Multi-Phase Learning for Jazz Improvisation and Interaction[1]

Judy A. Franklin
Computer Science Department
Smith College
Northampton, MA 01063
jfranklin@cs.smith.edu
http://www.cs.smith.edu/~jfrankli
(413)-585-3858

Abstract

This article presents a model for computational learning composed of two phases that enable a machine to interactively improvise jazz with a human. To explore and demonstrate this model, a working system has been built, called CHIME for Computer Human Interacting Musical Entity. In phase 1, a recurrent neural network is used to train the machine to reproduce 3 jazz melodies. Using this knowledge, CHIME can interactively play music with a human in real time by trading fours in jazz improvisation. The machine is further trained in phase 2 with a real-valued reinforcement learning algorithm Included in the paper are details of the mechanisms for learning and interaction and the results. The paper presentation includes real-time demonstrations of CHIME.

Introduction

The goal in developing CHIME was to use machine learning to enable a machine to learn to improvise jazz and to interact with a human player. Creating this kind of human/machine interaction has great aesthetic and philosophical appeal. CHIME starts as a beginner, but after training can "trade fours" with a human. That is each improvises (makes up original jazz melodies in real time) for four measures while the other listens. Then the other takes a turn and so on. These four measures are played over a chord structure. In trading fours, a player tries to incorporate some of the other player's music while adding new material.

The type of machine learning used is based in connectionist artificial networks and includes both supervised learning (recurrent back-propagation) and reinforcement learning. Connectionist or artificial neural network (ANN) learning has much to offer human-computer interaction as has been explored by (Griffith and Todd 1999, Todd and Loy 1991, and Hörnel and Menzel 1998). Complex mappings between inputs and outputs can be learned. Recurrent artificial neural networks are neural networks with a feedback loop from the output units to units at one of the preceding layers. In this way the network is given information about its past actions. The networks can be trained prior to use and are also capable of real-time learning (learning while doing). Reinforcement learning offers the ability to learn using indirect feedback about the effects of the network's outputs.

CHIME demonstrates a model for computational learning that consists of phases of learning. Currently there are two phases. In phase 1, the learning method is supervised learning (recurrent back-propagation) where the machine is trained to play three jazz tunes. In supervised learning, the correct output is always known and an error is formed that is the difference between the actual output of the network and the correct output. This error is used to train the network. Phase 1 has its basis in work by (Todd 1991). After phase 1 training, the machine is able to trade fours with the human, on-line and in real time.

In phase 2 the trained recurrent network is expanded and further trained via reinforcement learning. Reinforcement learning is a natural approach for learning to improvise because it uses a trial and error basis for learning that allows it to discover new solutions to problems. It does not need an exact error to learn. After this additional training the computer can also trade fours in real time.

Figure 1 shows the CHIME architecture for training, learning, and real-time interaction. The top box shows phase 1, where the initial recurrent network is trained on existing melodies. The inputs to the

[1] This work was funded in part by the U.S. National Science Foundation, under grant CDA-9720508 through the Human Computer Interaction Program. It was also supported (non-financially) by the John Payne Music Center in Brookline, Massachusetts, U.S.A. All opinions in the article are the author's own.

network are shown as boxes on the left. In phase 1, the network input is a decaying history of its own output (the recurrent feedback link), as well as a representation of the chord over which it is improvising, and a number representing which song it is learning. After phase 1 training, when it is interacting with a human, its input is a decaying history of the human's most recent improvisation and again the current chord. In phase 2, the network receives both the recurrent feedback link as well as the human's improvisation. During the phase 2 training period, it can use either a recorded human improvisation, or it can be trained while interacting with a human. In both phases the network is given the current chord over which it is improvising. A reinforcement value gives some indication of how "good" or "bad" the output is. At this stage of CHIME, a set of rudimentary rules for jazz is used to obtain the reinforcement value.

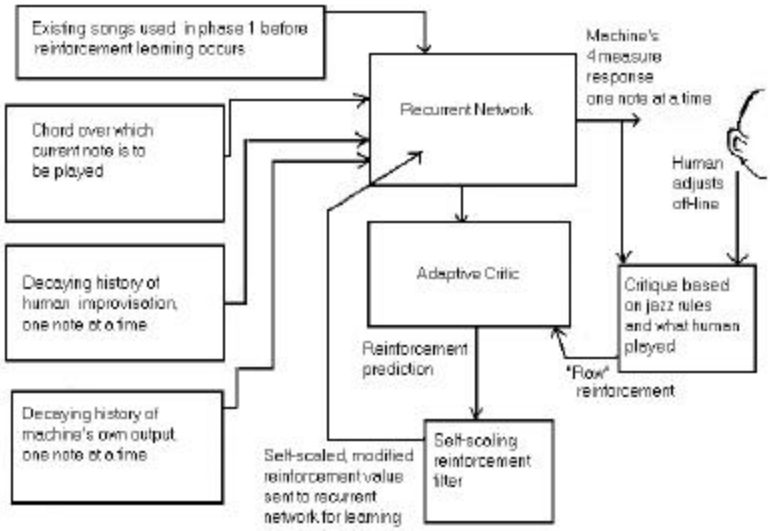

Figure 1. The CHIME architecture for learning, and interaction in real-time with a human.

