Learning Musical Structure Directly from Sequences of Music*

Douglas Eck and Jasmin Lapamle
University of Montreal
Department of Computer Science
CP 6128, Succ. Centre-Ville
Montreal, Quebec H3C 3J7 Canada
eckdoug,lapalmej@iro.umontreal.ca

Abstract

This paper addresses the challenge of learning global musical structure from databases of music sequences. We introduce a music-specific sequence learner that combines an LSTM recurrent neural network with an autocorrelation-based predictor of metrical structure. The model is able to learn arbitrary long-timescale correlations in music but is biased towards finding correlations that are aligned with the meter of the piece. This biasing allows the model to work with low learning capacity and thus to avoid overfitting. In a set of simulations we show that the model can learn the global temporal structure of a musical style by simply trying to predict the next note in a set of pieces selected from that style. To test whether global structure has in fact been been learned, we use the model to generate new pieces of music in that style. In a discussion of the model we highlight its sensitivity to three distinct levels of temporal order in music corresponding to local structure, long-timescale metrical structure and long-timescale non-metrical structure.

1 INTRODUCTION

In this paper, we present a music structure learner based on the LSTM recurrent neural network Hochreiter and Schmidhuber (1997). When trained on a corpus of songs selected from a musical style, this model is able to build a relatively low-capacity representation that captures important long-timescale aspects of the style such as chord structure. Because this global structure is at the heart of musical style, learning it directly from music sequences would be useful for many MIR-related applications such as similarity rating as well as for artistic applications such as automatic composition and improvisation. Our simulation results include generated music. However our focus is not on the quality or interestingness of this music. Instead we focus on relevant model details, including the music results as a demonstration that the model has successfully captured global stylistic constraints.

Finding long-timescale structure in sequences is difficult. Regardless of architecture (e.g. Hidden Markov Model, recurrent neural network, etc.) there is an explosion of possibilities that arises from a search for correlation at long timelags in a sequence. (A few details about why it is difficult are presented in Section 3.1.) Yet long-timescale structure is fundamental to music, as evidenced by its central role in theories like Lerdahl and Jackendoff (1983).

*Draft. Do Not Cite.

Our goal is to learn such structure directly from sequences using very little built-in knowledge. Such structure learning could aid in identifying stylistic similarities in musical examples that share little local structure. For example, consider several versions of *My Favorite Things* (Rodgers and Hammerstein) including the original sung version by Julie Andrews from "The Sound of Music" and the well-known bebop-jazz version by John Coltrane from the album "My Favorite Things". These songs have the same (or similar) melodies but are otherwise very different (more examples at `www.iro.umontreal.ca/~eckdoug/favorite_things`.)

Our current work is limited in two ways. First, it deals only with sequences drawn from MIDI files. Second, it does not treat performed music. However the core algorithms used in our model (autocorrelation-based meter detection and LSTM sequence learning) are well suited to working with digital audio Eck (2004) and are robust (Gers et al., 2002) to the kinds of temporal noise encountered in performance.

In previous work Eck and Schmidhuber (2002) demonstrated that a standard LSTM-based music learner can learn a fixed, simple chord structure. The novelty in the current model lies in the addition of time-delay connections that correspond to the metrical hierarchy of a particular piece of music. This meter information provides the LSTM network with musically-important temporal structure, freeing LSTM to learn other correlations in the input. With the addition of metrical structure, our model is able to capture some of the repetitive structure that is crucial to learning a musical style.

In Section 2 we discuss the importance of meter in music. In Section 3 we will describe details of the LSTM model. Finally in Section 4 through Section 6 we will describe our simulations and analyze results.

2 Meter

Meter is the sense of strong and weak beats that arises from the interaction among hierarchical levels of sequences having nested periodic components. Such a hierarchy is implied in Western music notation, where different levels are indicated by kinds of notes (whole notes, half notes, quarter notes, etc.) and where bars establish measures of an equal number of beats (Handel, 1993). For instance, most contemporary pop songs are built on four-beat meters. In such songs, the first and third beats are usually emphasized. Knowing the meter of a piece of music helps in predicting other components of musical structure such as the location of chord changes and repetition boundaries (Cooper and Meyer, 1960).

