrounded by a cloud of nonlinear units responsible for controlling the flow of information in and out of the CEC. Typically, a multiplicative *input gate unit* learns to protect the flow from perturbation by irrelevant inputs. Likewise, a multiplicative *output gate unit* learns to protect other units from perturbation by currently irrelevant memory contents. A *forget gate* learns to reset a memory cell when its content is obsolete. Learning is done by gradient descent method that uses a slightly modified, truncated BPTT and a customized version of RTRL. Output units use BPTT; output gates use the truncated version of BPTT; while weights to cells, input gates and forget gates use truncated RTRL. LSTM performance is improved in online learning situations by using

[2]Mozer cannot be faulted for overstating the achievements of the model. He suggest that the "*ER*" in CONCERT is meant to stand for "*ER*udite" but that it could also mean "*ER*satz" or *ER*ratic".

a Kalman filter to control weight updates [14].

Data Representation: We avoid psychologically realistic distributed encodings and instead represent the data in a simple local form (similar to [19]). We use one input/target unit per note, with 1.0 representing on and 0.0 representing off. (In later experiments we used the common technique of adjusting input units to have a mean of 0 and a standard deviation of 1.0.) Unlike CONCERT this representation leaves it to the network to learn an inductive bias towards chromatically and harmonically related notes. Despite this, we prefer a localist representation for several reasons. First it is implicitly multi-voice and makes no artificial distinction between chords and melodies. (In fact, we implement chords by simply turning on the appropriate notes in the input vector.) Second it is an easy task to generate probability distributions over the set of possible notes, with the flexibility to treat single note probabilities as independent or dependent from one another.

The representation of time is straightforward, with one input vector representing a slice of real time. The stepsize of quantization can of course vary; if the quantization is set at the eighth note level (as it is for all experiments in this study) then eight network time steps are required to process a whole note. This method is preferable for LSTM because it forces the network to learn the relative durations of notes, making it easier for the counting and timing mechanisms to work [6].

Two representational issues are ignored in this representation. First, there is no explicit way to determine when a note ends. This means that eight eighth notes of the same pitch are represented exactly the same way as, say, four quarter notes of the same pitch. One way to implement this without changing input and target data structures is to decrease the stepsize of quantization and always mark note endings with a zero. With this method, a quantization level of sixteen steps per whole note would generate unique codes for eight eighth notes and four quarter notes of the same pitch. A second method is to have special unit(s) in the network to indicate the beginning of a note. This method was employed by Todd [19] and is certainly viable. However, it is not clear how such a method would scale to data sets with multi-voice melodies.

In simulations for this study, a range of 12 notes were possible for chords and 13 notes were possible for melodies Figure 1. Though we divided chord notes from melody notes for these simulations, this division is artificial: Chord notes are represented no differently than melody notes and in future experiments we intend to blend the two in a more realistic manner.

Training Data: For the experiments in this study, a form of 12-bar blues popular among bebop jazz musicians is used. With a quantization stepsize of 8 notes per bar, this yields a single song length of 96 network time steps. The chords used did not vary from song to song and are shown in Figure 2. Chords inversions were chosen so that the chords would fit into the allocated range of notes. For Experiment 1, only these chords were presented. For Experiment 2, a single melody line was presented along with the chords. The melody line was built using the pentatonic scale (Figure 3)

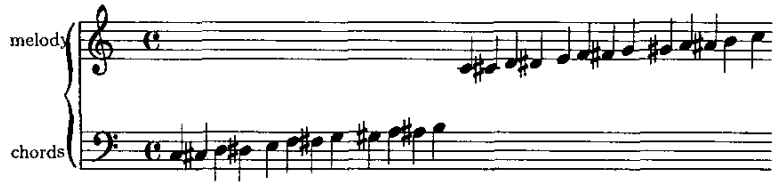

Figure 1: Possible note values for these simulations

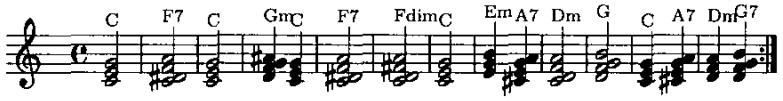

Figure 2: Chords for training data (transposed up one octave).

commonly used in this style of music. Training melodies were constructed

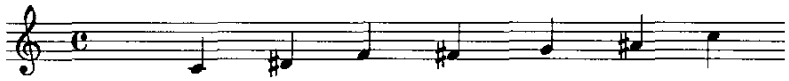

Figure 3: Pentatonic scale used for training data melodies.

by concatenating bar-long segments of music written by the first author to fit musically with each chord. Datasets were constructed by choosing randomly from the space of unique complete pieces ($n = 2^{12} = 4096$). Only quarter notes were used. No rests were used. Space constraints make it impossible to include examples. However several of these training pieces are provided as sheet music (Acrobat .pdf) and audio (MIDI, MP3 and wav) at `http://www.idsia.ch/~doug/blues/index.html`.

