Recurrent Neural Networks and Pitch Representations for Music Tasks

Judy A. Franklin

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

Abstract

We present results from experiments in using several pitch representations for jazz-oriented musical tasks performed by a recurrent neural network. We have run experiments with several kinds of recurrent networks for this purpose, and have found that Long Short-term Memory networks provide the best results. We show that a new pitch representation called Circles of Thirds works as well as two other published representations for these tasks, yet it is more succinct and enables faster learning.

Recurrent Neural Networks and Music

Many researchers are familiar with feedforward neural networks consisting of 2 or more layers of processing units, each with weighted connections to the next layer. Each unit passes the sum of its weighted inputs through a nonlinear sigmoid function. Each layer's outputs are fed forward through the network to the next layer, until the output layer is reached. Weights are initialized to small initial random values. Via the back-propagation algorithm (Rumelhart et al. 1986), outputs are compared to targets, and the errors are propagated back through the connection weights. Weights are updated by gradient descent. Through an iterative training procedure, examples (inputs) and targets are presented repeatedly; the network learns a nonlinear function of the inputs. It can then generalize and produce outputs for new examples. These networks have been explored by the computer music community for classifying chords (Laden and Keefe 1991) and other musical tasks (Todd and Loy 1991, Griffith and Todd 1999).

A recurrent network uses feedback from one or more of its units as input in choosing the next output. This means that values generated by units at time step t-1, say y(t-1), are part of the inputs x(t) used in selecting the next set of outputs y(t). A network may be fully recurrent; that is all units are connected back to each other and to themselves. Or part of the network may be fed back in recurrent links.

Todd (Todd 1991) uses a Jordan recurrent network (Jordan 1986) to reproduce classical songs and then to produce new songs. The outputs are recurrently fed back as inputs as shown in Figure 1. In addition, self-recurrence on the inputs provides a decaying history of these inputs. The weight update algorithm is back-propagation, using teacher forcing (Williams and Zipser 1988). With teacher forcing, the *target* outputs are presented to the recurrent inputs from the output units (instead of the actual outputs, which are not correct yet during training). Pitches (on output or input) are represented in a localized binary representation, with one bit for each of the 12 chromatic notes. More bits can be added for more octaves. C is represented as 100000000000. C# is 010000000000, D is 001000000000. Time is divided into 16th note increments. Note durations are determined by how many increments a pitch's output unit is on (one). E.g. an eighth note lasts for two time increments. Rests occur when all outputs are off (zero).

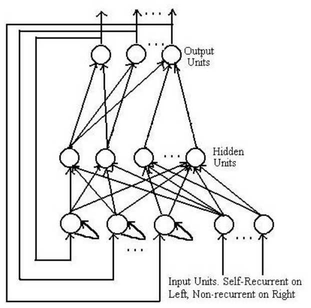

Figure 1. Jordan network, with outputs fed back to inputs.

(Mozer 1994)'s CONCERT uses a backpropagation-through-time (BPTT) recurrent network to learn various musical tasks and to learn melodies with harmonic accompaniment. Then, CONCERT can run in generation mode to compose new music. The BPTT algorithm (Williams and Zipser 1992, Werbos 1988, Campolucci 1998) can be used with a fully recurrent network where the outputs of all units are connected to the inputs of all units, including themselves. The network can include external inputs and optionally, may include a regular feedforward output network (see Figure 2). The BPTT weight updates are proportional to the gradient of the sum of errors over every time step in the interval between start time t_0 and end time t_1, assuming the error at time step t is affected by the outputs at all previous time steps, starting with t_0. BPTT requires saving all inputs, states, and errors for all time steps, and updating the weights in a batch operation at the end, time t_1. One sequence (each example) requires one batch weight update.

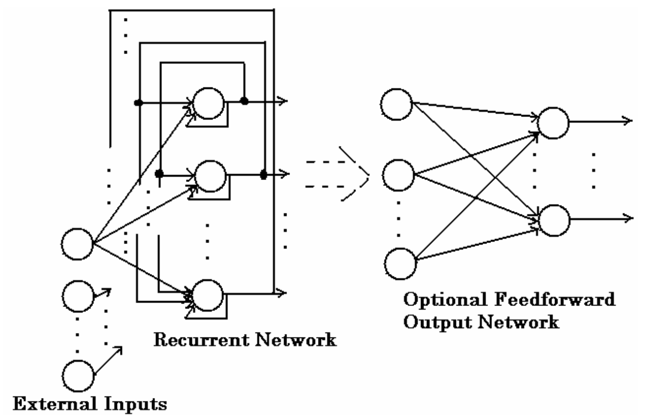

Figure 2. A fully self-recurrent network with external inputs, and optional feedforward output attachment. If there is no output attachment, one or more recurrent units are designated as output units.