Meter provides us with key information about musical structure. Music, at least popular Western music, tends to be chunked in ways that correspond to meter. Chord changes, for example, usually occur on metrical boundaries. Also, music tends to repeat at intervals corresponding to the metrical hierarchy. Repetition poses a particularly difficult challenge for models such as neural networks and graphical models (e.g. Hidden Markov models) because it requires memorization. A dynamic learner such as a recurrent neural network (details described below) could learn to repeat a fixed number of learned patterns, but it could not learn to repeat an *arbitrary* sequence because it has no way to implement content-addressable memory. Graphical models like Hidden Markov Models (HMMs) suffer the same constraint: to repeat an arbitrary pattern would require an explosive number of states.

Yet repetition is fundamental to music, with even children's music making heavy use of it. We suppose that one reason it is possible for children to master repetitive music is that the repetition

boundaries are aligned with the meter of the music. This provides a motivation for building into our model a bias towards metrical structure.

Our approach here is to provide meter to the network in the form of delayed inputs. Our representation of music as a quantized time series makes this relatively easy. We sample the music k times per measure. In the case of these simulations, $k = 8$ or every eighth note. Thus for a meter of 4/4 we can make it easy for the network to correlate metrically-relevant delays by providing time-delayed copies of the input input at, e.g., $t - 15$, $t - 31$ and $t - 63$ (corresponding to two measures, four measures and eight measures respectively). The network can still attend to other lags via its recurrent learning mechanism, but the lags related to meter are given special salience.

The model is not limited to pieces of music where the meter is provided. In cases where the meter is missing, it can be computed using a meter extraction algorithm from Eck (2004). This algorithm processes a MIDI file or audio file and returns a series of timelags corresponding to multiple levels in the metrical hierarchy. It works by searching through a space of candidate meters and tempos using an autocorrelation representation. The selected meter is then phase aligned with the music. Though this meter extraction method will occasionally be wrong, this poses no serious problem for our current model because LSTM works well with noisy datasets Gers (2001).

3 Model

In this section, we will explain how our model is trained and how we use the model to generate new songs. We will introduce recurrent neural networks (RNNs) and a specific kind of RNN called Long Short Term Memory (LSTM). We will also describe the simulation framework of next-step prediction.

3.1 Recurrent neural networks (RNNs)

A neural network is a statistical learning model that uses gradient descent to minimize prediction error via the adjustment of weighted connections. A *recurrent* neural network contains self connections. These connections are important for time-series prediction tasks because with them, a network can learn to take advantage of its own internal state (a set of numeric *activations*) as the time series evolves over time. By contrast, neural networks without recurrent self-connections (i.e. feed forward networks) are unable to discover correlations that depend on the order of pattern presentation making them ill-suited for time series tasks where there are important non-stationarities, such as in music, where there is a clear temporal evolution.

Recurrent neural networks can in principle learn arbitrary temporal structure. Unfortunately in practice this is not the case. The difficulty lies in correlating events that are separated by many timesteps. (For example, to learn a musical style such as blues, it is very important to predict when chord changes will occur. Yet chords can be separated by many notes.) To make the necessary long-timescale correlations, a recurrent network must propagate error "back in time". There are several strategies for achieving this, for example by transforming a recurrent network into a very large feed-forward network where layers correspond to timesteps (Back-Propagation Through Time, BPTT, (Williams and Zipser, 1995)) or by using tables of partial derivatives to store the same information (Real-Time Recurrent Learning, RTRL, (Robinson and Fallside, 1987). But all of them suffer from the flaw that the error signal becomes diluted as it is propagated backwards in time. This behavior (called the "vanishing gradient" problem) affects all systems that use standard gradient descent to

store information (Bengio et al., 1994; Hochreiter et al., 2001).

This explains the unsatisfactory results of one influential attempt to use recurrent networks to learn a musical style Mozer (1994). Even with sound RNN techniques and psychologically-realistic distributed representation, Mozer's "CONCERT" architecture failed to capture global musical structure. (In his excellent paper, Mozer cannot be faulted for making inflated claims. He described the output of his model as "music only its mother could love".) Though networks regularly outperformed third-order transition table approaches, they failed in all cases to find global structure. This also explains why neural networks designed to generate music such as Todd (1989) and Stevens and Wiles (1994) are successful at capturing local interactions between musical events but are unable to capture global musical structure.

