UNIVERSITY OF CALIFORNIA
SANTA CRUZ

SOLOS (DICE GAME) AND CONDUCTOR (NEURAL NETWORK)

A dissertation submitted in partial satisfaction
of the requirements for the degree of

DOCTOR OF MUSICAL ARTS

in

MUSIC COMPOSITION

by

Andre Marquetti

June 2015

The Dissertation of Andre Marquetti is approved:

_____
Associate professor Ben Leeds Carson, chair

_____
Professor Larry Polansky

_____
Professor Amy C. Beal

_____
Tyrus Miller
Vice Provost and Dean of Graduate Studies

Solos (Dice Game) and Conductor (Neural Network)
Table of Contents

Abstract

Andre Marquetti

Solos (Dice Game) and Conductor (Neural Network)

Solos (Dice Game) and Conductor (Neural Network) combines a multilayered environment made of solo pieces, ensemble music, and a digital network. A series of short interludes separate each solo and the *Main Section* of the composition. The *Main Section* re-unifies the instruments' solos by granting the performer the opportunity to improvise over a repeated vamp. Unique audio synthesis methods identify each solo instrument. The solo morphology for the clarinet is an open-notated piece derived from alternations between pitch-bend-, multiphonic-, and microtone-structures that includes real-time audio synthesis. The saxophone and violin solos make similar use of real-time audio synthesis with structures emerging from a dice-based compositional method. The process of selection for the musical material, and the game strategies used to generate their forms, consists of an algorithm combining randomness and choice.

The neural network that returns as output the identity of the input also generates "noise" internally; patterns of noise then converge toward the key pattern, and become a resource from which many audio synthesis attributes can be derived, like a mirror reflecting an object from a different perspective. In the digital network I developed an algorithm to parse pitch information from live audio MIDI pitch detection. At the core, I adapted a neural network to perform analysis on the pitch-class input vector. The neural network is a solution-based evolutionary algorithm that

simulates brain neural activity. In the saxophone solo particularly, I formulate a method to converge the dice game probabilities with the neural network activity. The digital score consists of selecting a pitch-class set for parsing pitch with the neural network key pattern. A key pattern requires a sample input pitch-class vector and a sample output to match the input. In performance, the music emerges from the particular choices of key pattern for solo compositions and solo improvisations in the *Main Section*.

Acknowledgments

I would like to thank the University of California Santa Cruz Music Department for their generous support in carrying out my research, my committee members Ben Leeds Carson, Amy C. Beal, Larry Polansky and Paul Nauert for their teaching; sfSoundGroup Matt Ingalls, Ben Kreith, Monica Scott, Brendan Lai-Tong and John Ingle who premiered the composition; fellow graduate students for their valuable input; my dear family Mirtes, Patricia, and Bruno to whom *Solos (Dice Game)* is dedicated.

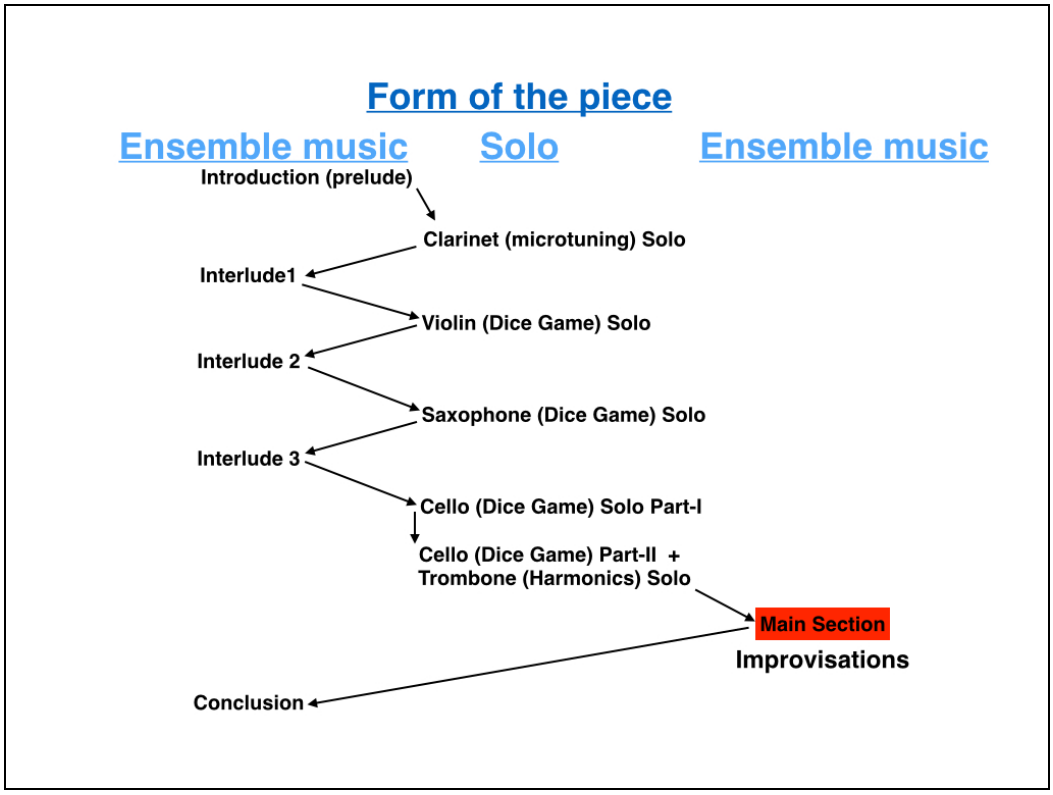


Figure 1. Composition form

I. Introduction

My composition *Solos (Dice Game)* and the digital performance environment *Conductor (Neural Network)* originated from three interrelated goals:

1. To compose solos for clarinet (Bb and bass), alto saxophone, trombone, violin and cello using an array of techniques and notations.

2. To implement a series of cognitive modules for parsing and interpreting the pitch from sound input.

3. To generate algorithmic music using various methods for audio synthesis.

The converging point of *Solos* using the dice game method to compose music and the *Conductors'* neural network for processing the music is based on pattern-extraction, recognition, generation, emergence and musical complexity.

I.1 Musical and Computing Influences

Table 1.1 lists some esteemed musical practitioners who influenced my composition. Complementary to the musical side of this work, Table 1.2 lists research influences that helped me gain a better understanding of networks. Some entries in the tables point sections in the thesis where their concepts (directly or indirectly) influenced my approach.

Table 1.1 Musical influences

A. Direct compositional influence:
- John Cage, chance operations and *HPSCHD* (1969) (prevalent Mozart Dice Game) (Brooks 2002, 136) => **Section IV**
- Morton Feldman music poetics => Interlude pieces, **Section II**
- Mozart, *Mozart Dice Game*
- Stockhausen (1989) intuitive (39) and stochastic music (66) formula (73) and group composition, and musical synthesis
- Anton Webern, pointillist pre and 12-tone compositions, palindromes techniques (dice game-serial strategies) => Violin solo **Section V.3.1**

B. Less direct compositional influence:
- Philippe Manoury, *Jupiter* (1992) pitch-tracking, score following, audio synthesis, segmentation and form (May 2006; Rowe 2002) => Solo audio synthesis, **Section V**
- Richard Teitelbaum (2006) *Solo for Three Pianos* (1982) (Rowe 1993, 2001) => Saxophone solo synthesis, **Section V.2.3**

C. Improvisational influence
- Derek Bailey (1983) style (10, 16, 63) and practice (5, 32,79)
- Anthony Braxton, solo music versus ensemble, language strategies, geometric schemes, synthesis, and identity states (Heffley 1996, 228–232)
- Miles Davis' electric music, *Circle in the Round* (1967), rhythmic concept, solo and group approach, 'transcendence' and 'inclusion' philosophies (Tingen 2001; Mandel 2008) => *Solos Main Section*: vamp and chordal improvisation, **Section V.4**

D. Music Theory:
- Robert Cogan and Pozzi Escot (1976) musical space (16–17), register (octave) expansion/contraction, analysis of Beethoven *Piano Sonata in Eb, Op.31 no.3* (42–49), spatial field theory, Schoenberg *Six Little Piano Pieces* (49–53) and musical language, Debussy *Syrinx* (100) => important ways, in which I combined the *Solos* with the audio synthesis, **Section V**

Table 1.2 Non-Musical influences

A. Computer Scientists:
- Marvin Minsky (1988) sensing the world around us (155), processing (111), sorting (25), and designing knowledge tree networks
- John Holland (1998) emergence (90)

B. Composers and Music Technologists:
- David Cope (2005) semantics network (276)
- Mark Leman (2008) sound, meta-cognition
- Robert Rowe (2001) *Cypher* (Rowe 1993, 219) and feature space representation

