

Bach-Prop: Modeling Bach's Harmonization Style with a Back-Propagation Network

Rob Meyerson

Cognitive Science Program, Indiana University Bloomington

(Adapted with permission from Indiana University Cognitive Science Program Undergraduate Paper Repository)

1. Introduction and Motivation

Artificial neural networks have been used to model many aspects of human-music interaction. Pitch perception, key identification, melody discrimination, and original composition are all tasks for which researchers have tried to use networks. For this project, however, I was most interested in the idea of using a neural network for harmonization. Since (good) harmonization is considered a difficult task for humans, I thought this task would prove challenging, but not impossible, for my networks.

I decided to use Bach's chorales for several reasons. I intended to use Bach because his music is traditional, somewhat predictable, and often used to train human music students to harmonize melodies. The idea was encouraged by Professor Erik Isaacson, at the music department of Indiana University. Some of the ideas presented in this paper are his, including that I use Bach's chorales, specifically. Professor Isaacson added that Bach was prolific in his chorale composition, and that it would be easy to find chorales for input. Finally, I chose Bach because I like his music, and knew that I'd most likely be listening to (or at least looking at) a lot of it.

2. Input Representation: Notes to Numbers

The networks were only trained on soprano and bass lines, so inner voices were ignored. For each piece, each note of the soprano and bass lines was converted to a number between one and twelve. The scale started on the tonic note (scale degree one) of the chorale, so that "one" represented the tonic, independent of the key of the piece. Thus, all the chorales used were transcribed to the same key before being inputted into a network. The octave of each note was also ignored, so if a piece was in the key of G, the note "B" was represented as "five," whether it was above, below, or within the staff.

The fact that artificial neural networks base their output on probabilities affected how notes were represented to the networks. If the network determined during training that "one" in the soprano was most often harmonized with "one" or "eight" (the dominant) in the bass, the network shouldn't have had the option of averaging to find an output of "four" or "five."

For this reason, all twelve half steps of the chromatic scale (within a given octave) were represented by separate nodes in both the input layer and the output layer.

In the hope that the networks would do more than just find the most probable output note for any given input note, three (or sometimes only two) notes of the soprano line were inputted to the network simultaneously for each cycle. On each cycle, the network was provided with a past note, a current note, and a future note. Thus, the network was actually making an association between the correct bass note (during training) and a *pattern* of soprano notes. The goal was that the harmonization of the soprano note at a given state would depend in part on the past and future soprano notes.

In order to facilitate this idea, the input layer was actually composed of thirty-six nodes. The past note was always represented as a node between one and twelve, the current note between thirteen and twenty-four, and the future note between twenty-five and thirty-six. At the beginning and end of each chorale, of course, only two input nodes were activated, since there is no past note before the first note of the piece, and no future note after the last note of the piece. Figure 1 summarizes how notes were represented as input.

Table 1
Input representation

Letter name for note in the key of D	Degree of note on chromatic scale, starting with D as 1	Input line given to network including past and future notes
D	1	13,25
D	1	1,13,36
C#	12	1,24,34
B	10	12,22,32
A	8	10,20,25

It is important to note that rhythm information was almost entirely discarded when notes were inputted into the networks. There are many problems with representing rhythm in the input, and the information was discarded mostly for the sake of simplicity. The problem was handled by only treating a time step as relevant if a note during that time step changed, or was articulated, in either the soprano or bass line. In other words, if both the soprano and bass line held the same note for four beats, or an eighth of a beat, it was represented to the network as one input group. If, however, the bass line held a half note, while the soprano line changed from one quarter note to another (whether or not the actual pitch changed), this was represented as two input groups with the same bass note. The opposite is true also, in that if the bass note changed while the soprano line held a note, this was represented in multiple input groups. This method of extracting pitch changes while discarding rhythm is illustrated in Figure 2.