3.2 Long Short-Term Memory (LSTM)

Eck and Schmidhuber (2002) improved on the state-of-the-art in music sequence learning by using an LSTM network designed to capture long-timescale dependencies. We demonstrated that a standard LSTM network can learn global musical structure using an input representation similar to that used by Mozer. Later these results were extended by Franklin to include different rhythm and pitch encodings Franklin (2004).

The success of LSTM can be explained in terms of the vanishing gradient problem. As will be described in more detail below, LSTM's architecture is designed to allow errors to flow backward in time without degradation. Special continuous values called *Cells* can use this error to build an internal state that is unbounded (save by the resolution of double-precision values in the computer) and persists over time. Of course, this is not a completely general solution to the vanishing gradient problem. (A completely general solution seems impossible.) However the compromises made by LSTM allow it to work very well in many instances where other learners fail. For a more in-depth analysis of why LSTM works, readers are referred to Gers et al. (2000); Schmidhuber et al. (2002).

In Figure 1 see an LSTM network consisting of several LSTM memory blocks. Errors the flow between blocks are truncated to the current timestep, resulting in blocks that function more or less independently. This has the benefit that any number of LSTM blocks can be employed depending on task complexity, and that LSTM blocks can be mixed with standard non-recurrent units in a hybrid network.

A single LSTM block is shown in Figure 2. At the core of the LSTM block is an unbounded Cell (in gray) whose value is never altered by a nonlinear squashing function. Normally such an unsquashed value would be unstable in a recurrent network. In LSTM stability is achieved using gating units that themselves are nodes in the network trained using gradient descent. The Input Gate modulates the flow of information into the Cell, allowing the Cell to ignore information at certain times. The Forget Gate allows the Cell to learn to empty its Cell contents under appropriate circumstances. The Output Gate allows the Cell to hide its contents from other units in the network. For example, a block that is not performing well in a particular context might learn to take itself offline using the Output Gate. The entire block, including gates, is trained using back-propagation. The details are out of the scope of this paper, but can be described as a combination of standard back-propagation at the Output and Output Gate combined with a truncated version of Real Time Recurrent Learning (RTRL) in the Cell, Forget Gate and Input Gate. For a complete treatment of the forward and backward pass Gers (2001).

Our LSTM music structure learner also made use of a standard feed-forward hidden layer in parallel to the LSTM blocks, generating a network similar to that in Figure 3. The feed-forward

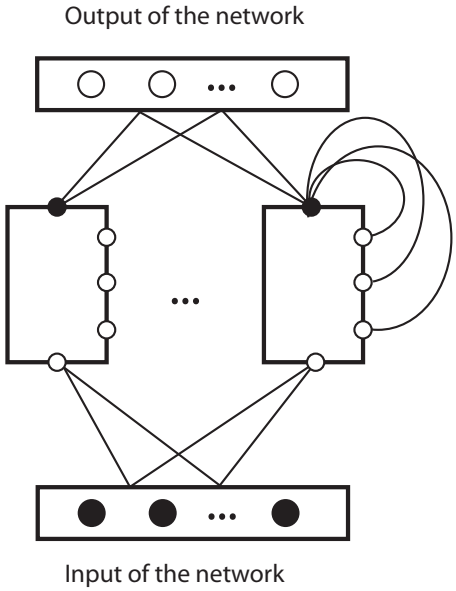

Figure 1: An LSTM network. For clarity, some edges are not shown. Black circles denote origins and white circles denote destinations. In a full network all origins and destinations would be connected. In addition the input and output layers can be connected directly.

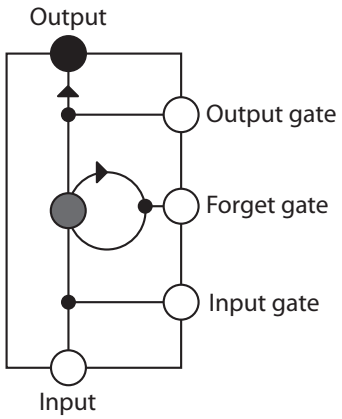

Figure 2: An LSTM block with a single Cell shown in gray. The Cell is not squashed and can obtain any positive or negative continuous value. When multiple Cells are used, they all share the same gates. The gates on the right are used to **multiply** information as it flows through the Cell. The multiplications are denoted with small black dots.

