Machine learning as meta-instrument: Human-machine partnerships shaping expressive instrumental creation

Dr. Rebecca Fiebrink
Department of Computing
Goldsmiths, University of London
r.fiebrink@gold.ac.uk

Introduction

The practice of building new musical instruments is predicated on the recognition that instruments matter: that the sort of music one can make with a xylophone is different than with a violin, which is different still from the music one can make with a computer. Instruments differ by more than just their sound qualities; acoustic instruments bring with them particular physical affordances, and these lead to idiomatic playing styles and repertoires.

The goal of many designers of digital musical instruments is to discover new idioms for expression, shrugging off old constraints of physical materials and acoustics. Each new configuration of sensors and sound synthesis algorithms patched together by software suggests a new way of being played.

Just as the instrument shapes the music that may be played, the tools for instrument creation shape the instruments that may be built. And just as each instrument demands its player develop a particular set of physical skills and musical knowledge to become competent, each instrument creation tool demands the cultivation of certain technical skills and ways of thinking in its users.

In this chapter, I will discuss how machine learning algorithms can shape the design of new instruments. Machine learning algorithms can facilitate new types of design *outcomes*: they enable people to create new types of digital musical instruments. But, I will argue, they are also valuable in facilitating new types of design *processes*, allowing the instrument creation process to become a more exploratory, playful, embodied, expressive partnership between human and machine. And these qualities of the design process in turn influence the final form of the instrument that is created—as well as the instrument creator herself.

My aims in this chapter are: (1) to provide readers new to these ideas an introductory understanding of how supervised learning algorithms can be used to build new digital musical instruments; (2) to demonstrate that supervised learning algorithms are valuable as design tools, bolstering embodied, real-time, creative practices; and (3) to argue that, because the nature of any new musical instrument is intimately tied to the process through which it was designed, a closer attention to the relationships between

instrument builders and instrument creation tools can deepen our understanding of new instruments as well as point to opportunities to design both new instruments and creative experiences.

New Instruments

Mappings and Mapping Creation Tools

Wanderley and Depalle (2004) use the following basic modular structure (illustrated in Figure 1) to frame discussion of the design of digital musical instruments: First, a gestural controller (or other sensing component) senses the actions of the performer(s); this may include custom sensing hardware, a microphone, a camera, biosensors, and so on. These sensors pass a real-time stream of data to a "mapping" component, which is typically a software program. This component determines how to control the parameters of a sound production component, based on the values of the sensors. The sound-making component might be controlled with low-level musical parameters (e.g., amplitudes and frequencies of sinusoidal components, filter coefficients, or physical modelling parameters) or higher-level ones (e.g., determining the tempo or style of an autonomous agent).

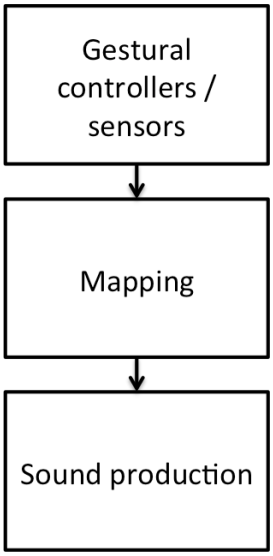

Figure 1: Components of a digital musical instrument

In acoustic instruments, the relationship between a performer's actions and the sound of an instrument is dictated by physics, but there are few constraints on how digital musical instrument mappings might link these together. The design of the mapping determines, in the words of Hunt et al. (2002), "the very essence of an instrument": it defines the ways a performer may move or act, the dimensions of musical engagement that are possible, the means for an audience to perceive the relationship between a performer's intention and the music, and so on. Designing a mapping can thus be understood as designing a space of musical possibilities, and a number of instrument builders see this process as one of musical composition, where the outcome is a system that "carr[ies] as much the notion of an instrument as that of a score" (Schnell and Battier 2002).