On the right of Figure 1 is 1) an adaptive critic that learns to predict reinforcement values in order to account for delays between actions and results, and 2) a mechanism to adaptively scale the reinforcement so that it can continue to learn after it has learned to perform reasonably well.

Use of Recurrent Artificial Neural Networks

(Todd 1991) used a recurrent network for classical melody learning and generation. The connection from output to input provides a memory of recently played notes. The network was taught, using back-propagation, to reproduce melodies. The network was able to distinguish between various melodies by noting the distinction in inputs called plan inputs. Interestingly, the network was able to generalize; when given a plan that it had not seen before, it generated a new melody. In phase 1, a recurrent network is first trained off-line using supervised learning. A diagram of a recurrent network, based on Jordan's work (Jordan 1986), is shown in Figure 2. It is similar to that used by Todd except that it has been augmented with the chord input. There are several other more subtle changes as well.

The network is composed of the input layer, the hidden layer, and the output layer. Each node in the output layer corresponds to a note that can be played, or a rest, or a new note indicator. When an output node value is above a fixed threshold it is a candidate note. A rest is included as a candidate. The output with the highest value above the threshold is chosen as the next note. The note indicator output indicates whether this is a new note (i.e. note indicator output is greater than the threshold) or if it is the same note being held for another time increment. The network has a range of two octaves (24 chromatic notes of Western tonal music). Each unit in the hidden and output layers utilizes the sum of its inputs, multiplied by weights. The output units are linear and their output is simply this sum.

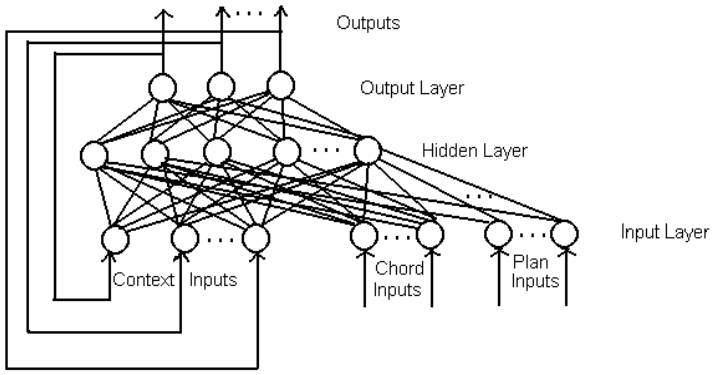

Figure 2. Recurrent Network for Phase 1, similar to network used by (Todd 1991).

In phase 1, the recurrent network is trained off-line to play three Sonny Rollins jazz melodies: "Blue Seven", "Tenor Madness", and "Solid" (Aebersold 1976). When training the machine to play melodies using *supervised learning*, errors are formed at each time step (16^{th} note) as the difference between the target (one output unit will have 1 as the target and the other units will have 0), and the units' output values. It is the difference between target note and the note played. The weight values are changed using back-propagation of the error through the network (Rumelhart, Hinton, and Williams 1986).

The hidden units are nonlinear, using the logistic function $f(x) = 1/(1 + e^{-x})$ where x is the weighted sum of the inputs. This function causes the output of each hidden unit to lie in the range between 0 and 1, with 0 and 1 being asymptotic values. Learning is accomplished by adjusting weight values. The inputs to the hidden units are from the context, chord, and plan inputs. The plan inputs indicate which one of the 3 songs is being played; of the three inputs, 1 has value 1 and the other two are 0. Chord inputs indicate the current chord. There are 12 chord inputs, corresponding to the chromatic scale, C up to B. A value of 1 means the corresponding note is on and a 0 means it is off. While jazz chords can have many different notes, the first, third, fifth, and seventh notes in the scale are good indicators of the chords that represent the three songs here. These notes correspond to the triad and the seventh[2]. When a chord is represented, it "wraps around" within the 12 inputs. For example, the F minor7 chord is F, Ab, C, Eb. It must be represented as C, Eb, F, Ab giving an input representation of 100101001000.

The context inputs, 26 in number, are shown as recurrent feedback from the outputs of the network. While the network is being trained of course the outputs are erroneous and this must be accounted for when using them as inputs to the network. In practice, while training there are two options to deal with this. One is back-propagation through time and another is called teacher forcing (Williams and Zipser, 1988). With teacher forcing the *target* output values for the *previous* step are used as the inputs, as if the network had output the correct target values. CHIME uses this method (as did Todd).

Results of Phase 1

Experimentation with 3 parameter values is required to find a solution. First is the number of hidden units. Todd used from 8 to15 hidden units and notes that with only 8 units the net requires 50,000 presentations of the two songs during training vs. 5,500 times with 15 hidden units. CHIME phase 1 uses 50 hidden units and 15,000 presentations are enough to learn each jazz melody individually, and even all three songs together. (Recall the advances in computing capability between 1991 and the present). The second parameter to vary is the learning rate α that determines what fraction of the error gradient is used to update the weights. While the learning rate can be different between layers, or even among individual units, a value of 0.075 is used successfully for all units. Todd's typical learning rate is 0.05. The third parameter is called the eligibility rate. It is the rate of decay of a decaying average of past inputs that is

[2] The triad and seventh work for now, but more chord tones will be needed for a more sophisticated improviser.

added to the present input at the context units. This decaying average, called an eligibility trace, gives the hidden layers a history of notes played is critical to learning sequences of notes. The eligibility rate that works the best is 0.9 (Todd uses around 0.8). Because of the larger number of rests in the Rollins tunes, the output units learn to turn off all the time unless a dedicated rest output unit is used. The Todd alternative is to assume a rest when the note units are all "turned off."