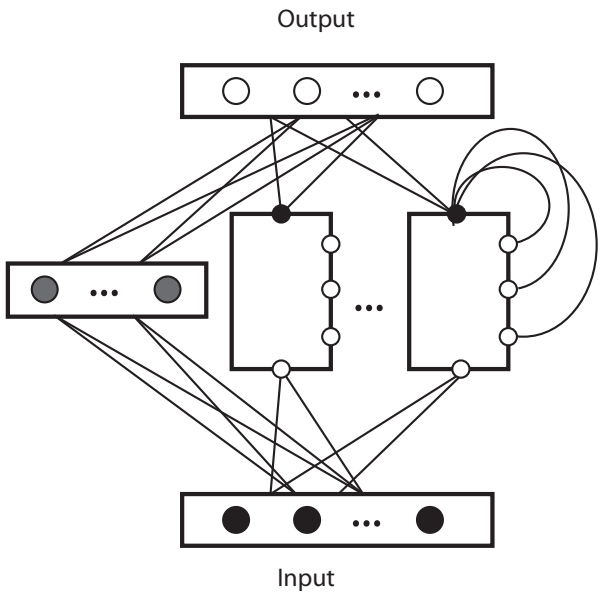

Figure 3: A slightly more complex LSTM network with a standard feed-forward layer in parallel. The feed-forward layer accelerated training by learning local dependencies more quickly than LSTM could alone. This had some positive smoothing effects on the performance of the network.

layer helped by quickly capturing local dependencies, thus allowing the LSTM cells to handle longer timescale dependencies. This resulted in melodies with smoother contours. Our simulations showed that neither a feed-forward network alone nor an LSTM network alone could outperform the combined network.

3.3 Next-step prediction

Following Mozer (1994) we will train the network to predict the probability density over all possible notes at time t using as input the note (and chord) values at time $t-1$. This general approach is called next-step prediction. In the current model, the network receives as input not only the sequence delayed by a single lag $(t-1)$ but also delayed by lags corresponding to the metrical structure of the piece.

Multiple songs are presented to the network as a single long sequence. However we truncate the flow of error at song boundaries so that the network does not learn spurious correlations from one song to another.

3.4 A Generative Model

Once trained, we can generate music with the model by presenting it with the first few notes of a song that it has never seen in training and then using network predictions to generate network inputs. Network predictions are conditioned using a softmax function, ensuring that the sum of the output vector is 1.0. This allows us to interpret the output vector as a probability estimation

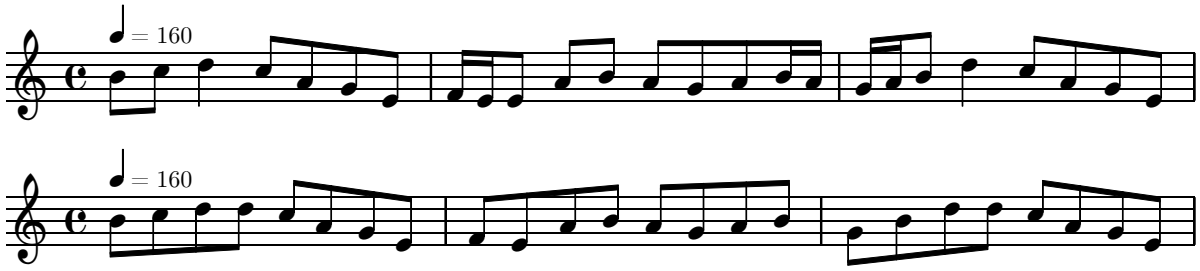

Figure 4: On the top staff, a segment from the original dataset; on the bottom, the quantized version.

from which we can select the next note. The selected note is then presented to the network at the next timestep as an input.

For some simulations, we applied a threshold to our note generation, ensuring that very low probability notes would not be chosen. The threshold we used was $1/N$ where N is the cardinality of the output vector. We recognize that this heuristic departs from an interpretation of the output vector as a probability estimation. See Section 6 for a discussion of this choice.

