

Learning The Long-Term Structure of the Blues^{*}

Douglas Eck and Jürgen Schmidhuber

Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
Galleria 2, CH 6928 Manno, Switzerland
doug@idsia.ch, juergen@idsia.ch, <http://www.idsia.ch/>

Abstract. In general music composed by recurrent neural networks (RNNs) suffers from a lack of global structure. Though networks can learn note-by-note transition probabilities and even reproduce phrases, they have been unable to learn an entire musical form and use that knowledge to guide composition. In this study, we describe model details and present experimental results showing that LSTM successfully learns a form of blues music and is able to compose novel (and some listeners 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.

*A note to referees: As the output of the network is musical, it is difficult to gain an appreciation by just reading the paper. The text references a URL containing helpful musical examples (www.idsia.ch/~doug/blues/index.html). If this paper is accepted at ICANN, the authors look forward to playing some of these passages as part of the presentation. **This note will be deleted before publication! The paper is 6 pages long without this note.***

1 Introduction

The most straight-forward way to compose music with an RNN is to use the network as single-step predictor. The network learns to predict notes at time $t+1$ using notes at time 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 [12, 1] and later used by others [11, 9].

A feed-forward network would have no chance of composing music in this fashion. With no ability to store past information, 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. In practice, however, RNNs do not perform very well at this

^{*} Work supported by SNF project 21-49144.96

task. As Mozer [9] 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.” In short, RNNs do not excel at finding long-term dependencies in data — the so-called *vanishing gradient* problem [6] — and so are relatively insensitive to the global structure that defines a particular musical form.

Long Short-Term Memory (LSTM) [7] solves the problem of vanishing gradients by enforcing constant error flow. In doing so, LSTM is able to find long-term dependencies in data. Recent research [3] has already shown that LSTM can solve certain rhythmical timing and counting tasks. In the current study we ask the question, “Can LSTM compose music?” The answer is a guarded “Yes.” Though the experiments presented here are very preliminary, LSTM was able to find global musical structure and use it to compose new pieces. Section 2 describes the music composition model, Section 3 presents two experiments in music composition and Section 4 discusses the results.

2 An LSTM Music Composer

LSTM Architecture. Due to space constraints it is impossible to describe LSTM in depth. See [4, 5] 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 to overcome the problem of error decay or error explosion. Each unit has a fixed self-connection and is surrounded by a cloud of nonlinear gating units responsible for controlling information flow. Learning is done by a gradient descent method that is a combination of modified BPTT and modified RTRL [10].

Data Representation. Inputs and targets are represented in a simple local form (similar to [12]). We use one input/target unit per note, with 1.0 representing ON and 0.0 representing OFF. Inputs are then adjusted to have mean=0.0 and a standard deviation=1.0. The representation is multi-voice and makes no distinction between chords and melodies. Time is encoded implicitly, with one input vector representing a slice of real time. This is a major difference from Mozer’s CONCERT system [9] which used two different distributed representations and processed a single *note* per timestep regardless of duration.

Note that the start (or end) of a note is not marked. This means that (with eighth-note quantization) a series of eight identical eighth notes are encoded the same way as four identical quarter notes. This can be remedied by decreasing the stepsize of quantization and marking note offsets with a zero. Alternatively, special unit(s) can indicate the start of note(s) [12]. 12 notes (starting an octave below middle C and ascending by half steps) were allocated for chords and 13 notes for melodies (continuing up by half steps from middle C). For these simulations, melody notes never occurred in the range for chords or vice-versa.

Training Data. For the experiments in this study, a form of 12-bar blues popular among bebop jazz musicians is used (Figure 1). With 8 time steps per bar, each song was 96 time steps long. For Experiment 1, only these chords



Fig. 1. Bebop-style blues chords used for training data (transposed up one octave).

were presented. For Experiment 2, a single melody line was also presented. This melody was built using the pentatonic scale (Figure 2) commonly used in this style of music. Training melodies were constructed by concatenating



Fig. 2. Pentatonic scale used for training data melodies.

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 and audio at www.idsia.ch/~doug/blues/index.html.

3 Experiments

EXPERIMENT 1 – Learning Chords. We test the ability for LSTM to learn and reproduce the global structure of bebop-jazz style blues. Musical structure is presented to the network in the form of chords only. This ensures that the model is truly sensitive to long time lag structure and is not exploiting local regularities that may be found in the melody line. With a quantization of eight time steps per bar, this dataset could not be learned using RTRL or BPTT [8].