C. Neurologists:
- Frank R. Wilson (1988) hand and brain network (63)

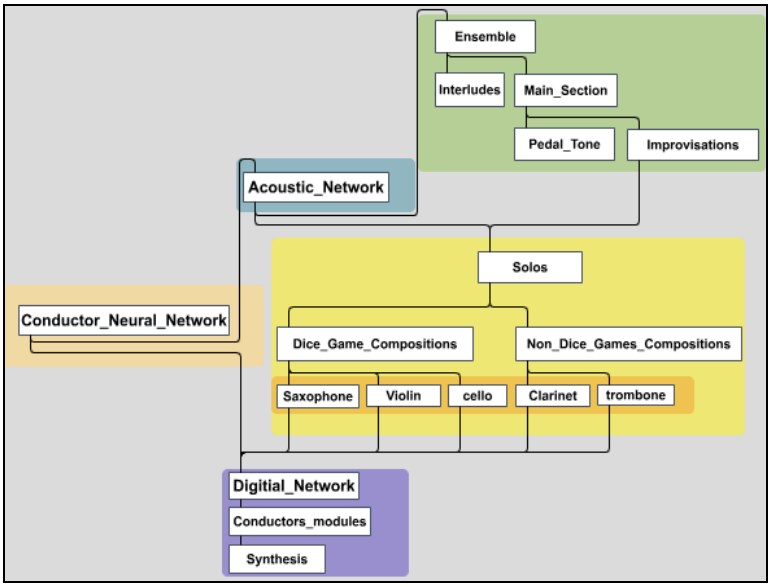


Figure 2. Overview of compositional components

II. *Solos (Dice Game)* Composition

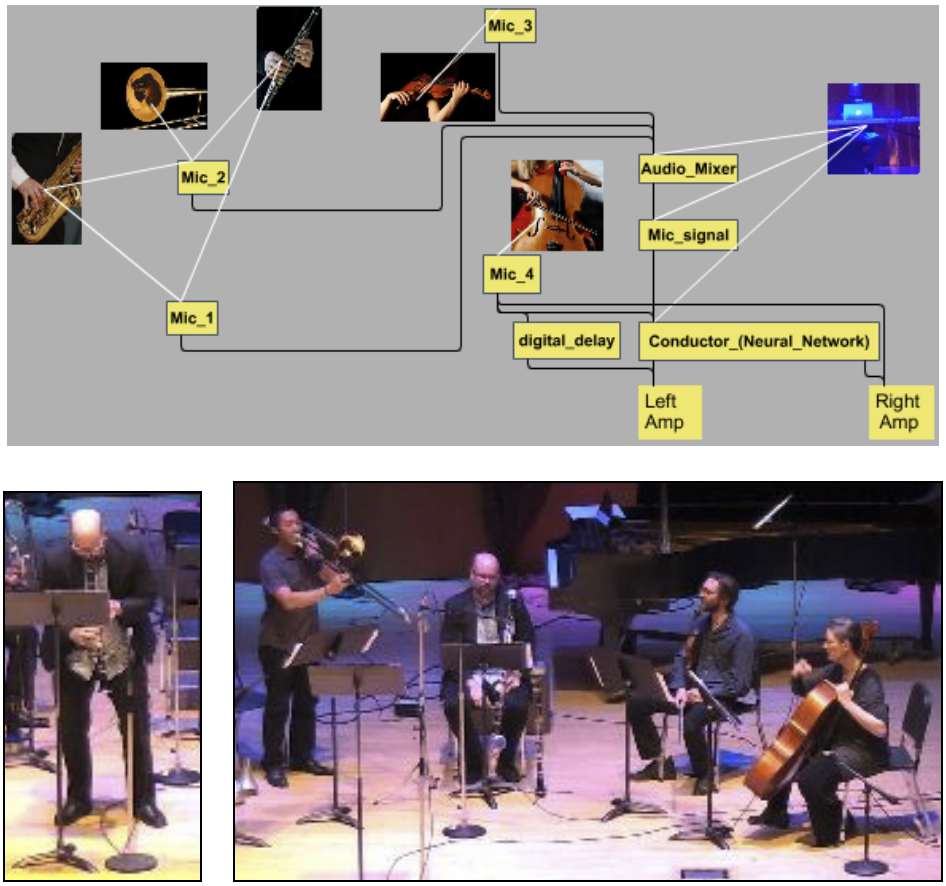
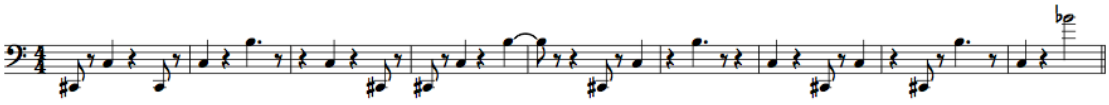


Figure 2.1 Performance network

The composition consists of solo (Table 2.2) and ensemble sections (Table 2.1). Each solo is scored according to a different notation. The violin and trombone are more conventionally notated, whereas the clarinet score shows the relative pitch range, portamento, multiphonic and microtuning notation open to the performer's interpretation. In the saxophone solo appears an event for the player to improvise his or her own musical material. The cello score contains various non-pitched events, realized by striking the palm on the back of the instrument or fingertip tremolos,

knocking and gliding the hand on the board. For the saxophone, violin and cello solos, I rolled a pair of dice to compose the music. The specific compositional procedures are introduced in **Section IV**, and explained in **Section V**.

The ensemble section titled *Main Section* features solo improvisations. During these passages, a smaller group of instruments plays a vamp while the improvisation goes on. In addition, a series of short ensemble interludes bind the solos and *Main Section* together. These miniature pieces were composed by correlating a fixed pitch, interval and register with a rhythm, combining a note of a specific value with a rest of the same value, and alternating the value freely. The wave pattern in Example 2 is composed of eighth note + eighth rest + quarter note + quarter rest … dotted quarter note + dotted quarter rest … I selected a base pitch C# for the shortest rhythm of an eighth note and the major seventh interval above the base pitch C#-C corresponds to the next quarter note rhythm. The pitch and register continues to expand this way C#-C-B for each added eighth note value, towards the longest half note rhythm, Bb.



Example 2. Interlude 2

In the final form (Table 2.3), I alternated solo and ensemble sections within the ideal overall duration of thirty minutes. I initially composed every solo, interludes and *Main Section* independently of this form. To retain the composition within the thirty minutes mark, every effort was made to keep solos and ensemble sections concise. My general form scheme reflects a loop, between ensemble compositions

(interludes) and solos (one-per-instrument) with the *Main Section* at the end. The *Main Section* loops a pedal tone (F4) with chords made by stacking major and minor 3rds intervals.[1] The collections of pitches found in these thirds arpeggios do not include the pedal tone pitch.

An important aspect of the composition involves chance procedures for the solos. The aspect of pitch and timbre between a solo and interlude is a consequence of chance operations inspired by Mozart and Cage. My interludes were conceived variously for trios, quartet and full ensemble.

Table 2.1 Ensemble music components

A. The introduction (prelude), ending (conclusion) and 3 interludes between solos musical materials consist of:

1. A note with a rest of the same value alternates sixteenth to half-note rhythms
2. 2- to 3-part canons, a half note apart at the unison or in a major seventh
3. Major seventh intervals (Example 2), the lowest pitch corresponds to the shortest rhythm or sometimes the highest; as a pitch ascends by a major 7 the rhythm lengthens. Each interlude extends, expands or contracts a core set of pitches. Interlude 2, Example 2, between violin and saxophone solo, modulates to a new set of pitches. The sequences of pitches link the first to last in a circle
4. Timbre modulation processes: 1. a pitch/rhythm unit is sustained on the same instrument; 2. changes from instrument to instrument, or 3. a combination of A.4.1 and A.4.2 with instruments playing a melody

B. A main section alternates *tutti* and *solo* improvisations:

1. All instruments sustain pedal tone (F4)-as long as the player can in one breath or single bow with a fermata shown above the note-in groups of three attacks points
2. Maj/min 3rds arpeggios
3. Solos improvisation over chords determined from B.2
4. Vamps over solos on the chords from B.3
5. Quotations (taken from my original piano piece based on *Mozart Dice Game*, example 4.1-4.3), upward major sevenths fast flurries, played above and below pedal tone (F4), and octave contours over a dominant seventh (Ab)

[1] The chords to improvise derived from the stacked thirds structure.

Table.2.2 Solos music components

A. Clarinet (**Section V.1.1**) comprises:

1. Graphic material, pitch bending plus multiphonics
2. A microtone melody derived from plotting points on a two-staff system. The top staff, five lines indicates a micro-intervals (1/8-1/4-1/2-3/4-1). I selected an initial pitch from the lower staff, three lines; showing the relative clarinet pitch-range, high, middle and low. I added to this pitch the micro-interval found in the upper staff, in a line moving up or down. For example the first phrase, graph indicates low range initial pitch + 1/8 + 1/8 + 1/8 + 1/8 + 1/8 upwards; next phrase, middle range pitch + 1/8 + 1/4 + 1/8 + 1/4 upwards. In the final score, I removed precise tuning. I compromised the tuning with a more general indication letting the player determine a ratio smaller than semitone.
3. Embellishments

B. Alto Saxophone (**Section V.2.1**):

1. *Mozart Dice Game* dictated basic cells, their sequences and transformations. Cell configuration included: number of notes, pitch, accidentals, articulation and dynamics
2. Improvisations: the player generates a cell/phrase event to remember and repeats the same event (closely as possible) as shown in the score

C. Trombone:

1. Harmonic series slide positions

D. Violin (**Section V.3.1**):

1. Two types of articulation, pizzicato and bowed
2. *Mozart Dice Game* dictated random selection of pitches from a scale and register distribution.
3. 12 tone series, envelope signature, determined from D.1
4. *Mozart Dice Game* selected 'all four strings' simultaneous chords from a table of all possibilities; the chords are strummed and bowed

E. Cello:

1. The *Mozart Dice Game* selected a hand striking method on the back (or rear) of instrument body from a list of possible variations

A. Basic form:

Prelude=>clarinet solo=>interlude 1=>violin=>interlude 2 =>alto saxophone=>interlude 3 =>cello solo overlapping into trombone solo=> Main Section=>conclusion

B. *Main Section* form:

Pedal tone=>violin=>pedal=>clarinet/saxophone duet=>pedal=>trombone=>saxophone=>pedal=> clarinet=>pedal=>cello=>pedal=>extended conclusion

Figure 2.2 Composition forms

III. Digital Network

III.1 *Conductor* Types

The *Conductor* network consists of modules that are designed to translate aspects of music from a performer's ongoing activity via a microphone, for example, identifying a key, chord, melody, and inducing a common meter.[2] These modules include a pitch-class vector filter, tonal saliency, pattern matcher and quantizer. The implementation of these modules with the composition is described in **Section V** and **VI**.

During a performance, a user influences these modules and mixes the processed (pitch-tracked) live sound by selecting a few features like context, training, window size for grouping pitches (i.e. given a monophonic signal) and tuning system.

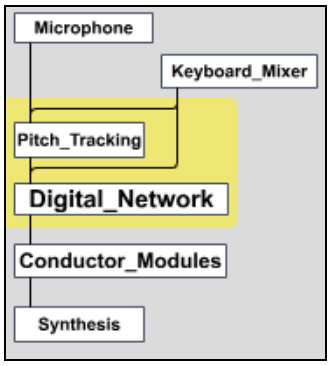


Figure 3.1 Digital network

III.2 *Conductors*

When a sound event is played, an algorithm takes the resulting MIDI input and parses it according to a pitch-class vector then feeds the vector into the *Conductors'* modules

[2] The algorithms are drawn from Robert Rowe (2001) *Machine Musician* in C# program. I customized the code to Java and Max/MSP (mxj) Java external converting their function in a working object.

(figure 3.2). Pitch-tracking a sound to MIDI introduces several inaccuracies (Rowe 1993; Gjerdingen 1994).[3] Regardless, Miller Puckette's (1998) Fiddle MSP object works well. The user can specify several parameters to improve the pitch-tracker overall performance, for example: highest partial analysis, 7th overtone (default); the attack portion of a sound, limitation of volume between 10 to 100 db; minimum MIDI velocity; and vibrato (50 ms=>0.5, or a half-tone). Any value outside the specified attributes temporarily halts the MIDI pitch sound-conversion. My algorithm contains another measure of reassuring accuracy. It waits for a new MIDI pitch to continue on parsing a vector.

An additional factor to consider in the MIDI parser is how the pitch-data from the monophonic signal is recorded and segmented. The user can specify a window size for any sequences from single pitch, dyad, triad, or tetrachord pitch-class set. Since each new note of a melody can have a repeated pitch before the previous, a tetrachord pitch-class set may result in a triad, or dyad, for example a tetrachord 60-61-60-61 yields a dyad 60-61.[4] If the window size is very large pitch-classes will be repeated or there is a chance the analysis will yield an all-pitch-class vector. A single pitch-window processes the fastest rate of analysis or pulse in which the audio

[3] Microphone pitch tracking: "A delay of over 30 ms is required in the best case for identification…the attack portion of an instrumental waveform will be the least regular part of it" (Rowe 1993, 16) and "the abstraction of MIDI already throws away a good deal of timbral information about the music it represents" (Rowe 1993, 120). Gjerdingen (1994) points out the virtual pitch representation of a signal is a model that assumes "a strict form of prior determination in which musical tones excites" (143).

