Jazz Melody Generation from Recurrent Network Learning of Several Human Melodies

Judy A. Franklin

Smith College
Computer Science Department
Northampton, MA 01063
jfranklin@cs.smith.edu

Abstract

Recurrent (neural) networks have been deployed as models for learning musical processes, by computational scientists who study processes such as dynamic systems. Over time, more intricate music has been learned as the state of the art in recurrent networks improves. One particular recurrent network, the Long Short-Term Memory (LSTM) network shows promise as a module that can learn long songs, and generate new songs. We are experimenting with using two LSTM modules to cooperatively learn several human melodies, based on the songs' harmonic structures, and the feedback inherent in the network. We show that these networks can learn to reproduce four human melodies. We then introduce two harmonizations, constructed by us, that are given to the learned networks. i.e. we supply a reharmonization of the song structure, so as to generate new songs. We describe the reharmonizations, and show the new melodies that result. We also use a different harmonic structure from an existing jazz song not in the training set, to generate a new melody.

LSTM Networks as Modules in a Music Learning System

Recurrent neural networks are artificial neural networks that have connections from the outputs of some or all of the network's nonlinear processing units back to some or all of the inputs. These networks are trained by repeatedly presenting inputs and target outputs and iteratively adjusting the connecting weights so as to minimize some error measure. The advantage of recurrent neural networks is that outputs are functions of previous states of the network, and sequential relationships can be learned. However, this very facet causes the weight update equations to be much more complex than simple non-recurrent neural networks, to correct for using erroneous outputs in previous time steps. And it is difficult to design a stable network that can learn long sequences. Yet, this is necessary for musical learning systems.

In recent publications (Franklin 2004a, Franklin 2004b, Franklin and Locke 2005a), we have shown that a particular recurrent neural network, the long short-term memory network (LSTM), can learn to distinguish musical pitch sequences, and can learn long songs. Here we present a two-module LSTM system that can learn both pitch and duration of notes in several long songs, and can subsequently be used to generate new songs. While we have developed systems before based on similar ideas, the LSTM-based system is much more precise and stable, and can learn much longer songs. Figure 1 shows our two-module LSTM configuration for learning songs. This configuration is inspired by Mozer's (1994) CONCERT system that uses one recurrent network, but with two sets of outputs, one for pitch and one for duration. It is also inspired by Eck and Schmidhuber's (2002) use of two LSTM networks for blues music learning, in which one network learns chords and one learns pitches; duration is determined by how many network iterations a single pitch remains on the output.

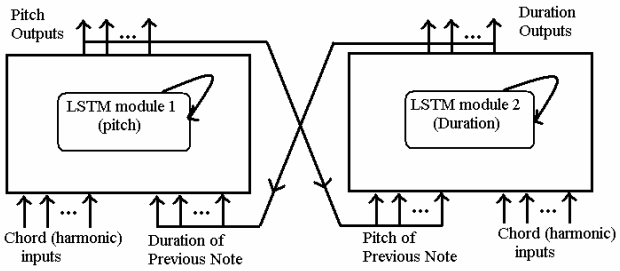

Figure 1. The two-module LSTM system. One LSTM module learns pitches. The other learns note durations. The recurrence in each LSTM network is shown internally.

Each LSTM module contains an LSTM neural network. An LSTM neural network is a kind of recurrent neural network with conventional input and output units, but with an unconventional, recurrent, hidden layer of memory blocks (Hochreiter and Schmidhuber 1997, Gers et al. 2000). Each memory block contains several units (see Figure 2). First there are one or more self-recurrent linear

memory cells per block. The self-recurrence on each cell enables it to accumulate numerical values over a series of iterations of the network. The accumulated data (cell output) is passed through a nonlinear function. Second, each block contains three gating units that are typical sigmoid units, but are used in the unusual way of controlling access to the memory cells. One gate learns to control when the cells' outputs are passed out of the block, one learns to control when inputs are allowed to pass in to the cells, and a third one learns when it is appropriate to reset the memory cells. LSTM's designers were driven by the desire to design a network that could overcome the vanishing gradient problem (Hochreiter et al. 2001). Over time, as gradient information is passed backward to update weights whose values affect later outputs, the error/gradient information is continually decreased by weight update scalar values that are typically less than one. Because of this, the gradient vanishes. Yet the presence of an input value way back in time may be the best predictor of a value far forward in time. LSTM offers a mechanism where linear units can accumulate and hold important data without degradation, and release it many iterations later.