Architecture and 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 -0.1, -0.3, -0.5 and -0.7. 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 was set at .9. Weights are burned after every timestep. Experiments showed that learning is faster if the network is 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 [3] 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. The network was trained using cross-entropy as

the objective function. Notes are treated independently, as multiple notes are on at once. The network was trained until it could successfully predict the entire chord sequence multiple times. This performance was tested by starting the network with the inputs from the first timestep of the dataset and then using network predictions to generate the next input. Notes were predicted using a decision threshold of 0.5.

Results. LSTM easily handled this task under a wide range of learning rates and momentum rates. As it is already well documented that LSTM excels at timing and counting tasks [3], success at this task is not surprising. Fast convergence was not a goal of this study, and learning times were not carefully collected.

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 examples.

Architecture and 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 other 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. Forget gate, input gate and output gate biases for the four blocks dedicated to processing chords are set at -0.1, -0.3, -0.5 and -0.7. Gates for processing melodies are biased in exactly the same way. All other parameters are identical to those in Experiment 1.

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 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 seeding the network with a single note or series of notes (up to 24) as was done in Experiment 1.

Results. LSTM composed music in the form of blues. It learned the long-term chord structure in training and used that structure to constrain its melody output in composition mode. Because it is difficult to evaluate the performance objectively—this point is commonly made in AI art research, e.g., [9]—we urge the reader to visit www.idsia.ch/~doug/blues/index.html. On that page are examples of network blues composition in sheet music and audio. The network compositions are remarkably better sounding than a random walk across the pentatonic scale (this is also available for listening at the website). Unlike a random walk, 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” too far away from the form: the long-term structure reflected by chord changes always bring them back. Also, an informal survey in our lab indicates that the compositions are at times quite pleasant. It is striking, to the authors at least, how much the compositions sound like real (albeit not very good) bebop jazz improvisation. In particular, the network’s tendency to intermix snippets of known melodies with less-constrained passages is in keeping with this style.

4 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 [9] 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. This made it easier for LSTM to generate appropriately-timed chord changes. Furthermore, quantization stepsize for these experiments was rather high, at 8 time steps per whole note. As LSTM is known to excel at datasets with long time lags, this should 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 discriminated from a single eighth note.

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 more interesting training set would allow for more interesting compositions. Finally, recent evidence suggests [2] that LSTM works better in similar situations using a Kalman filter to control weight updates. This should be explored.

5 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 able to learn and reproduce long-term global music structure in the form of chords. 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. However by demonstrating that an RNN can capture both the local structure of melody and the long-term structure of a of a musical style, these experiments represent an advance in neural network music composition.

References

1. J. J. Bharucha and P. M. Todd. Modeling the perception of tonal structure with neural nets. *Computer Music Journal*, 13(4):44–53, 1989.
2. F. A. Gers, J.A. Perez-Ortiz, D. Eck, and J. Schmidhuber. DEKF-LSTM. In *Proc. 10th European Symposium on Artificial Neural Networks, ESANN 2002*, 2002. submitted.
3. 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.
4. F. A. Gers and J. Schmidhuber. LSTM recurrent networks learn simple context free and context sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001.
5. F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000.
6. 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, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.
7. Sepp Hochreiter and Juergen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.
8. Michael. C. Mozer. Induction of multiscale temporal structure. In D. S. Lippman, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 4*, pages 275–282. San Mateo, CA: Morgan Kaufmann, 1992.
9. Michael C. Mozer. Neural network composition by prediction: Exploring the benefits of psychophysical constraints and multiscale processing. *Cognitive Science*, 6:247–280, 1994.

10. A. J. Robinson and F. Fallside. The utility driven dynamic error propagation network. Technical Report CUED/F-INFENG/TR.1, Cambridge University Engineering Department, 1987.
11. C. Stevens and J. Wiles. Representations of tonal music: A case study in the development of temporal relationship. In M.C. Mozer, P. Smolensky, D.S. Touretsky, J.L Elman, and A. S. Weigend, editors, *Proceedings of the 1993 Connectionist Models Summer School*, pages 228–235. Erlbaum, Hillsdale, NJ, 1994.
12. Peter M. Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 13(4):27–43, 1989.