GrooveTransformer: A Generative Drum Sequencer Eurorack Module

Nicholas Evans*
Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
nicholas.evans@upf.com

Behzad Haki*
Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
behzad.haki@upf.com

Sergi Jordà
Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
sergi.jorda@upf.com

ABSTRACT

This paper presents the *GrooveTransformer*, a Eurorack module designed for generative drum sequencing. Central to its design is a Variational Auto-Encoder (VAE), around which we have designed a deployment context enabling performance through accompaniment and/or user interaction. This module allows the user to use the system as an accompaniment generator while interacting with the generative processes in real-time. In this paper, we review the design principles and technical architecture of the module, while also discussing the potentials and short-comings of our work.

Author Keywords

Real-time Music Generation, Drum Generation, Transformer, Eurorack

CCS Concepts

•**Applied computing** → **Sound and music computing;** Performing arts; •**Information systems** → *Music retrieval;*

1. INTRODUCTION

In recent years, our research has been predominantly focused on the advancement of real-time, performance-oriented deep generative systems for symbolic music [7]. Our main aim has been to design and build lightweight systems deployable as software plugins to allow for a user-friendly interaction with state-of-the-art generative models.

While deploying a symbolic generative system as a MIDI plugin promotes accessibility and ease of use, it also bears inherent limitations that are worth examining. Plugins that are specifically designed for integration within digital audio workstation (DAW) environments, typically have a rigid set of controls and functionalities. Consequently, users may find themselves confined to a defined application scope and unable to experiment beyond the intended use of the plugin. For instance, consider a drum pattern generator that has been trained on human performances to generate "human-like" patterns through intricate manipulations of velocity and micro-timing dynamics of a generated score/pattern [4]. Once deployed as a MIDI plugin, the most conventional use of this generative system is to be paired with a virtual drum kit plugin to generate human-like drum performances. In such a setting, certain aspects of the generations such as voice mappings, velocities, and micro-timings are automatically handled and interpreted by the virtual instrument. Consequently, the user may be predisposed to use the trained generative model only in the context for which it was developed.

Our aim is to challenge this conventional use and inspire a more expansive and exploratory approach to symbolic generative models. Rather than adhering strictly to the originally intended application, we sought to develop a deployment strategy in which the users were encouraged to redefine the musical semantics associated with the generated patterns. By enabling users to define the application of these musical attributes, they can expand the functionality of the generative systems beyond their original intended use cases. For example, in the case of the previously discussed drum generation system, the velocity of events in a drum pattern could be repurposed to control other musical parameters or to influence entirely different aspects of a composition. This new perspective invites users to engage with these generative models in a broader, more creative context than what was originally envisioned. In our pursuit of an expansive, exploratory approach to the application of neural network (NN) based generative models, we found the Eurorack modular synthesis platform to be a particularly effective medium. In this format, every module, connection, and interaction is determined by the user, leading to uniquely curated and patched systems. In using the Eurorack format, each aspect of the generated patterns can be provided to the user as a sequence of discrete voltage gates and control voltages (CV) which the user routes within their system. Subsequently, the user is encouraged to redefine and re-purpose the musical semantics associated with the generated patterns.

In this work, we specifically focused on adapting a previously developed generative system, originally designed to provide symbolic real-time drum accompaniments [7] (see Section 3), to the Eurorack format. Rather than merely cloning the existing system into a hardware format, we re-conceptualized its design to achieve the objectives mentioned above: (1) to foster a more exploratory and user-driven approach in using the drum generation model, and (2) to enable users to re-define and creatively re-purpose the musical semantics of the generated patterns. The culmination of this work is an open source hardware prototype (see Figure 1), constructed according to our revised design principles. This paper discusses our design and adaptation

*Equal contribution

NIME'24, 4–6 September, Utrecht, The Netherlands.

process, highlighting the potential of this approach to harnessing deep generative models for musical expression.

Figure 1: The final developed module

2. RELATED WORK

Before discussing the relevant works, it is important to emphasize that our project did not aim to investigate or optimize the embedding of deep learning models into microcontrollers, a topic thoroughly explored in several studies [9, 15, 13, 3, 14]. Instead, our objective was to develop a conceptual prototype that allows for real-time interaction with our generative model and to understand different hardware deployment strategies. Therefore, in this section, we focus on relevant works that explore interaction in the context of deep generative or machine learning based music models.

As interactive deep generative systems become more prevalent in sound and music composition, we expect to see new paradigms of control and interaction emerge. One of the clearest examples of this is AI-terity, a deformable, non-rigid musical interface that allows for complex and continuous control over a Generative Adversarial Network (GAN) [18]. Bending and deforming the instrument allows the performer to navigate the latent space of the network for real-time audio generation while applying different levels of pressure controls the parameters of a granular synthesizer engine.