Below are results of four different experiments. In all experiments, 15,000 presentations of the songs were made. Each song has 192 16th note events. All songs are played at a fixed tempo. Weights are initialized to small random values. These results are representative of the runs. The value given for the squared error is the average squared error over one complete presentation of the song.
Experiment 1. Song: Blue Seven. Squared error starts at 185 and decreases to 2.67.
Experiment 2. Song: Tenor Madness. Squared error starts at 218 and decreases to 1.05.
Experiment 3. Song: Solid. Squared error starts at 184 and decreases to 3.77.
Experiment 4. 3 songs.Order: Solid, Blue 7, Tenor Madness. Squared error starts at 185, decreases to 36. More "finessing" of the network may improve these values. The songs are easily recognized and in fact an exact match could impair the network's ability to further learn. Figure 3 shows the results for "Solid."

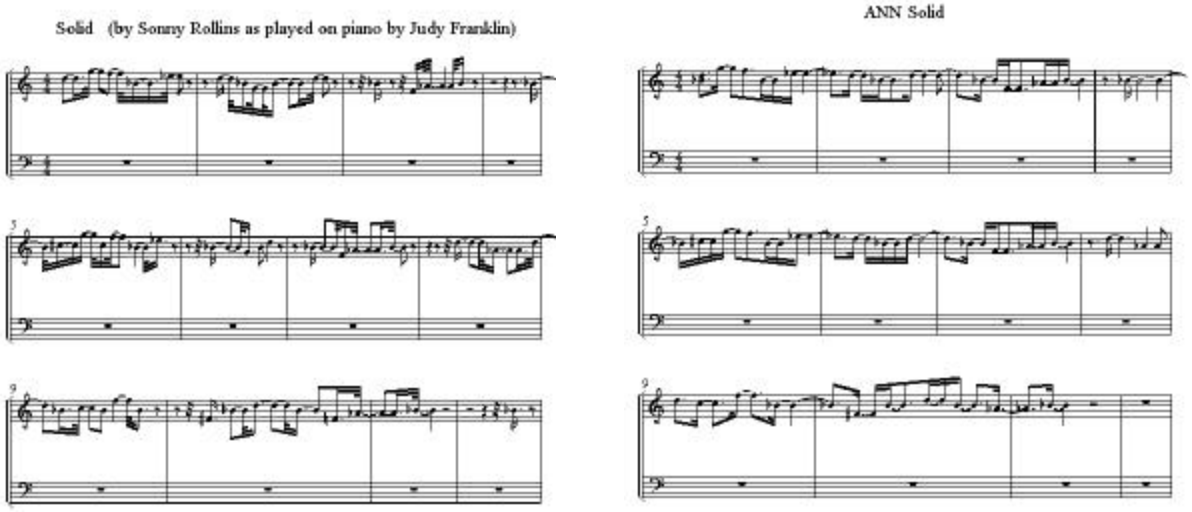

Figure 3. At left, the original song played by a human and at right, the song as learned by the machine.

Phase 1 Human Computer Interaction in Real Time

A human can play with the trained network via a digital piano and note events can be brought in using the MIDI interface. A human can play an acoustic instrument and have pitches (analog waves) converted to MIDI note events. Pitch to MIDI conversion is sought after by acoustic musicians and is largely unsolved, except for guitars. Software called Sound2Midi (AudioWorks 1998,1999) works reasonably well. It obtains wave input from a microphone attached to a sound card and converts it to MIDI, acting as a MIDI device (Windows95/98 OS). Sound2Midi has been used with voice and flute. A rhythm section (drums, bass, and piano) was produced by software called Band-in-a-Box (PG Music). Part of the technical work was to merge the rhythm section's MIDI events with those generated by human/machine. To use the trained network as a real-time interactive improviser, the recurrence is severed; i.e. the machine no longer has the recurrent feedback of its own outputs. Rather, notes a human plays for four measures are recorded and given as input, one at a time. All plan inputs are set to 1.0. The chord inputs follow the same pattern as the songs on which the net was first trained, although that is not an inherent constraint. The machine generates its four measures, one note at a time. The notes are played in real time as they are generated. Then the human plays again, etc. The trained network uses the human's melody (as input) plus its Sonny Rollins knowledge (stored as the weight values) to play its own melody.

Figure 4 shows 16 measures of a phase 1 interaction. The chords of the Rollins blues form are shown. These chords are used in all experiments and are examined later. Interpreting why a complex

artificial neural network produces specific outputs is difficult. However, we can examine its output and draw some conclusions. At each 16th note interval the machine is supplied with the human's input from the corresponding 16th note interval in the human's solo, passed of course through the eligibility trace. The machine's improvisations are shown in the 2nd and 4th lines of Figure 4. In measure 5 the flat 9 of the Eb7 chord appears; it is the E. E appears over both the Eb7 and the Bb7 chord in Blue Seven, albeit as a passing tone. The machine plays a D in measure 5 over the Eb7 chord. D is the natural 7 over Eb7 and should be used carefully over a dominant seventh. D is a note that Rollins uses heavily in all three songs, and even once over the Eb7. One conjecture is that it is responding to the rest and the Bb played by the human in measure 1. D follows a rest or the Bb in many places in both Tenor Madness and Solid.