[4] A continuous sequence of a repeated pitches, or ostinato pattern is discarded. Tremolos are allowed.

changes. A larger window slows down the pulse from there. The window size and music both determine the rate in which *Conductors'* modules perform.[5]

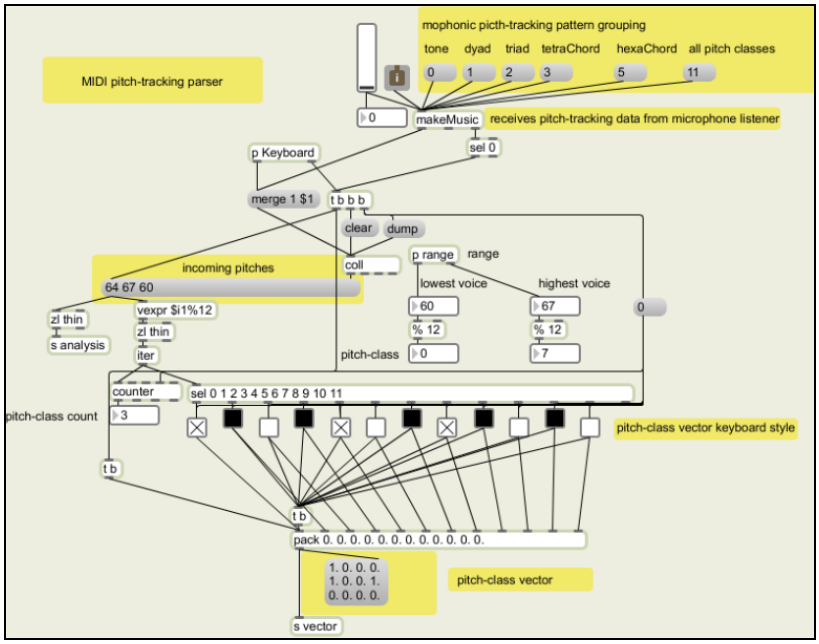


Figure 3.2 MIDI pitch-class set parser

The initial sound approach I implemented for the *Conductors'* modules is to normalize the resulting analyses between 0.0 to 1.0 to control the amplitudes of sine

[5] During sequencing, consecutive pitches become parsed in one group; a dyad, for example, follows 1-2, 3-4, 5-6, 7-8. As a result of this operation, there are the ignored relationships of importance, 2-3, 4-5, 6-7. A larger window takes into consideration simultaneously 1-2-3-4-5-6-7-8 sequencing. Smaller collections are more favorable. There is no simple way of predicting exactly how the data will be parsed from the live input.

waves from middle C4, pc 0, 262Hz to B4, pc11, 494Hz.[6] The densest cluster occurs when all pitch class amplitudes in a vector are 1.0.[7]

III.3 Neural Network Application to Real-time Pitch Composition and Improvisation

The neural network I have adapted is a supervised learning type; one makes a text file of input and output matching corresponding data (see Example 3.1).[8] A function called *Train* evaluates the data repeatedly in the neural network. Once this process is completed the neural network performs real-time pattern matching.[9]

```
12;                                                                    Pitch-Class
C    C#   D    D#   E    F    F#   G    G#   A    A#   B    Transposition
1.000 0.000 0.000 0.000 1.000 0.000 0.000 1.000 0.000 0.000 0.000 0.000 ; input set  0
1.000 0.000 0.000 0.000 1.000 0.000 0.000 1.000 0.000 0.000 0.000 0.000 ; output set  0
```

Example 3.1 C major (m3-M3) match its identity key pattern

The pitch-class vectors matching an input to output can be any pattern.[10] The simplest adaptation is a melodic pattern that recognizes itself in all keys. In the *Conductor* network, C major is mapped to C major. For any given input the neural network will output a chord identity that reflect the intervals of a C major triad (M3,

---

[6] I implemented the tonal saliency module first (Virtual Pitch, VP). This module estimates the relative importance of each pitch-class in a vector. The model was conceived based on empirical studies (Rowe 2001, 47-49) of chord-root saliency in a major or minor context, a key, lowest tone and overtones series. Users have the option to set any of these parameter strategies: lowest tone on, overtones series on, and C major context on. An interesting side effect of the tonal saliencies is that offset pitch from C major, like F#, can exert some influence.

[7] See Hunt and Wanderley (2002) for a similar approach (104).

[8] I customized the Quickprop program (Rowe 2001, 97).

[9] My network is composed of 12 nodes x 3 layers: 12 input, 12 hidden, and 12 output nodes. Every node in one layer connects to the others. The hidden layer evaluates a pattern set (input and output) during training the network. If the sum of weights per connection at the input exceeds the node threshold in the hidden layer causes that neuron to fire, similarly, from hidden to output. The logic of a 12 x 12 x 12 topology naturally comes from music 12 chromatic notes, 12 keys or 12 tone-series table transposition, inversion and retrograde. A more focused, musical application, should consider setting up just-enough layers to avoid non-wanted, side effect from pattern transpositions.

[10] Rowe (2001) associates I-IV-V relationship in a given key with a group of notes (98–101).

m3, 5th), and a likely transposition thereof. If the chord played is complex the net will return an analysis, retaining some of the major triad characteristic intervals while introducing others that match a most likely transpositions.[11] Apart from the neural network matching major triads, all others matches cannot be foreseen.

The degree to which the neural network recognizes a pitch class can vary from a likely recognition if the output is greater than 0.7 to full recognition or 1.0, partial recognition 0.4 to 0.6, and some recognition 0.1 to 0.3. Table 3.1 shows one-to-one pitch matching with common patterns (dyads and chords) presented to the neural network that I played on a MIDI keyboard. The blue areas indicate strong neurons firing and yellow weaker. The table shows the weights trend during running the neural network. The weights trend appears to favor certain pitches highlighted in green (stronger) and gray (weaker). Table 3.2 shows the same procedure for a major triad matching its identity. Re-running the neural network will yield different weights; Table 3.1 and 3.2 are not the same trend.

[11] Pitch C can occur in 3 different degrees C major, F major and Ab Major.The network can output the minor triad C, A, and F since their 3rd is the inverted major intervals. The neural network does not understand their semantic differences. It looks for numerical relations in where a major triad intervals and minor are the same (see Table 3.2).

Table 3.1 Single pitch-training C=>C

	C	C#	D	Eb	E	F	F#	G	G#	A	A#	B	
C	1.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	;
G	0.	0.	0.	0.	0.	0.	0.	1.	0.	0.	0.	0.	;
C-G	0.69	0.	0.	0.	0.	0.	0.	0.8	0.	0.	0.	0.	;
G-D	0.	0.	0.99	0.	0.	0.	0.	0.97	0.	0.	0.	0.	;
C-E-G	0.75	0.	0.	0.	0.99	0.	0.	0.37	0.	0.	0.	0.	;
G-B-D	0.	0.	1.	0.	0.	0.	0.	0.1	0.	0.	0.	0.19	;
C-E-G-Bb	0.36	0.	0.	0.	0.87	0.	0.	0.11	0.	0.	0.04	0.01	;
G-B-D-F	0.	0.	0.99	0.	0.	0.02	0.	0.	0.	0.	0.	0.3	;
C-E-Bb-D	0.	0.	0.93	0.	0.84	0.	0.	0.	0.	0.	0.43	0.	;
G-B-F - A	0.	0.	0.	0.	0.	0.27	0.	0.02	0.	0.68	0.	0.	;
E-G-Bb-D	0.	0.	0.99	0.	0.84	0.	0.	0.25	0.	0.	0.02	0.	;
B-D-F-A	0.	0.	0.98	0.	0.	0.88	0.	0.	0.	0.11	0.	0.	;
G-Bb-D	0.	0.	0.97	0.	0.	0.	0.	0.41	0.	0.	0.53	0.	;
D-F-A	0.	0.	0.99	0.	0.	0.48	0.	0.	0.	0.73	0.	0.	;

=>D<=>E<=>G<=>C<=>A<=>F<=>A#(Bb)<=>B<=

Table 3.2 Triad pitch-training C-E-G=>C-E-G

	C	C#	D	Eb	E	F	F#	G	G#	A	A#	B	
C	1.	0.	0.	0.01	0.99	0.	0.	0.98	0.01	0.	0.	0.	;
G	0.03	0.	0.1	0.	0.04	0.	0.	1.	0.	0.	0.02	0.	;
C-G	1.	0.	0.	0.	0.99	0.	0.	1.	0.	0.	0.	0.	;
G-D	0.	0.	1.	0.	0.	0.	0.	1.	0.	0.	0.71	0.	;
C-E-G	1.	0.	0.	0.	1.	0.	0.	1.	0.	0.	0.	0.	;
G-B-D	0.	0.	1.	0.	0.	0.	0.	1.	0.	0.	0.	1.	;
C-E-G-Bb	0.99	0.	0.	0.	0.99	0.	0.	1.	0.	0.	0.05	0.	;
G-B-D-F	0.	0.	1.	0.	0.	0.88	0.	0.83	0.	0.1	0.	0.06	;
C-E-Bb-D	0.14	0.	0.33	0.	0.75	0.	0.	1.	0.	0.	0.04	0.	;
G-B-F - A	0.01	0.	0.99	0.	0.21	0.94	0.	0.58	0.	0.85	0.	0.02	;
E-G-Bb-D	0.	0.	0.98	0.	0.05	0.	0.	1.	0.	0.	0.99	0.	;
B-D-F-A	0.	0.	1.	0.	0.	0.07	0.55	0.	0.	0.87	0.	0.19	;
G-Bb-D	0.	0.	0.99	0.	0.	0.	0.	1.	0.	0.	1.	0.	;
D-F-A	0.02	0.	1.	0.	0.	1.	0.01	0.	0.	0.93	0.05	0.	;

=>G<=>D<=>E<=>C<=>A/F<=>A#(Bb)<=>B<=>F#<=

A neural network is designed to perform real-time pattern matching, time quantization, and the recognition of styles and gestures (Cope 2005; Rowe 1993 & 2001; Mulder 1999). David Cope (2005) points out the motivations for using a neural network: “Inputting two or more quite different examples and desired output into a network during training can achieve an interesting mix of musical styles or materials” (71). Another key pattern could be the dominant to tonic motion, or a G major triad resolving to C major in a harmonic progression (Example 3.2).

```
12;                                                                  Pitch-Class
C    C#   D    D#   E    F    F#   G    G#   A    A#   B    Transposition
1.000 0.000 0.000 0.000 1.000 0.000 0.000 1.000 0.000 0.000 0.000 0.000 ; input set  0
1.000 0.000 0.000 0.000 0.000 1.000 0.000 0.000 0.000 1.000 0.000 0.000 ; output set  0
```

Example 3.2 C major yields F major key pattern

The neural network can be adapted as a harmonizer. For example, a single pitch input matches the root, third, and seventh above a chord (Example 3.3).

```
12;                                                                  Pitch-Class
C    C#   D    D#   E    F    F#   G    G#   A    A#   B    Transposition
1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ; input set  0
0.000 0.000 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 1.000 0.000 ; output set  0
```

Example 3.3 Bebop chord-style accompaniment –root harmonizes 3rd and 7th

Any complementary relation in training, like black and white note clusters of piano, or hexachord combinatorial 12-tone aggregates will generate complementary pitch relations to embellish, harmonize or extend a played melody (Example 3.4).[12]

```
12;                                                                  Pitch-Class
C    C#   D    D#   E    F    F#   G    G#   A    A#   B    Transposition
1.000 1.000 1.000 1.000 1.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 ; input set  0
0.000 0.000 0.000 0.000 0.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 ; output set  0
```

Example 3.4 Twelve-tone complementary hexachords key pattern

For composition the neural network outputs either the expectations or non-determinate chords that are somewhat related to the training the user sets. Another useful feature is the neural network's robust ability to remember any notes played on a keyboard with the same matching output. Trainings can be changed relatively quickly, making a text file of input and output set is an automated easy procedure. A neural-net may be connected sequentially to many sub-networks. Its modularity leads to intricate and complex musical webs.