An example of an interactive, generative model deployed to an embedded system is the NSynth Super. This device features the NSynth generative model which uses a WaveNet-style autoencoder to learn meaningful representations of instrument sounds from a dataset of 300k notes from 1000 instruments [2]. It uses MIDI control to trigger note events while also incorporating a 4-quadrant X/Y touch pad that morphs the timbre of the generated sound by interpolating the characteristics of four assigned instruments. A similar concept is applied to Neurorack, a Eurorack module that features a real-time generative audio model deployed to a single NVIDIA Jetson Nano board [12]. This model generates impact sounds based on the distribution of seven adjustable descriptors.

The notion of abstracting away traditional methods of composition and control in favor of newly defined perceptual representations need not be confined solely to audio generations and tactile interfaces. A previous study by Gómez-Marín et. al. explores symbolic generations and develops a list of perceptual rhythm descriptors that are used to build a meaningful, navigable rhythm space that interpolates between a collection of drum patterns. Patterns are organized by similarity and modeled according to human ratings [5].

The *Mutable Instruments* Grids applies a similar rhythm space concept but in the Eurorack format. This 3-channel trigger generator features a 2-dimensional map, generated through machine learning techniques, that encapsulates drum patterns extracted from an extensive set of drum loops [11].

Latent Drummer, another symbolic rhythm generation model designed for Eurorack, is a classic 16-step, 5-channel drum sequencer that uses a Variational Autoencoder (VAE) to generate drumming styles, and a set of Markov Models to improvise within the generated style [19].

3. METHODOLOGY

This section will outline the evolution of the *GrooveTransformer* Eurorack module. At the core of our system is a generative Variational Auto-Encoder (VAE) model that we had previously utilized in our research [7]. Illustrated in Figure 2, this model is adept at transforming single-voice rhythmic patterns, containing timing and velocity information, into multi-voice, humanized drum sequences.

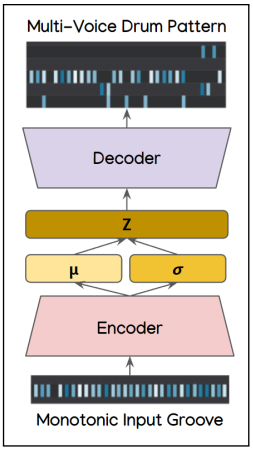

Figure 2: Generative Model

We initially employed this architecture to develop a real-time drum accompaniment system [7]. The system was designed to generate drum patterns that emulated human performance, conditioned by an input groove extracted from real-time instrumental performances. This initial application showcased the model's ability to interact dynamically with live inputs, a feature we aimed to preserve and enhance as we adapted it to the Eurorack format.

3.1 Adaptation for Eurorack

Our previous real-time drum accompaniment system was designed to respond dynamically to a live instrumental performance. This live input from the performer conditioned the model to generate a rhythmically compatible accompanying pattern. This behaviour was designed to simulate a traditional improvisational environment.

In contrast to a performance with traditional instruments in which tight timing and coordination are key, the emphasis of a Eurorack performance is typically more focused on exploring sonic and rhythmic possibilities achievable mainly by applying various methods of automation, control, and modulation. Modules are used to generate sequences, complex waveforms, or control voltages that drive and interact with other parts of the system. Commonly, Eurorack performers use CV to automate or inject randomness into these processes and allow for autonomous and complex behavior to develop.

Although the real-time drum accompaniment model can still be used in a Eurorack system, the inherent difference in interaction modality and purpose between these two systems presented an opportunity to consider how to leverage the generative system in a Eurorack environment. In considering the *GrooveTransformer*, we identified potential use cases of the VAE model's latent space that could expand the system's capabilities beyond its original application. Specifically, we implemented a set of controls that allow the performer to navigate the latent space for the purpose of unconditioned pattern generation rather than solely allowing for conditioned pattern generation as is the case with real-time drum accompaniment.

3.1.1 *Latent Space Interpolation Control*

To facilitate the construction and navigation of the latent space, the interface allows a performer to randomly generate and store two separate drum patterns, each associated with a latent vector, respectively represented by points Z_A and Z_B in Figure 3. Once these points are selected, interpolation between these two points leads to an intermediate pattern (Z_{AB}). As such, the interpolation parameter α shown in Figure 3 allows the performer to manipulate the generated pattern while maintaining a degree of rhythmic similarity between two known patterns Z_A and Z_B.