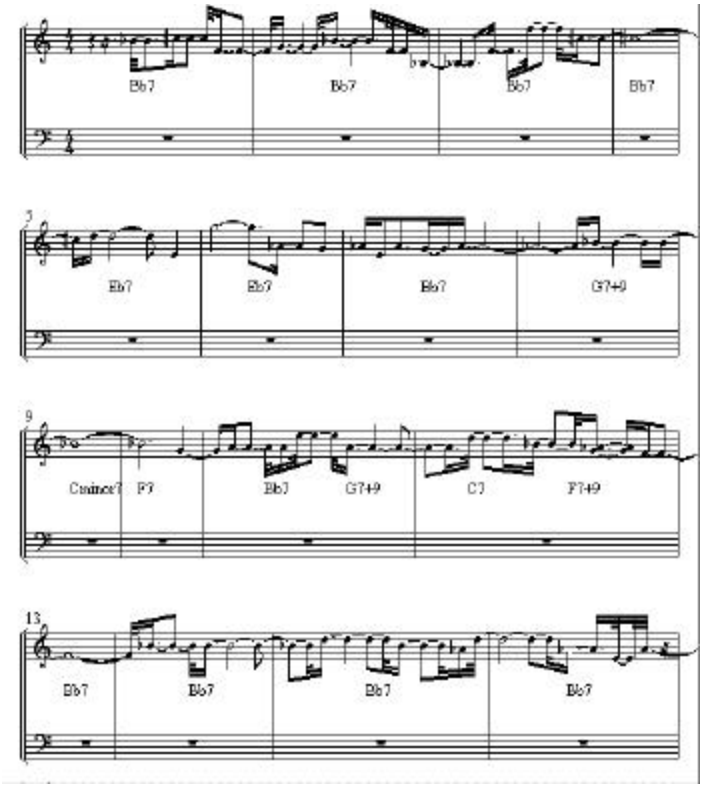

Figure 4. Phase 1: 4 measures of human improvising then 4 of machine, then 4 of human, 4 of machine.

In measure 6, the long G and the Ab (the third then the fourth of Eb7 chord) figure prominently in Solid. The focus of these next 2 measures is repeated jumping between the 3rd and 4th of the Eb7. At the beginning of measure 7 is the 2-note sequence Ab-E that appears in exactly the same place in the song Blue 7. At the end in measure 8 the machine plays the Ab then the Bb of the G7alt scale. In measures 13-16 the machine plays longer tones, as did the human before it in measures 9-12. In sequence, the tones are the 5th, the root, the 3rd, the root, the flat 7, the 3rd, and the 7th briefly, then the raised 4th. All are chord tones except for the last 2. The machine finds notes that work on the chords, uses notes from the Rollins melodies, is driven by the human's playing and "quotes" an interval in a Rollins melody.

Further Training with Reinforcement Learning - Phase 2

In phase 2 the learned network is used in a new configuration. In the actor-critic approach to reinforcement learning (Barto, Sutton, & Anderson 1983, Williams 1992 & Werbos 1995), the actor chooses state-dependent actions (such as next note, based on previous notes). The critic receives a reinforcement that reflects ramifications of the action taken. The critic learns to modify this value so the actor can use it to adjust its weights. Figure 5 shows a network for phase 2 with 100 hidden units, twice the number as the phase 1 network. The inputs are: the recurrent connection of 26 inputs, 3 plan inputs, and 12 chord inputs. 26 added inputs provide human input to the network. The total number of inputs is 68. CHIME's critic uses the output of the hidden units from the actor's recurrent network as inputs, an experimental configuration that eases the computation.

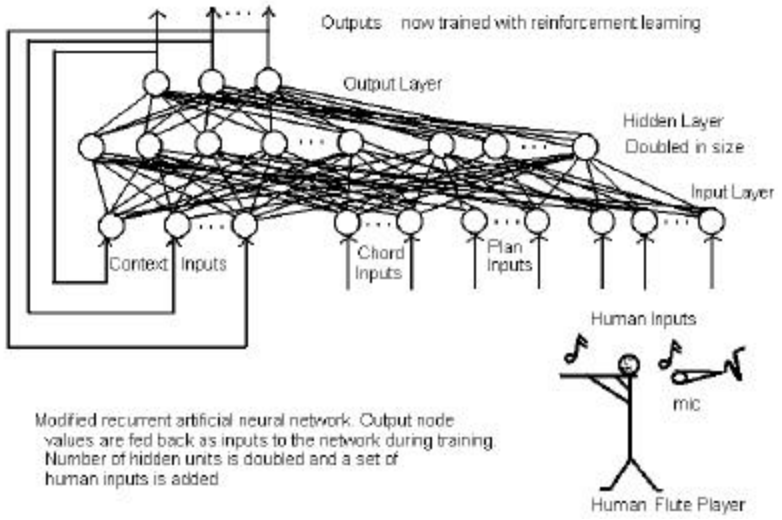

Figure 5. Experimental network for Phase 2 with reinforcement learning.

