International Journal on Artificial Intelligence Tools

RECURRENT NEURAL NETWORKS FOR MUSICAL PITCH MEMORY AND CLASSIFICATION

JUDY A. FRANKLIN

Smith College, Computer Science Department
Northampton, Massachusetts 01063, United States
jfranklin@cs.smith.edu

KRYSTAL K. LOCKE

Smith College, Engineering Program
Northampton, Massachusetts 01063, United States
klocke@smith.edu

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

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. We then discuss limited results using other types of networks on the same tasks.

Keywords: recurrent networks; LSTM; music processing

1. 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,[1] 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. These networks have been explored by the computer music community for classifying chords[2] and other musical tasks.[3,4]

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[5] uses a Jordan recurrent network[6] 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.[7] 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 (has value one). E.g. an eighth note lasts for two time increments. Rests occur when all outputs are off (zero).

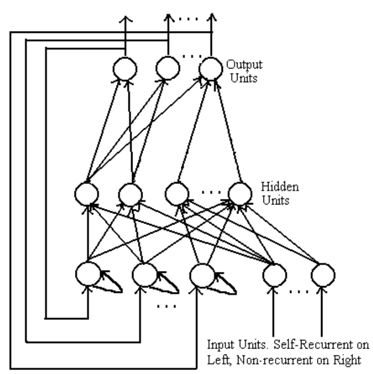

Fig. 1. Jordan network, with outputs fed back to inputs.

Mozer's CONCERT[8] 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[9,10,11] can be used with a fully recurrent network where the outputs of all units are connected to the inputs of all units, including themselves (see Figure 2). The network can include external inputs and optionally, may include a regular feedforward output network. 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. This "unfolding of time" is BPTT's alternative to teacher forcing. 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.

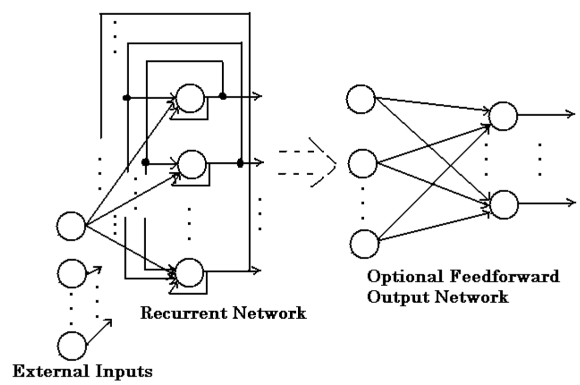

Fig. 2. A fully self-recurrent network with external inputs, and optional feedforward output attachment. If there is no output attachment, one or more recurrent units are designated as output units.

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. Figure 3 shows the chromatic circle (CC) and the circle of fifths (CF), used with a linear octave value or pitch height (PH) for CONCERT's pitch representation, called PHCCCF. 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.

For chords, CONCERT uses the overlapping subharmonics representation of Laden and Keefe.[2] 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. Using these representations, CONCERT learns waltzes over a harmonic chord structure.

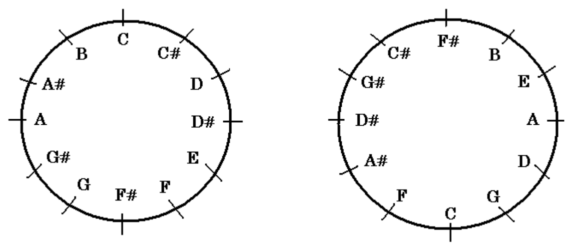

Fig. 3. Chromatic Circle on Left, Circle of Fifths on Right. Pitch position on each circle determines its representation.

Eck and Schmidhuber[12] use Long Short-term Memory (LSTM) recurrent networks to learn and compose blues music. An LSTM network[a] 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. 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.

Weight updates are based on gradient descent, with multiplicative gradient calculations for gates, and approximations derived from the truncated BPTT[15] and Real-Time Recurrent Learning (RTRL)[6] 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 train 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, C7 is represented as 100010010010 (C,E,G,B-flat), and F7 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.

[a]See Hochreiter and Schmidhuber 1997[13]. Also see Gers et al., 2000[14] for succinct pseudo-code.