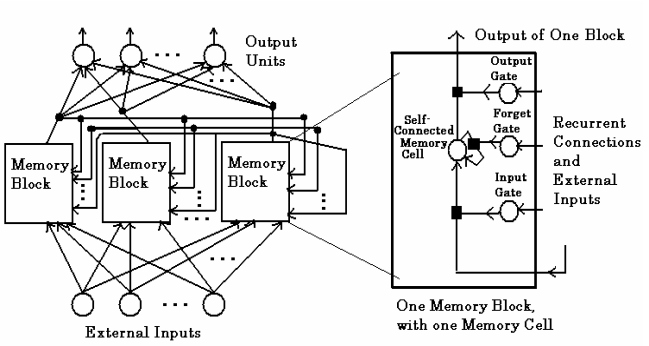

Figure 2. An LSTM network, and an enlargement of a memory block containing one memory cell, and the three gating units.

The complete equations for the LSTM network are beyond the scope of this paper. Hochreiter and Schmidhuber 1997 provide the detailed derivation., Gers et al. 2000 add the forget gates, and also provide detailed pseudo-code that is very valuable for implementation. However, a brief summary of the algorithm is given in Figure 3.

As shown in Figure 1, one LSTM network learns to reproduce the pitches of one or more songs, and a second one learns to reproduce the corresponding durations. The dual system contains recurrence in three places: the inter-recurrence at the network level, the recurrence of the hidden layer of memory blocks, and the self-recurrence of each memory cell. We showed in previous work (Franklin 2005b) that a similar system, with two LSTM networks that are not inter-recurrent, can learn both a human rendition of the song "Afro Blue", as well as a score-based version.

Symbol Definitions

c_j^v	v^{th}cell of memory block j
i_m	m^{th}input to memory block j
$s_{c_j^v}(t)$	output of memory cell c_j^v
$y^{in_j}(t)$	input gate output, $0 < y^{in_j}(t) < 1$
$y^{\phi_j}(t)$	forget gate output, $0 < y^{\phi_j}(t) < 1$
$y^{out_j}(t)$	output gate output, $0 < y^{out_j}(t) < 1$
$y^{c_j^v}(t)$	v^{th}output of memory block j
$y^k(t)$	output unit outputs
$(t^k(t) - y^k(t))$	k^{th}output unit target $-$ k^{th}output unit output
$w_{km}(t)$	general form: weights connecting memory-block outputs to network outputs
h	sigmoid function ranging from -1 to 1

Feed-forward Pass

$$g(net_{c_j^v(t)}) = g(\sum_1^m w_{c_j^v m} i_m)$$
$$s_{c_j^v}(t) = y^{\phi_j}(t) s_{c_j^v}(t-1) + y^{in_j}(t) g(net_{c_j^v(t)})$$
$$y^{c_j^v}(t) = y^{out_j}(t) h(s_{c_j^v}(t))$$
$$y^{out_j}(t) = f(net_{out_j}(t))$$
$$y^{\phi_j}(t) = f(net_{phi_j}(t))$$
$$y^{in_j}(t) = f(net_{in_i}(t))$$

Errors and Delta Feedback

$$e_k(t) = f_k'(net_k(t))(t^k(t) - y^k(t))$$
$$\Delta w_{km}(t) = \alpha\, e_k(t) h_m(t) y^{out_j}(t)$$
$$\Delta w_{out_j m}(t) = \alpha\, f'_{out_j}(net_{out_j}(t)) \sum_{k \in outputunits} e_k(t) \sum_{v=1}^n w_{kc_j^v} h(s_{c_j^v}(t)) x_m(t)$$
$$e_{s_{c_j^v}}(t) = y^{out_j}(t) h'(s_{c_j^v}(t)) \sum_{k \in outputunits} w_{kc_j^v} e_k(t)$$
$$\Delta w_{c_j^v m} = \alpha e_{s_{c_j^v}} \frac{\partial s_{c_j^v}(t)}{\partial w_{c_j^v m}}$$
$$\Delta w_{\phi_j m} = \alpha \sum_{v=1}^n e_{s_{c_j^v}} \frac{\partial s_{c_j^v}(t)}{\partial w_{\phi_j m}}$$
$$\Delta w_{in_j m}(t) = \alpha \sum_{v=1}^n e_{s_{c_j^v}} \frac{\partial s_{c_j^v}(t)}{\partial w_{in_j m}}$$
$$\frac{\partial s_{c_j^v}(t)}{\partial w_{c_j^v m}} = \frac{\partial s_{c_j^v}(t-1)}{\partial w_{c_j^v m}} y^{\phi_j}(t) + g'(net_{c_j^v}) y^{in_j}(t) x_m(t)$$
$$\frac{\partial s_{c_j^v}(t)}{\partial w_{\phi_j m}} = \frac{\partial s_{c_j^v}(t-1)}{\partial w_{\phi_j m}} y^{\phi_j}(t) + s_{c_j^v}(t-1) f'_{\phi_j}(net_{\phi_j}(t)) x_m(t)$$
$$\frac{\partial s_{c_j^v}(t)}{\partial w_{in_j m}} = \frac{\partial s_{c_j^v}(t-1)}{\partial w_{in_j m}} y^{\phi_j}(t) + g(net_{c_j^v}(t)) f'_{in_j}(net_{in_j}(t)) x_m(t).$$