The weights obtained by training the phase 1 network are used to initialize this network. The weights connecting the plan and chord inputs retain the same values. The weights connecting the recurrent feedback from outputs into inputs are halved. The same weights, halved, also connect the 26 human inputs to the hidden layer. Now there are 100 inputs into each output unit, corresponding to the 100 hidden units. The original 50 connecting weights are halved and used as initial values of the 2 sets of 50 weights connecting hidden units to output units.

The reinforcement learning algorithm is the SSR algorithm (Benbrahim&Franklin 1997). When the network is trained in phase 1 via supervised learning, each output unit's output is a linear combination of its inputs. In contrast, in phase 2, reinforcement learning stochastically allows a machine to try new actions. Each output unit now uses the linear sum of inputs as the mean action μ. A Gaussian distribution centered at the mean stochastically chooses the actual action. The action is compared to a threshold as in phase 1 to determine if it should be the note. The Gaussian standard deviation, σ is changed, depending on the amount of exploration desired. This depends directly on how much reinforcement is received. In supervised learning the goal is to minimize a function of the error between output value and target value. In reinforcement learning the goal is to maximize a reward value. Each SSR unit must learn to be the highest output above the threshold only if its note receives the highest reward. If an action is "rewarded" the network is updated so that the action is more likely to be taken in the future. If it is "punished", it is updated so that action is less likely to be taken in the future. Actions are state and input dependent; an action rewarded for one set of inputs may be punished for another set.

In phase 2, human recordings were used as human input to the network. A critique is made of the machine's playing according to a set of rules. This critique is the "raw" reinforcement value (Figure 1) passed to an adaptive critic that modifies it in order to compensate for delays in reward. Next, a self-scaling mechanism scales the reinforcement value from the critic, looking for abnormally high or low reinforcement values that indicate a discovery. These algorithms are in the appendix.

Jazz Theoretical Basis and Rules

The Sonny Rollins songs are 12 measure blues forms. One method for improvising over a standard blues form is to play the blues scale over the whole form. However, improvising in one scale over a whole song can bore the listener (Sabatella 1996, Coker 1987). Sonny Rollins' songs Blue Seven, Tenor Madness, and Solid are in 4/4 time with a major or dominant tonality (Reeves 1995).

Blue Seven is a composition on the basic blues form. The chord progression for Blue Seven, listed as measures 1-4, 5-8, and 9-12 is at left and in Roman numeral notation for chord progressions on the right:

Bb7	Eb7	Bb7	Bb7		I	IV	I	I
Eb7	Eb7	Bb7	Bb7		IV	IV	I	I
F7	Eb7	Bb7	F7		V	IV	I	V

The I is the Bb dominant seventh chord. The IV chord is a dominant seventh chord, starting on the fourth degree of the Bb major scale. In Solid and Tenor Madness, Rollins uses a blues form that is closer in style to the form as it was developed in the bebop era (chords on left, Roman notation on right):

Bb7	Bb7	Bb7	Bb7		I	I	I	I
Eb7	Eb7	Bb7	G7+9		IV	IV	I	VI
Cm7	F7	Bb7 G7+9	C7 F7+9		ii	V	I VI	II V

The II-V-I and VI-ii-V-I progressions are added as variations to the original form. Rollins adds the G7alt (G7+9) chord substitution for the dominant 7th, for the VI and F7alt for the V. And he uses the dominant 7th to make a II-V turnaround to the I of the first measure for repetition. CHIME uses this form.

The original set of rules for critiquing in phase 2 is given below. This set had to be revised for reasons given following. A summary of the theory that provides the basis for these rules is presented just after.

Original Rules:

1) *Given a chord, playing notes in the corresponding scale is good (except as noted in rule 3).*

2) *Given a dominant seventh chord (typical for jazz) adding the 9, flat 9, sharp 9, #11, 13 or flat 13 is very good, unless one has already been played within the last 2 beats.*

3) *Given a dominant seventh chord, playing the natural 4 is bad.*

4) *Given a dominant seventh chord, playing the natural 7 is very bad.*

5) *Resting is bad if the total amount of resting time so far is over half of the playing time.*

Except for rule 5, these are rules obtained from John Payne, a professional jazz musician and 25-year jazz teaching veteran[3]. Actually the notes in rule 2 were cast as good to play in a "hip" situation. A machine improviser at this level cannot possibly understand hip. Hip could be construed that the notes should be played within a certain context of other notes or even in a certain atmosphere of playing. This idea is clearly necessary in the future for a more advanced player, but requires more planning than exists in the current system. Rather than reinforcing the machine for playing these notes when it is "hip," they are instead reinforced if they are played with definite but infrequent occurrence. Rule 5 was added to "punish" the machine when it did not play any notes.

Choosing one among several scales was not an option in this phase of training CHIME. Rather each chord was assigned one scale in order to interpret rule 1. And, CHIME was not trained over many sets of chord progressions, although there are not any inherent limitations to doing this. It is certainly possible for CHIME to play notes that are not in the assigned scale because of the influence of rule 2, and by the nature of reinforcement learning. These rules do not provide training to play outside the changes on songs that are not based on the dominant 7th, although rule 1 would train the system to play notes in the scales.