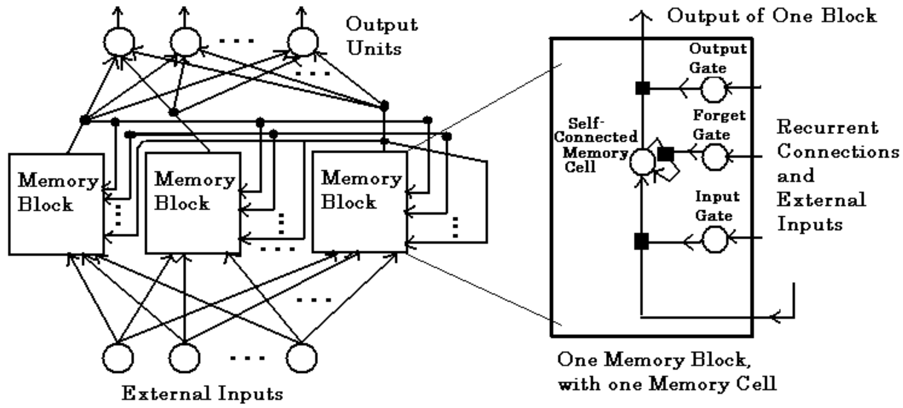

Fig. 4. An LSTM network on the left and a one-cell memory block on the right, with input, forget, and output gates. Black squares on gate connections show that the gates can control whether information is passed to the cell, from the cell, or even within the cell.

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,[16,17] 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 target 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 [6], 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.

2. 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 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.

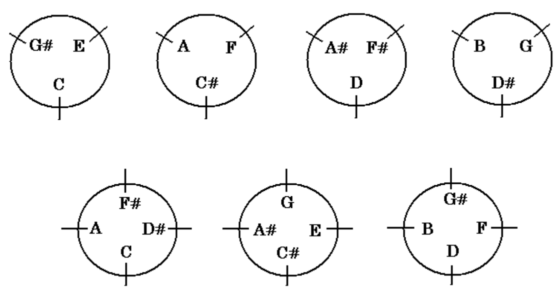

Fig. 5. At top, circles of major thirds. At bottom, circles of minor thirds. A pitch is uniquely represented via these circles, ignoring octaves.