CONCERT is a combination of BPTT with a layer of output units that are probabilistically interpreted, and a maximum likelihood training criterion (rather than a squared error criterion). There are two sets of outputs (and two sets of inputs), one set for pitch and the other for duration. One pass through the network corresponds to a note, rather than a slice of time. We present only the pitch representation here since that is our focus. Mozer uses a psychologically based representation of musical notes.

Figure 3 shows the chromatic circle (CC) and the circle of fifths (CF), used with a linear octave value for CONCERT's pitch representation. Ignoring octaves, we refer to the rest of the representation as CCCF. Six digits represent the position of a pitch on CC and six more its position on CF. C is represented as 000000 000000, C# as 000001 111110, D as 000011 111111, and so on. Mozer uses -1,1 rather than 0,1 because of implementation details.

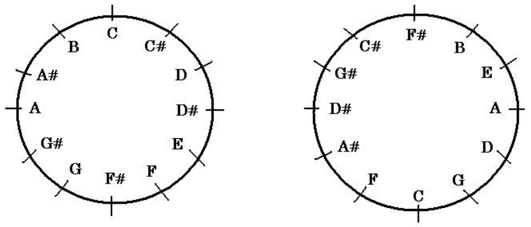

Figure 3. Chromatic Circle on Left, Circle of Fifths on Right. Pitch position on each circle determines its representation.

For chords, CONCERT uses the overlapping subharmonics representation of (Laden and Keefe, 1991). Each chord tone starts in Todd's binary representation, but 5 harmonics (integer multiples of its frequency) are added. C3 is now C3, C4, G4, C5, E5 requiring a 3 octave representation. Because the 7th of the chord does not overlap with the triad harmonics, Laden and Keefe use triads only. C major triad C3, E3, G3, with harmonics, is C3, C4, G4, C5, E5, E3, E4, B4, E5, G#5, G3, G4, D4, G5, B5. The triad pitches and harmonics give an overlapping representation. Each overlapping pitch adds 1 to its corresponding input. CONCERT excludes octaves, leaving 12 highly overlapping chord inputs, plus an input that is positive when certain key-dependent chords appear, and learns waltzes over a harmonic chord structure.

Eck and Schmidhuber (2002) use Long Short-term Memory (LSTM) recurrent networks to learn and compose blues music (Hochreiter and Schmidhuber 1997, and see Gers et al., 2000 for succinct pseudo-code for the algorithm). An LSTM network consists of input units, output units, and a set of memory blocks, each of which includes one or more memory cells. Blocks are connected to each other recurrently. Figure 4 shows an LSTM network on the left, and the contents of one memory block (this one with one cell) on the right. There may also be a direct connection from external inputs to the output units. This is the configuration found in Gers et al., and the one we use in our experiments. Eck and Schmidhuber also add recurrent connections from output units to memory blocks. Each block contains one or more memory cells that are self-recurrent. All other units in the block gate the inputs, outputs, and the memory cell itself. A memory cell can "cache" errors and release them for weight updates much later in time. The gates can learn to delay a block's outputs, to reset the memory cells, and to inhibit inputs from reaching the cell or to allow inputs in.

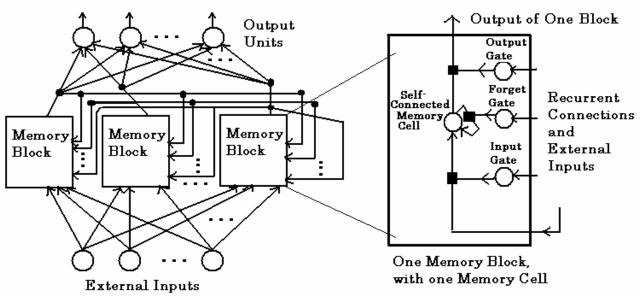

Figure 4. An LSTM network on the left and a one-cell memory block on the right, with input, forget, and output gates. Black squares on gate connections show that the gates can control whether information is passed to the cell, from the cell, or even within the cell.