Currently, computer programming is the de facto tool for creating an instrument mapping. Programming allows the creation of any imaginable mapping, in theory—just as a Theremin allows one to play nearly any imaginable melody, in theory. However, the practice of programming strongly encourages the creation of certain types of instrument mappings and discourages others. It is easiest to program mappings in which each sensor input controls a single sound synthesis parameter, and in which each synthesis parameter value is likewise impacted only by this single sensor; Hunt and Wanderley (2002) term such configurations "one-to-one mappings." Furthermore, it is easiest to program mapping functions that are simple (e.g., linear) and deterministic. The easiest instrument to build is therefore often analogous to a mixing desk: a set of independent sliders, each with an easy-to-reason-about control mechanism wherein the usable range of the sensor is mapped onto the useful range of a single sound control parameter.

This type of mapping naturally supports particular types of interactions between performer and instrument at the expense of others. Problematically, Hunt and Kirk (2000) found evidence that such simple, "one-to-one" mappings may present barriers to effective musical use when compared with more complex mappings. They found that mappings in which multiple dimensions of input affected multiple sound parameters simultaneously—"many-to-many" mappings—were more engaging to the user, offered more effective control over complex tasks, facilitated more effective learning of the interface over time, allowed people to think about sound gesturally, and were sometimes even considered to be more fun.

Researchers have developed various approaches to facilitate mapping creation through means other than programming, and a number of these approaches make it easier to create complex and many-to-many mapping functions. This work includes a variety of mathematical approaches to function generation, including matrix operations (Bevilacqua et al. 2005), interpolation (e.g., Garnett and Goudeseune 1999; Bencina 2005), and machine learning, which I discuss in the next section.

Human-computer interactions with digital instruments: Control versus partnership

The idea that "mappings" are a useful concept for framing the design or analysis of digital musical instruments is not without its detractors. For instance, Chadabe (2002) is critical of the paradigm, as it assumes a one-directional, simplistic relationship between human and instrument where the aim is control by the human over the sound. To employ a fixed, deterministic mapping function can be seen as ignoring the true potential for digital instruments to facilitate truly new forms of music making. Instead of taking advantage of computers' capacity for complex, non-deterministic processes, employing a static mapping function underutilizes the computer as simply a means of mimicking acoustic instruments, "to make the performer powerful and keep the performer in complete control" (Chadabe 2002).

In this chapter, I argue that the act of composing the instrument, like Chadabe's vision for the act of performing with an instrument, presents opportunities for new forms of relationships between humans and computers. The machine learning approaches I will discuss next create deterministic mapping functions that might be lacking interest on their own, at least in Chadabe's assessment; however, they support a rich dialogue and journey of co-discovery between human and machine throughout the process of

creating a mapping. This process may unfold for months or years before a performance, or it may happen live on stage, making the mapping-building process a performative instrument in its own right. In either case, the quality of relationship between human and machine in the instrument composition process has significant aesthetic and practical consequences, as I will discuss.

Machine Learning and The Wekinator

Supervised learning algorithms are capable of learning functions from examples. An instrument mapping can be understood as such a function, whose inputs are sensor readings and whose outputs are sound synthesis parameter values. An algorithm can learn this mapping from a set of training examples, where each training example contains one set of sensor readings, paired with the set of sound synthesis parameter values that the designer would like to produce when those sensor readings are seen during performance (Figure 2).

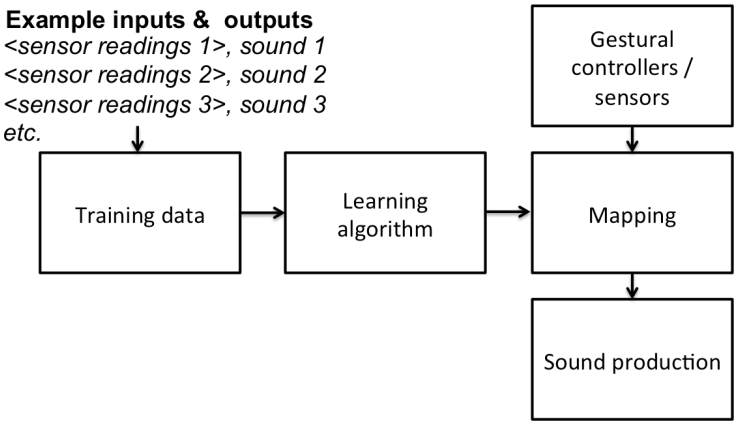

Figure 2: A supervised learning algorithm can create a mapping from a set of training examples.