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 could represent chords as 4 separate pitches, each encoded as 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, the dominant seventh chord C7 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   (C7 chord)
```

Notice that the dominant seventh chord is composed of the root, third, fifth, and flat seventh of the major scale. An interesting facet of this representation is that the third, fifth, and (flat) seventh of the dominant seventh chord all have the same minor circle. The root and third have the same major circle. This is true of all dominant seventh chords and is a straightforward ramification of the chord structure. Other chords will share similar commonalities among their notes in this representation.

3. Learning Tasks

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

3.1. *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, then the third, then the fifth, and then the (flat) seventh of the chord as output. For example, the chord C7 is presented as input for 4 increments of the network, and the four outputs, one on each increment, should be 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.

3.2. *Task 2 - Chromatic Lead-in*

As jazz educator Berg[18] 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[18], and all occur over the Cma7 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.

3.3. *Task 3 - AABA Melody*

We created a melody that has the form AABA. The A form is an 8-pitch arpeggio over the Cma9 chord: C,D,E,G,G,E,D,C. The B form is an 8-pitch improvisation over the same chord, containing auxiliary pitches[18] as follows:
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.

4. Results

Table 1 shows, for the 3 pitch 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

Table 1. Comparison of representations on three music tasks.

Task	Represen-tation	Cells/blk &Num blks	Learning Rates	Epochs	Max Error	Direct I/O Link?
1-Chord	CT	1 cell, 10 blks	.5 blk, .2 out	10000	.05	yes
Tones	CCCF	1 cell, 10 blks	.5 blk, .2 out	10000	.11	yes
	Binary	1 cell, 10 blks	.5 blk, .2 out	10000	.04	yes
2-Chromatic	CT	1 cell, 10 blks	.5 blk, .2 out	10000	.03	yes
Lead-in	CCCF	1 cell, 10 blks	.1 blk, .05 out	10000	.07	no
	Binary	3 cells, 15 blks	.15 blk, .05 out	15000	1	yes
3- AABA	CT	2 cells, 15 blks	.15 blk, .05 out	15000	.06	yes
Melody	CCCF	2 cells, 15 blks	.15 blk, .05 out	15000	.07	yes
	Binary	2 cells, 15 blks	.15 blk, .05 out	15000	.08	yes

Note: CT stands for Circles of Thirds representation

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. 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.

5. Results For Feedforward, Elman, and BPTT Networks

Once we had shown that the LSTM network, with the Circles of Thirds representation could learn to carry out the three tasks, we compared it to both a Feedforward network (FFN) as well as two other well-known recurrent networks, an Elman network, and a Back-Propagation Through Time network (BPTT). Recall Figure 2 shows a fully recurrent network, with an optional feedforward output attachment. The BPTT network we used has this configuration, but with a single layer of non-linear feedforward output units. It should be noted that the BPTT network alone without the feedforward layer could not learn any of the three tasks. The FFN is not recurrent, and is just the optional part alone, as shown in Figure 2.

An Elman network[19], as shown in Figure 6 has recurrent links between the hidden units and a set of input units. The network receives, as input, its last internal hidden unit values, along with the external inputs provided at each iteration of the network. It is trained by gradient descent, without any unfolding in time to correct past errors (so is unlike BPTT in that way).

The FFN and Elman simulations were carried out in the Matlab Neural Networks Toolbox.

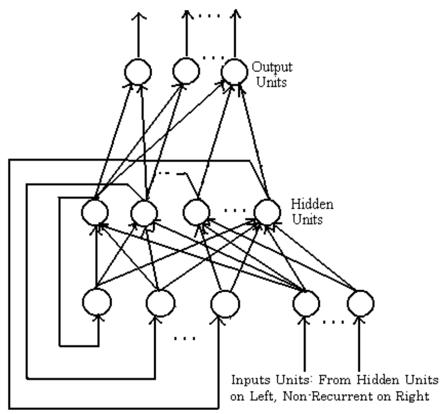

Fig. 6. Elman network, with hidden unit outputs fed back to inputs.

5.1. *Task 1. Chromatic Lead-in*

5.1.1. *FFNs*

When using feedforward networks on the chromatic lead-in task, all five pitches are presented simultaneously as inputs using the Circles of Thirds representation. The FFNs used in this experiment classified all fourteen note sequences correctly, both when single sequences were classified and when all fourteen were classified by the same network. The FFNs were efficient in this classification. In order to obtain clear separation between a good and bad result, the FFNs were trained to an extremely low error, 1e-10. Even with such a tiny error, the FFNs were able to achieve their training goal with a reasonable number of training epochs and number of hidden units within the networks. The learning rate was set at 0.05, which allowed the network to remain stable in its training while still training quickly. The average number of training cycles to reach the desired error was 756 epochs. The maximum number of epochs needed in the 25 trials was 1214 epochs, and the minimum number was 466 epochs. Fifteen neuron units in the FFN hidden layer allowed for training success for 25 out of 25 trials.

5.1.2. *BPTT*

In the recurrent network trials, each 5 pitch sequence is presented, one pitch at a time. The target output is 0 at each network iteration, except the fifth or last one, where it is 1 for a positive sequence, and 0 for a negative one. The BPTT network was run on the chromatic lead-in experiment with both the Circles of Thirds and the CCCF representations. Using Circles of Thirds, it could learn to classify the 7 pairs of examples exactly. For learning rates of .05 for the recurrent units, and .1 for the output units, it correctly classified after 15000 epochs, using 15 recurrent units, and one feedforward unit, corresponding to the one output for classification. With only 10000 epochs, and 10 recurrent units, analogous to the size of the successful LSTM, it could not learn the classification. The BPTT with the feedforward unit could not learn the task using the CCCF representation. A BPTT network with only recurrent units, one set up to produce the output, could not learn the task with either representation.