[12] Some triads loop back to the initial C pitch-class. The audio playback will reflect their inversion. For example, Bb-D-F sounds D-F-Bb, the second inversion of that major triad, see **Section V.1.2**.

My digital network consists of two main interfaces: for processing sound (figure 3.4) and *Conductor* network modules (figure 3.3). I have listed in a Table 3.3 all the active components for the main *Conductor* module only. Sound is described in **Section V** and **VI**.

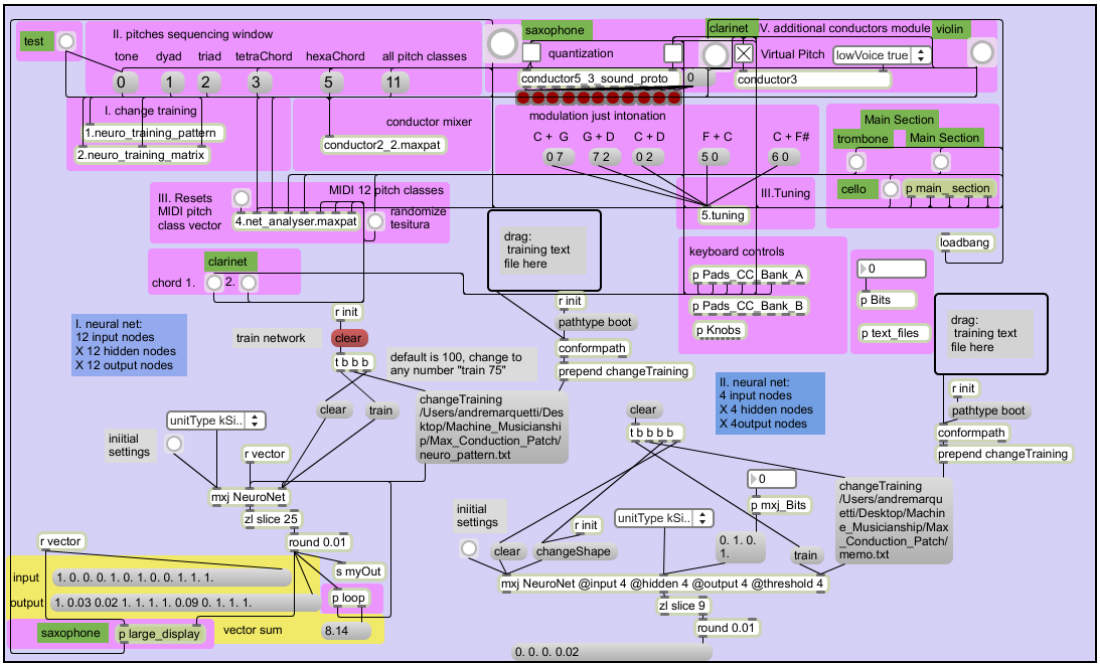


Figure 3.3 Main *Conductor (Neural Network)* modules interface

Table 3.3 Main interface components

conductor4_2: main patch

conductor2_2: parses, MIDI pitch tracking, sequence the user determines (dyad, triad…) to a pitch-class vector

mxj NeuralNet (Java code): performs analyses, matching likely patterns (pitch-class amplitudes) with input patterns to the key patterns the user feeds the neural network

mxj Tessitura (Java code): randomizes all pitch-class in a given register band

4.net_analyser: formats MIDI pitch to frequency, from mxj Tessitura, with mxj NeuralNet pitch-class amplitudes vector

1.neuro_training_pattern: generates a neural network, learning file-expected input matches to its identity. The user selects the pitch-class vector in real-time, parsed from conductor2_2

2.neuro_training_matrix: generates a neural network, learning file-expected input matches to output. The user can select any input/output correspondence realized on a keyboard style interface

mxj VirtualPitch (Java code): Analyses a pitch-class vector-pitch strength, from the context-lowest note, overtone series, major or minor mode and key

conductor3: feeds forward conductor2_2 vectors and the user configures the mxj VirtualPitch context

3.stochastic_velocity: Normalizes MIDI velocity 80 to the mxj VirtualPitch with greatest strength; all others pitch-class are scaled accordingly

6.vp_resonance: formats together frequencies, amplitudes and virtual pitch analysis in a mxj Noise (java code) object. The data is fed to the sound synthesis sinusoids resonance

mxj tuningRatio (Java code): morphs 2 Just intonation scale

For example:

C Just Tuning:	1 0 1 0 1 1 0 1 0 1 0 1
F#Just Tuning:	0 1 0 1 0 1 1 0 1 0 1 1
C + F#:	1 1 1 1 1 1 1 1 1 1 1 1

mxj freqMidi (Java code): MIDI pitch conversion to Frequency back to MIDI

5.tuning: User's Just intonation interface-mxj tuningRatio and mxj freqMidi goes to 4.net_analyser

mxj DesainQuant (Java code): Desain and Honning quantizer (1994)

conductor5_3_Sound: Interface mxj NeuralNet triggers 11 IOI mxj DesanQuant melody recorded from performer

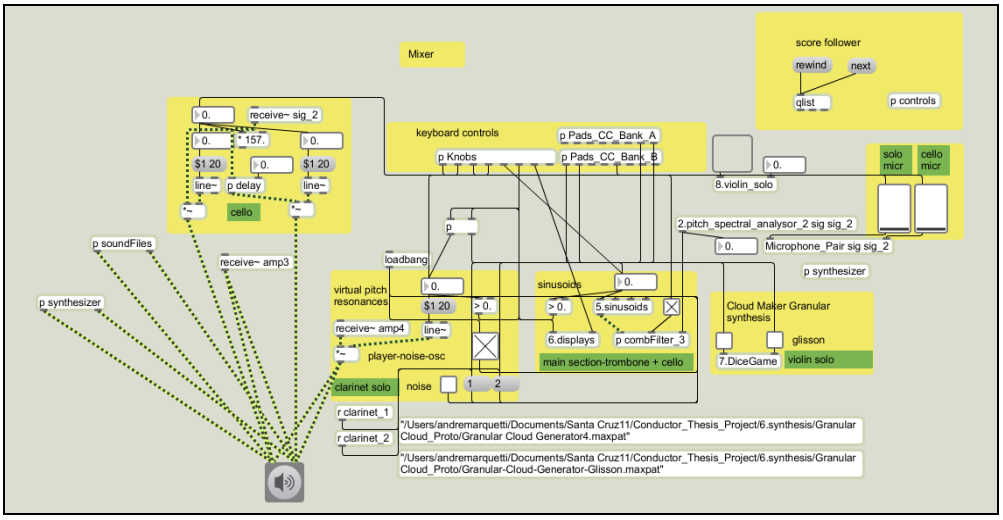


Figure 3.4 Audio interface

IV. *Dice Game* and Composition Genesis

Musical dice game briefly popularized from 1757 to 1812.[13] Mozart is credited to have conceived a work considered the standard game.[14] A pair of dice is rolled to select musical numbered bars from the composition attributed to Mozart. The music is composed of simple arpeggios in treble and root-fifth bass accompaniment on a I-IV-V progression in the key. There is no clear melody. Pitches underline the chord tones to the waltz basic eighth note pulse. The game consisted of the music numbered for each bar. A numbered table consisting of eleven rows dice combinations and eight columns referred to the corresponding bar in the music. The table is used to combine the rolled dice matching music in a version ready for performance.

[13] See Stephen A. Hedges (1978) *Dice Music in the Eighteenth Century*.
[14] *Mozart Musikalisches Würfelspiel* Edition published by Nikolaus Simrock of Bonn in 1796 (Hedges 1978, 183).

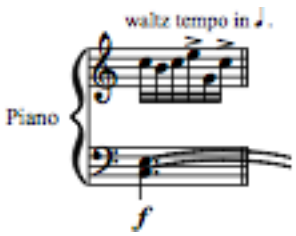
These games were done in salons, where a person without music education could compose his or her version and passed it on to an available pianist. The game represents a simple form of automated composition and a tool to generate chance combinations from any musical material the composer determines. In *HPSCHD* (1967) John Cage "substituted" the original *Mozart Dice Game* music with music quotations from Cage, Mozart, Satie and collaborator Lejaren Hiller. Hiller devised a computer program version of the game (Brooks 2002, 136). The correlation, computer and dice game, indeed "cannot help but sound reasonable" (Todd & Werner 1999, 315). For my *Solos (Dice Game)* composition I reinvented the game altogether. Music composition entails selection, designing combinatory schemes and foreseeing continuities.

IV.1 Quotation of Mozart's Game

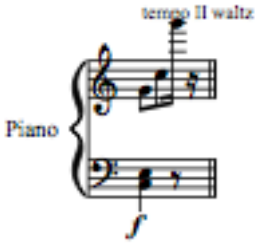
My first experiment was a piano work using the original *Mozart Dice Game* music. I developed a process by extending the possibilities given from the game. Each Mozart musical fragment in the game is transposed a half-step up making the piece bi-tonal. If I rolled a repeated numbered fragment, a full rest was substituted instead of music. In the next repetition of this numbered fragment, I selected the transposed version. The process continued with a rest before the following stage. Several remaining stages featured Mozart's waltz motives with the register an octave up for treble and down for bass and the music from one clef swapped with the other. I introduced a

chord to signal the end of my process. I imagined a continuum of music in two keys (a half tone apart) and pauses emerging in a non-determinate order.

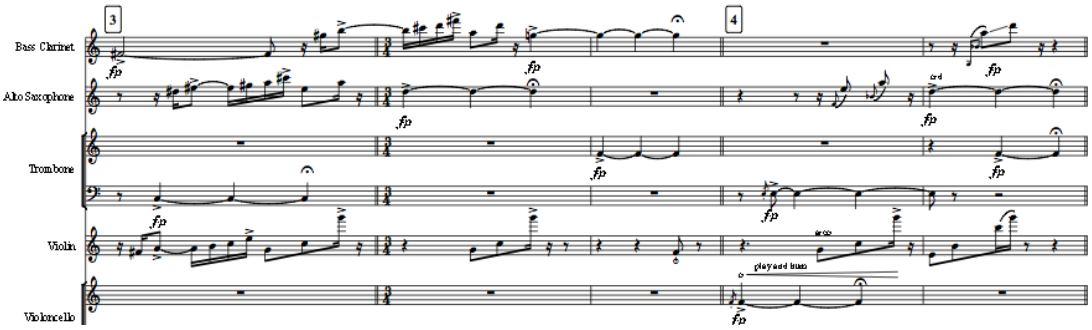
The *Main Section* of the *Solos (Dice Game)* composition features direct quotes (Example 4.3) from my piano version of the *Mozart Dice Game* music (Example 4.1) where I extensively added musical commentaries during rests (Example 4.2) and introduced major and minor third arpeggios between sections of the waltz (Example 4.4). These arpeggios form the basis of the *Main Section* improvisations (Example 4.5). In my piano piece, their oscillation with a musical section foreshadows my *Solos (Dice Game)* compositional grand plan of combining solos and ensemble sections.