Rules 1 and 3 can be justified by recalling that there is a scale associated with each chord. For example, the scale for Gmaj7 is G-A-B-C-D-E-F#-G. The scale used over the dominant seventh chord when there are no chord substitutions is the mixolydian mode in which the seventh is lowered. The C7 mixolydian mode is C-D-E-F-G-A-Bb-C. The fourth of the major scale is often called an avoid note over a major seventh chord. (Playing it may be done consciously, with an awareness of the dissonance). Like the major scale, the fourth is somewhat of an avoid note for the mixolydian mode. However it may be played over suspended chords that use the fourth instead of the third. In this case, the third becomes the avoid note. At present CHIME is being trained on songs that do not use suspended chords. Playing dissonant notes is part of what makes jazz so interesting. However, if done at an amateur level, the song will simply sound bad. Thus at present the fourth remains an avoid note (rule 3).

Similarly, a scale called the dominant bebop can be played over the dominant seventh (Sabatella 1996). This adds the major seventh to the dominant seventh scale. The result for C7 is C-D-E-F-G-A-Bb-B-C. Sabatella indicates that the major seventh (the B here) may be played if it is used as a passing tone between the C and the Bb. Using the major seventh note over the dominant seventh chord may take two levels of planning. First is planning the three note sequence of Bb-B-C and also is its use as a raised

[3]These rules arose during an informal discussion and do not represent John Payne. As his student, I was not given rules to follow to improvise. My exercises overlapped significantly with (Sabatella 1996 & Reeves 1995).

fourth in the next chord (the I), if the dominant seventh chord is the V of a ii-V-I progression. Rule 4 of the original rules reflects the decision to avoid the major seventh on a dominant seventh chord, at the present amateur level.

"The most effective way to add interest to the ii-V progression is to alter the dominant (V) chord" (Sabatella 1996). The C7 in the Gm7-C7-Fmaj7 example may be replaced by a C7#11, a C7+ chord, or a C7b9b5 or C7alt chord. The substitutions of the C7#11 and the C7+ as well as the substitutions of the C7b9b5 or the C7alt form the theoretical basis for rule 2. The C whole tone scale is generally played over the C7+ chord and it makes available the flat 5 (or sharp 11), flat 6 (or flat 13) and of course the flat 7^{th} to be used for improvising over the dominant seventh chord. The scale used with the C7#11 is called the lydian dominant scale and it makes the raised fourth (sharp 11) available to the improviser as well. The additions of the sharp 11 and the flat 13 are part of rule 2.

The C7b9b5 substitution is a dominant 7^{th} chord with a flat 9 and flat 5. The "usual" corresponding scale is called the C HW diminished (C HalfWhole Diminished). It makes available the natural sixth (or the 13 an octave higher) and the raised ninth and is part of the John Coltrane post-bop sound. The altered scale, used over the altered seventh chord, provides the root, third and flat 7^{th} of the C7 as well as the flatted 9^{th}, raised 9^{th}, flat 5 and raised 5. This chord and scale are also written as a 7+9 (e.g. C7+9) chord and the scale may be called the diminished whole tone scale. According to (Sabatella 1996) this substitution is one of the most important sounds in post-bop jazz. Both the alt scale and the HW dim scale contain lowered fifths and lowered and raised ninths and are often used interchangeably, and again substantiate the "hip" notes of rule 2.

Phase 2 Results

One discovery in this experimentation is that the original rules did not work with the given architecture and algorithm. The learning algorithm either became unstable and the weights grew without bound, or it would settle on one note forever. Two modifications arose. First, the rules are changed and are shown below. Most of the change is focused on not letting the network stay in one output state too long. Second, limits are placed on the value of the standard deviation σ. Not only does this limiting constraint usually prevent the algorithm from producing huge growing weights, it also prevents the algorithm from settling on one note. In addition, it makes the action choice uncertain enough so different "hip" notes can be played at different times, for the same network state. Usually the goal of reinforcement learning is to find *the* best action for a given state, with uncertainty employed to allow further exploration for a possible even better action. Here, reinforcement learning is being employed to find the best *set* of actions for a given state and uncertainty is employed to choose among them to add variety to the output.

Reinforcement values: very bad (-1), bad (-0.5), a little bad (-0.25), ok (0.25), good (0.5), very good (1).

New Rule Set:

1) *Any note in the scale associated with the underlying chord is ok (except as noted in rule 3).*

2) *Given a dominant seventh chord, adding the 9, flat 9, sharp 9, #11, 13 or flat 13 is very good. But, if the same hip note is played 2 time increments in a row, it is a little bad. If any 2 of these hip notes is played more than 2 times in a row, it is a little bad.*

3) *If the chord is a dominant seventh chord (typical for jazz), a natural 4^{th} note is bad.*

4) *If the chord is a dominant seventh chord, a natural 7^{th} is very bad.*

5) *A rest is good unless it is held for more than 2 16^{th} notes and then it is very bad.*

6) *Any note played longer than 1 beat (4 16^{th} notes) is very bad.*

In a typical example using the phase 1 network prior to phase 2, the average reinforcement value according to the modified rule set is -.3716 (on a scale from –1 to 1). After using reinforcement learning, the average reinforcement value is .1875, after 8 presentations of the human solo as input. The human solo has 1800 note events (with the 16^{th} note increment). Each network's performance was evaluated, *after learning*, on the *original* rule set. Each had a better evaluation with the phase 1 network receiving .2432, and the phase 2 network achieving .45. A future goal is the automation of this rule modification.