5.1.3. *Elman nets*

Effective Elman networks proved to be more difficult to create for this task. In order to facilitate efficient training, the acceptable error was raised from 1e-10 to 1e-3. Any error larger than this led to an inability to distinguish good from bad results. Stability and error reduction issues led to a low learning rate, 0.0001.

Successful training results were only occasionally obtained for up to three sequences, using 100 hidden units. Most trials ended with the error reduction gradient reaching its minimum well before the desired error was reached, around an error of 0.01. This error level was not sufficient to correctly classify the sequences. The inclusion of more training note sequences required the addition of more hidden layer units, which in turn seriously slowed the training process, and often led to software instability.

However, if only the first three notes of each sequence were used, Elman networks did prove to be effective for this task. In other words, the data set used for training this network was modified. Instead of sets of 5 notes per sequence, the first three notes of each sequence were used. We refer to these as sequence segments. The Elman network used for this experiment consisted of 11 hidden layer units using the tansig function, and 1 logsig output layer unit. In order to facilitate efficient training, the acceptable error was raised from 1e-10 to 1e-3. Any error larger than this led to an inability to distinguish good results from bad. Stability and error reduction issues led to a low learning rate, 0.001.

While it was certainly possible to include the full 14 sets of sequence segments of three notes each in the Elman training sets, it was not necessary. Using the first 4 pairs was sufficient to train the network to classify any sequence in the 14 as good or bad. New sequences which were consistent with the training set were very strongly classified, within 0.09 of their expected results.

5.2. *Task Two. Chord Tone Separation*

Using FFNs to separate the chord representation into individual notes was successful. The FFN is given the chord as input using the Circles of Thirds representation. It has 28 output units, corresponding to the four chord pitches, to be output simultaneously. An FFN with 30 hidden layer neurons and a learning rate of 0.5 is able to correctly extract the individual notes from the chord representation with an error of 1e-10. Over 25 trials, the average number of training epochs needed to reach the desired error was 38462 epochs.

The BPTT network could not learn this task, whether using Circles of Thirds or the CCCF representation. Either the outputs (numbering 7 or 12, respectively) were simply incorrect, or the network was unstable and oscillatory, although the error did generally decrease by half, from the initial value given the initial random weights. This behavior applied for various numbers of epochs, number of recurrent units, and learning rates.

The Elman network could not learn this task either, in its sequential form. It exhibited similar behavior to the chromatic descent experiments, being able to learn to produce perhaps 2 or 3 of the sequences. However, we tried using the Elman network to extract the notes from the chord all at once (all 12 chords as input, and all 12 sets of four notes output simultaneously) and the Elman network was able to separate each given chord, in the Circles of Thirds representation, into the component notes. The network used for this task had 84 input units, 16 hidden layer units using a logsig function, and 336 output units which also used the logsig function. The acceptable error was set to 1e-3, and the learning rate was set to 1. The high number of inputs and outputs made a higher learning rate preferable. The higher learning did not noticeably affect stability. The Elman networks performed the tasks more efficiently than the FFN networks, both in terms of the number of hidden layer units and the average number of training epochs required to perform the tasks.

5.3. *Task Three. AABA Melody*

The FNN cannot learn this task simply because it is so sequential a task that it does not make sense to use an FFN for it.

The BPTT network was incapable of learning this task using either representation, and exprimenting with a wide range of number of epochs, number of recurrent units, and various learning rates. the output squared error was always large after any number of epochs tried.

The Elman network was also used to attempt to recreate the simple AABA melody given just the starting note, C.

The first network configuration used had 12 units in its hidden layer and 8 output units. The learning rate was set to 0.05. The network was given a single input, C, and was trained to output the A segment of the melody, C,D,D,E,G,G,E,D,C. During training, the desired error level was never reached. Adjustment of network parameters did not significantly improve performance, and the network was not able to create the desired note sequence from the single note input. The partially trained network was able to create the first and last note of the sequence, but could not correctly create the other 6 notes. The 6 incorrect outputs were interesting in that they were either zeros, or they were notes that were misclassified as the input note, C. The notes that were misclassified as C were only notes that shared a major third circle, that is, the note E.

The whole AABA sequence was tested, using an Elman network that had 4 input units, 40 hidden layer tansig units, and 32 output tansig units. The input training set was C C D C. The network had a learning rate of 0.05 and an error level of 1e-3, as did the other Elman network used for this task. The results for the second partially trained network show that the first and second note in the A segments of the melody were correctly classified, while no other notes were.