Table 2.
Extracting pitch changes and discarding rhythm

Letter name for soprano note in the key of D	Letter name for bass note in the key of D	Input line	Degree of bass note, given for training
F#	F#	3,17,30	5
G	F#	5,18,32	5
A	F#	6,20,30	5
G	B	8,18,30	10
G	C#	6,18,29	12
F#	D	6,17,27	1
E	C#	5,15,29	12
F#	C#	3,17,30	12

3. Network Architectures

Three network architectures were used. All the networks were created in Tlearn, a free program available for download on the internet. All the networks had the same input and output nodes, but differed in their middle layers. Each network had thirty-six input nodes. Each cycle (or “sweep,” as they’re referred to in Tlearn) had either two or three notes of input, and each was represented by activating the appropriate node in the input layer. Similarly, all the networks had twelve output nodes. In all three networks, every input node was connected to every output node. The weights on these connections were initially set to random numbers, but once the random values for the weights were determined, they were kept consistent every time the networks were reset. Figure 3 shows the input and output layers of all three networks. The arrows from the input layer to the output layer represent connections from every input node to every output node. Keep in mind that the network shown in Figure 3 is simply an illustration of the nodes and connections that all three of my networks had in common, and that the exact network shown in Figure 3 was not actually used.

3.1 Recurrent network

The first network I used was a recurrent network, in that its output from one cycle was used as input in the next cycle. This was done via twelve “copy-back” nodes, which simply took the exact activation values of the output nodes in one cycle and sent them as input for the next cycle. Recurrence seemed like a good idea for a harmonizing network because Bach’s bass-lines often move in what is known as *stepwise motion*. Stepwise motion is simply the musical term for music that moves along a scale one step at a time, without skipping directly from a low note to a high note, or vice versa. By allowing the network to be aware of its output on the previous cycle, the hope was that it would learn to associate one cycle’s output with a new output that was only one step away from the previous output (assuming this is what Bach most often did in this situation).

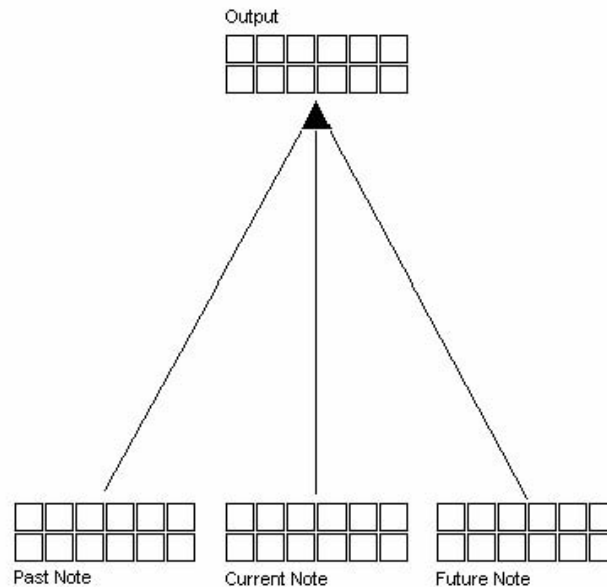


Figure 3. Nodes and connections held in common by all three networks used

Another goal of the recurrent network was to learn *resolution*. In Bach's pieces, and most traditional music, certain notes are extremely good predictors of the next note. For example, when chromatic scale degree twelve (the subtonic) moves to degree one (the tonic), it is said to "resolve." This resolution from scale degree twelve to one is very powerful, and listeners (those used to Western music at least) can predict it even with no knowledge of musical composition. At the very least, I hoped the recurrent network would learn this common resolution. A diagram of the recurrent network can be seen in Figure 4.

3.2 Pre-wired network

Another technique used in creating these networks was pre-wiring. In this network, I created a layer of twenty-one hidden nodes, seven for each set of twelve input nodes. Each node in the hidden layer was connected from three of the twelve input nodes directly below it. This is shown in Figure 5, but only three of the connections are shown so that the diagram is understandable. Each of the seven nodes in each of the three sections of the hidden layer represents a major scale degree. Unlike the chromatic scale, the major scale has only seven notes, and each of these notes has a chord associated with it. In a major piece, the chords associated with the first through seventh major scale degrees are major, minor, minor, major, major, minor, and diminished, respectively. So the first node in the hidden layer is connected from input nodes one, five, and eight, composing the major chord associated with scale degree one. The second hidden node is connected from input nodes three, six, and ten, and so on.