Figure 3. Synopsis of forward and backward pass of the LSTM algorithm.

The two networks can also learn Afro Blue with and without the harmonic structure (the chords) given as input. To expand the system to be able to generate music, we now tie the pitch and duration networks together, so each network receives the outputs of the other network, for the previous note. We also present the harmonic structure corresponding to the example song to be learned to each network's input units. These inputs are the chord over which the current melody notes are played. The representations for duration, pitch, and chords are described in a later section. A small amount of beat information is given to the networks as one input with value 1 only if the beginning of a new measure has passed. Another set of inputs not shown in the diagram are a binary encoding of the number of the song being learned; one binary input for each of the four songs, that is 1 only if that song is the current training example.

The Four Songs

The songs learned by the two-module system are "Summertime", "Watermelon Man", "Blue Bossa", and "Cantaloupe Island." Each song is presented here, as a musical score of a human rendition of the melody, with the chord structure. Figures 4 and 5 show the songs Summertime and Watermelon Man. The human renditions were obtained from MIDI files found on the web. The chords for the songs are provided by (Aebersold 2000).

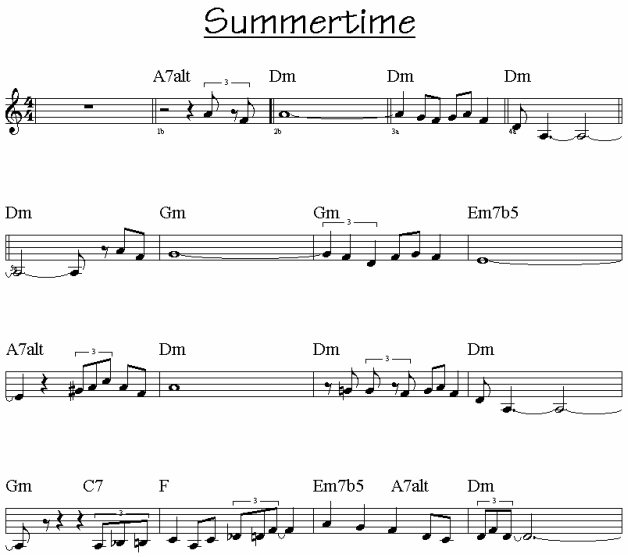

Figure 4. Summertime score, showing human rendition and harmonic structure.

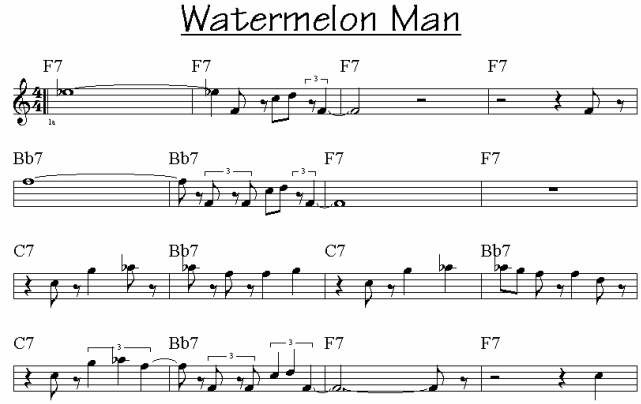

Figure 5. Watermelon Man score, showing human rendition and harmonic structure.