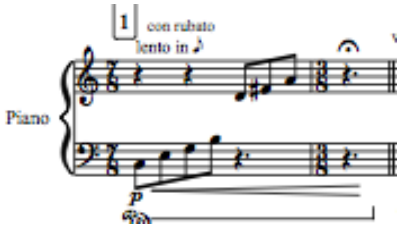
Example 4.1 Original *Mozart Dice Game* cell



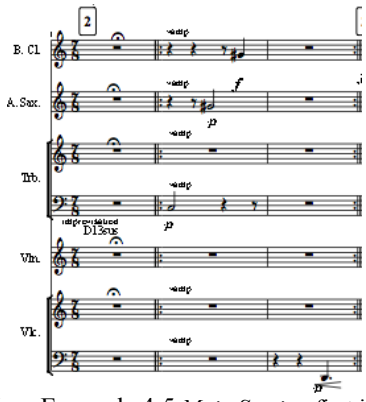
Example 4.2 *Mozart Dice Game* commentary cell



Example 4.3 *Solos (Dice Game) Main Section*, rehearsal mark 3 and 4 are dice game piano quotations



Example 4.4 Initial arpeggio before Mozart's music



Example 4.5 *Main Section* first improv. and vamp

The quotation material (Example 4.3) is combined with the main pitch (F4) of the piece. The motive G-C-G, Example 4.2, appears several times in the course of the *Main Section*, and in one large section, starting from rehearsal mark 17, to the end. The piano dice game composition and *Main Section* share several common features.

The violin, saxophone solo and cello were subsequent experiments with the dice using original music and designing entirely different games.

IV.2 Dice Game Probabilities

Each numerical combination of the dice is an empty slot that can be allocated to anything. For example, one set of rolls determines the pitch, another the timbre, and yet another determines rhythm.

For the solo violin music, all slots were used, but I decided to simplify my process for the saxophone solo. Finding 11 dynamics or 11 types of accents become unnecessarily cumbersome. I combined pairs of odd numbers (dice combinations) in 6 slots. I generated 10 musical cells + 1 open improvisation, with 6 odd number

combinations. Next, I selected their sequence and cell transformation based on the 11-combinatory scheme (2-12 odds).

Table 4.1 Odds from two dice

Violin (brown):	odds	Saxophone (gold):	odds
2(1&1)	=>1	2(1&1)	=>1
3(1&2)	=>1	3(1&2)+4(3&1,2&2)	=>3
4(3&1,2&2)	=>2		
5(4&1,3&2)	=>2	5(4&1,3&2)+6(5&1,4&2,3&3)	=>5
6(5&1,4&2,3&3)	=>3		
7(6&1,5&2,4&3)	=>3	7(6&1,5&2,4&3)+8(6&2,5&3,4&4)	=>6
8(6&2,5&3,4&4	=>3		
9(6&3,5&4)	=>2	9(6&3,5&4)+10(6&4,5&5)	=>4
10(6&4,5&5)	=>2		
11(6&5)	=>1	11(6&5)+ 12(6&6)	=>2
12(6&6)	=>1		

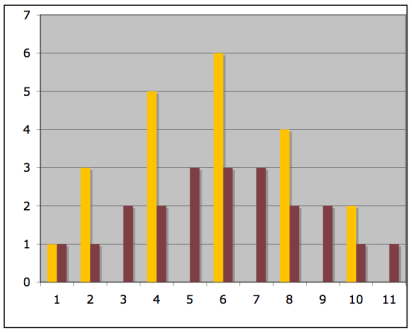


Figure 4.1 Saxophone and Violin Odds

Given the chart above, I refined the scope of the game, diversity and saturation with compositional aims catered to optimize a solo expressivity; **Sections V.2** and **V.3** describes my methods. With the dice game, I match music arguments with dice roll probabilities. On the other hand, the neural network matches music with patterns. The dice game is a tool performed manually while the neural network is digital. Their nuances will be elaborated in what follows and in the context of my composition.

V. *Solos* (*Dice Game)* Composition Morphology

In the following **Sections V.1, V.2** and **V.3**, I describe the *Solos (Dice Game)* composition (dice game or non-dice game models) in the order: clarinet, saxophone violin, and *Main* ensemble *Section* featuring solo improvisations, **Section VI.2**. My strategy for selecting this order is based on the degree to which a solo instrument influences the audio synthesis, the music notation is open to interpretation and the music is determined from the dice probabilities.[15]

Each discussion first considers the music morphology followed by the *Neural Network* audio synthesis and performance practice. These sections embody many compositional concepts. I have widened the scope of the saxophone solo (**Section V.2**) to introduce a new idea to converge the neural network with the dice game and audio synthesis. For each solo, a critical procedure involves configuring the key patterns the user feeds the neural network. The specifics for the conclusion are described in **Section VI.1** and **VI.2**. The conclusion broadens the violin solo and neural network discussion with the *Main Section* of the composition. Table 6.1 summarizes all my key patterns intended for a particular performance of the composition.

[15] The final ensemble music or *Conclusion* that proceeds after the *Main Section* is composed with a similar pointillist material found in the introduction and interludes one, two and three that bind *Solos* together. The motivation for combining ensemble music with solos reflects the dichotomy between the individual and the group common to many genres of musical styles.

V.1 Clarinet Solo (Non-*Dice Game* Model)

V.1.1 Clarinet Composition Morphology

For the clarinet solo, I selected three types of technique from my sketches and layered each semi-randomly together in a single score system.

These techniques include:

1. Pitch-bending:
A short (shaded) and longer stem note (non-shaded) indicating notes' relative values were drawn on 2 5-line staves. The lower-staff, 3 lines, indicates, the player to select a relative pitch, low, middle or high. Next, the player micro-bends the chosen pitch further up or down as shown in the upper-staff, 2 lines. In a micro-bend the player jumps to the new pitch by executing a narrow portamento.

2. Multiphonics:
Similarly, I plotted stemless shaded notes on 2 5-line staves. The upper staff specifies a relative spectrum (bright or dark), and the lower staff, an ideal register (low, middle or high). The spectrum should converge progressively to the register and timbre as indicated in the score. For example in bar 1, a simultaneous low and dark, multiphonic appears and changes to middle and bright; next, the brightness remains, while the player attempts to center the multiphonic progressively toward the lower or bottom frequencies. In bar 11 and 16, the timbre inflection appears before the range, the player generates any dark multiphonic and gradually focuses the spectrum towards a lower frequency. In the process the spectrum itself may somewhat change. All multiphonics are relative and different. The performer has to coordinate the upper and lower line notes in-or-off phases changing the spectrum brightness, depth and volume.

3. Microtonal melodies:
The third lower staff represents a stricter notation derived from 1.

I arranged the above in a wave fashion such that no systems repeat and the music changes linearly from one system, pitch-bending to the next system, either multiphonic or microtonal melody: 1-2-3-2-1-3-2-1-2-1…

V.1.2 Clarinet Audio Synthesis and Solo Composition

As explained in **Section III.2**, the neural network pitch-class vector output is mapped to sinusoids of amplitude 0.0 to 1.0.[16] A pitch-class can be in any octave. The user establishes a range (i.e. clarinet lowest note and subjective highest) in the *Main Conductors'* interface (numbered III in figure 3.3) and triggers a button that spreads the sinusoidal pitch-class frequencies randomly within this range.

The clarinet sound pulsates a filter resonator bank whose arguments are frequencies, amplitudes and decays specified in Hertz. The neural network supplies the frequencies, pitch-classes, amplitudes and the virtual pitch (VP) conductor, the decays. The virtual pitch module estimates each pitch-class weight overall correlation (see footnote 6) based on lowest MIDI pitch-class, harmonic series, major and minor key context. For the clarinet music, I selected the lowest pitch and harmonic series, turning off the major and minor key context.

The Max/MSP, CNMAT Resonances object allows additional spectral controls, for scaling sinusoids: amplitudes evenly; decay; transposition; and adding inharmonic frequencies. I established three presets for the clarinet solo: the frequencies band; non- scaled; and scaled sinusoids. The last preset is noisier and nasal. During performance the user re-randomizes the frequencies and changes the spectrum back and forth from the original to the scaled. The user triggers any three settings (random frequencies, non-scaled or scaled) when a clarinet system changes

[16] CNMAT/MSP spectral synthesis externals: sinusoids and resonances object.

(micro-bending to multiphonic) or at the end of any phrase involving two or more systems.

V.2 Saxophone Solo (Dice Game)

V.2.1 Saxophone Composition Morphology

The saxophone solo belongs to a series of pieces composed by probabilities *Mozart Dice Game*. Any musical material can be allocated to eleven available slots pair of dice combination. (**Section IV.2**). My game consisted of generating eleven musical cells (see saxophone score, cells labeled 1 to 11). Each cell was first composed reducing the pair of dice odds to 6 previously explained in **Section III.2** and obtaining six rolls in the order of moves listed in Table 5.1 and as needed per notes determined from the first roll (3 notes; 3 pitches; 3 accidentals; 3 octave). The accent and dynamic apply for the entire note sequence.

Table 5.1 Saxophone solo cells (dice game) generation

Odds(roll)	1(2)	2(3&4)	3(5&6)	4(7&8)	5(9&10)	6(11&12)
I. Number of notes	1	2	3	4	5	7
II. Pitch	b4	c5	d5	e5	f5	g5
III. Accidental	x	#	n	b	(#/b)	microtone
IV. Octave	2+	0	+	0	-	0
V. Accent	>	.	^	ST	…	Slur
VI. Dynamic	P	F	FFF	Cresc	Dim	PPP
Articulation signs are described in the score.						

For cell no.8, the player improvises an event that he or she can remember. On next appearance, the player recalls the same improvisation as closely as possible.

Once I completed the process of composing cells, I rolled the dice to determine which of the cells would be first transformed. The dice determined cell no. 3. I recomposed a new cell no. 3 from the Table 5.1 above. I rolled a pair of dice to select the fragments, from cells no.1 to no.11, replacing no.3 with the new one, and continued the same way, ending when all cells transformed at least once. I imposed some limits to disallow the game to become long and repetitive. If I rolled a transformation for cell no. 8, the player improvises another event. The new improvisation is indicated in the score labeled *Event 2* enclosed in the square. For musical reasons, I altered an entire unpractical articulation's dice probable occurrence, like ST=slap tongue for a more manageable one of unlikely probability such as strong accent (>) that would not alter the essence of the game.

V.2.2 Saxophone Neural Network Application

In this section of the piece, I present an application of the neural network that deviates from sinusoid amplitude mappings. I will reiterate this methodology when I describe the violin solo. After, extensively playing the *Mozart Dice Game*, my approach to the neural network changed. There are 3-neural network outcomes that are likely to occur with the key pattern the user specifies (for example C major = C major) including an additional one that enforces the fact that an input pattern will always yield the same analysis.[17]

[17] Neural network key patterns can be selected from the live input pitch-class set . Once the neural network has been trained, the weights are static. The neural network ability to match a unique pattern for a given gesture is what motivated me to use this technology for music composition.