Figure 6 shows 12 measures of a human solo (at left) and 12 measures of a machine solo (at right). The machine is given the human solo as input, and its solo is produced by this phase 2 network. The solo is significantly different from the 4 measure solos produced in phase 1. The machine plays chord tones in many places, such as the Bb and D in measures 1 and 2 (chord tones of the Bb7 chord). The high G is the 13 of Bb7, a hip note. In measures 3 and 4 it plays C# (or Db), a hip note (the #9 of the Bb7 chord) and again the high G. These notes are similarly played in measures 9 and 11 over the Bb7. In measures 5 and 6 the 9 and 13 (F and C) are played over the Eb7 chord. The natural 7 (D) is played as well and, as in phase 1, can probably be attributed to Rollins' heavy use of this note in the three melodies used for training in phase 1. Other hip notes show up later as in measure 9 over the Cminor7 chord, e.g. the 13 (G) and the 9 (D). The Cminor7 chord differs from the C dominant7 by only one note, a single bit to the machine. In the second half of measure 11 G is played 3 times over the G7+9 chord and the "hip" flat 9 (the Ab) is played as well. In the second half of measure 12, the Eb is played over the F7+9 chord and this is the dominant 7th note for that chord. As expected, the note durations are much shorter, reflecting the modification made in the second set of rules to prevent settling onto one note.

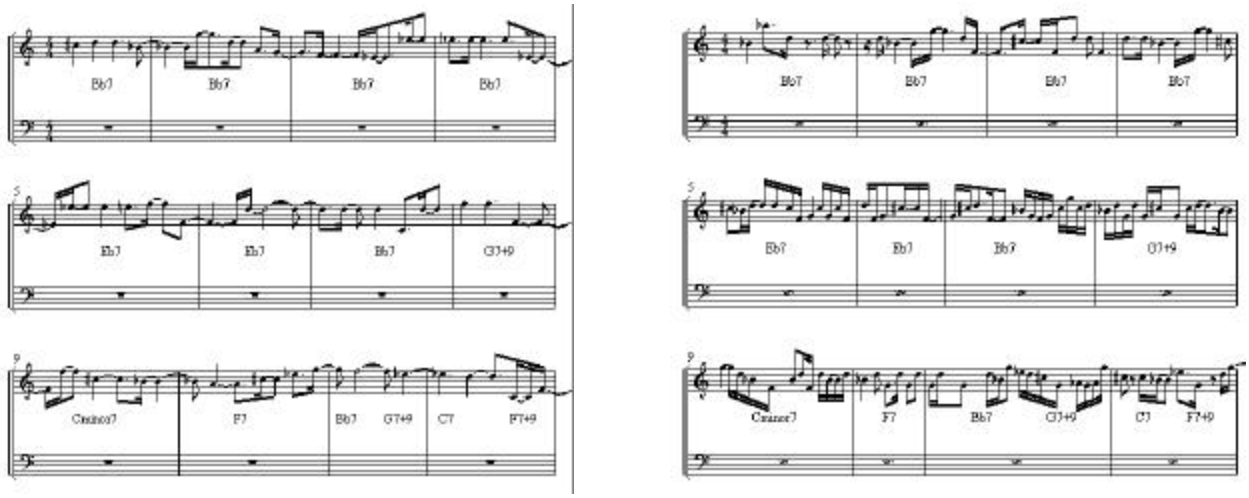

Figure 6. On left, 12 measures of human solo. On right, 12 measures the machine plays in response.

In several places the machine starts at the G at the top of the staff and descends through several chord tones. This occurs in measures 2, 4, 7, 9, and 10 of the solo at the right in the figure. After learning in phase 2, the recurrent network can produce sequences of notes and repeat the sequence or one similar to it. These sequences can be likened to recurring motifs. Many of Sonny Rollins' own solo improvisations (Blue Seven) have structural unity that most solos by other musicians lack (Bailey 1976) and this is achieved in part through his use of recurring motifs (Bailey 1976 and Schuller 1979).

Conclusion

The phase 2 network has been used to trade fours with a human player as in phase 1, except that the recurrence of the network is not necessarily severed; the network has a separate set of inputs for the human's music. Also, the network can continue to learn on-line during its interaction with the human. The (actor) network plus critic and reinforcement-scaling mechanisms are computationally taxing except for the fastest of today's PCs, but it is encouraging that today's machines can be used for this purpose. Recurrent networks have proven useful in the past for computer music generation. This work takes them a step further and uses them in real-time to interact with a human and it employs reinforcement learning to improve performance. So CHIME has a stochastic element that allows it to play "outside the chord changes." However, a big research problem is to enable it to do this more pointedly or purposefully. The current rudimentary rule set could be developed into a complex oracle for future use but learning from human feedback would be even better. Phase 2 demonstrates the ability of reinforcement learning to enable a recurrent network to learn from a set of rules that put local constraints on its behavior. These rules do not encompass developing a statement or creating a shape. The CHIME demonstration gives a basis to begin work on an improvisation's characteristics over a longer time period. It will be beneficial to tap the literature on models of human/machine interaction to explore the exchange of musical ideas.