Different learning algorithms employ different strategies for learning a function from the training examples. However, the learning process can be roughly characterized as finding a mapping function which, upon seeing input values similar to those in a given training example, tends to produce output values similar to those in that training example.

Supervised learning has been used to create mappings for new musical instruments since the early 1990s. Neural networks—a type of supervised learning algorithm—were used by Lee, Freed, and Wessel (1991) to control the timbre of synthesised sound using a MIDI keyboard, and by Fels and Hinton (1995) to control speech synthesis using a sensor glove.

In 2008, I began to build a general-purpose machine learning tool that could be used by composers[1] to create a variety of new digital instruments. By that time, seventeen

[1] In this chapter, I use the word "composer" to refer to people who build new musical instruments and create customized controller mappings, rather than referring to them as instrument builders or musicians. This word choice reflects an understanding of instrument building as an act of musical composition (cf. Schnell and Battier 2002, discussed above). It also accommodates the fact that there

years after Lee and Wessel's experiments, many composers had laptops which could easily train neural networks in a few seconds (or even faster). They had a wealth of sensors and game controllers, as well as fast audio and video feature extractors from which to obtain information about performers' actions. They had easy-to-use communication protocols such as Open Sound Control (Wright and Freed 1997) to patch these sensors to powerful, real-time sound synthesis software such as Max/MSP. However, composers did not have access to easy-to-use machine learning software tools. Outside of music, toolkits such as Weka (Hall et al. 2009) were beginning to make it easier for people without extensive machine learning expertise to experiment with off-the-shelf machine learning algorithms, using graphical user interfaces (GUIs) that did not require computer programming. However, general-purpose GUI toolkits such as Weka did not typically support real-time applications such as music performance.

I named my real-time machine learning toolkit Wekinator, in honor of Weka's achievements in making machine learning accessible to wider groups of users, and also because Wekinator used Weka's implementations of several learning algorithms. Fundamentally, Wekinator is a tool for building mappings like those in Figure 1. In real-time performance, Wekinator receives input values from sensors or other input sources via Open Sound Control (OSC) messages, and it sends output control values to any sound synthesis program (or even animation program, game engine, etc.) via OSC. Wekinator provides a GUI for recording new training examples, training supervised learning algorithms (including neural networks and linear and polynomial regression for creating continuous mapping functions, as well as other methods), running trained models, and configuring various aspects of the machine learning process (e.g., specifying which sensor values will be used in computing each one of the synthesis parameters).

Interactive machine learning as design tool

In most conventional machine learning applications, the goal of using machine learning is to build an accurate model from the set of training examples. For example, the goal might be to build a model that predicts whether a medical treatment is likely to be effective for a new patient, using a training dataset with information about previous patients (the model function inputs) and the efficacy of the treatment on them (the model function's output). The set of training examples is often assumed to be fixed, and much of the human work of applying machine learning focuses on finding the algorithm that most accurately models the patterns in the given training set. Typically, the human practitioner relies on established quantitative metrics in order to compare alternative models and choose the best.