1. Perfect matches
2. Probable matches
3. Fuzzy matches
4. Consistencies

I concluded that the neural network resembles the *Mozart Dice Game*. Both automated compositional methods (manual and machine) converged in one. I reinterpreted the dice game using the neural network. The music composed would yield likely odds of 1, 2, 3 listed above and always 4. The neural network functioned as a way to reconstruct the solo dice game music probabilities back to map the audio synthesis. In short, the dice game reinforces the neural network probability matches, processing the performer's input and finding out a corresponding pattern. The neural network vector output, in turn, triggers and supplies changing values to the synthesis parameters.

In this new conception the neural net can be thought of a probability matrix for pattern generation. A note (C) selected by a dice roll yields a neural network chord output dice probability, such as E-Bb-A. Note (C) has a frequency dice-chances pattern. Another note (C#) will yield another neural network chord (perhaps F-B-Bb) and alternates its dice probability of appearance with note (C). Each note selected dice probability can generate one kind of neural network chord probability (based on the neural network own static pattern). But both dice and neural network probabilities complicate when the pitch input patterns presented to the neural network increase. Furthermore, the neural network can generate an infinite range of patterns.

In the process of questioning my own motivations, I realized the dice game could embody a hypothetical case application of the neural network's idea.

The dice game is about statistic or probability choices and so the neural network. At some point my objective became to merge the two together. I first realized of their potential convergence as I was developing the saxophone solo and from this point, it became easy to dissociate pitch to more abstract approaches mapping the neural network output vector to audio.

V.2.3 Saxophone Audio Synthesis and Quantizer Conductor's Module Application

The approach I formulated in the preceding section appears to work well in composition. The saxophone audio synthesis relies on probabilistic dice game odds to generate MIDI sounds. The synthesis incorporates a quantizer's module modified to randomize a sampled melody in real-time according the neural network output vectors.[18]

A melody consisting of eleven pitches and note values, selected by the user, is stored in the program. This melody is replicated ten times. Each replication links a different trigger point (amplitude > 0.0) in the note sequence of the melody. The neural network (amplitudes) vector maps a pitch-class index to the melody MIDI velocity and playback (figure 5.1).

[18] The quantizer's program records 11 *inter-onset-intervals* (IOI =>time between two adjacent events) and the corresponding pitch, and compares ratios between consecutive durations and rescales their relationship to define a common denominator to use an underlying unit of pulse (Rowe 2001, 112-118).

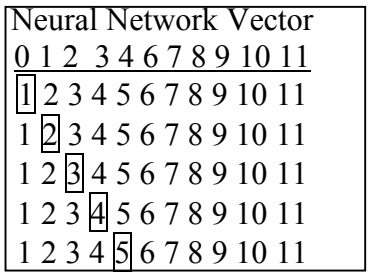


Figure 5.1 Trigger melody points

The neural network pitch-class 0 to 11 triggers the sequence of quantized melodies note 1 to 11, with exception of pitch-class 5 (*Main Section* pedal tone).[19]

An example of how the quantizer's module works, a major triad neural network match C-E-G will trigger and start the melody notes at 1-4-7. The remaining note sequences are played each with a MIDI velocity (in this case 1-1-1) set from the neural network amplitudes vector. If the same triad is matched within a short period of time, it will trigger the melody on these same notes even if the previous notes have not finished their course. The amplitudes for the running melodies get skewed from the last recorded MIDI velocities. At any time, the user can record a new melody or let the continuous mode melody sampling, and choose a different tempo for their playback. In my application, the first note melody sequence runs the fastest (eleven notes) and the last note value is seven times longer (Figure 5.2). All other trigger points in the melody note sequence speed are scaled in between the fastest note value multiplier and longest.

[19] The number of pitch-class is not equal to the number of available quantizer note units.

The quantized melody is scaled to the series below:

Note	1	2	3	4	5	6	7	8	9	10	11	
Speed	0.25	0.5	0.66	1	1.5	2	3	3.33	5	5.5	7	x IOI quantized duration

Figure 5.2 Quantizer, 11 tempos scale division 1/16-1/8-2/3-1-3/2-2-3-10/3-5-11/2-7

V.2.4 Saxophone Solo Composition and Synthesis Network

In performance, the improvised cell no.8 event is recorded. The playback melodies are turned on; performing in the way the module was designed, with the rest of the cells. When a new no. 8 event changes, the player improvises a new melody recorded over the old quantized melody. Cell no. 8, *Event 1* has several appearances. If the improvised events are near each other, I simply left the continuous sampling melody mode on. The process is described in figure 5.3. At the same time, I triggered different key patterns to feed the neural network and changed the pitch-class set or window in real-time from tetrachord to single and eleven pitches. The key pattern neural network output pattern matches the identity of the input pattern selected.

Improvisation Cell no.8	Quantization	Window
Event 1.1	Event 1.1	Tetrachord
Event 1.2	Event 1.2	
Event 1.3		
Event 1.4	Event 1.4	
Event 2.1		
Event 2.2	Event 2.2	Eleven + tetrachord
Event 2.3	Event 2.3	Single + tetrachord
Event 3.1	Event 3.1	
Event 3.2		
Event 3.3	Event 3.3	Eleven + single
Event 4.1	Stop	None
Event 4.2		

Figure 5.3 Quantization of events for saxophone solo

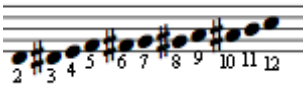
V.3 Violin Solo (Dice Game)

V.3.1 Violin Composition Morphology

The violin solo music is composed of two large sections. Each section contrasts two timbres. Section-1 plucked/bowed tone rows unfold 3 fields.

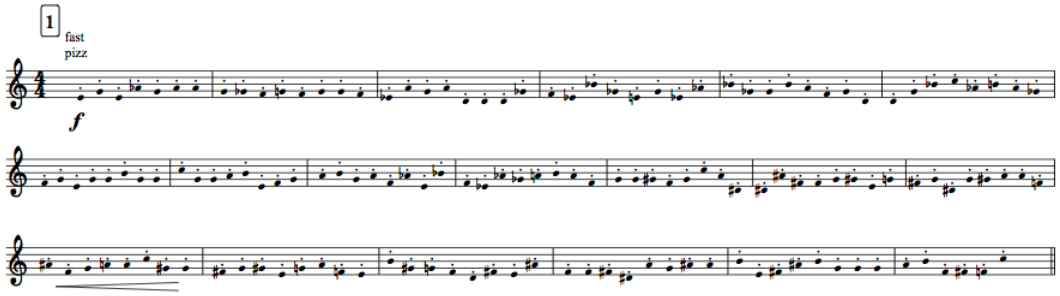
Field-1, Pizzicato=>Field-2, Arco=>Field-3, Pizzicato + Arco

Field-1 Pizzicato was composed rolling dice to select pitches from the scale below:



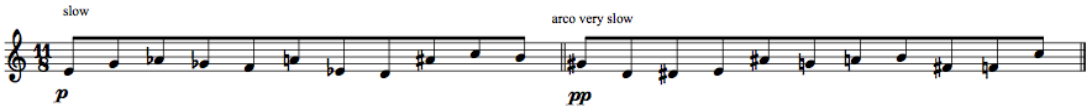
Example 5.1 Field-1 scale

I rolled one dice to determine a limit of how far I should go on before ending a game. For example, I rolled a 5. I'd roll the dice to select notes. I ended once every pitch in the scale appeared at least 5 times. Rolls 6-7-8 have a higher frequency probability to populate their pitches compared to 4-5-9-10 beyond five times, and a tendency to retard the spread of tones progressing on the way towards the lower 2-3 and upper 11-12 boundaries (Example 5.2).



Example 5.2 Field-1 dice rolls

I added musical commentaries to the tone row above. A twelve-tone series traced each new pitch generated from dice rolls (Example 5.3.1).[20] The order of pitches in Example 5.1 is rearranged in the order of likely dice probabilities first occurrence. A game begins or ends with each note of the scale's first or last appearance. I refer to these 12-tone alike series as the section-envelope signatures. A signature can be in the retrograde (prime) form, tracing backwards another twelve-tone series (Example 5.3.2). These signatures are always bowed which means that Field-1 (Pizzicato) shortly changes to Field-2 (Arco). The tones row (Example 5.2) and 12 tone-scale (Example 5.3.2) ending Field-1 have been discarded from the final score version. The remaining sections in Field-2 and Field-3 are composed basically the same way.



Example 5.3.1 Field-1 first 12 tones

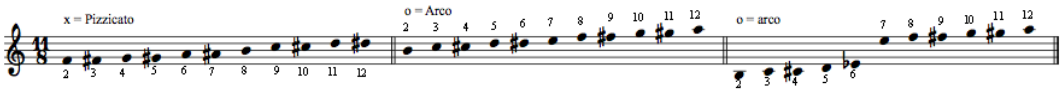
Example 5.3.2 Field-1 last 12 tones in retrograde

After the opening Field-1 envelope signature, bar 2 introduces Field-2 (Arco) with a chromatic scale rising from E4 (dice roll 2) to D5 (dice roll 12) (Example 5.1 transposed by a major second). I rolled one dice (returned 6) to set a threshold for the least amount of scale-pitch repetition to end the round. Rather than filling-in any pitch occurrence beyond 6, I substituted rests. This process ensures that each tone appears 6 times only. While the relation between pauses and scale tones can be predicted from the roll's odds, the shape emerging from each operation cannot.

[20] The dice slot combinations can accommodate 11 tones, not 12. I am using the idea that there are just enough dice slots to resemble a 12 tone-series. Missing one or 2 pitches from the row does not alter the serial procedures significantly.

In Field-2, Field-1's opposite emerges. The least likely odd pitch 2-3 and 11-12 dice roll will generate lounger rests. The more likely odd rolls will pack middle pitches of the scale quickly. The entire tone row progresses from a high-density mass in the center, fast pulsation to sparse, prolonging and accentuating fixed pitches and register towards the extreme points of the scale (E4 to D5). At some point, it became pointless to continue rolling dice. I added long fermatas above ending notes to finish up the required repetition and embellished the target highest pitch with envelope 12-tone signature grace notes.

Each time I refined a compositional process, I added a degree of complexity. Field-3 Pizzicato and Arco combination pitches were selected from a 3 scale partition:



Example 5.4 Field-3, 3 scales partition, 2 x Field-2 + 1 Field-1

I proceeded the same way I did for Field-1 (x=Pizzicato) and Field-2 (o=Arco) (see score). I notated each dice roll resultant pitch in a linear succession alternating x-o-x-o-x-o… mixing Field-1 and 2, and ended when their occurrences satisfied an initial single roll 6-3-3 for scale 1,2 and 3. The first scale plucked field populates the entire Field-3 as shown in figure 5.4, the overall shape of the solo. The second bowed field occupies 1/3 of Field-3. At that point I decided to introduce a third bowed scale that by chance approximates the 2/3 remaining to end plucked Field-1 tone row. The layering of the scales is asymmetrical due to the extended rest in Field-2 and Field-1 opposing dice odd strategies.