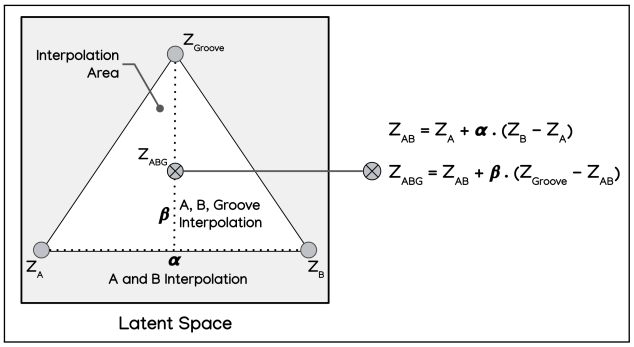

Figure 3: Latent Space Bilinear Interpolation

Similarly, the encoder of the VAE model can be used to encode an instrumental performance into a latent vector, denoted as Z_{Groove} in Figure 3. When this latent point is present, a bi-linear interpolation between Z_A, Z_B and Z_{Groove} will allow for generating an intermediate pattern that is correlated with these points. The amount of interpolation between Z_{AB} and Z_{Groove}, represented by β, determines how closely the system follows the input groove (i.e. the instrumental performance). At maximum ($\beta = 1$), the system acts fully as a drum accompaniment to the input groove ignoring the preset states A and B. On the other hand, at minimum ($\beta = 0$), the output of the system acts independently of the input groove and the output will directly correspond to the value of α. Therefore, by controlling the β parameter, the user can specify whether and how much the generator should follow the instrumental performance.

3.2 Interaction and Control

The *GrooveTransformer* Eurorack module utilizes two distinct control signals, CV and MIDI, each of which influences two primary aspects of the system. Firstly, they can be used to condition the outputs of the generative VAE model; secondly, they can facilitate interaction and communication with external devices. To condition the model with an input groove, as described in the prior discussion concerning the drum accompaniment system, a user can send a sequence of voltage gates (such as those produced by a Eurorack sequencer) to represent note events, as well as a separate CV sequence to represent the velocity for each sequenced event. This pairing of voltage gates and CV is utilized in a similar manner when routing the generated patterns to external devices. In this case, each of the module's output voices has a separate gate and CV output corresponding to the generated sequence and associated velocities. While the user can utilize these output signals however they see fit, a conventional approach would be to use the gate output to trigger an amplitude envelope and to use the CV output to control some parameter that affects the loudness or sonic characteristic of the sound. Alternatively, to allow compatibility with non-Eurorack devices, input grooves and pattern outputs may also be transmitted or received via MIDI. Finally, in typical Eurorack fashion, the *GrooveTransformer* module also allows for the use of CV automation over certain model parameters that are exposed to the user on the module's interface, particularly those that facilitate navigation of the latent space.

3.3 Hardware Design

As previously mentioned, the aim of the current work was not to explore model optimization, but rather to develop a hardware prototype, that would allow us to implement our previously discussed ideas in a real-world Eurorack environment. Therefore, our primary objective was to create a functional and reliable hardware interface that could seamlessly interact with the generative model. To this end, we decided to use dedicated hardware for managing the interface while running the model inference (generation) on a separate Linux-based single-board computer (SBC).

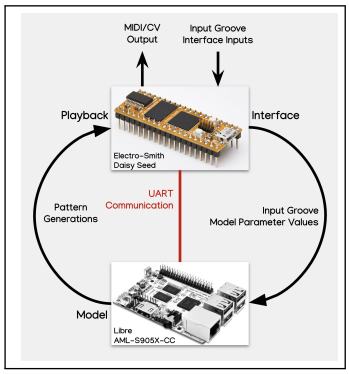

Figure 4: Hardware Communication

To be more specific, the *GrooveTransformer* implements a dual-board approach using both an *Electrosmith* Daisy Seed [1] micro-controller and a *Libre* AML-S905X-CC [8] single board computer. The Daisy Seed, making use of the libDaisy and DaisySP C++ libraries, handles CV and MIDI inputs on the module interface, manages MIDI and CV playback of the patterns, and serves as the master clock. The AML-S905X-CC is used to host the PyTorch model and perform real-time inference. These two boards communicate via a two-way UART connection. Parameter control values are transmitted from the interface (Daisy Seed) and received by the model (AML-S905X-CC). As the model receives new input grooves and/or parameter control values, it transmits newly generated patterns to the module interface. This communication architecture is illustrated in Figure 4.