3.5 Preprocessing and Representation

We presume that our dataset is encoded in standard MIDI. This is not a severe constraint as most other input encodings such as Humdrum and ABC can easily be converted to MIDI. We built input and target vectors by sampling (quantizing) the MIDI file at eighth-note intervals. An example of the quantization is shown in Figure 4. We limited the number of octaves available, folding notes that fall outside of that octave to the nearest allowed octave. For these simulations we chose the interval between C3 and C5 indicating that, for example, a D2 in the dataset would be transformed into at D3.

Our quantization strategy allows us to represent time implicitly. That is, there are no units in the input or output dedicated to representing note duration. This strategy is identical to that used in Eck and Schmidhuber (2002) but differs from the approaches of others such as Mozer (1994) and Franklin (2004).

Notes are represented locally using a one-hot vector (i.e. every note in the corpus receives a dedicated input and output dimension in the vector). Notes that never appear in the corpus are not represented.

Chords are also represented using local units in a one-hot vector. Thus, for example, an Fmaj7 would be encoded in a single input unit rather than as several units representing the notes that make up the chord. This is a departure from Eck and Schmidhuber (2002) where chords were represented in a distributed manner. The current work has the advantage that the network can more quickly learn the chord structure but has the disadvantage that the network cannot generalize to unseen chords.

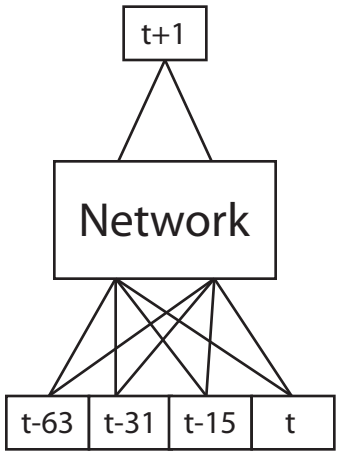

Figure 5: The one-hot output vector is shown at the top. The one-hot input vectors of chords and notes corresponding to the delayed versions of the input are at the bottom.

3.6 Encoding Meter Using Time Delays

In order to encode metrical structure for the network we add several additional one-hot vectors to the input layer of the network corresponding to time-delayed versions of the input. For the simulations in this paper we used the notated meter found in the MIDI file.

See in Figure 5 an example of an LSTM network with the input and output structures described above.

Though a network like LSTM can in principle identify these lags by itself, it proves in practice to be very difficult. This is better understood by observing that LSTM is searching for repetition at *any* lag. This is at least as difficult as correctly identifying strings in the simple context free grammar A^nB^n where n is unbounded. LSTM can in fact do this very well Schmidhuber et al. (2002), perhaps better than any other dynamical learning system. However by providing the metrical structure to the network in the form of delayed inputs, this unbounded search (in the space of possible values of n) is bounded to be a search for strings A^kB^k where k is one of the lags identified in the metrical structure. In short, LSTM still looks for repeating structures, but the lag at which LSTM will likely look is strongly biased towards metrical boundaries. We believe this implements a musically-reasonable and particularly powerful prior on the search. At the same time we observe that LSTM can always search at other lags in the input using its own dynamical gating mechanism (the same mechanism it used to solve the A^nB^n problem) to identify other important long-timescale dependencies that do not align with metrical structure.

The basic idea of using time delays in recurrent neural networks is not new. In fact time delay neural networks are themselves a large and well-studied class of dynamical learning models. See Kolen and Kremer (2001) for an overview. What makes our approach special is our use a musically-motivated preprocessing method to provide the "correct" delays to the network (where "correct" means metrically salient).

3.7 Postprocessing

To listen to the output of the network, we translate network predictions into standard MIDI using our own software. Because note duration is not encoded explicitly, it is unclear whether to interpret, e.g., eight consecutive D3s as a single D3 whole note, four D3 quarter notes or eight D3 eighth notes. We resolve this by always choosing the longest possible note duration suggested by the output. In addition we use a strategy first employed by Mozer to break all notes at the measure boundary That is, we disallow tied notes. This postprocessing seemed to work well with the current datasets but would need to be addressed for other forms of music. We address this in Section 6.