Figures 6 and 7 show the other two of the four songs, Blue Bossa, and Cantaloupe Island. Each song has a different harmonic structure, although there is some overlap in the individual chords that appear. Each song is in 4/4 time, with four beats per bar, and each has 16 bars. Three of the songs have lead-in notes before the first bar. The presentation of the songs as examples includes one lead-in measure so as to include the lead-in notes.

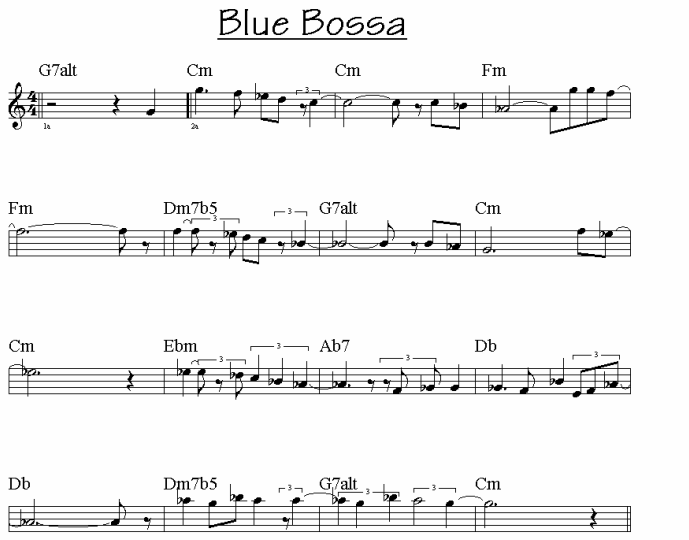

Figure 6. Blue Bossa score, showing human rendition and harmonic structure.

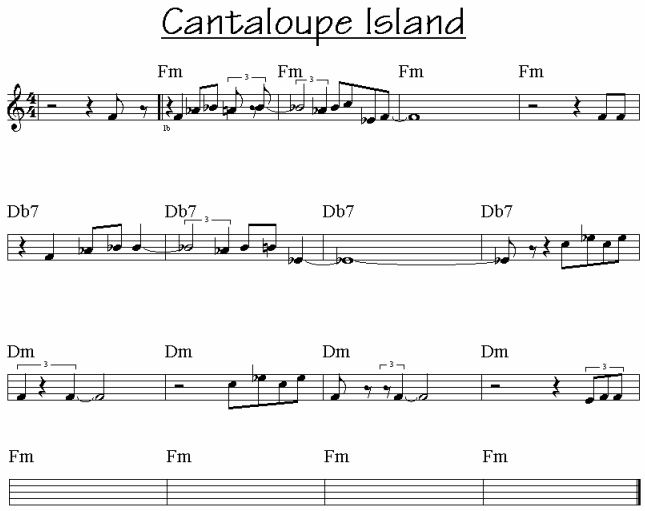

Figure 7. Cantaloupe Island score, showing human rendition and harmonic structure. Notice that the melody does not contain any notes in the last four bars of the song, although the song has a 16-bar harmonic structure.

Experimental Details

This section describes the representation of pitch and duration, as well as the learning parameters for the experiments.

Pitch Representation

The pitch of the note corresponds to the note's semi-tone, from Western tonal music. Pitch must be represented

numerically, and there are many ways to do this from a musicological point of view (Selfridge-Field 1998). Since our melody sources are MIDI-based, we often think of pitches as having an integer value, one value for each semi-tone, with 60 representing middle-C. But the pitch must be represented in a way that will enable a recurrent network to easily distinguish pitches. We have developed one such representation called Circles of Thirds. We have experimented with this representation on various musical tasks, with successful results (Franklin 2004a, Franklin and Locke, 2005) and have compared it to others such as those found in (Todd 1991, Mozer 1994, Eck and Schmidhuber 2002).

Figure 8 shows the four circles of major thirds, a major third being 4 half steps, and the three circles of minor thirds, a minor third being 3 half steps.

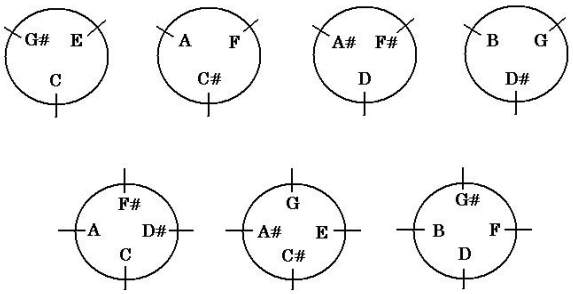

Figure 8. At top, circles of major thirds. At bottom, circles of minor thirds. A pitch is uniquely represented via these circles, ignoring octaves.