Weight updates are based on gradient descent, with multiplicative gradient calculations for gates, and approximations from the truncated BPTT (Williams and Peng 1990) and Real-Time Recurrent Learning (RTRL) (Robinson and Fallside 1987) algorithms. LSTM networks are able to perform counting tasks in time-series.

Eck and Schmidhuber's model of blues music is a 12-bar chord sequence over which music is composed/improvised. They successfully trained an LSTM network to learn a sequence of blues chords, with varying durations. Splitting time into 8th note increments, each chord's duration is either 8 or 4 time steps (whole or half durations). Chords are sets of 3 or 4 tones (triads or triads plus sevenths), represented in a 12-bit localized binary representation with values of 1 for a chord pitch, and 0 for a non-chord pitch. Chords are inverted to fit in 1 octave. For example, C^7 is represented as 100010010010 (C,E,G,B-flat), and F^7 is 100101000100 (F,A,C,E-flat inverted to C,E-flat,F,A). The network has 4 memory blocks, each containing 2 cells. The outputs are considered probabilities of whether the corresponding note is on or off. The goal is to obtain an output of more that .5 for each note that should be on in a particular chord, with all other outputs below .5.

Eck and Schmidhuber's work includes learning melody and chords with two LSTM networks containing 4 blocks each. Connections are made from the chord network to the melody network, but not vice versa. The authors composed short 1-bar melodies over each of the 12 possible bars. The network is trained on concatenations of the short melodies over the 12-bar blues chord sequence. The melody network is trained until the chords network has learned according to the criterion. In music generation mode, the network can generate new melodies using this training.

In a system called CHIME (Franklin 2000, 2001), we first train a Jordan recurrent network (Figure 1) to produce 3 Sonny Rollins jazz/blues melodies. The current chord and index number of the song are non-recurrent inputs to the network. Chords are represented as sets of 4 note values of 1 in a 12-note input layer, with non-chord note inputs set to 0 just as in Eck and Schmidhuber's chord representation. Chords are also inverted to fit within one octave.

24 (2 octaves) of the outputs are notes, and the 25th is a rest. Of these 25, the unit with the largest value is the current note. The 26th output indicates if this is a new note, or the same note held another time step (16th note resolution). This network is similar to, and based on Todd's. After learning the Rollins melodies, the network is expanded and further trained by reinforcement learning (RL) according to a set of rules useable by an amateur improvisor. In RL, the network is not explicitly given the right outputs. It is given a reinforcement value that reflects the effects of its outputs. The reinforcement is positive if a note is a chord note, in a chord's scale, or if it is a note, not too frequently played, that is appropriate according to other jazz theory. To learn to improvise, the CHIME network must generate sequences of notes in the jazz idiom and be able to interpret sparse feedback, i.e. reinforcements, that may be delayed in time.

Seeking improved networks, we experimented with LSTM networks, comparing them with Jordan nets, BPTT and RTRL, and finding LSTM to be at least as accurate, and more stable. LSTM's algorithm is efficient, and does not rely on teacher forcing. It is incremental (BPTT is a batch operation), so is useable with incremental reinforcement learning. We revisited pitch and chord representations, pursuing a better musical representation. We present the results of three sets of experiments, using LSTM networks, and three different pitch and chord representations: Mozer's CCCF and chord representation, the localized binary representation of Todd, Eck and Schmidhuber, and Franklin, and a new one we call Circles of Thirds.

Circles of Thirds Representation

The Circles of Thirds representation is inspired by both the binary and CCCF representations, and Laden and Keefe's chord representation. It includes a pitch as well as a chord representation, and results in a 7-digit value for a pitch or a chord. Figure 5 shows the three circles of major thirds, a major third being 4 half steps, and the four circles of minor thirds, a minor third being 3 half steps.

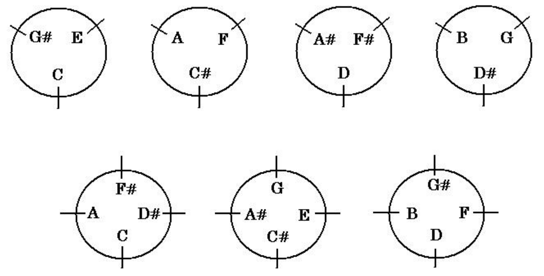

Figure 5. At top, circles of major thirds. At bottom, circles of minor thirds. A pitch is uniquely represented via these circles, ignoring octaves.