EXPERIMENT 1 — LEARNING CHORDS

In this experiment we show that LSTM can learn to reproduce a musical chord structure. Our motivation is to ensure that the chord structure of the song can in fact be induced in absence of its melody. Otherwise it is unclear whether LSTM is taking advantage of the local structure in melody to predict global chord structure. This is especially a risk when input data is generated using a random recombination of a relatively small number of musical examples, as was done here.

Network Topology and Experimental Parameters: The chords used are the ones described in Section 2. No melodies are presented. The quantization timestep is eight events per whole note. In the network four cell blocks containing 2 cells each are fully connected to each other and to the input layer. The output layer is fully connected to all cells and to the input layer. Forget gate, input gate and output gate biases for the four blocks are set at -0.5, -1.0, -1.5 and -2.0. This allows the blocks to come online one by

one. Output biases were set at 0.5. Learning rate was set at .00001. Momentum rate [15] was set at .9. Weights being burned after every timestep. Experiments showed that learning was faster if the network was reset after making one (or a small number) of gross errors. Resetting went as follows: on error, burn existing weights, reset the input pattern and clear partial derivatives, activations and cell states. Gers et al. [6] use a similar strategy. The squashing function at the output layer was the logistic sigmoid with range [0,1].

Training and Testing: The goal was to predict at the output the probability for a given note to be on or off. For predicting probabilities root mean squared error (RMSE) is not appropriate. Instead the network was trained using cross-entropy as the objective function. The error function E_i for output activation y_i and target value t_i is $E_i = -t_i ln(y_i) - (1 - t_i) ln(1 - y_i)$. This yields a δ term at the output layer of $(t_i - y_i)$. See, e.g., [11] for details. By using a series of binomial formulations rather than a single multinomial formulation (softmax) we treat outputs as statistically independent of one another. Though this assumption is untrue, it allows the network to predict chords and melodies in parallel and also allows for multi-voice melodies. The network was tested by starting it with the inputs from the first timestep and then using network predictions for ensuing time steps. Chord notes were predicted using a decision threshold of 0.5. Training was stopped after the network successfully predicted the entire chord sequence.

Results: LSTM easily handled this task under a wide range of learning rates and momentum rates. Once a network could successfully generate one full cycle through the chord sequence, it could generate any number of continuing cycles. This indicates that there was no reason to continue learning for a longer time. As it is already well documented that LSTM excels at timing and counting tasks [6] , success at this task is not surprising. Fast convergence was not a goal of this study, and learning times were not carefully collected. Informal timing tests show learning times on the order of 15 minutes to 45 minutes of processor time on a 1Ghz Pentium depending on parameter settings and initial random weights.

EXPERIMENT 2 — LEARNING MELODY AND CHORDS

In this experiment both melody and chords are learned. Learning continues until the chord structure is learned and cross-entropy error is relatively low. Note that there are far too many melodies for the network to learn them all. Once learning has been stopped, the network is started with a seed note or series of notes and then allowed to compose freely. The goal of the study was to see if LSTM could learn chord structure and melody structure and then use that structure to advantage when composing new songs.

Network Topology and Experimental Parameters: The network topology for this experiment differs from the previous task in that some cell blocks processed chord information while other cell blocks processed melody

information. Eight cell blocks containing 2 cells each are used. Four of the cell blocks are fully connected to the input units for chords. The other four cell blocks are fully connected to the input units for melody. The chord cell blocks have recurrent connections to themselves and to the melody cell blocks. However, melody cell blocks are only recurrently connected to melody cell blocks. That is, melody information does not reach the cell blocks responsible for processing chords. At the output layer, output units for chords are fully connected to cell blocks for chords and to input units for chords. Output units for melody are fully connected to cell blocks for melody and to input units for melody. The implications of such a topology are discussed below in Section 5. Forget gate, input gate and output gate biases for the four blocks dedicated to processing chords are set at -0.5, -1.0, -1.5 and -2.0. Gates for processing melodies are biased in exactly the same way. All other parameters are identical to those described for Experiment 1 in Section 3.