The representation consists of 7 bits. The first 4 indicate the circle of major thirds in which the pitch lies, and the second 3, the circle of minor thirds. The number of the circle the pitch lies in is encoded. C's representation is 1000100, indicating major circle 1 and minor circle 1, and D's is 0010001, indicating major circle 3, and minor circle 3. D# is 0001100. Octave information is indicated by two additional bits, one for octave C2-B2 and one for octave C4-B4. Only one of these bits would be nonzero at a time. If both are zero, then octave C3-B3 is indicated.

Because the 7th chord tone is so important to jazz, our chords are the triad plus 7th. In using Circles of Thirds to represent chords, we could represent chords as four separate pitches, each with seven bits for a total of 28 bits. However, it would be left up to the network to learn the relationship between chord tones. We borrowed from Laden and Keefe's (1991) research on overlapping chord tones as well as Mozer's (1994) more concise representation for chords. The result is a representation for each chord that is 7 values. Each value is the sum of the number of on bits for each note in the chord. For example, a C^7 chord in a 28 bit Circles of Thirds representation is

1000100	1000010	0001010	0010010
C	E	G	B-flat

The overlapping representation is:

1000100	(C)
1000010	(E)
0001010	(G)
+0010010	(B-flat)
2011130	(C^7 chord)

This is scaled to produce seven input values between zero and one: 0.6, 0, 0.3, 0.3, 0.3, 0.9, 0.

Duration Representation

We have also experimented with duration representations (Franklin 2004b). In our system, the entire note duration is the output of one LSTM module on one iteration. In our Modular Duration representation, beat length is divided by 96 giving 96 clicks per quarter note, 48 per eighth note, 32 per eighth note triplet note, etc. We can represent triplets and swing, and duration variations that occur in human MIDI performance (Thomson 2004), a step toward interpreting expressive MIDI performances. Our representation is a set of 16 binary values. Given a duration value, dur, the 16th bit is 1 if dur/384 >= 1, where 384 = 96*4, is the duration in clicks of a whole note. Then the 15th bit is 1 if (dur%384)/288 >= 1. In other words if the remainder after dividing by 384 and then dividing by 288 is greater than or equal to 1. The 14th bit is 1 if (dur%384%288)/192 >= 1. The modulo dividers are 384, 288, 192, 96, 64, 48, 32, 24, 16, 12, 8, 6, 4, 3, 2, and 1, corresponding to whole note, dotted half, half, quarter, dotted eighth, eighth, eighth triplet, sixteenth, sixteenth triplet, thirty-second, and then 6, 4, 3, 2, and 1 for completeness. Any duration that exactly matches, in clicks, one of these standard score-notated durations can be represented, as can combinations of them, or human–performed approximations to them.

Two example durations (in clicks) from Summertime are 86 and 202. The duration 86 is 64+16+6, represented as 0000100010010000, and the representation of 202 is 192+8+2, or 0010000000100010.

Experimental Results

We base the choice of parameters for the two LSTM modules on those values that worked best in the past on specific musical tasks and on the learning of the pitch and duration of the melody of Afro Blue. Consequently, both of the two LSTM modules contain 20 memory blocks, with four cells each. The set of four songs is presented for 15000 epochs. The two-module network learns to reproduce the four songs *exactly*, with a learning rate of 0.15 on the output units, and a slower rate of 0.05 on all other units. A larger rate on the output units produces consistently stable and accurate results in our previous experiments as well. Once the four songs were learned, we began experimentation with generating new melodies. In this network configuration, a straightforward way to do

this is to give the networks a whole new chordal structure. We keep the inter-recurrent connections and set the four song inputs all equal to one (all on).