The benefit of the chosen design was twofold. Firstly, given our limited experience in embedded systems, this setup

allowed for a clear delineation between the model-inference process and the hardware interface. Utilizing a Linux-based *Libre* board provided the flexibility to deploy our model without the complexities of compiling or porting the source code to a specific microcontroller. Simultaneously, the *Electrosmith* Daisy platform, being specifically designed for audio and music applications, offered numerous utilities for managing hardware electronics and handling essential communication protocols such as MIDI, UART, and I2C. This separation of tasks would enable us to focus on each aspect of the system with the appropriate tools and environments, streamlining development and troubleshooting.

Secondly, each Daisy board uses the STM32 micro-controller [16]. *STMElectronics* provides the *STM32Cube.AI* [17] toolbox to convert simple neural networks into optimized code to run on certain STM32 micro-controllers. As such, while not the focus of our initial development, in the future we would be able to explore the development of light-weight simple networks that would be directly deployable on the Daisy Seed board without requiring a Linux-based SBC.

4. DISCUSSION AND FUTURE WORK

The *GrooveTransformer* module is our first attempt at hardware based deployment of generative models. While we have successfully developed a functioning prototype, it is pertinent to acknowledge that certain decisions made along the way notably influenced the trajectory and outcome of our project. In this section, we introspect on these decisions in an effort to guide our future work and to inform similar endeavors that may be undertaken by novice researchers.

One of the primary challenges we faced was the complexity involved in debugging a system built on a two-board setup. This architecture, while offering advantages in terms of performance and specialization, significantly complicates the debugging process. The inter-board communication adds an extra layer of complexity, making it difficult to isolate and resolve issues efficiently. A top priority of any future hardware prototypes would be to deploy the system to a powerful single board computer such as the NVIDIA Jetson Nano [12].

Additionally, we realized that it would be beneficial to design an interface that places a greater emphasis on adaptability. The current interface is not flexible enough to accommodate significant feature changes that may be implemented in future iterations of the model. In the next version of our prototype, we would opt for a more adaptable design, possibly incorporating a touchscreen in conjunction with physical controls. An adaptable framework such as this could facilitate future updates to the model and increase the longevity of the hardware prototype.

Finally, we intend to continue expanding our capabilities for user testing. Initially, we were limited to a single hardware prototype. This meant that end-user testing could only be conducted onsite and our ability to gather a diverse range of user feedback was limited. However, to mitigate this and to facilitate faster prototyping of ideas, we developed a VST plugin version of the module. Albeit in a different environment than the intended hardware deployment, this software version can be used within virtual Eurorack environments or paired with MIDI/CV converters to interact with hardware Eurorack modules. Now, we are working closely with a local musician to test and evaluate our system in a series of live performances. We anticipate that the feedback and insights gained from these performances will play a key role in shaping the future iterations of our model.

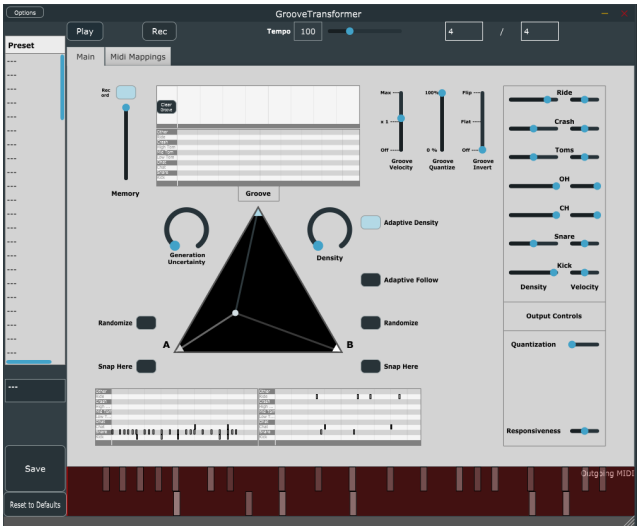

Figure 5: Software implementation of the *GrooveTransformer* module. Developed using the NeuralMidiFx VST Wrapper [6]

5. CONCLUSION

In this paper, we discussed how we adapted an existing generative system to the Eurorack format in a way that is conducive to real-time live performance. We showed that by employing bi-linear interpolation between two pre-selected drum patterns and a pattern associated with a performed groove, users can perform with the generative system in ways ranging from manual manipulation of generation control parameters all the way to generating patterns by providing the system a real-time instrumental performance.

We plan to develop an improved version of this module using a more generalizable interface that would accommodate future versions of the generative engines as well. Also, for the upcoming prototypes, we aim to use a single board setup to improve the integration and compactness of the module. Lastly, the existing module and the future iterations will be tested in performance settings throughout a series of upcoming improvisational live shows.