Training and Testing: The goal was to predict at the output the probability for a given note to be on or off. For chords, the same method as Experiment 1 is used: the network applies a decision threshold of 0.5 for all chord notes. For melodies we restrict the network to choosing a single note at any given timestep. This is achieved by adjusting melody output activations so that they sum to 1.0 and then using a uniform random number in the range [0,1] to choose the appropriate next note. The implications of this decision are discussed below in Section 5. The network was trained until it had learned the chord structure and until objective error had reached a plateau. Then the network was allowed to freely compose music. Music was composed by providing a single note or series of notes (up to 24) from the training set as input. After those were presented, network outputs were presented as inputs at the next timestep. No algorithmic or statistical method was used to evaluate the musical quality of the network. In all cases the network succeeded in reproducing chord structure while in parallel improvising new melodies.

Results: LSTM composed music in the form of blues. It learned the chord structure in training and used that structure to constrain its melody output in composition mode. Because is difficult to evaluate the performance objectively—this point is commonly made in AI art research, e.g., [13] — we urge the reader to visit `http://www.idsia.ch/~doug/blues/index.html`. On that page are examples of network blues composition in sheet music form (Acrobat .pdf) and audio (MIDI, MP3 and wav). It can be said that the network compositions are remarkably better sounding than a random walk across the pentatonic scale because the network compositions follow the structure of the musical form. They do diverge from the training set, sometimes significantly. But due to the influence of the chord structure, they never "drift" away from the form: the chord changes always bring it back. Also, an informal survey in our lab indicates that the compositions are at times quite pleasant. One jazz musician[3] is struck by how much the compositions sound like real bebop jazz improvisation over this same chord structure. In

[3]Admittedly, this musician isn't particularly good, and also happens to be the first author.

particular, the network's tendency to intermix snippets of known melodies with less-constrained passages is in keeping with this style.

DISCUSSION

These experiments were successful: LSTM induced both global structure and local structure from a corpus of musical training data, and used that information to compose in the same form. This answers Mozer's [13] key criticism of RNN music composition, namely that an RNN is unable to compose music having global coherence. To our knowledge the model presented in this paper is the first to accomplish this. That said, several parts of the experimental setup made the task easier for the model. More research is required to know whether the LSTM model can deal with more challenging composition tasks.

Training Data: There was no variety in the underlying chord structure. For this reason it is perhaps better to talk about network performance as improvisation over a predefined (albeit learned) form rather than composition. This lack of variation made it easier for LSTM to generate appropriately-timed chord changes. Furthermore, quantization stepsize for these experiments was rather low, at 8 time steps per whole note. As LSTM is known to excel at datasets with long time lags, this does not pose a serious problem. However it remains to be seen how much more difficult the task will be at, say, 32 time steps per whole note, a stepsize which would allow two sixteenth notes to be disambiguated from a single eighth note.

Network Architecture: There network connections were divided between chords and melody, with chords influencing melody but not vice-versa. We believe this choice makes sense: in real music improvisation the person playing melody (the soloist) is for the most part following the chord structure supplied by the rhythm section. However this architectural choice presumes that we know ahead of time how to segment chords from melodies. When working with jazz sheet music, chord changes are almost always provided separately from melodies and so this does not pose a great problem. Classical music compositions on the other hand make no such explicit division. Furthermore in an audio signal (as opposed to sheet music) chords and melodies are mixed together.

These are preliminary experiments, and much more research is warranted. A comparison with BPTT and RTRL (and other candidate RNNs) would help verify the claim that LSTM performance is better. A more interesting training set would allow for more interesting compositions. Finally, recent evidence suggests [5] that LSTM works better in similar situations using a Kalman filter to control weight updates. This should be explored. Finally, the current architecture is limited to working with symbolic representations (i.e. modified sheet notation) of music. If the architecture were to be extended to handle real-time performed music (i.e. MIDI or audio) it would have potential as a tool for interactive improvisation. This would require an ability to deal with temporal noise found in real performed music. One possibility is to

apply research on oscillator beat tracking models [4, 3] to LSTM in order to create an inductive bias towards coupling with sometimes-noisy rhythmical elements in the input.

CONCLUSION

A music composition model based on LSTM successfully learned the global structure of a musical form, and used that information to compose new pieces in the form. Two experiments were performed. The first verified that LSTM was not relying on regularities in the melody to learn the chord structure. The second experiment explored the ability for LSTM to generate new instances of a musical form, in this case a bebop-jazz variation of standard 12-bar blues. These experiments are preliminary and much more work is warranted. For example, we have yet to compare LSTM performance to non-RNN approaches such as HMMs and graphical models. Also, we report on LSTM behavior for a single set of parameters; a more methodical exploration of parameter space is warranted. However by demonstrating that an RNN can capture both the local structure of melody and the long-term structure of a musical style, these experiments represent an advance in neural network music composition.