A composer using supervised learning to build a new instrument is faced with a very different type of application. She most likely does not begin the design process with an appropriate training set in hand—she must build a training set from scratch, creating examples that encode her understanding of how performer gestures or actions will be mapped to musical control parameters. While quantitative metrics may be helpful in assessing whether a model has accurately captured the patterns in the

---

may not be a clear or consistent distinction between the notions of instrument, "preset" or mapping, and composition. For instance, at least two of the composers discussed here (Dan Trueman and Laetitia Sonami) have used the same controllers or sensors to play different musical pieces, but designed a different gesture-to-sound mapping for each piece.

training data, these metrics cannot always reflect all of a composer's priorities for a trained mapping (Fiebrink et al. 2011). For instance, she might want the mapping to provide access to a range of sounds that fits the desired aesthetic of the piece, and to make these all accessible using a set of performer gestures that are comfortable to perform; or perhaps she wants to create a mapping that is easy (or difficult) for a performer to learn to play without making undesired sounds. The composer therefore cannot rely only on quantitative metrics of how well a model fits the training data to know whether a mapping is any good, or whether one alternative is better than another; she must use other means to evaluate a mapping, such as experimenting with it herself and listening to how it responds to her actions.

If a composer is dissatisfied with the mapping built by a supervised learning algorithm, changing the training examples is often an effective way to improve the mapping. For instance, if she wants a particular sound to be more easily playable using her mapping, she can provide additional training examples, pairing that sound with easy-to-demonstrate performer gestures, then retrain to build a new model. If she is unhappy with the outcome, she can delete those training examples and replace them with different ones. When supervised learning is used to build new musical instrument mappings, the training examples act as the conduit through which a composer communicates her intention to the computer. In more conventional machine learning applications, however, changing the training data is not a reasonable action to take to improve a model, because the training dataset is assumed to be a (more or less) accurate representation of some phenomenon in the world. This is the case with the medical treatment prediction example above, where the dataset recording treatment outcomes for previous patients is a valuable source of information about the problem domain.

For these reasons, Wekinator's user interface is designed to facilitate certain interactions between humans and supervised learning algorithms which are not part of more conventional machine learning processes: Users can create new training examples in real-time, by demonstrating performer actions along with the sound synthesis parameters they would like to be associated with those actions. Users can evaluate trained mappings by hands-on experimentation, observing how the mapping changes the sound as they change the input values. Users can iteratively add and remove training examples, and seamlessly move between these phases of editing data, re-training, and evaluating the effects of changes they make to the mappings (Figure 3). This type of approach to machine learning in which a human user steers model behaviours through iterative and strategic changes to the training data is often called "interactive machine learning" (Fails and Olsen 2003).

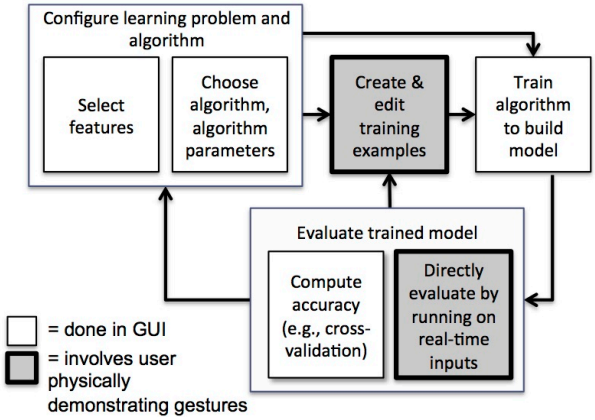

Figure 3: Interactive workflow with Wekinator

Machine Learning as Design Tool

In the eight years since developing Wekinator, I have observed it being used to create new instruments by dozens of professional composers, computer music and computer science students, "hackers" and "makers," and people with disabilities, and I have also used it in my own compositions and performances (Figures 4–6). Previous publications describe how I have used participatory design processes and surveys (Fiebrink et al. 2010), workshops (Katan et al. 2015), interviews (Fiebrink 2011), analysis of software logs (Fiebrink et al. 2011), and reflection on my own work (Fiebrink et al. 2009) to understand how people use Wekinator and why. This work all suggests that the most important benefits of Wekinator pertain to the way that it changes the design process, facilitating the creation of new kinds of instruments while also making design accessible to new people.