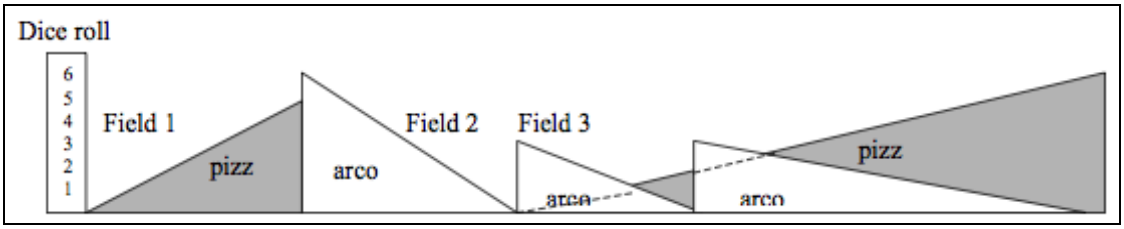
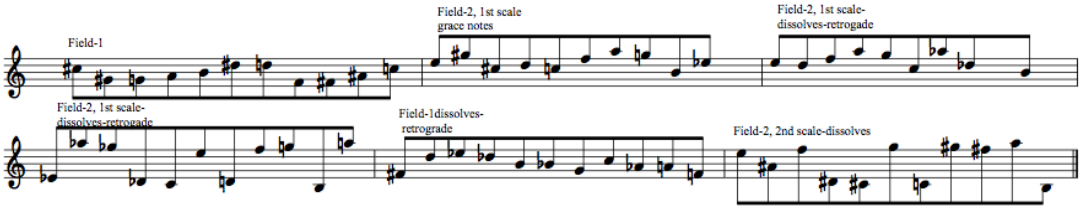


Figure 5.4. Violin solo overall shape

The 12-tone series envelope signature appears as introduction, conclusion and broad grace notes (summarized in Example 5.5).



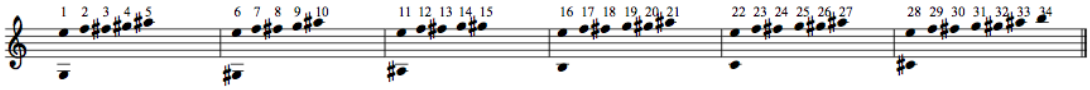
Example 5.5.Serial-rows in field-3

In addition I made musical commentaries by overlapping one field bowed into another plucked, mixing both fields.

Section-2 of the violin solo unfolds two fields:

Field-1 is a quadruple stop strummed upward=>Field-2 quadruple stop bowed fast with pitch permutations. I catalogued all quadruple open strings arpeggio possibilities and I used the dice game to select their sequences. The process resembles the original *Mozart Dice Game*, where each quadruple stop is numbered according to three sub-tables. Table-1 catalogues quadruple stops based on adjacent open string. The upper string aligns a partial chromatic scale (to a fourth/tritone) with one scale degree, lower string ascending line (Example 5.6). An octave generated with the lower or open string above is discarded. For Table-1, I generated a total of 103 possible intervals. Table-2 (64 intervals) continues the same process, skipping one

open string (i.e. D-E). Table-3 (34 intervals) skips two open strings leading to one possibility G-E. Example 5.6 shows the shortest Table-3 (34 intervals):



Example 5.6 Quadruple stops Table-3

I calculated a cell-grid that optimizes the total number of intervals per table columns and dice probability row 2 to 12. Table-1, 103 intervals, 11(dice rolls) x 10 columns = 110 cells table. For the 7 additional cells, I substituted a non-available interval for a rest. Rows are the rolls and columns are the time signatures for successive beat and cell numbered quadruple stop selection. For example, in figure 5.5: a 4/4 measure; 1 = first beat; 2 = second beat; and so on. If I rolled 2 for the third beat, the figure shows (X) = a rest. I tossed dice rolls pairing with the 103 numbered intervals for Table-1 randomly in the grid cells (similar to figure 5.5). The other Table-2 (64 intervals) followed the same way and a variation of Table-3 (34 intervals) for Field-2 (bowed arpeggio) appears in figure 5.5. I played the game twice, and devised the music into two large sub-sections, Field-1 (strummed arpeggio) and Field-2 (bowed). Each field repeats a number of cycles with the same pattern 10/4 (2 x 5/4) correspond the 1st table selection, 6/4 the 2nd table and 4/4 the 3rd table:

	Grid-cells
Field-1=>6 cycles	Table-1=>5/4-5/4=>Table-2=>6/4=>Table-3=>4/4
Field-2=>3 cycles	Table-1=>5/4-5/4=>Table-2=>6/4=>Table-3=>4/4

Example 5.7 Quadruple stop selection order

For Field-2, I modified the grid cell arrangement to four quadrants (figure 5.5). The table numbered arpeggios were tossed in their sequence over 4 cell groups: A (row numbered 2 - column 1 to row numbered 6 - column 2); C (7 - 3 to 12 - 4); B (all X = rest); and D (7 - 1 to row numbered 12 - row 2).

	1	2	3	4
2	1	7	x	x
3	8	2	x	x
4	3	6	x	x
5	10	4	x	x
6	5	9	x	x
7	33	34	11	18
8	31	32	13	20
9	29	30	15	22
10	27	28	21	16
11	25	26	19	14
12	23	24	17	12

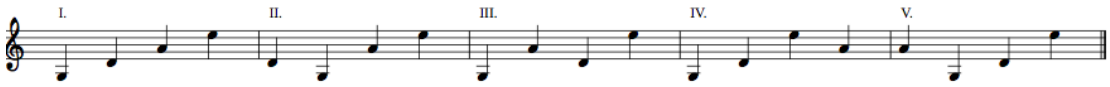
Figure 5.5 Field-2, Table-3 dice rolls and
beat numbered tetrachord (refer to Example 5.6)

To select Field-2 order of pitch permutations, I reduced the dice odds further (explained in the context of the saxophone):

	I.	II.	III.	V.	VI.
Roll	2 3	4 5	6 7 8	9 10	11 12
Odd	2	4	9	4	2

Figure 5.6 Pitch permutation dice odds

I rolled one additional dice pair to select a permutation:



Example 5.8 Open strings permutation

In the final score version, I devised a Field-3 made up of 6 cycles, by combining 3 cycles from Field-1 with the 3 cycles from Field-2. Figure 5.7 shows my scheme: shaded areas = strummed; and blank = bowed arpeggios:

Figure 5.7 Section 2, Feld-3 scheme combining Field-1 (strummed) and Field-2 (bowed)

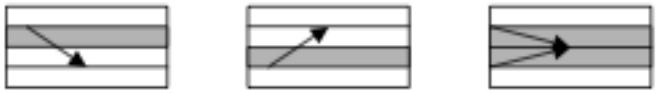
V.3.2. Violin Audio Synthesis and Solo Composition

The violin solo uses two types of audio synthesis: granular synthesis; and Glisson. Both methods are described in Curtis Roads (2001) *Mircrosound* (86–91; 121–123). The Max/MSP application emulates Road's cloud generator program. For generating a cloud, the vector from the neural network maps granular parameters, such as grain (with Gaussian envelope) duration, density, band, loudspeaker spatialization, and reverberation.

Table 5.2 Sinusoidal grain cloud thresholds

1. Grain band range: 0–15000 Hz (random grains)
2. Grain frequency: 30–7000 Hz (if this frequency is 7000, and the band 15000, the randomized band will be 8000 Hz above 7000. Compressing randomized grain in a single frequency (eventually, the random generator yields the same) produces interesting sustained sounds and beats modulation
3. Grain density: 5 to 200 grains
4. Grain duration: 50 to 150 ms
5. Stereo field left pan: 0–127 MIDI value
6. Stereo field right pan: 0–127 MIDI value
7. Reverberation: 30–127 MIDI value

In a Glisson cloud, a grain travels with a short glissando (grain duration) from the grain-band (random grains) towards some frequency (offset). Grains will travel down for a grain-band set in the high register and frequency offset in low (figure 5.8.1). If the frequency offset is set in the middle of the band, random grains within the spectrum will travel both ways toward the frequency offset (figure 5.8.3).



1. 2. 3.

Figure 5.8 grains glissando direction-band to frequency offset

Table 5.3 Glisson cloud thresholds

1. Grain band range: 200–15000 Hz (random grains) 2. Grain frequency: 400–10000 Hz (if this frequency is 7000, and the band 15000, the randomized band will be 8000 Hz above 7000. Compressing randomized grain in a single frequency (eventually, the random generator yields the same) produces interesting sustained sounds and beats modulation 3. Grain density: 5 to 150 grains 4. Grain duration: 60 to 200 ms 5. Stereo field left pan: 0–127 MIDI value 6. Stereo field right pan: 0–127 MIDI value 7. Reverberation: 30–127 MIDI value 8. Offset: 20–1000 Hz frequency glissando ending

The above thresholds were found experimentally to avoid audio clicks and speaker noise distortion. During performance the user triggers: 1. grain cloud for Field-1 (pluck notes/strummed arpeggio); 2. Glisson cloud for Field-2 (bowed) and both synthesis methods for Field-3 combining plucked and bowed notes/arpeggios. During an envelope signature dividing a Field-1 section and a Field-2, the user turns off both audio modules.

VI. Conclusion

VI.1 Violin Neural Network Configuration

The user must specify an output and input key pattern relationship for the neural network (see **Section III.3**). This pattern is transposed in all twelve keys. There are several possible means to feed learning patterns to the neural network:

1. The user can select a pitch-class vector happening in real-time. The output pattern is the same pattern matching the identity from the live input pattern (saxophone solo, **Section V.2.4**).
2. The user enters a separate pattern for input and output (using a piano style interface, figure 6.1). Their patterns may have no relationship to each other or determined randomly.
3. The user combines both methods 1 and 2 (live input with user selected output pattern or vice versa).

Figure 6.1 Neural Network keyboard input style

Sinusoids amplitude, pitch-class, neural network key patterns depend on well-defined contexts:

1. Music theory: chord progressions and resolutions; identifying a key (Rowe 2001, 98–101).
2. Harmonization: embellishment of a note with 3rd and 7th bebop style comping; or enhance a chord with extended dissonances (#7, b/#9, suspension, 13, etc.).
3. Timbre harmonization: fundamental; and overtones combination.
4. Extract structures from composition or player's performance.
5. 12-tone serial aggregates, inversion, retrograde…

In contrast, the dice odds convergence with the neural network does not depend on the contexts listed above.

These contexts can be any random input and output key patterns. The saxophone solo still relies on some degree of context dependency, whereas the violin does not. For this reason, transparency and a high degree of variation among emerging patterns enhances the audio synthesis. All patterns are continually turned on and lead quickly to the same sound saturation if the neural network output fires continually all pitch-class. This idea of low entropy is central to the composition and neural network configuration. It is possible to extract learning key patterns directly from the compositional structure both in real-time and before hand to bring about a unity among components, digital synthesis and in performance. The analysis is doubtful. The structure exists in a state of flux. Another game composition or clarinet solo performance will lead to an entirely different piece and sound.

My approach to identify and delineate musical regions (i.e. Field-1, 2 or 3) is intuitive, and intuition does not always agree with theory. The digital score consists of selecting:

1. The pitch-class set or window for parsing pitches
2. A key pattern for the neural network
3. The pitch-class frequency band
4. Audio synthesis

The sounds generated can be independent of my specific choices for the above.