ACKNOWLEDGEMENTS

We would like to thank Mike Mozer for answering many questions about his model and simulation methods. The division of labor between authors is as follows: The first author devised and constructed the datasets, implemented the program code, ran the simulations and wrote the paper. The second author wrote the grant proposal and eventually obtained a grant (SNF 21-49144.96) for doing this type of work. He provided guidance on how to structure the task and how to best use LSTM; he also edited the paper.

REFERENCES

[1] J. J. Bharucha and P. M. Todd, "Modeling the perception of tonal structure with neural nets," **Computer Music Journal**, vol. 13, no. 4, pp. 44–53, 1989.

[2] G. Cooper and L. B. Meyer, **The Rhythmic Structure of Music**, The University of Chicago Press, 1960.

[3] D. Eck, "A Network of Relaxation Oscillators that Finds Downbeats in Rhythms," in G. Dorffner (ed.), **Artificial Neural Networks – ICANN 2001 (Proceedings)**, Berlin: Springer, 2001, pp. 1239–1247.

[4] D. Eck, "Finding Downbeats with a Relaxation Oscillator," **Psychological Research**, vol. 66, no. 1, pp. 18–25, 2002.

[5] F. A. Gers, J. Perez-Ortiz, D. Eck and J. Schmidhuber, "DEKF-LSTM," in **Proc. 10th European Symposium on Artificial Neural Networks,**

ESANN 2002, 2002.

[6] F. A. Gers and J. Schmidhuber, "Recurrent Nets that Time and Count," in **Proc. IJCNN'2000, Int. Joint Conf. on Neural Networks**, Como, Italy, 2000.

[7] F. A. Gers and J. Schmidhuber, "LSTM recurrent networks learn simple context free and context sensitive languages," **IEEE Transactions on Neural Networks**, vol. 12, no. 6, pp. 1333–1340, 2001.

[8] F. A. Gers, J. Schmidhuber and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," **Neural Computation**, vol. 12, no. 10, pp. 2451–2471, 2000.

[9] S. Hochreiter, " Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München," 1991, See http://ni.cs.tu-berlin.de/~hochreit/papers/-hochreiter.dipl.ps.gz.

[10] S. Hochreiter, Y. Bengio, P. Frasconi and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," in S. C. Kremer and J. F. Kolen (eds.), **A Field Guide to Dynamical Recurrent Neural Networks**, IEEE Press, 2001.

[11] M. Joost and W. Schiffmann, "Speeding up backpropagation algorithms by using cross-entropy combined with pattern normalization," **International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems**, vol. 6, no. 2, pp. 117–126, 1998.

[12] B. Laden and D. H. Keefe, "The representation of pitch in a neural net model of chord classification," **Computer Music Journal**, vol. 13, no. 4, pp. 44–53, 1989.

[13] M. C. Mozer, "Neural network composition by prediction: Exploring the benefits of psychophysical constraints and multiscale processing," **Cognitive Science**, vol. 6, pp. 247–280, 1994.

[14] J. A. Pérez-Ortiz, F. A. Gers, D. Eck and J. Schmidhuber, "Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets," **Neural Networks**, 2002, Accepted pending minor revisions.

[15] D. C. Plaut, S. J. Nowlan and G. E. Hinton, "Experiments on learning back propagation," Techn. Report CMU–CS–86–126, **Carnegie–Mellon University**, Pittsburgh, PA, 1986.

[16] A. J. Robinson and F. Fallside, "The Utility Driven Dynamic Error Propagation Network," Techn. Report CUED/F-INFENG/TR.1, **Cambridge University Engineering Department**, 1987.

[17] R. N. Shepard, "Geometrical approximations to the structure of pitch," **Psychological Review**, vol. 89, pp. 305–333, 1982.

[18] C. Stevens and J. Wiles, "Representations of Tonal Music: A Case study in the development of temporal relationship," in M. Mozer, P. Smolensky, D. Touretsky, J. Elman and A. S. Weigend (eds.), **Proceedings of the 1993 Connectionist Models Summer School**, Hillsdale, NJ: Erlbaum, pp. 228–235, 1994.

[19] P. M. Todd, "A connectionist approach to algorithmic composition," **Computer Music Journal**, vol. 13, no. 4, pp. 27–43, 1989.

[20] R. J. Williams and D. Zipser, "Gradient-based learning algorithms for recurrent networks and their computational complexity," in Y. Chauvin and D. E. Rumelhart (eds.), **Back-propagation: Theory, Architectures and Applications**, Hillsdale, NJ: Erlbaum, chap. 13, pp. 433–486, 1995.