Neural Audio Synthesis for Non-Keyboard Instruments

Franco Caspe[1]*, Andrew McPherson*[2]*, Mark Sandler*[1]

[1]Centre for Digital Music, Queen Mary University of London, London, UK
[2]Dyson School of Design Engineering, Imperial College, London, UK

***Abstract*—Interaction with musical instruments can take as many shapes as there are musicians and instruments. However, in many production tasks such as music sketching, composition in Digital Audio Workstations, and sound design, instrumental interaction occurs mainly around keyboard interfaces and MIDI. Although this combination is highly versatile, when musicians who play other instruments, such as guitarists or singers, approach these tasks, they lose a dimension of expression as they cannot convey their usual articulations and their rhythmic sense on these interfaces. In this demonstration, we present a neural audio system for sound-based transformation of musical instruments that aims to provide capabilities similar to those of synthesizers for non-keyboard instruments, allowing musicians access to a wide range of high-quality sounds that can be used for sound production, sketching, and live playing. We achieve this in two ways: first, we employ a novel waveform autoencoder that learn control features that can convey the rich expression of musical instruments, as well as the ambiguities and uncertainties of their sound in ways that respect the nature of the source instrument. Furthermore, we run this model using an efficient, custom C++ neural network engine that supports high frame rates, making it able to be used for live playing in current audio production workflows in an audio plugin. We will demonstrate our system by sketching rhythms and melodies of different instruments with a guitar and a microphone, recording and playing them live.**

1. THE PROBLEM: DRIVING SYNTHESIZERS WITH NON-KEYBOARD INTERFACES

Synthesizers are ubiquitous in music production, composition, live use, and sound design. Still, their design enables interaction modalities that are amenable to only a part of music players, i.e., those who play instruments that center the scope of interaction in *discrete key presses* (ON/OFF), such as piano-like keyboards, grid controllers, piano rolls in Digital Audio Workstations, to name a few. We refer to all these as *keyboard* interfaces.

Since its inception, the synthesizer has been closely related to keyboard interfaces [1], which had become the de facto interface for novel ways of producing sound, long before the dawn of electronic music [2]. This greatly influenced the way electronic musical instruments are interconnected, which eventually led to MIDI, a communication protocol designed to exchange discrete key presses between controllers and sound generators. This discrete communication archetype has seen little modification through the years, still being included in MIDI's newer polyphonic expression protocol [3].

The MIDI standard does a great job of modeling the ballistic action of a hammer hitting the strings of a piano. However, for a wide variety of musicians, who sing or play string or wind instruments, instrumental interaction is not of a discrete nature, making them unable to fully leverage the flexibility of synthesizers, and having to resort to audio effects for sound design and playing live, which, when combined are difficult to program and of limited versatility, usually being bound to the timbre of the original instrument.

We argue the difficulty of successfully implementing MIDI in other musical interfaces, as in, for instance, MIDI guitars, which have seen limited adoption, arises from its core building blocks: MIDI's interconnection standard aims to exchange *notes*, which are not an inherent part of a musical instrument, and instead, part of music notation [4]. Musical instruments generate *sounds* rather than notes, and sounds can be messy and ambiguous, have fluctuating sonic characteristics, be too noisy, or too soft, or present ambiguous or multiple pitches, or be produced through different, continuously evolving articulations. This way, using sound as a control source for a synthesizer could be a good idea that can help convey many of the sound variations that can be done on an instrument.

However, when considering such ambiguity and subjectivity of musical instrument sounds, it follows that their characteristics cannot be unequivocally analyzed and digested to be incorporated into explicit MIDI controls to drive a synthesizer. Instead, an approach that tries to do so may end up breaking the natural rapport between instrument and musician: as discussed in Human-Computer-Interaction (HCI) literature, such feature extraction processes can push such uncertainties and ambiguities to a different modality of the interaction, through a process known as ambiguity shift [5].

Given these challenges, we believe that designing an interface that can accommodate a wide variety of acoustic musical instruments requires conveying a big part of their ambiguity of sound *through* the interface, without attempting to inequivocally explain it. This can also preserve the natural relationships that are pre-existing between players and instruments through *meaning-making* [6].

In this demo, we present a novel neural audio system that uses sound as a control source of a synthesizer, in ways that preserve ambiguity and uncertainty, allowing a natural interaction. The system effectively transforms the timbral characteristics of source musical instruments to those of other instruments, in real-time, and with a very low latency of about 10 ms, enabling effective audio production and sound design on non-keyboard instruments, such as guitars, singing voice, or percussion, while being able to play these new sounds live. Musicians trained on such instruments can preserve their proficiency, intuition, and thinking process while leveraging a wider sound palette.

2. SYSTEM DESCRIPTION AND DEMO PROPOSAL

Our neural audio system is based on a novel waveform autoencoder architecture called BRAVE, presented originally in [7], which is designed to minimize input-output latency and thus can allow for intimate control of musical instruments [8]. Such models are then trained with datasets of single musical instruments, which are curated in ways to ensure they convey a wide variety of content, accounting for rhythms, melodies, or chords, depending on the instrument. This ensures the models are capable of interpreting a wide range of articulations, music events, and gestures that could be present in the input.