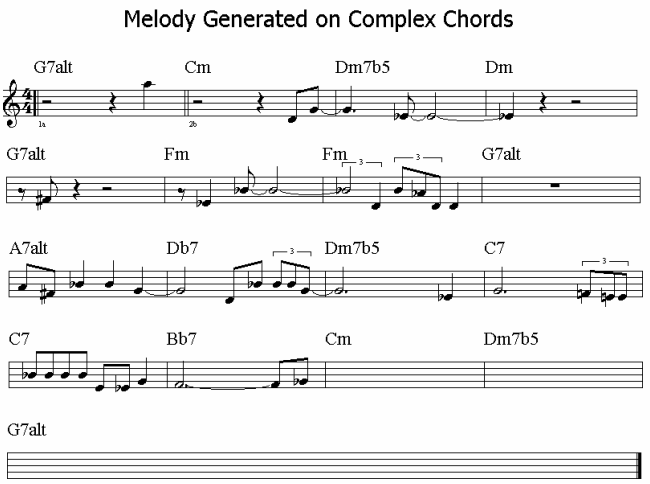

Figure 9. The melody generated by the dual-network system, over a complex chord structure.

We show melodies that are generated over three different harmonic structures, in Figures 9-11. The figures also show the harmonic structure as before. One bar of a pick-up or lead-in chord is given in each chord structure, since three out of four training songs had lead-in notes. Figure 9 shows a melody generated over a fairly complex harmonic structure that we derived from the structures of the four learned songs. There is a new chord in every bar except for the occurrence of F-minor (shown as Fm) two bars in a row in the second line. The melody depicted is a close approximation of the actual melody output by the networks. The approximation is made by the software used, Band-in-a-Box (PG Music 2004), to enter in the chords, to import the MIDI file, and to generate the scores as shown in the figures.

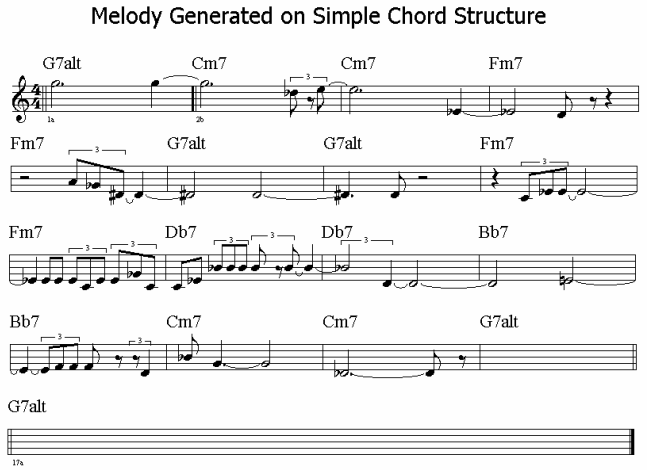

Figure 10. The melody generated by the dual-network system, over a simpler chord structure.

Figure 10 shows a much simpler chord structure also derived from the four original songs. All chords are carried over two bars (except the lead-in). The simpler chord structure results in a melody that is more rhythmic, and contains more notes. Note the use of grace note-like triplets that is an influence of the human musicians' style of playing. While there are a couple of notes out of place, such as the initial A on the G7alt chord in the lead-in bar, and the F# on the G7alt four bars later, the melody notes are derived from the scales one might associate with the chords when improvising, and the rhythm is quite reasonable.

Figure 11 shows the melody generated over a chord structure of an existing jazz composition, Song for My Father. This melody is by far the most pleasing to the ear, due in part to Horace Silver's (composer of Song for My Father) experienced use of the F-minor blues chords. But also, since these chord changes follow patterns and sequences that occur in the training songs, the network should be more likely to generate a better melody on them. We note two bars with a flurry of musical activity, the Eb7 in line 3 where the flurry is rhythmic, and on the Gminor chord in line 4. These are attractive because human musicians will often play such riffs in an improvised solo, but also because they occur within smoother, more melodic contexts.

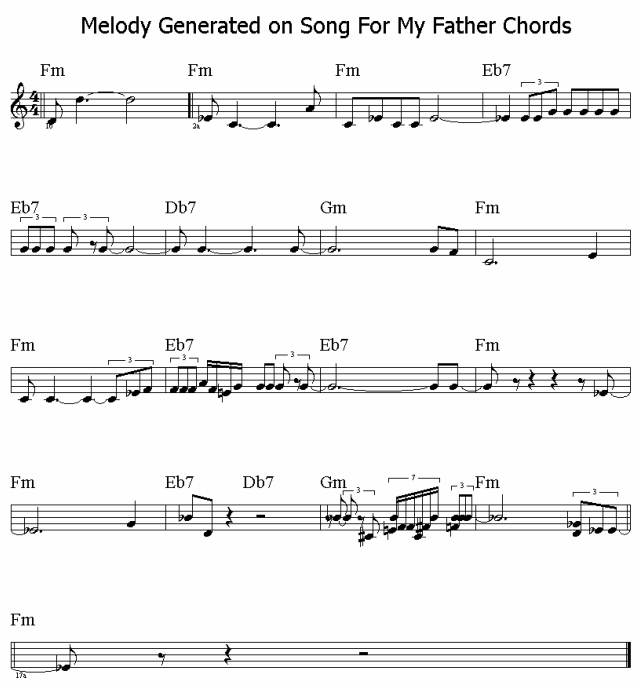

Figure 11. The melody generated by the dual-network system, on the AB part of the AAB structure of Song For My Father.