In order to share our current progress and any future iterations, we have developed a software version of the *GrooveTransformer* module. Source code, documentation and demos of both the hardware Eurorack module and the plugin implementation are available in the following website:

`https://groovetransformer.github.io/`

6. ACKNOWLEDGMENTS

We express our sincere appreciation to Alba G. Caballer for her invaluable contributions to the design of the faceplate and artwork (alba.gonzalez.caballer@gmail.com).

This research was partly funded by the Secretaría de Estado de Digitalización e Inteligencia Artificial, and the European Union-Next Generation EU, under the program Cátedras ENIA 2022. "IA y Música: Cátedra en Inteligencia Artificial y Música" (Reference: TSI-100929-2023-1).

7. ETHICAL STANDARDS

Our research adheres to the ethical guidelines and standards set forth by the NIME Principles & Code of Practice on Ethical Research [10]. Throughout the duration of this project, we have avoided any circumstances that could potentially lead to a conflict of interest.

Recognizing the inherent challenges in the accessibility of the hardware developed for this project, we have developed and publicly shared an open-source software emulation of the hardware module. This initiative is aimed at broadening accessibility for both the wider NIME community and the general public, thereby fostering inclusive participation and engagement.

8. REFERENCES

[1] Electro-Smith. Electro-smith daisy seed. https://www.electro-smith.com/daisy, 2021.

[2] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan. Neural audio synthesis of musical notes with WaveNet autoencoders. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1068–1077. PMLR, 06–11 Aug 2017.

[3] M. Faizan, I. Intzes, I. Cretu, and H. Meng. Implementation of deep learning models on an soc-fpga device for real-time music genre classification. *Technologies*, 11(4), 2023.

[4] J. Gillick, A. Roberts, J. Engel, D. Eck, and D. Bamman. Learning to groove with inverse sequence transformations. In *Proc. of the 36th International Conference on Machine Learning*, pages 2269–2279, California, USA, May 2019.

[5] D. Gómez-Marín, S. Jordà, and P. Herrera. Drum rhythm spaces: From polyphonic similarity to generative maps. *Journal of New Music Research*, 49(5):438–456, 2020.

[6] B. Haki, J. Lenz, and S. Jorda. NeuralMidiFx: A Wrapper Template for Deploying Neural Networks as VST3 Plugins. In *Proceedings of the 4th International Conference on on AI and Musical Creativity*, Sept. 2023.

[7] B. Haki, M. Nieto, T. Pelinski, and S. Jordà. Real-Time Drum Accompaniment Using Transformer Architecture. In *Proceedings of the 3rd Conference on AI Music Creativity*. AIMC, Sept. 2022.

[8] Libre Computer. Aml-s905x-cc. https://libre.computer/products/aml-s905x-cc/. Accessed: 2024-01-26.

[9] A. McPherson. Bela: An embedded platform for low-latency feedback control of sound. *The Journal of the Acoustical Society of America*, 141(5):3618–3618, 2017.

[10] F. Morreale, N. Gold, C. Chevalier, and R. Masu. NIME Principles & Code of Practice on Ethical Research, Jan. 2023.

[11] Mutable Instruments. Grids eurorack module. https://mutable-instruments.net/modules/grids/, 2014. Accessed: 2024-01-26.

[12] NVIDIA. Nvidia jetson. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/. Accessed: 2024-01-26.

[13] T. Pelinski, R. Diaz, A. L. B. Temprano, and A. McPherson. Pipeline for recording datasets and running neural networks on the bela embedded hardware platform. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Mexico City, Mexico, May 2023.

[14] T.-V. H. Rong-Guey Chang. Constructing an ai compiler for arm cortex-m devices. *Computer Systems Science and Engineering*, 46(1):999–1019, 2023.

[15] A.-M. Solomes and D. Stowell. Efficient bird sound detection on the bela embedded system. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 746–750, 2020.

[16] STMicroelectronics. Stm32 32-bit arm cortex mcus. https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html. Accessed: 2024-01-26.

[17] STMicroelectronics. Stm32cube.ai. https://stm32ai.st.com/stm32-cube-ai/. Accessed: 2024-01-26.

[18] K. Tahiroğlu, M. Kastemaa, and O. Koli. Ai-terity 2.0: An autonomous nime featuring ganspacesynth deep learning model. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Shanghai, China, June 2021.

[19] N. Warren and A. Çamci. Latent drummer: A new abstraction for modular sequencers. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Auckland, New Zealand, June 2022.