Pitch-class:	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.00
Granular Cloud	Minimum	Band	Density	Size	Left Pan	Right Pan	Reverb					
Granular Glisson	Offset				Right Pan	Left Pan		Minimum	Band	Density	Size	Reverb

Figure 6.2 Granular synthesis, parameters and pitch-class mapping

I implemented a visual tool to aid the user in making intuitive choices. The process I describe here is best worked out beforehand with the music and performers.

First, I made an audio file from the violin score. The audio file is pitch-tracked, parsed and analyzed for the sum of each input and output vector. A sum does not identify an actual vector trend. Finally, a graph plots the history of the sums.

For example, I configured the neural network with a major triad identity-learning key pattern and tetrachord pitch-class set (window of pitch parsing, figure 6.3.1). I made a second graph the same way with random patterns to compare the general curve shape with the first graph (figure 6.3.2). The green curve is the input, the red output. Using the graph's information to select patterns, the user can infer the neural network effect input (stable) and output (changing) rising or sinking trends in the course of the piece. The graph below shows the dramatic contrast, using a simple pattern, emergence, versus a random (complex) pattern saturation.

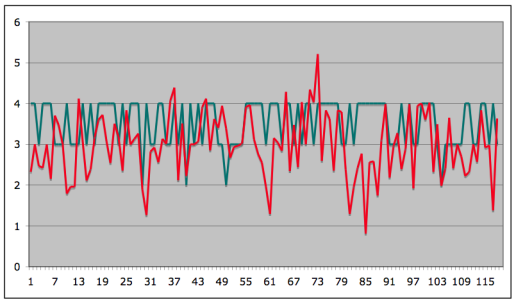


Figure 6.3.1 Triad identity vector

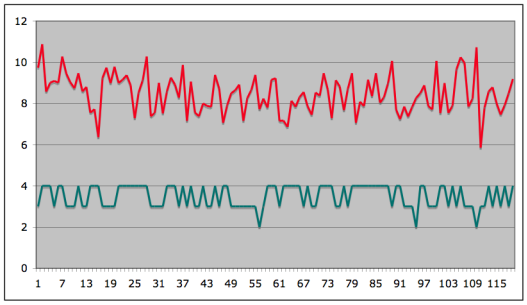


Figure 6.3.2 Random patterns vector

VI.2 *Main Section* and Overall Neural Network Configuration

An objective of the *Main Section* is to unify the individual solos and their audio synthesis by the means of ensemble music. In the *Main Section* the performers are given the opportunity to improvise or reflect on prior solos. I determined a chord symbol for the player to improvise from pitches in a collection based on a major and

minor third interval sequence. These sequences vary in length and their pitch collections vary in size. Many chord interpretations can be deduced. I composed a series of vamps from the pitches spelled in these chords. Each vamp loops while an improvisation goes on. The players agree upon the number of loops. Improvisations and vamps divide one pedal tone (F4) region from another. To summarize, the *Main Section* contains the sustained central pedal tone pitch F4 (Table 2.1), Mozart's quotations (**Section IV.1)**, and improvisations.

In the *Main Section*, each improvisation recapitulates the audio syntheses corresponding to that soloist (**Sections V.1.2, V.2.3, V.3.2)**. In Rehearsal Five, the saxophone and clarinet improvise together with their audio synthesis combined. Sinusoidal synthesis accompanies the trombone improvisation. This same synthesis reappears during the cello improvisation. I used Miller Puckette's (1998) Bonk MSP objects to perform live spectral analyses. The average of Bonk eleven bins spectrum breakdown change the delay time of Max/MSP Comb filter object.

The trombone solo and the cello solo overlap before the *Main Section*. There, the trombone solo has no audio synthesis and the cello is amplified in one channel delayed by a second on the other channel loudspeaker. The performers begin together and proceed independently of each other. At some point, the cellist completes his or her solo and the trombonist is left alone to finish up. The *Main Section* begins immediately after.

The *Main Section* improvisation vamps still recall some pointillist techniques featured in the interludes. The interludes arpeggiate like a wave, a major 7^{th} intervals

sequence; in the *Main Section* improvisations, the vamps arpeggiate pitches from the chords based on the pitch collection of third interval sequences. There are many connections among themes just as there are connections among modules in the digital *Conductors'* network.

	1.	2.	3.	4.	5.	6.
Rehearsal mark	2	5	8	9	12	14
Chord change	D13sus	B13#9sus	A13b5b9sus	D13sus-A13b5b9sus	Eb1/2dim b9	B13#9sus-Eb1/2 dim b9
Instrument	Vln.	B.Cl.+Sax.	Trb.	Sax.	B. Cl.	Vlc.
Spectral range	none	Orch.	Trb.	none	none	Vlc.
Audio synthesis	Granular	Resonance + MIDI	Sinusoid	MIDI	none	Sinusoid

Figure 6.4 Improvisation chords and syntheses

Solo compositions and improvisations in the *Main Section* digital attributes are preset in the main *Conductors*' interface and interface for processing audio (figure 3.3 and 3.4). Presets include the pitch window for parsing the input sound pitch-class vector, the audio band in which pitch-class frequencies are distributed randomly, and the neural network patterns. Figure 6.4 shows the different range listed in the Spectral range row. The violin and saxophone synthesis do not require any frequency bands. In rehearsal mark twelve, the clarinet improvises with no accompanying digital audio. Table 6.1 lists all the neural network key pattern configurations and window pitch-class set analysis. Finally the *Main Section* improvisation presets in the table shows all the main features described in this section. These include the thirds interval sequences and pitch collections, chords and vamps.

Table 6.1 Neural Network and window pitch parser configurations

Clarinet solo presets:

Neural network key pattern :

Window, see **figure 3.3** box 2: **Hexachord**

12;												**Pitch-Class**
C	C#	D	D#	E	F	F#	G	G#	A	A#	B	**Transposition**
1.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.000	0.000	0.000	0.000	0.000	0.000;	input set 0
0.000	0.000	0.000	0.000	0.000	0.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**;	output set 0

Violin solo presets:

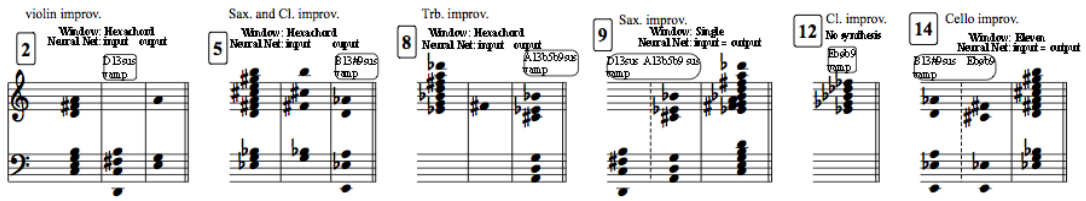
Neural network key pattern:

Window, see **figure 3.3** box 2: **Tetrachord**

12;												**Pitch-Class**
C	C#	D	D#	E	F	F#	G	G#	A	A#	B	**Transposition**
1.000	0.000	0.000	0.000	**1.000**	0.000	0.000	**1.000**	0.000	0.000	0.000	0.000;	input set 0
1.000	0.000	0.000	0.000	**1.000**	0.000	0.000	**1.000**	0.000	0.000	0.000	0.000;	output set 0

Saxophone solo presets: See **Section V.2.4**

Main Section improvisations presets:



VIII. Bibliography

Bailey, Derek (1993), *Improvisation Its Nature and Practice in Music* (New York: Da Capo Press).

Brooks, William (2002), *Music II: from the late 1960* in *The Cambridge Companion to John Cage*, ed. Nichols, David (Cambridge, UK: Cambridge University Press). 128–147

Chadabe, Joel (1997), *Electric Sound the Past and Promise of Electronic Music* (Upper Saddle River, N.J.: Prentice Hall).

Cogan, Robert and Escot, Pozzi (1976), *Sonic Design the Nature of Sound and Music* (Englewood Cliffs, N.J.: Prentice-Hall).

Cope, David (2005), *Computer Models of Musical Creativity* (Cambridge, Mass. MIT Press).

Gjerdingen, Robert (1994), *Apparent Motion in Music?* In *Parallel Distributed Perception Musical Networks and Performance*, ed. Griffith, Niall and Todd, Peter M (Cambridge, Mass.: MIT Press). 141–173

Griffith, Niall and Todd, Peter M. (1999), *Parallel Distributed Perception Musical Networks and Performance* (Cambridge, Mass.:MIT Press).

Hedges, Stephan A. (1978), *Dice Music in the Eighteenth Century*, Music and Letters, Vol 59 (2), 180–187.

Heffley, Mike (1996), *The Music of Anthony Braxton* (Westport, Conn.: Greenwood Press).

Holland, John H. (1998), *Emergence from Chaos to Order* (New York: Basic Books).

Hunt, Andy and Wanderley, Marcelo M. (2002), Mapping performer parameters to synthesis engines, Organized Sound, 7 (2), 97–108.

Leman, Marc (2008), *Embodied Music Cognition and Mediation Technology* (Cambridge, Mass.: MIT Press).

Levitin, Daniel J. (2006), *This is Your Brain on Music the Science of a Human Obsession* (New York, N.Y.: Dutton).

Llinás, Rodolfo R. (2001) *I of the Vortex from Neurons to Self* (Cambridge, Mass.: MIT Press).

Lovine, John (1998), *Understanding Neural Networks* (Indianapolis: Prompt).

Mandel, Howard (2008), *Miles, Ornette, Cecil Jazz Beyond Jazz* (New York: Routledge).

Minsky, Marvin Lee (1986), *The Society of Mind* (New York: Simon and Schuster).

Mulder, Axel G. E. (1999), *Design of virtual three-dimensional instruments for sound control*, (Bibliothèque Nationale du Canada).

Nicholls, David (2002), *The Cambridge companion to John Cage* (Cambridge, UK: Cambridge University Press).

Puckette, M. S., & Apel, T. (1998). *Real-time audio analysis tools for Pd and MSP* (UCSD).

Roads, Curtis (2001), *Microsound* (Cambridge, Mass.: MIT Press).

Rowe, Robert (1993), *Interactive Music Systems : Machine Listening and Composing* (Cambridge, Mass.: MIT Press).

Rowe, Robert (2001), *Machine Musicianship* (Cambridge, Mass.: MIT Press).

Stockhausen, Karlheinz (1989), *Towards a Cosmic Music* (Longmead, Shaftesbury, Dorset: Element).

Tingen, Paul (2001), *Miles Beyond the Electric Explorations of Miles Davis, 1967-1991 (*New York: Billboard Books).

Todd, Peter M. & Werner, Gregory M. (1999) *Frankenstein Methods for Evoultionary Music Composition.* In *Parallel Distributed Perception Musical Networks and Performance*, ed. Griffith, Niall and Todd, Peter M. (Cambridge, Mass.: MIT Press). 311–339

Whitmer, T. Carl (1934), *The Art of Improvisation* (New York: M. Witmark & Sons, Dept. of Standard and Educational Publications).

Wilson, Frank R. (1998), *The Hand How its Use Shapes the Brain, Language, and Human Culture* (New York: Pantheon Books).