References

1. Bailey, P. 1976. Introduction to Aebersold, J. 1976. You can play Sonny Rollins. *A New Approach to Jazz Improvisation* Vol 8. Jamey Aebersold, New Albany, IND.

2. AudioWorks. 1998. 1999. Sound2Midi. Collins, P. http://www.audioworks.com/index.html.

3. Benbrahim, H. and Franklin, J. 1997. Biped walking using reinforcement learning. *Robotics and Autonomous Systems* 22. 283-302. Elsevier Science, The Netherlands.

4. Coker, J. 1987. *Improvising Jazz*. Simon & Schuster. New York, NY.

5. Gullapalli, V., Franklin, J., and Benbrahim, H. 1994. Acquiring robot skills via reinforcement learning. February *IEEE Control Systems Magazine*. IEEE Press.

6. Hörnel,D.&Menzel,W.1998.Learning musical structure and style with neural networks. Computer Music J.Winter

7. Jordan, M. I. 1986. Attractor dynamics and parallelism in a connectionist sequential machine. *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*. Erlbaum Associates. Hillsdale N.J.

8. Messick, P. 1998. *Maximum MIDI*. Manning Publications. Greenwich CT.

9. PG Music. Band-in-a-Box http://www.pgmusic.com/bandbox.htm

10. Reeves, S. 1995. *Creative Jazz Improvisation*, Second Ed. Prentice Hall. Upper Saddle River NJ.

11. Rumelhart, D. E., Hinton, G.E., and Williams, R. J. 1986. Learning internal representations by error propagation. *Parallel Distributed Processing. Vol 1*. MIT Press. Cambridge MA.

12. Sabatella, M. A 1996. Whole Approach to Jazz Improvisation. A.D.G. Productions. Lawndale CA.

13. Schuller, G.1979.Sonny Rollins and Thematic Improvising. *Jazz Panorama* Williams, M. ed. Da Capo Press NY

14. Sutton, R. S. and Barto, A. G. 1998. *Reinforcement Learning*. MIT Press. Cambridge MA.

15. Todd, P. M., 1991. A connectionist approach to algorithmic composition. *Music and Connectionism*. P. M. Todd and D. G. Loy eds. MIT Press. Cambridge MA.

16. Werbos, P. J. 1998. Multiple models for approximate dynamic programming and true intelligent control: why and how. *Proceedings of the Yale Workshop on Intelligent Control*. Yale University. New Haven CT.

17. Williams, R. J. and Zipser, D. 1998. A Learning algorithm for continually running fully recurrent networks. ICS Report 8805. Cognitive Science. University of California San Diego. La Jolla CA.

18. Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*. Vol. 8. 229-256. Kluwer Academic Publishers. The Netherlands.

Appendix - SSR and Critic Algorithms

Each output unit has standard deviation σ_j and mean $\mu_j = \sum_{i=0}^{n} w_{ij} x_i$ and w_{ij} is the weight connecting hidden unit i to output unit j for x_i the output of hidden unit i. The Gaussian chooses output y_j using μ_j and σ_j. The critic uses reinforcement r to produce r_c. A self-scaling algorithm uses r_c to produce $\hat{r}$ by calculating the max and min reinforcement via a moving average: $\hat{r}$ =exp(r_c-rmax)-exp(rmin-r_c).
rmax(t+1) = max{rmax(t),r_c}, and then rmax(t+1) = λrmax(t+1)+(1-λ)r_c
rmin(t+1) = min{rmin(t),r_c},and then rmin(t+1) = λ rmin(t+1) + (1-λ)r_c
The weights and the standard deviation for each output unit area updated using $\hat{r}$ as follows (subscripts dropped):
w(t+1) = w(t) + α$\hat{r}$ (y-μ)∂μ/∂w = w(t) + α$\hat{r}$ (y-μ)x
σ(t+1) = max{ maxsigma, min{γσ(t) + (1-γ)*(rmax-rmin), minsigma}}
The difference between action and mean action is (y-μ). If the difference between max and min reinforcement stays large, σ will increase and allow more exploration. σ is never more than maxsigma. When rmax-rmin is small, σ shrinks, but is never less than minsigma. These bounds on σ are specific to the phase 2 implementation. The speed with which σ changes depends on the value of γ. The actor's hidden units are updated by back-propagating a value through the network, here the value $\hat{r}$*(y-μ). The SSR algorithm (Benbrahim & Franklin 1997) has roots in (Williams 1992 & Gullapalli 1994). The output of the critic unit is a prediction p(x, t) of future reward, for the current state x at time t. The system uses this prediction of future consequences to reward current actions.
r_c = r + γ_c*p(x(k),t) – p(x(k-1), t-1), for 0 < γ_c < 1. If the system moves from state x(k-1) to state x(k) that has a higher prediction of success, the value r_c increases over r. The critic is a linear function of its inputs: p(x(k),t) = w(t)•x(k) for w(t) the vector of weights at time t and x(k) the vector of input values at time k. These weights are updated incrementally: w(t+1) = w(t) + β*r_c*∂p/∂w and r_c is an "error", a difference between consecutive predicted rewards. The value ∂p/∂w is just input x, or rather its eligibility trace.