Next, the trained models are exported to a custom C++ inference library, designed for execution with a low real-time factor even with short audio frames, which yields a working system that we implement in a plugin for audio production and live use. Once the models are trained and exported, audio from a different musical instrument (source instrument) can be fed to the encoder, which generates a latent set of features that are immediately consumed by the decoder, which acts as a synthesizer. The decoded audio can preserve fine-grained temporal characteristics of the input, such as rythms and melodies,

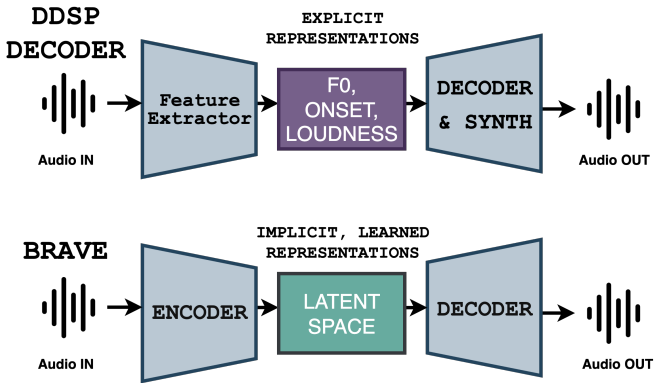

Fig. 1: A comparison between explicit and implicit representations in two neural audio architectures trained with the same objectives.

while presenting new timbre features that are characteristic of the training dataset, such as different transients and a new a new a new instrument identity.

Unlike other interactive audio systems that rely on traditional Music Information Retrieval (MIR) feature extractors to drive a synthesizer, such as fundamental frequency trackers, onset detectors, or loudness extractors [9]–[11], our waveform autoencoder learns to preserve ambiguities and micro temporal articulations of the input instrument through implicit representations in the latent space of the autoencoder. Through data curation, a trained autoencoder can learn a highly expressive representation that, without the need to fully explain a specific event in the audio input, can convey a wide range of expressive actions on the source instrument. A comparison is presented in Figure 1, showing two different neural audio architectures that are used for audio processing of musical instruments, and trained with the same objectives, i.e., re-synthesizing an input signal at the output, namely the DDSP Decoder [12] and BRAVE [7], that use explicit representations from MIR feature extractors and implicit representations from the autoencoding scheme, respectively.

To showcase our models' flexibility, we propose a live demonstration of three use cases, including song sketching, sound design, and live playing, all three driven by an unmodified electric guitar and voice. Visitors can observe how the natural sound of both instruments is transformed in real-time, which allows them to sketch different parts of a song, including drums, bass, saxophone, and piano, and how guitarists or vocalists can easily convey rhythms, melodies, and harmonies with both guitar and microphone, seamlessly transforming these on the target instruments. Please refer to the video accompanying this document, which showcases the live use of the system for both guitar and voice.

Participants are invited to try the system and reflect on its potential. We aim to show how domain knowledge can guide neural network and DSP design, and how a holistic approach that considers instrumental interaction requirements, neural architectures, sources of latency, data curation, and real-time DSP engineering can lead to a solid interactive experience. Interestingly, this is only possible when all elements are considered together.

3. REQUIRED EQUIPMENT

We propose to demonstrate our system at a small table, big enough to hold a laptop, a small USB audio interface, a headphone splitter, and three sets of headphones. A microphone sits also on top of the table, and an electric guitar to the side of it. The first author will demonstrate the use cases with both a guitar and a microphone, inviting visitors to try the system. The item list is shown as follows:

- Table or stand, of at least 70 cm x 50 cm
- Laptop
- USB Audio Interface
- Headphone Splitter
- Headphones x3
- USB-C cable
- Dynamic Microphone
- Electric Guitar
- XLR Cable for Microphone (1 m)
- Plug Cable for guitar (2 m)

In case there is no space for a guitar, which should not occupy that much space since it sits next to the table, and is supported by it, the demo could also be conducted just with one or two microphones, still showcasing the ability of the system to play live, along with other musicians.

REFERENCES

[1] T. Pinch and F. Trocco, "The Social Construction of the Early Electronic Music Synthesizer," *Icon*, vol. 4, pp. 9–31, 1998.

[2] E. I. Dolan, "Toward a Musicology of Interfaces," *Keyboard Perspectives 5 (2012)*, 2012.

[3] Rory Dow, "The ABC Of MPE," https://www.soundonsound.com/sound-advice/mpe-midi-polyphonic-expression, Jun. 2021.

[4] A. McPherson, L. Morrison, M. Davison, and M. M. Wanderley, "On mapping as a technoscientific practice in digital musical instruments," *Journal of New Music Research*, vol. 53, no. 1-2, pp. 110–125, Mar. 2024.

[5] C. N. Reed, A. L. Benito, F. Caspe, and A. P. McPherson, "Shifting Ambiguity, Collapsing Indeterminacy: Designing with Data as Baradian Apparatus," *ACM Trans. Comput.-Hum. Interact.*, vol. 31, no. 6, pp. 73:1–73:41, Dec. 2024.

[6] W. W. Gaver, J. Beaver, and S. Benford, "Ambiguity as a resource for design," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '03. New York, NY, USA: Association for Computing Machinery, Apr. 2003, pp. 233–240.

[7] F. Caspe, J. Shier, M. Sandler, C. Saitis, and A. McPherson, "Designing Neural Synthesizers for Low-Latency Interaction," *Journal of the Audio Engineering Society*, vol. (In Press), 2025.

[8] D. Wessel and M. Wright, "Problems and Prospects for Intimate Musical Control of Computers," *Computer Music Journal*, vol. 26, Dec. 2001.

[9] G. Magenta, "DDSP-VST," https://magenta.tensorflow.org/ddsp-vst, 2023.

[10] V. Verfaille, U. Zolzer, and D. Arfib, "Adaptive digital audio effects (a-DAFx): A new class of sound transformations," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1817–1831, Sep. 2006.

[11] C. Pöpel and R. Dannenberg, "Audio Signal Driven Sound Synthesis," in *Proceedings of the International Computer Music Conference*. Barcelona, Spain: Michigan Publishing, Sep. 2005.

[12] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable Digital Signal Processing," in *8th International Conference on Learning Representations*. Addis Ababa, Ethiopia: ICLR, Jan. 2020.