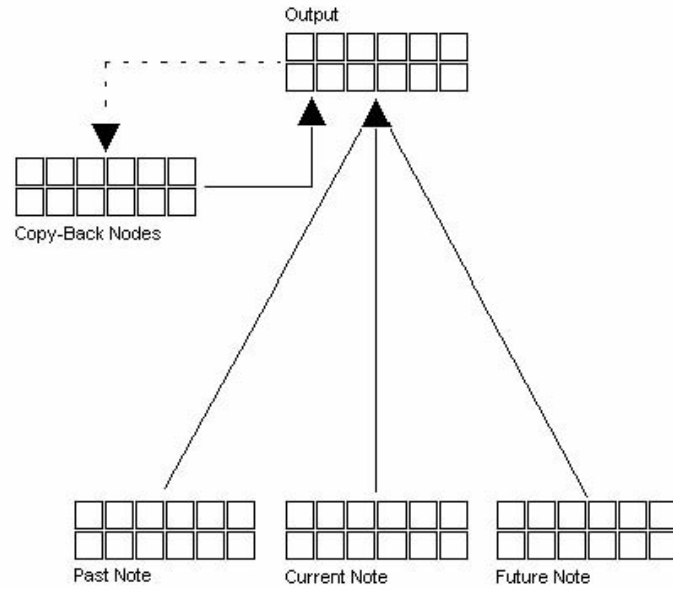


Figure 4. The recurrent network

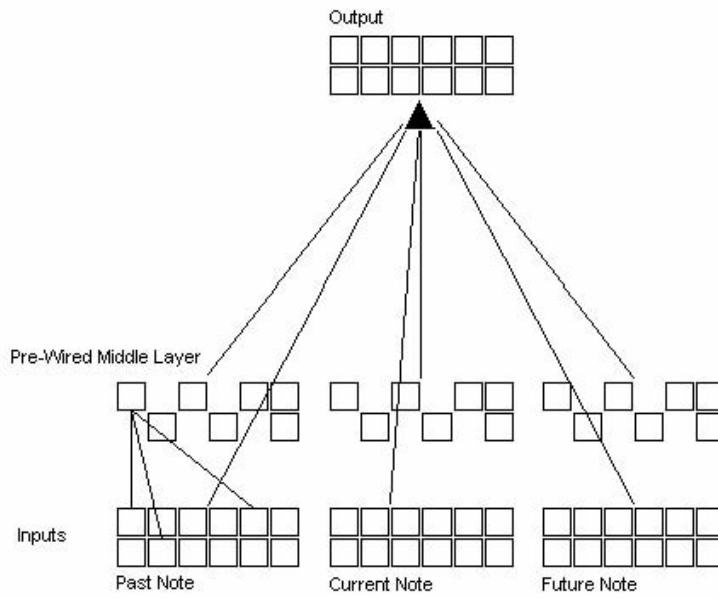


Figure 5. The pre-wired network

I chose to pre-wire a network for two reasons. First, I simply hoped that the network would be able to make associations between notes of the same chord within the key, since this could conceivably make the network a better harmonizer. Secondly, since I was, in part, attempting to model the human ability to harmonize Bach chorales, I thought it only fair that my network be explicitly taught some characteristics of music the way most humans are when they're learning to harmonize music.

3.3 Recurrent and pre-wired networks

This third network is simply a combination of the previous two. I created it in the hope that it would outperform both the recurrent and pre-wired networks, by using both techniques to find the best harmonization. This network is represented in Figure 6.