Figure 4: Laetitia Sonami plays the Spring Spyre, an instrument she created with Wekinator (2015)

Figure 5: The Sideband ensemble performs Anne Hege's composition *From the Waters*, in which Wekinator was used to create several GameTrak-controlled instruments

Figure 6: *Nets0* was one of the first pieces written for Wekinator, and it requires performers to train new mappings for their own controllers live on stage.

Speeding up implementation of complex mappings

One of the most immediately apparent benefits of using Wekinator to build mappings is the speed and ease with which composers can build a new instrument and modify it. Once the sensors or input devices are sending data to Wekinator via OSC, and a sound synthesis program is ready to receive control messages from Wekinator, the process of training a machine learning algorithm to create a mapping from input values to sound can take as little as a few seconds. This is true even for complicated, many-to-many mappings (the default type of mapping created by Wekinator) in which each sound control parameter is influenced by many input dimensions in possibly non-linear ways. Thus, using supervised learning encourages the creation of mapping types that have been shown to be more engaging, learnable, and controllable than those that are easiest to create using coding (Hunt and Kirk 2000).

Supporting prototyping and exploration

Reducing the time it takes to create a viable instrument does not necessarily mean that composers using Wekinator spend less time building instruments. Instead, composers I have observed typically use their time to make many different variants of an instrument. They iterate many times, making slight or dramatic changes to the training data, as well as to the input devices and the sound synthesis software. Sometimes, these iterations are attempts to fix a problem with the mapping or otherwise improve the instrument according to a clear set of criteria. In these cases, changing a supervised learning model via changes to the training data can be a much faster way to fix a mapping or adapt it to a change in input or sound synthesis, compared to changing manually-written programming code.

However, these iterations are often the result of the composer intentionally exploring alternative designs in an effort to better understand what sort of instrument he really wants to make and how to make it. Prototyping and iterative refinement are recognized as activities that are critical to design in any domain (Resnick et al. 2005; Buxton 2010). Prototypes are physical manifestations of design ideas, and experimentation with a prototype helps a designer better understand the merits of the idea as well as potential ways to improve it. By reducing the time and effort needed to instantiate a prototype for a new idea, Wekinator encourages prototyping and allows composers to explore more ideas, and more refined ideas, over the process of building an instrument. In contrast, several composers I surveyed described how creating instruments by writing code often led to them using instruments they were unhappy with: changing a design using code incurred enough time and effort that they were discouraged from exploring new ideas, and they chose instead to accept instruments that limited them in problematic ways.

Supporting surprise and discovery

Creating an instrument can be understood as an example of what design theorist Horst Rittel (1972) described as a "wicked" design problem: the definition of the problem (What sort of instrument should I make? How will it be played, and what sort of sounds will it produce?) is not known in advance. It is only by designing the instrument that the problem becomes clear: the final instrument design embodies both the composer's final understanding of what the goals of the design process are, as well as the method of achieving them.

Composers using Wekinator to build instruments have often intentionally used machine learning in ways that will help them refine this "problem definition," to evolve their understanding about what kinds of instruments are possible to build, and what kind of instruments they ultimately want. A common strategy for a composer creating a new mapping with Wekinator is to "sketch out" the rough boundaries of the gestural and sonic space using the initial training dataset, then discover what sounds and gesture-sound relationships the supervised learning algorithm builds into the mapping trained from this dataset. A composer can construct this first training set by choosing a set of sounds she thinks she might want to play using the instrument, and a set of different input actions that span a comfortable range of control, then pair these together in a small number of training examples. A mapping created from these examples immediately allows the composer to discover new sounds that might exist in between and beyond the input values (e.g., gestures) she placed in the training set. When using this strategy, experimenting with the resulting mapping is really a process

of discovering unexpected sounds and behaviors, rather than "testing" whether the mapping has learned the "right" behaviors from the given training examples. One Wekinator user described his rationale for this process thus: "There is simply no way I would be able to manually create the mappings that the Wekinator comes up with; being able to playfully explore a space that I've roughly mapped out, but that the Wekinator has provided the detail for, is inspiring."