Here, a pitch 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 index of the pitch's circle is encoded, unlike CCCF, which encodes the position on the circle. 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. Because the 7th chord tone is so important to jazz, our chords are the triad plus 7th. Circles of thirds can represent chords as 4 separate pitches, each with 7 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 research on overlapping chord tones as well as Mozer's more concise representation. The result is a chord representation that is 7 values. Each value is the sum of the number of on bits from the Circles of Thirds representation 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)

Learning Tasks

Our initial experiments combining LSTM networks with the Circle of Thirds representation are quite promising. The three tasks are as follows.

Task 1- Chord Tones

Each of the twelve Dominant 7 chords, C7, C#7, etc. is presented, one at a time, as input for four increments. The network must first output the tonic, the third, the fifth, and then the (flat) seventh of the chord as output (e.g. chord C7 for 4 increments, and output C, E, G, B-flat). The network is trained, then tested afterward without weight updates. Thus in testing the network is given only the chord for 4 increments and its own recurrent connections.

Task 2 – Chromatic Lead-in

As (Berg 1990) points out, one effective technique of improvisation is to use chord tones in creating a melody, and to lead-in to a chord tone with chromatic pitches just below or just above the chord tone. In task 2, the network is given a set of 7 pairs of sequences that it must classify. Each sequence contains five pitches. In the first sequence, the third note is a chromatic tie between the second note and the fourth note. Both the fourth and fifth notes are chord tones. The network should output a 0 at each time step, except the last, when the target is 1 if there is a chromatic lead-in, and 0 otherwise. The positive sequences are from Berg, and all occur over the Cma^7 chord (with the 6th and 9th included as chord tones). There is no chord input however. The 7 pairs of sequences, each sequence labeled with its correct final target, are:

c,d,d-,c,c	1	c,d,d,c,c	0
c,d,e-,e,e	1	c,d,d,e,e	0
g,g-,f,e,e	1	g,f,f,e,e	0
e,g,a-,a,a	1	e,g,g,a,a	0
a,b,b-,a,a	1	a,b,b,a,a	0
a,a,b-,b,b	1	a,a,a,b,b	0
d,e,e-,d,d	1	d,e,e,d,d	0

During testing, the network receives each sequence note by note, and the outputs are recorded, with no weight updates.

Task 3 - AABA Melody

We created a melody that has the form AABA. The A form is an 8-pitch arpeggio over the Cma^9 chord: C-D-E-G-G-E-D-C. The B form is an 8-pitch improvisation over the same chord, containing auxiliary pitches (Berg 1990): C-F-D#-E-F#-A-G#-F#. This 32 note melody is presented as one example with 32 time steps to the network. The only external input is the representation for the C pitch, at each increment. When testing occurs, the network receives only the constant C pitch and does not see the target melody.

Results

Table 1 shows, for the 3 representations, results for the 3 tasks. Recall both the CCCF and Binary representations require 12 inputs, for either a pitch or chord, and 12 output units if pitches are the outputs. The Circles of Thirds representation requires 7. There is a bias term used in LSTM that enables the blocks (specifically the gates) to be “activated’’ one by one over time. We found a bias of -.1 to work the best for these tasks. The bias value of block 1 is 0, block 2 is -.1, block 3 is -.2, etc. The more negative the bias value, the longer it takes the weight updates to enable non-zero values on the output of the units in the block. An epoch is one presentation of all examples, and the max error is the maximum output error.

The results are the best we could obtain, varying learning rates (one for output units, one for blocks), whether there is a direct connection from input to output units, number of epochs, and number of blocks and cells per block. We aimed for an output error less than .1. If a .5 threshold is chosen as criterion, learning would take a fraction of the epochs shown in the table. The LSTM network perfectly learns the Chord Tones task, with all three representations, with the same parameters. LSTM perfectly learns the Chromatic Lead-in task with the Circle of Thirds and CCCF representations, but with binary representation it is not able to classify all of the examples. Thinking of them as pairs, it correctly classifies 4 or 5 of the 7 pairs, misclassifying the positive example as 0. Finally, on the AABA Melody task, all three representations work perfectly on this difficult task. This achievement made us very impressed with the LSTM network. Here, the LSTM network needs two cells per block. With 1 cell per block, no representation worked in learning the B part of the melody without error. Overall, it is clear that LSTM shows a lot of promise for musical tasks, and that the Circles of Thirds representation compares well to the CCCF and both are preferable to the binary representation. And the Circles of Thirds uses the fewest inputs.