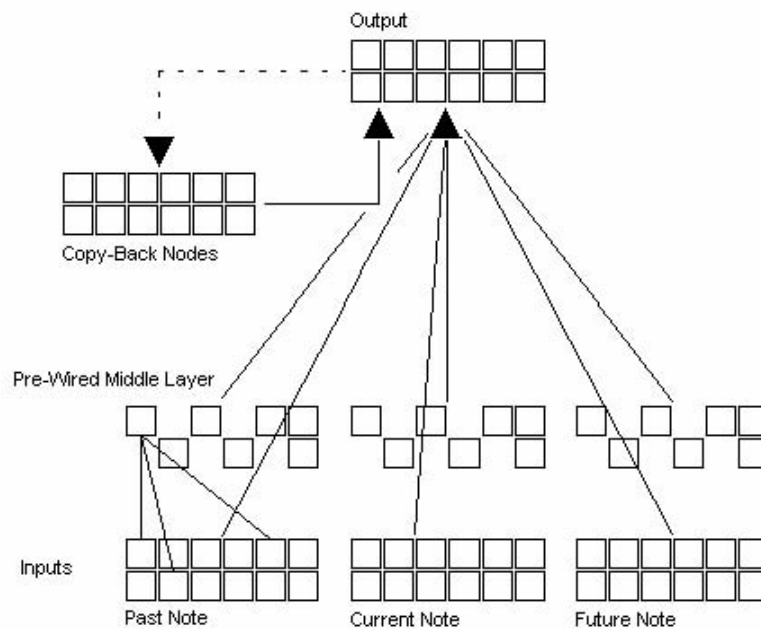


Figure 6. The recurrent and pre-wired network

4. Training and Output Representation: Numbers to Notes

The networks were trained on a total of seventeen chorales. Chorales were not used if they were in a minor key, or if I felt certain that there was a key change within the piece. The seventeen chorales provided the networks with a total of 839 lines of input, however I also stopped and tested the networks' performance after 471 lines of input, so that I could measure the networks' improvement as input was added. Before testing, the networks were all trained on five-thousand sweeps, and the learning rate was set to 0.1. I tested the networks on two pieces that I felt were indicative of Bach's harmonization style. Their titles are "Open wide for me the portal" and "My trust is bold." When tested, the networks provided me with the activations of the output nodes for each input line of testing. I chose to use the maximum activation of the output layer for each input line as the networks' "best

guess” as to the correct bass note. After determining the maximally activated output node for each input line, I regarded the number of that node as the chromatic scale degree for the output note. I used Finale (music software) to place these notes onto a staff, conserving Bach’s soprano line, and his rhythm and octave distinctions as much as possible. Finale also allowed me to make audio recordings of these pieces as MIDI files.

5. Tests of Success: Percent Accuracy and Average Step Size

I developed two methods for testing the success of my networks’ outputs. My first approach, percent accuracy, seemed like the most straight-forward method. I simply compared every output value to Bach’s original bass line. For each piece, I divided the number of outputs that matched Bach’s to the total number of outputs. Figure 7 shows the percent accuracy of each network after 839 lines of input. The percent accuracy is averaged over the networks’ output for both “Open wide for me the portal” and “My trust is bold.” As seen in Figure 7, the network that was both pre-wired and recurrent outperformed the other two networks, as predicted.

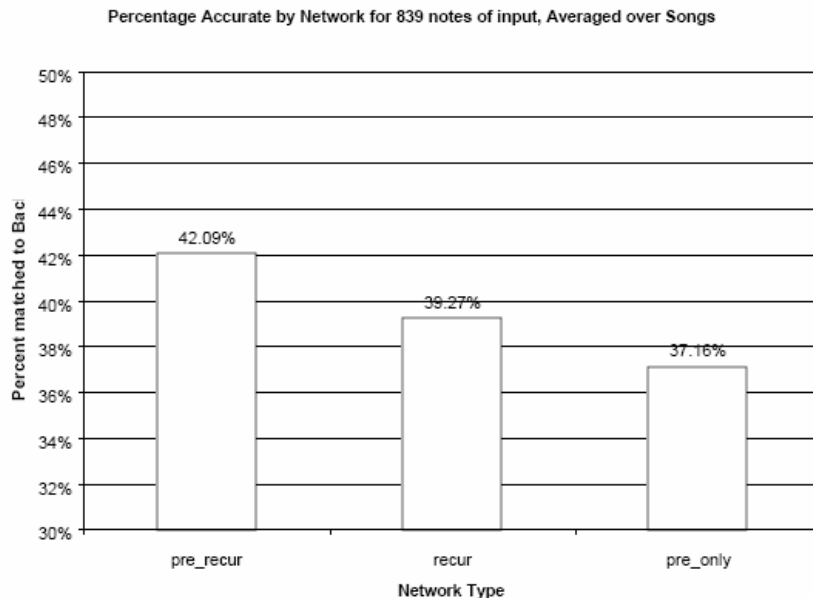


Figure 7. Percent accuracy for each network after 839 lines of input

It should be noted, however, that there may have been a significant problem with using percent accuracy as a measurement of the output’s “goodness.” The percent accuracy of a piece was determined independently of any given note’s duration or position within the piece. So although two outputs may have the same percent accuracy, as determined here, one of the outputs may actually sound much better because the notes that were correctly guessed are longer and in crucial places within the piece. A potential solution would be measuring percent accuracy for time steps, so that each half beat, for example, would be equally

weighted in the calculation of percent accuracy. Furthermore, certain time steps could be somehow weighted as more important than others based on their position in the song. For example, the first and last notes of a piece often play a large role in how “good” the piece sounds to a human listener (this is at least in part due to primacy and recency effects) and could therefore be weighted as more important than other notes when determining percent accuracy.