Wekinator's support of interactive supervised learning allows composers to edit training examples to modify the mapping in response to the discoveries they then make. When a composer discovers a new sound she likes, she can reinforce this sound in her instrument by adding new training examples with this sound into the training set. When she discovers a sound she doesn't like, she can change the training examples in that region of the input gesture space to correspond to a more favorable sound.

Having access to surprise and discovery can fundamentally change the way a composer understands their relationship to the computer as well as the qualities of the instrument that they build. In particular, professional composers who have used Wekinator in their work have described how it allows them to move away from a paradigm of control over a computer into one where the computer is a collaborator. Laetitia Sonami, who has been using Wekinator for five years in the development of the Spring Spyre (Figure 4), says in a lecture about her use of machine learning:

> "…in a way, you don't want the instrument to perform like a well-trained animal circus, you kind of want it to be a little wild, and you want to adapt to it somehow, like riding a bull... I think the machine learning allowed more of this…fun of exploring, instead of going 'I have to have a result right away, this thing is going to do that,' and then leaving it at that. This… allows for a kind of flexibility that I think is essential for artists and musicians to… open up some kind of unknown and really create… things that excite you. I'm not sure about exciting the audience, but actually hopefully exciting the person who's making it, at least! And then you hope that it gets conveyed." (Sonami 2016)

Supporting embodied design practice

Another critical difference between designing instruments using machine learning and designing instruments by writing code is that composers are able to use their bodies directly in the design process. Instead of reasoning about what sort of movement-sound relationships he might want in an instrument, then deriving a mathematical function that he thinks will facilitate those relationships in a mapping, a composer can simply demonstrate examples of movements and movement-sound pairs that feel and sound right to him.

The ability to draw on embodied understanding of movement and sound in the process of designing an instrument is vitally important to many composers who work with Wekinator; the use of the body changes both the experience of composition and the type of instrument that can be created. Composer Michelle Nagai used the Wekinator to create an instrument, the MARtLET, from a piece of tree bark with embedded light sensors. She describes her experience:

> "I have never before been able to work with a musical interface (i.e. the MARtLET) that allowed me to really 'feel' the music as I was playing it and developing it. The Wekinator allowed me to approach composing with electronics and the computer more in the way I might if I was writing a piece for cello, where I would actually sit down with a cello and try things out." (Excerpt from interview, published in Fiebrink 2011)

Composer Dan Trueman, who used Wekinator to create game controller instruments for his piece *CMMV* writes:

> "With [the Wekinator], it's possible to create physical sound spaces where the connections between body and sound are the driving force behind the instrument design, and they *feel* right. It's very difficult to do this with explicit mapping for any situation greater than 2–3 features/parameters [i.e., inputs and outputs], and most of the time we want more than 2–3 features/parameters, otherwise it feels too obvious and predictable. So, it's very difficult to create instruments that feel embodied with explicit mapping strategies, while the whole approach of [the Wekinator], especially with playalong, is precisely to create instruments that feel embodied. I like to think of digital instrument building as a kind of choreography. Choreographers are hands-on—they like to push, pull, hold their dancers, demonstrate how things should go, in order to get what they want, and the resistance and flow of their dancers in turn feeds back into their choreography. This is quite similar to the approach that [the Wekinator] engenders, and radically different than what explicit mapping strategies [i.e., mappings created with programming] enable." (Excerpt from personal correspondence, published in Fiebrink 2011)

Supporting accessibility

Wekinator allows people to build new instruments without programming. In addition to making the instrument-building process faster for programmers, this means that non-programmers have the ability to create new instruments for themselves and others to perform. As an educator, this has been helpful in teaching students about computer music performance and interaction design. Students can easily explore different designs, start to reason about design trade-offs, and experience the satisfaction of building and performing with a new instrument even if they are not confident programmers (Morris and Fiebrink 2013).