Future Work

An obvious next step is to include octave information, and of course rhythm. We expect to tap Mozer’s useful work in this area. We are working with reinforcement learning for jazz improvisation and are experimenting with temporal difference (TD) learning to predict success or failure at a point in the middle of a musical sequence. For example, in the chromatic lead-in experiments, a TD algorithm can be used on the output units to predict the classification. Our preliminary results are promising. We are also encouraged by discovering work (Bakker 2002) in combining a form of reinforcement learning called advantage learning, that encompasses TD learning, with LSTM applied to 2 classic non-Markov tasks (ball balancing and T-maze following).

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Nos. CDA-9720508 and 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 John Payne Music Center, Brookline, MA for human jazz learning.

TASK	Represen-tation	Num. Blocks Num. Cells per blk	Learn-ing Rates	Num. Epochs & Max Output Error	Direct Input-Output Link?
=======	=======	====	===	====	====
1-Chord Tones	Circles of Thirds	1 cell each 10 blks	.5 blk .2 out	10000 .05	yes
	CCCF	1 cell each 10 blks	.5 blk .2 out	10000 .11	yes
	Binary	1 cell each 10 blks	.5 blk .2 out	10000 .04	yes
2-Chrom-atic Lead-in	Circles of Thirds	1 cell each 10 blks	.5 blk .2 out	10000 .03	yes
	CCCF	1 cell each 10 blks	.1 blk .05 out	10000 .07	no
	Binary	3 cells each 15 blks	.15 blk .05 out	15000 1 pair (of 7) in-correct	yes
3- AABA Melody	Circles of Thirds	2 cells each 15 blks	.15 blk .05 out	15000 .06	yes
	CCCF	2 cell each 15 blks	.15 blk .05 out	15000 .07	yes
	Binary	2 cell each 15 blks	.15 blk .05 out	15000 .08	yes

Table 1: Results for all experiments.

References

Bakker, B., 2002, Reinforcement Learning with Long Short-Term Memory, *Advances in Neural Information Processing Systems, 14,* eds. Dietterich, T.G., Becker, S. and Ghahramani, Z., Cambridge, MA: MIT Press.

Berg, S., 1990, Jazz Improvisation: the Goal-Note Method, Second Ed., Delevan, NY: Kendor Music, Inc.

Campolucci, P. 1998, A Circuit Theory Approach to Recurrent Neural Network Architectures and Learning Methods, Dottorato di Ricerca in Ingegneria Elettrotecnica, Università Degli Studi di Bologna.

Eck, D. and Schmidhuber, J., 2002, Learning the Long-Term Structure of the Blues. ICANN 2002, 284-289

Franklin, J. A., 2000, Multi-phase Learning for Jazz Improvisation and Interaction. *Proc. Eighth Biennial Connecticut College Symposium on Arts and Technology*, New London, CT.

Franklin, J. A., 2001, Learning and Improvisation, *Neural Information Processing Systems 14*, eds. Dietterich, T. G., Becker, S., & Ghahramani, Z. Cambridge, MA: MIT Press.

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

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

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

Jordan, M., 1986, Attractor Dynamics and Parallelism in a Connectionist Sequential Machine. *Proc. Eighth Annual Conf. of the Cognitive Science Society*, Amherst, MA.

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

Rumelhart, D., Hinton, G., and Williams, R., 1986, Learning Internal Representations by Error Propagation, *Parallel Distributed Processing 1*, eds. Rumelhart, D. et al, Cambridge, MA: MIT Press, pp. 318-362, 1986.

Todd, P. M., and Loy, E. D., 1991, *Music and Connectionism*, 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.

Werbos, P. J., 1988, Generalisation of Backpropagation with Application to a Recurrent Gas Market Model, *Neural Networks*, 1: 339-356.

Williams, R. J., and Peng, J., 1990, An Efficient Gradient-based Algorithm for On-line Training of Recurrent Network Trajectories, *Neural Computation*, **2**(4): 490-501.

Williams, R. J. and Zipser, D., 1988, A learning algorithm for continually running fully recurrent networks. Tech Report ICS-8805, Univ of Calif., San Diego, La Jolla.