The second method of measuring an output’s “goodness” was the average step size between notes in the output. As mentioned previously, traditional compositions such as Bach’s often include a lot of stepwise motion. As a result, the average step size between notes is very small. In fact, the average step size of the bass line in both Bach’s “Open wide for me the portal” and his “My trust is bold” is only about 2.4 (half steps). Figure 8 shows the method I used to determine average step sizes for both Bach’s pieces and the output of my networks. Figure 9 shows the average step sizes for Bach’s pieces as compared to my three networks. Contrary to my predictions, the pre-wired network had smaller average step sizes in its output than the recurrent network. However, another goal of the recurrent network was to see resolutions from chromatic scale degree twelve to one, and this does occur in the recurrent network’s output more than the output of the other two networks.

<p>IF $x_i - x_{i+1} \leq 6$</p> <p>THEN $s = x_i - x_{i+1}$</p> <p>ELSE $s = 12 - x_i - x_{i+1}$</p> <p>Where s equals the step size (not the average step size), x is equal to the chromatic scale degree of the note, and i is equal to the serial position of the note within the chorale.</p> <p>And now $S = \frac{s}{n-1}$</p>
--

Figure 8. Determining average step size

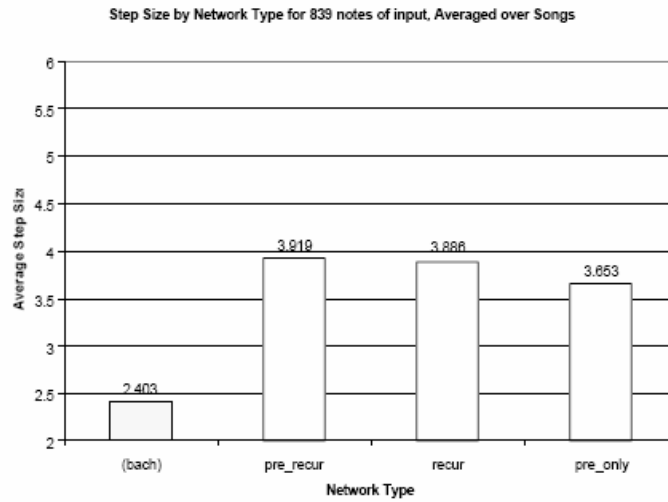


Figure 9. Average step sizes for Bach and all three networks after 839 notes of input, also averaged between both test songs

For further analysis, I've included graphs of the percent accuracy for the song "Open wide for me the portal" in Figure 10. This figure shows the percent accuracy for all three networks, after both 471 lines of input, and after 839 lines. Although the network that was both recurrent and pre-wired improved with more input, the other two networks had higher percents of notes matched to Bach's after only 471 lines of input. One possible explanation for this is that the chorales entered after 471 lines of input had more erratic bass lines that were not as easy for the recurrent networks to assimilate. Figure 11 shows the average step size for the same test piece, and as expected the average step size decreased with more input.