Discussion: Wekinator as Meta-Instrument

I describe Wekinator as a meta-instrument: an instrument for creating instruments (Fiebrink et al. 2009). Like anyone learning a new instrument, users of Wekinator must begin by mastering the fundamental techniques of training, testing, and modifying models, but they soon reach a point where their attention is no longer on the algorithms but on using them to achieve a creative vision. Building an instrument with Wekinator then becomes, fundamentally, a real-time process of self-expression, sculpting a unique space of musical possibilities that will afford creative engagement

by oneself and/or others. In designing this space, just like in performing an instrument, a creator draws on a foundation of established musical practices while also seeking to imbue his work with an individual style, all the while being influenced by affordances of the tool which subtly encourage certain idiomatic ways of working and not others.

Understanding composition tools as instruments—whose affordances are vitally tied to the musical potential of the instruments created with them—invites us to bring aesthetic and philosophical considerations pertaining to the role of computers in musical performance to bear on the analysis and creation of composition tools as well. Composers have written of the value of creating "potential for change in the behaviors of computer and performer in their response to each other" (Moon 1997), of interfaces in which "interaction transcends control" (David Rokeby as described by Rowe et al. 1993), becoming more "like conversing with a clever friend" (Chadabe 1997, p.287) or "sailing a boat on a windy day and through stormy seas" (Drummond 2009).

My work with composers suggests that a meta-instrument that supports these interactive qualities, as Wekinator does, can make the process of composition more engaging and musically satisfying. A meta-instrument that encourages playful exploration and discovery can help a composer navigate the wicked design problem of instrument building, sculpting the instrument to better meet her goals while simultaneously evolving those goals in response to the instrument. When the process of exploration and engagement is physical, rather than abstracted into mathematical functions and programming code, composers are able to engage in tight, *enactive* (Wessel 2006) action-feedback loops which further inform their embodied understanding of the instrument and their own musical aims.

Supervised learning algorithms are not the only computational tools which might give rise to these interactive qualities during instrument building or other compositional activities, and Wekinator's user interfaces are far from the only way to link human creators to supervised learning processes. Alternative approaches might facilitate faster exploration of more diverse instrument designs, or take advantage of additional information that composers could communicate through the body (such as examples of comfortable movement sequences or evocative sounds) without requiring a composer to format these as supervised learning training examples. Particular interaction qualities might be intentionally designed into tools, for instance making the "seas" of interaction even stormier with algorithms that make it difficult for composers to build instruments similar to those they have built before, or that introduce indeterminacy into more aspects of the tool. Those of us who are composers of meta-instruments have many new ideas to explore, ourselves, as we design new spaces of musical interactions for the composers who use our tools.

References

[Bencina, 2005] Bencina, R. (2005). The Metasurface: Applying natural neighbor interpolation to two-to-many mapping. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME),* pages 101–104.

[Bevilacqua et al., 2005] Bevilacqua, F., Müller, R., and Schnell, N. (2005). MnM: A Max/MSP mapping toolbox. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, pages 85–88.

[Buxton, 2010] Buxton, B. (2010). *Sketching user experiences: Getting the design right and the right design*. Morgan Kaufmann.

[Chadabe, 1997] Chadabe, J. (1997). *Electric Sound: The Past and Promise of Electronic Music*. Prentice Hall, Upper Saddle River, New Jersey.

[Chadabe, 2002] Chadabe, J. (2002). The limitations of mapping as a structural descriptive in electronic instruments. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*.

[Drummond, 2009] Drummond, J. (2009). Understanding interactive systems. *Organised Sound*, 14(2):124–133.

[Fails and Olsen, 2003] Fails, J. A. and Olsen, Jr., D. R. (2003). Interactive machine learning. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI '03)*, pages 39–45.

[Fels and Hinton, 1995] Fels, S. S. and Hinton, G. E. (1995). Glove-Talk II: An adaptive gesture-to-formant interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 456–463.

[Fiebrink and Cook, 2011] Fiebrink, R. A. and Cook, P. R. (2011). *Real-time human interaction with supervised learning algorithms for music composition and performance*. PhD thesis, Princeton University, USA.