4 Experiments

We use the model to learn sequences using next-step prediction. However, the training and testing error of such an exercise is not of great value because it confounds the learning of local structure with the learning of global structure. Given our goal of focusing on global structure learning, we used the task of music generation to test the performance of the model. As we have already pointed out, our focus is not on the artistic quality or interestingness of the compositions themselves, but rather on their ability to reflect the learning of global structure.

All of our experiments employed the meter time-delay strategy for encoding metrical structure. For less-satisfying results using no meter time delays see Eck and Schmidhuber (2002). We performed three sets of experiments using the following models and datasets:

- A baseline standard feed-forward neural network trained on melodies
- An LSTM recurrent neural network trained on melodies
- An LSTM recurrent neural network trained on chords and melodies

4.1 Databases

For our experiments we used examples of traditional Irish reels. Reels are relatively fast 4/4 songs used for accompanying dance.

A first selection of reels was taken from `http://www.thesession.org`, a repository of Irish folk music encoded using a music typesetting language called ABC (`http://abc.sourceforge.net`. At the time of writing this paper, there were over 1700 reels available at the website. We selected all the reels in the keys of C major and E major, yielding a subset of 56 songs. We trained our model using songs from only one key at at time. For this database, we were unable to obtain a large set of songs having labeled chords and so used only melodies.

A second selection of 435 reels was taken from the Nottingham database found at `http://www.cs.nott.ac.uk/~ef/music/database.htm`. For this dataset, we transposed all songs into the same key. In addition to melodies, this database also provides chord information that we used in our simulations.

4.2 Melodies with baseline feed forward network

To compute a baseline, we used a standard feed-forward network neural network. The network contained a single hidden layer and used standard logistic sigmoid activation functions. Note that

the extent to which this baseline model succeeds at capturing any repetition structure at all is thanks to the meter time-delayed inputs.

We trained the model with the E-major reels from the Session database. The hyperparameters used were as follows:

Hidden units	Stages	Batch size	learning rate
4	500	50	0.05
12	500	50	0.05
16	500	50	0.05

4.3 Melodies with LSTM

We compared the performance of the baseline model to an LSTM network constructed as described in the sections above. Here our dataset consisted of the C-major reels from the Session database. We used the following hyperparameters, with learning rate fixed at .05 and batch size fixed at 50 as in the baseline model:

Sets	Hidden units	LSTM blocks	Cells in each LSTM block	Stages
Cmaj	0	2	1	500
Cmaj	4	2	1	500
Emaj	0	2	1	500
Emaj	4	2	1	5000

4.4 Melodies and chords with LSTM

In this last set of experiments, we add the chords to see if LSTM can generalize the melodies as well as the chords. The input representation change a little bit. Chords were represented in a one-hot vector as described in sections above. Here our dataset consisted of the reels from the Nottingham database, all transposed into C major. We used the following hyperparameters:

Hidden units	LSTM blocks	Cells in each LSTM block	Stages
4	2	1	100
4	2	1	500

5 Results

Compared to previous attempts using neural networks, including our previous work with LSTM, the results were impressive. We invite readers to visit our website of examples of generated songs at `http://www-etud.iro.umontreal.ca/~lapalmej/ismir/lstm.html` There you will find examples in MIDI and in Mp3 format for all three models described above. Beware that for all songs first 8 measures were used to "seed" the generative model and are taken directly from songs in the

validation set! The generation of an original sequence begins after these initial 8 measures. Note that none of the models exhibit such over-fitting that they simply repeat the seed sequence.

5.1 Baseline Feed Forward Model on Melodies

The melodies generated by the simple feed-forward are quite good for such a simple model. The model is able to take advantage of the time delay connections, as evidenced by repeated themes. However after some time, most of the baseline models become stuck in a repeating loop of, say, 16 notes.

5.2 LSTM on Melodies

LSTM does a better job of generating elaborations around a core melody. To our ear, the results were pleasant. Surely not everyone will love these melodies — nor do we like them so much that we put them in our lab's MP3 database — but they are interesting to hear. We did have to take care not to over-fit the dataset with too high network capacity (too many nodes). careful not to "over-fit" with too much capacity or too much training because it will just repeat constantly the same notes.