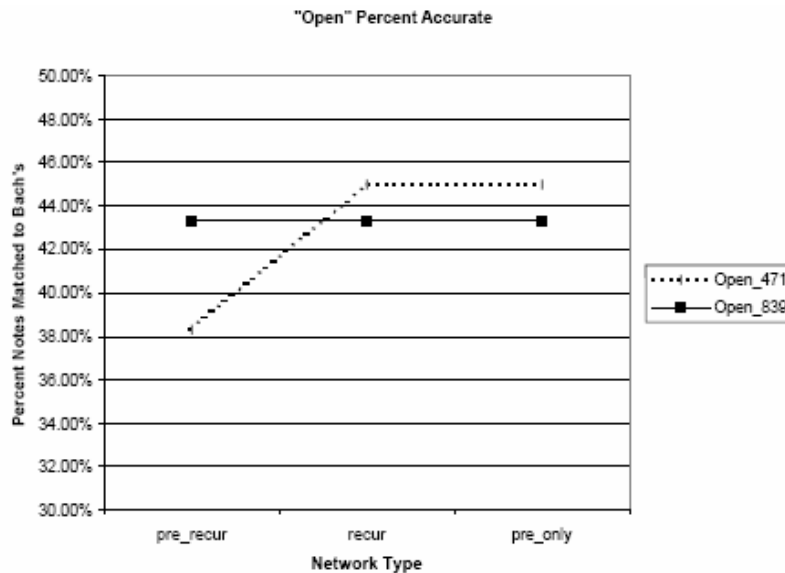


Figure 10. Percents accurate for "Open wide for me the portal."

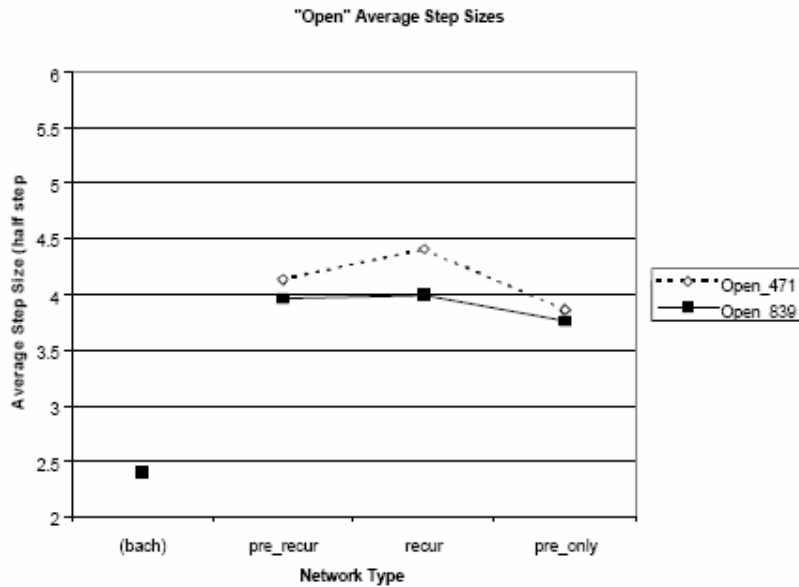


Figure 11. Average step sizes for “Open wide for me the portal.”

Figures 12 and 13 are similar to Figures 10 and 11. They present the same information, respectively, for the chorale “My trust is bold.” Once again the average step size decreases with more input, as seen in Figure 13. Figure 12 shows that the percent accuracy improves with more input, except for the recurrent network. This fact further supports the possibility that the later chorales inputted to the networks were less appropriate for the recurrent network, and more suited for the pre-wired networks.

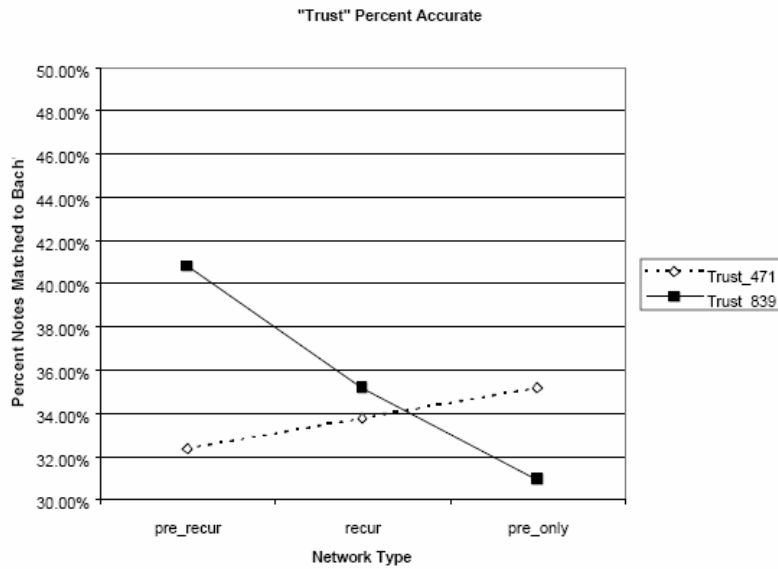


Figure 12. Percents accurate for “My trust is bold.”