6. Discussion

After testing and comparing the LSTM network to other recurrent networks on several simpler tasks, we discovered that the LSTM network is much more stable and accurate than other recurrent networks. We proceeded to test it on the three music tasks described in this article, using three different representations. Finally, we compared the LSTM network with a feedforward network and two other well-known recurrent networks, BPTT and Elman nets, on the same three tasks. There is really no comparison. The LSTM network outperforms the other nets and can very accurately learn to produce the desired behavior. We are convinced that the LSTM network is the best recurrent neural network for these kinds of tasks, and probably for many other domains. Our next steps are to more rigorously study its generalization capability. We are also encouraged by our results with the Circle of Thirds pitch representation and can imagine combining it with a form of CCCF to obtain the best features of both. We have experimented with a note duration representation that enables learning of MIDI-based performances.[20] An impetus for our study of these recurrent networks for specific musical tasks is our work in reinforcement learning. We are working with reinforcement learning for jazz improvisation and require a recurrent network that can learn specific notes with high accuracy. Furthermore, we are currently experimenting with temporal difference (TD) learning to predict success or failure at a point in the middle of a musical sequence. We are encouraged by discovering Bakker's work[21] 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). In the chromatic lead-in experiments, a TD algorithm can be used on the output units to predict the classification, based on Bakker's derivation. Our preliminary results are promising.[22]

Acknowledgments

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 authors and do not necessarily reflect the views of the National Science Foundation. The first author thanks the John Payne Music Center, Brookline, MA for human jazz learning.

References

1. D. Rumelhart, G. Hinton, and R. Williams, Learning Internal Representations by Error Propagation, Parallel Distributed Processing 1, eds. Rumelhart, D. et al., pp. 318-362, (MIT Press, Cambridge, MA, 1986).
2. B. Laden and D. H. Keefe, The Representation of Pitch in a Neural Net Model of Chord Classification. Music and Connectionism, eds. Todd, P. M. and Loy, E. D., (MIT Press, Cambridge, MA, 1991).
3. P. M. Todd, and E. D. Loy, Music and Connectionism, (MIT Press, Cambridge, MA, 1991).
4. N. G. Griffith, and P. M. Todd, Musical Networks: Parallel Distributed Perception and Performance, (MIT Press, Cambridge, MA, 1999).
5. P. M. Todd, A Connectionist Approach to Algorithmic Composition, Music and Connectionism, eds. Todd, P.M. and Loy, E. D., (MIT Press, Cambridge, MA, 1991).
6. M. Jordan, Attractor Dynamics and Parallelism in a Connectionist Sequential Machine. Proc. Eighth Annual Conf. of the Cognitive Science Society, (Amherst, MA, 1986).
7. R. J. Williams and D. Zipser, Gradient-based learning algorithms for recurrent networks and their computational complexity. In: Y. Chauvin and D. E. Rumelhart (Eds.) Back-propagation: Theory, Architectures and Applications, (Erlbaum, Hillsdale, NJ 1995).
8. M. C. Mozer, 1994, Neural Network Music Composition by Prediction: Exploring the Benefits of Psychophysical Constraints and Multiscale Processing. Connection Science, 6, 247-280, (Taylor and Francis Ltd, Abingdon, UK, 1994).
9. R. J. Williams, and D. Zipser, A learning algorithm for continually running fully recurrent networks. Tech Report ICS-8805, (Univ of Calif., San Diego, La Jolla, 1988).
10. P. J. Werbos, Generalisation of Backpropagation with Application to a Recurrent Gas Market Model, Neural Networks, 1: 339-356., (Elsevier Ltd., UK, 1988).
11. P. Campolucci, A Circuit Theory Approach to Recurrent Neural Network Architectures and Learning Methods, Dottorato di Ricerca in Ingegneria Elettrotecnica, (Universita Degli Studi di Bologna, 1998).
12. D. Eck and J. Schmidhuber, Learning the Long-Term Structure of the Blues, 284-289, (ICANN 2002).
13. S. Hochreiter, and J. Schmidhuber, Long Short-Term Memory, Neural Computation, 9(8):1735-1780, (MIT Press, Cambridge, MA, 1997).
14. F. A. Gers, J. Schmidhuber, and F. Cummins, Learning to Forget: Continual Prediction with LSTM. Neural Computation 12(10): 2451-2471. (MIT Press, Cambridge, MA, 2000).
15. R. J. Williams, and J. Peng, An Efficient Gradient-based Algorithm for On-line Training of Recurrent Network Trajectories, Neural Computation, 2(4): 490-501, (MIT Press, Cambridge, MA, 1990).
16. J. A. Franklin, Multi-phase Learning for Jazz Improvisation and Interaction. Proc. Eighth Biennial Connecticut College Symposium on Arts and Technology, (New London, CT, 2000).
17. J. A. Franklin, Learning and Improvisation, Neural Information Processing Systems 14, eds. Dietterich, T. G., Becker, S., & Ghahramani, Z. (MIT Press, Cambridge, MA, 2001).
18. S. Berg, Jazz Improvisation: the Goal-Note Method, Second Ed., (Kendor Music, Inc., Delevan, NY, 1990).
19. J. L. Elman, Finding Structure in Time, Connection Science 14: 179-211, (Taylor and Francis Ltd, Abingdon, UK, 1990).

20. J. A. Franklin, Predicting Reinforcement of Pitch Sequences via LSTM and TD, Proc. 2004 International Computer Music Conference. (ICMA, San Francisco, CA, 2004).
21. B. Bakker, Reinforcement Learning with Long Short-Term Memory, Advances in Neural Information Processing Systems, 14, eds. Dietterich, T.G., Becker, S. and Ghahramani, Z., (MIT Press, Cambridge, MA, 2002).
22. J. A. Franklin, Computational Models for Learning Pitch and Duration Using LSTM Recurrent Neural Networks. in Proceedings of the Eighth International Conference on Music Perception and Cognition. (Society for Music Perception and Cognition, Evanston, IL, 2004).