[Fiebrink et al., 2011] Fiebrink, R., Cook, P. R., and Trueman, D. (2011). Human model evaluation in interactive supervised learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 147–156.

[Fiebrink et al., 2010] Fiebrink, R., Trueman, D., Britt, C., Nagai, M., Kaczmarek, K., Early, M., Daniel, M. R., Hege, A., and Cook, P. R. (2010). Toward understanding human-computer interaction in composing the instrument. In *Proceedings of the International Computer Music Conference (ICMC)*.

[Fiebrink et al., 2009] Fiebrink, R., Trueman, D., and Cook, P. R. (2009). A meta-instrument for interactive, on-the-fly machine learning. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*.

[Garnett and Goudeseune, 1999] Garnett, G. and Goudeseune, C. (1999). Performance factors in control of high-dimensional spaces. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 268–271.

[Hall et al., 2009] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The Weka data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

[Hunt and Kirk, 2000] Hunt, A. and Kirk, R. (2000). Mapping strategies for musical performance. In Wanderley, M. M. and Battier, M., editors, *Trends in Gestural Control of Music*. IRCAM—Centre Pompidou.

[Hunt and Wanderley, 2002] Hunt, A. and Wanderley, M. M. (2002). Mapping performer parameters to synthesis engines. *Organised Sound*, 7(2):97–108.

[Hunt et al., 2002] Hunt, A., Wanderley, M. M., and Paradis, M. (2002). The importance of parameter mapping in electronic instrument design. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*.

[Katan et al., 2015] Katan, S., Grierson, M., and Fiebrink, R. (2015). Using interactive machine learning to support interface development through workshops with disabled people. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 251–254.

[Lee et al., 1991] Lee, M., Freed, A., and Wessel, D. (1991). Real-time neural network processing of gestural and acoustic signals. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 277–280.

[Moon, 1997] Moon, B. (1997). Score following and real-time signal processing strategies in open-form compositions. *Information Processing Society of Japan, SIG Notes*, 97(122):12–19.

[Morris and Fiebrink, 2013] Morris, D. and Fiebrink, R. (2013). Using machine learning to support pedagogy in the arts. *Personal and Ubiquitous Computing*, 17(8):1631–1635.

[Resnick et al., 2005] Resnick, M., Myers, B., Nakakoji, K., Shneiderman, B., Pausch, R., Selker, T., and Eisenberg, M. (2005). Design principles for tools to support creative thinking. In *Report of Workshop on Creativity Support Tools*, Washington, DC, USA.

[Rittel, 1972] Rittel, H. W. (1972). On the Planning Crisis: Systems Analysis of the 'First and Second Generations'. *Institute of Urban and Regional Development*.

[Rowe et al., 1993] Rowe, R., Garton, B., Desain, P., Honing, H., Dannenberg, R., Jacobs, D., Pope, S. T., Puckette, M., Lippe, C., Settel, Z., and Lewis, G. (1993). Editor's notes: Putting Max in perspective. *Computer Music Journal*, 17(2):3–11.

[Schnell and Battier, 2002] Schnell, N. and Battier, M. (2002). Introducing composed instruments, technical and musicological implications. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*.

[Sonami, 2016] Sonami, L. (2016). Lecture on machine learning, within online class "Machine Learning for Musicians and Artists" by R. Fiebrink, produced by Kadenze, Inc.

[Wanderley and Depalle, 2004] Wanderley, M. M. and Depalle, P. (2004). Gestural control of sound synthesis. *Proceedings of the IEEE*, 92(4):632–644.

[Wessel, 2006] Wessel, D. (2006). An enactive approach to computer music performance. In Orlarey, Y., editor, *Le Feedback dans la Creation Musical*, pages 93–98. Studio Gramme, Lyon, France.

[Wright and Freed, 1997] Wright, M. and Freed, A. (1997). Open Sound Control: A new protocol for communicating with sound synthesizers. In *Proceedings of the International Computer Music Conference (ICMC)*.