Discussion

The melodies generated by the trained network are interesting and for the most part pleasant. However, there are several "rough spots" that reveal the inexperience of the dual LSTM system, in which it "finds itself" in unknown musical territory. Two possible ways to decrease these rough spots are 1) to train the network on more songs, and 2) to employ a reinforcement learning (RL) mechanism to improve the melody generation. How can this be done? An RL agent could monitor the phrase structure produced by a network, such as noticing the two similar phrases in Figure 9 that both start with C and rise to G, each over a Fm to Eb7 (ii-I) chord transition, and reward that network output in some way. We have done some preliminary work in combining LSTM with a reinforcement prediction algorithm in which the LSTM equations are directly altered.

Another idea is to use a simpler, even non-recurrent RL agent that controls the dual LSTM networks. This agent could control several networks that are each trained on several possibly overlapping songs. The RL agent could choose which network's output to use for each note, or phrase. It could also learn to control the network by e.g. varying the threshold used in the duration network to choose which outputs are considered to contribute to the final duration value.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. IIS-0222541 and by Smith College. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation. The author acknowledges the excellence of the John Payne Music Center, Brookline, MA for human jazz learning.

References

Aebersold, J. 2000. Maiden Voyage (Vol 54). New Albany, IN : Jamey Aebersold.

Eck, D. and Schmidhuber, J. 2002. Learning the Long-Term Structure of the Blues. Proceedings of the 2002 International Conference on Artificial Neural Networks (ICANN). 284-289.

Engelmore, R., and Morgan, A. eds. 1986. Blackboard Systems. Reading, Mass.: Addison-Wesley.

Franklin, J. 2004a. Recurrent Neural Networks and Pitch Representations for Music Tasks, in Special Track: Artificial Intelligence and Music, in Proceedings of the Florida Artificial Intelligence Research Symposium, Miami, FL.

Franklin, J. 2004b. Computational Models for Learning Pitch and Duration Using LSTM Recurrent Neural Networks. in Proceedings of the 8th International Conference on Music Perception and Cognition (ICMPC8), Chicago, IL.

Franklin, J. and Locke, K. 2005a. Recurrent Neural Networks for Musical Pitch Memory and Classification. International Journal on Artificial Intelligence Tools (IJAIT). To appear, March 2005. World Scientific Publishing Co. Pte. Ltd. SINGAPORE

Franklin, J. 2005b. Franklin, J. Recurrent Neural Networks for Music Computation. in INFORMS Journal on Computing, Special Cluster on Music and Computation. To appear 2005.

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

Griffith, N. and Todd, P. 1999. Musical Networks: Parallel Distributed Perception and Performance. MIT Press,Cambridge MA.

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

Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. A Field Guide to Dynamical Recurrent Networks. IEEE Press, New York, NY.

Laden, B., & Keefe, D.H., 1991. The Representation of Pitch in a Neural Net Model of Chord Classification. Music and Connectionism, Todd,P.M., Loy,E.D., eds.,Cambridge, MA. MIT Press.

Mozer, M. C., 1994. Neural Network Music Composition by Prediction: Exploring the Benefits of Psychophysical Constraints and Multiscale Processing. Connection Science, 6, 247-280
Todd, P. M., & Loy, E. D., 1991. Music and Connectionism, Cambridge, MA: MIT Press.

Selfridge-Field, E. 1998. Conceptual and Representational Issues in Melodic Comparison. In Melodic Similarity. Concepts, Procedures, and Applications. Computing in Musicology 11. Hewlett, W. and Selfridge-Field, E., eds. Cambridge MA, MIT Press.

Todd, P. M., 1991. A Connectionist Approach to Algorithmic Composition, Music and Connectionism, eds.: Todd, P.M. and Loy, E. D., Cambridge, MA, MIT Press.