5.3 LSTM on Melodies and Chords:

Here we used more capacity to allow the model to learn both the melodies and the chords. Of particular interest was whether the model could learn and reproduce the chord structure such that generated sequences were coherent examples of the reel style.

Here results were mixed but very promising. The LSTM model can generate new interesting melodies. The chords changes were better than previous attempts and were reasonable, tending to follow metric boundaries, but were not perfect. One interesting quality of the compositions is that (perhaps not surprisingly) the melodies do follow the chord structure suggested by the model. This makes the compositions more interesting to listen to and also suggests that improvements in learning the chord structure will indeed result in better compositions. More importantly it reveals that the model has captured a slow-changing chord structure and is able to synchronize faster-changing melodic structure with those chords.

6 Discussion

We believe that the learning of musical style takes place at several timescales, something our model is particularly well-suited to address. We also believe that in the long run models that learn by example, such as ours, show great promise due to their ability to identify statistical regularities in a training set, thus lessening the need to provide expert-level prior knowledge We have addressed this challenge by building a model that responds to musical structure on at least three levels: Local structure is learned using standard feed-forward connections. Hierarchical metrical structure is learned via the time delay connections provided in the input layer. Non-hierarchical long-timescale structure is learned using LSTM.

When any of these are lacking, the ability to learn musical style suffers. Models that use only local structure, such as a feed-forward network with no time delays or N-gram models, lack all high-level musical structure and can only produce "aimless" music with interesting note combinations.

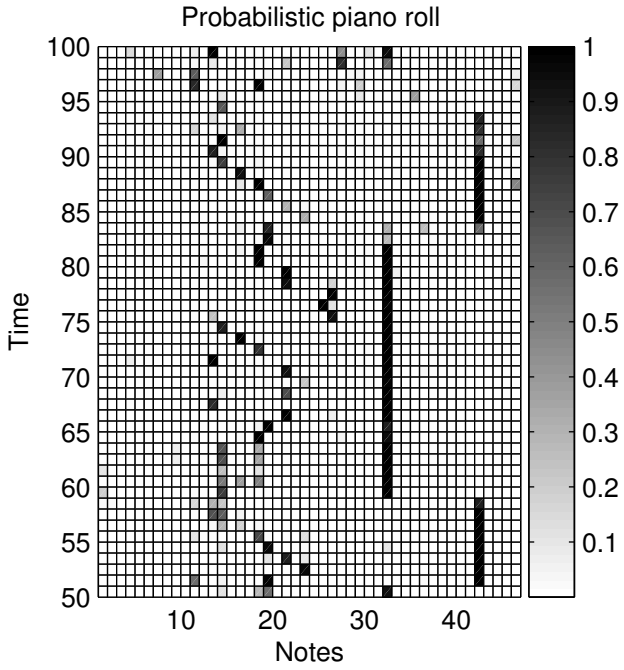

Figure 6: The probability of notes (columns 1 to 26) and chords (columns 27 to 48)). Time flows upwards. The darker points represent higher probability for selection. Rows with darker points can be interpreted as corresponding to parts of the song where the network has high certainty.

Models that use only fixed metrical structure such as the feed-forward network with time delayed inputs tend to create another kind of "aimless" music that produces fixed-length repeated loops. Though our LSTM model is flawed, it does generate music that manages to incorporate several levels of musical structure.

7 Possible applications and future work

Despite having presented music compositions as a measure of model performance, in our view the least interesting use for such a model is standard automatic music composition. We see at least two other applications of the model. First, as discussed in the introduction, the model could be used to rate music similarity. For example, different networks trained on different styles could rate novel pieces for goodness-of-fit in their learned style. Second, the model could be used as part of a music analysis tool. Here the ability to predict the probability density of possible notes in time could be used to provide a picture uncertainty in an unfolding performance. This uncertainly is only implicit in the current version but could be explicitly computed in future work. As an example of this kind of application we computed a "probabilistic piano roll" of one of the reels in the database. This was generated by running the correct note inputs through the trained network and storing the predicted outputs. See Figure 6 for an example.