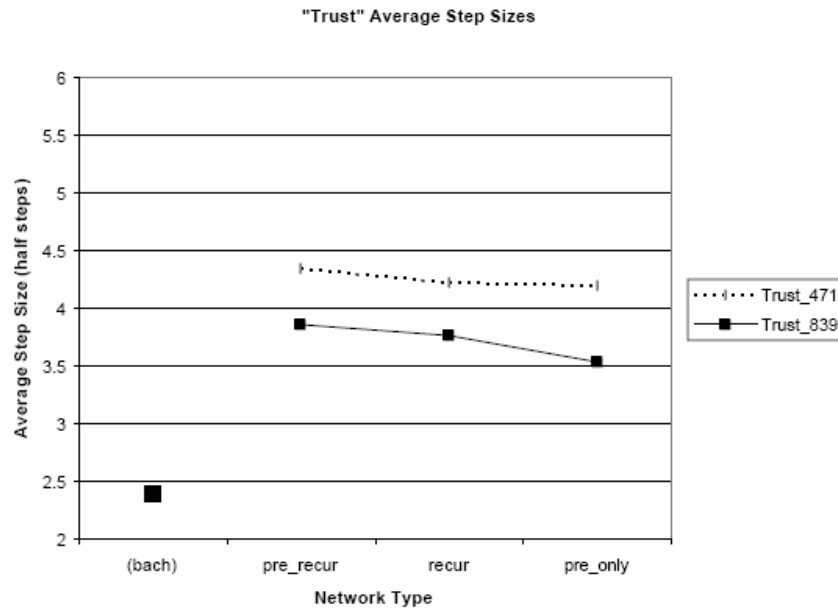


Figure 13. Average step sizes for “My trust is bold.”

6. Problems and Possible Solutions

Artificial neural networks find the most probable outputs by strengthening associations between inputs and correct outputs during training. The network’s ability to propose “creative” output is significantly hindered, as it is, in essence, designed to do the exact opposite. Thus, my networks never chose notes other than major scale degrees as output. Bach, on the other hand, occasionally used accidentals (notes that are not major scale degrees) to add to the creativity of his pieces. A possible solution to this would be to have occasionally insert some randomness into the network’s output, however this would probably end result in some unpleasing outputs. Another possibility that might provide more creative outputs would be to use a loose choice rule to determine the output for each input line, as opposed to the maximum activation rule I chose to use. This way a little randomness is used, but the likeliness of an output node being chosen is weighted by its activation. This could provide a nice output, since output that the network is very confident of would most likely not change, but when activation is a little more evenly dispersed throughout the output layer, more interesting choices may be made.

Another problem with the networks’ outputs was the inability to handle runs in the soprano and bass. A “run” in music is usually a fast-paced (eighth notes or shorter) section of a melody or harmony, and usually contains a lot of stepwise motion. Because my networks only had information about one note in the past or future at most, they were unable to model composition of runs in the bass. Again, the result is a less “artistic” harmonization. In the future, it would be interesting to let the network see several steps into the past and future of the soprano line, and at least two steps into the past of the bass line. In the hopes of modeling

human harmonization, it seems only fair to provide the network with this information, since obviously any human attempting to complete the same task would have such knowledge.

Another problem already mentioned was the difficulty with representing rhythm to the network. A way of fixing this problem might be to add nodes to the input layer that would represent the duration of each note. So along with representing the pitch of each note in the input, another one of five nodes (perhaps) would be activated that would tell whether the note was a fermata or whole note, half note, quarter note, eighth note, or shorter. The problem with this solution, however, is that it does not allow the bass line to move freely of the soprano line. The ability for one soprano note to be harmonized with several moving bass notes is one of the most important parts of harmonization in Bach's style, so this solution would have to be altered if not entirely abandoned.

Despite the problems with my networks, I am pleased with the results. The fact that their output approached fifty-percent accuracy to Bach's notes is promising, to say the least. In the future, I hope to improve both the networks' percent accuracy and average step size by implementing some of the ideas I've mentioned here.

7. Acknowledgments

Special thanks to Erik Isaacson, Sean McLennan, Robert Goldstone of Indiana University Bloomington and the COGS Q400 students of Spring, 2001.