Second the model could form the core of an online music generator for video games. One could train the network on musical examples that are labeled by their level of (for example) "danger"

or "safety". By training the network on both the music and a parameter corresponding to danger/safety level, it should be possible to build a model that can generate "dangerous" music as game context becomes tense and "safe" music as game context becomes calmer, provided the game designer can provide a parameter at game time corresponding to this value.

Finally the model could easily be used in the context of music improvisation. It is easier to train the model on either chords or on melodies. By training the model to produce chords in response to melodies, it would be possible to create an automatic accompaniment system. By reversing this and producing melodies in response to chords, one could build an automatic melody improviser. Either could respond in real time to a performing musician.

8 Conclusion

There is ample evidence that LSTM is good model for discovering and learning long-timescale dependencies in a time series. By providing LSTM with information about metrical structure in the form of time-delayed inputs, we have built a music structure learner able to use global music structure to learn a musical style. Such a model has potential in the domain of music similarity, especially for identifying similarity based on long-timescale structure. The model had two basic components, the meter time-delayed inputs supplied by an autocorrelation meter detection algorithm and the LSTM network. Our simulations demonstrated that the full model performs better than a simpler feed-forward network using the same meter time-delayed input and better than an LSTM network without the delays. We argue that the model is conceptually interesting because it is sensitive to three distinct levels of temporal ordering in music corresponding to local structure, long-timescale metrical structure and long-timescale non-metrical structure.

References

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Cooper, G. and Meyer, L. B. (1960). *The Rhythmic Structure of Music*. The Univ. of Chicago Press.

Eck, D. (2004). A machine-learning approach to musical sequence induction that uses autocorrelation to bridge long timelags. In Lipscomb, S., Ashley, R., Gjerdingen, R., and Webster, P., editors, *The Proceedings of the Eighth International Conference on Music Perception and Cognition (ICMPC8)*. Causal Productions.

Eck, D. and Schmidhuber, J. (2002). Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In Bourlard, H., editor, *Neural Networks for Signal Processing XII, Proc. 2002 IEEE Workshop*, pages 747–756, New York. IEEE.

Franklin, J. (2004). Computational models for learning pitch and duration using lstm recurrent neural networks. In Lipscomb, S., Ashley, R., Gjerdingen, R., and Webster, P., editors, *The Proceedings of the Eighth International Conference on Music Perception and Cognition (ICMPC8)*, Adelaide, Australia. Causal Productions.

Gers, F., Schraudolph, N., and Schmidhuber, J. (2002). Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research (JMLR)*, 3:115–143.

Gers, F. A. (2001). *Long Short-Term Memory in Recurrent Neural Networks*. PhD thesis, Department of Computer Science, Swiss Federal Institute of Technology, Lausanne, EPFL, Switzerland.

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471.

Handel, S. (1993). *Listening: An introduction to the perception of auditory events*. MIT Press, Cambridge, Mass.

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

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Kolen, J. and Kremer, S., editors (2001). *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press.

Lerdahl, F. and Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. MIT Press, Cambridge, Mass.

Mozer, M. C. (1994). Neural network composition by prediction: Exploring the benefits of psychophysical constraints and multiscale processing. *Cognitive Science*, 6:247–280.

Robinson, A. J. and Fallside, F. (1987). The utility driven dynamic error propagation network. Technical Report CUED/F-INFENG/TR.1, Cambridge Univ.

Schmidhuber, J., Gers, F., and Eck, D. (2002). Learning nonregular languages: A comparison of simple recurrent networks and LSTM. *Neural Computation*, 14(9):2039–2041.

Stevens, C. and Wiles, J. (1994). Representations of tonal music: A case study in the development of temporal relationship. In Mozer, M., Smolensky, P., Touretsky, D., Elman, J., and Weigend, A. S., editors, *Proceedings of the 1993 Connectionist Models Summer School*, pages 228–235. Erlbaum, Hillsdale, NJ.

Todd, P. M. (1989). A connectionist approach to algorithmic composition. *Computer Music Journal*, 13(4):27–43.

Williams, R. J. and Zipser, D. (1995). Gradient-based learning algorithms for recurrent networks and their computational complexity. In Chauvin, Y. and Rumelhart, D. E., editors, *Back-propagation: Theory, Architectures and Applications*, chapter 13, pages 433–486. Hillsdale, NJ: Erlbaum.