R-VAE: Live latent space drum rhythm generation from minimal-size datasets

Gabriel Vigliensoni[1], Louis McCallum[1], Esteban Maestre[2], and Rebecca Fiebrink[1]

[1] Creative Computing Institute, University of the Arts London, UK
[2] Department of Music Research, McGill University, Montréal, QC
gabriel@vigliensoni.com

Abstract. In this article, we present R-VAE, a system designed for the modeling and exploration of latent spaces learned from rhythms encoded in MIDI clips. The system is based on a variational autoencoder neural network, uses a data structure that is capable of encoding rhythms in simple and compound meter, and can learn models from little training data. To facilitate the exploration of models, we implemented a visualizer that relies on the dynamic nature of the pulsing rhythmic patterns. To test our system in real-life musical practice, we collected small-scale datasets of contemporary music genre rhythms and trained models with them. We found that the non-linearities of the learned latent spaces coupled with tactile interfaces to interact with the models were very expressive and led to unexpected places in musical composition and live performance settings. A music album was recorded and it was premiered at a major music festival using the VAE latent space on stage.

Keywords: Rhythm, meter, latent space, music visualization, dimensionality reduction

1 Introduction

In this article, we present research on customizing a variational autoencoder (VAE) neural network to play with musical rhythms encoded within a latent space. To allow the encoding of a wide range of rhythms, we implemented a data structure that is capable of accommodating the onsets of rhythms in simple and compound meter while also encoding their velocities and their deviations from the grid—known as *microtiming*. To facilitate the exploration and performance with the rhythmic models, we designed and implemented a visualizer that relies directly on the pulsing rhythmic patterns to provide a dynamic representation of latent spaces. The system comes packaged as an instrument for a popular digital audio workstation and can generate models using minimal training data—as few as one dozen MIDI clips with rhythms. To the best of our knowledge, this is the first time that a network architecture devoted to the modeling of drum patterns has been used to encode rhythms in simple and compound meter, and the first time that a dynamic visualization has been implemented to observe latent spaces learned from rhythmic patterns.

We examined and interrogated our approach to rhythm modeling in real-life music production and performance by collecting a series of small-scale datasets of rhythms, training models with them, and exploring the resulting latent spaces in improvisation and live performance contexts. A music album was recorded and premiered at a major music festival using the VAE latent space on stage. We found that beat making and rhythm generation with latent representations can lead to unexpected and surprising musical places because datapoints in the space are organized in non-linear ways. We also observed that being able to model rhythm representations with little data makes the human influence on the learning process more clear. This can have huge implications in creative practice because performers can understand where they have agency and increase their sense of ownership when training and performing with a machine learning model.

This article is organized as follows: in Section 2 we present background information on musical genre, rhythm, meter, and the modeling of rhythms using machine learning. We also comment on some of the biases that arise when choosing specific data representations and network architectures. In Section 3 we explain the data representation and network topology we use to model a wide range of rhythms while using little training data. We detail the rationale and implementation of the rhythmic latent space visualizer and talk about how it uncovers features of the latent space. We devote Section 4 to explaining how we validate our design and implementation by using it in real-life musical practice. Finally, in Section 5 we offer our thoughts, insights, conclusions, and future work after developing and performing with our system.

2 Background and Related Work

In this section, we present background information and previous research on rhythm generation systems, the concept of meter, and how different genres—both traditional and contemporary—use different meters. We follow on introducing how some machine learning systems have been used to learn compact, latent space representations of rhythm, and how they have overlooked encoding rhythms in compound meter, both due to the datasets and data representation chosen. Finally, we comment on biases introduced when not considering data and data representations able to model diverse music, and the need of using simple network architectures to allow creative practitioners in general, and musicians in particular, to learn and customize rhythmic models.

2.1 Rhythm generation systems

Rhythm generation and beat making are an intrinsic part of contemporary electronic music production. Common tools available to the music producer for generating, programming, or processing rhythmic patterns are trackers, sequencers, and drum machines; but looping systems based on audio or MIDI clips have also become extremely popular. The former set of tools have been part of professional music production for the past 40 years (since the release of the programmable drum machine Roland CR78 in 1978 and the successful sample-based Linndrum LM-1 in 1980), and the latter for the past 20 years (since the release of Ableton Live in 2001). Musicians have used these tools and instruments extensively in all sorts of music genres.

Machine-learning-driven musical systems for rhythm generation have started to appear but are not yet commonly used in contemporary rhythm generation and beat making. There are currently a few standalone and web applications that allow musicians to play and manipulate rhythms using latent space representations, such as Magenta Studio (Roberts et al., 2019), DrumNet (Lattner & Grachten, 2019), Beat Blender,[3] RhythmVAE (Tokui, 2020), and DrumVAE.[4] Within the low-dimensional latent space representations of these applications, the distance of the rhythmic patterns is based on similarities learned from the data through an iterative process of optimization at training time. The resulting space is continuous and (ideally) easy to navigate after reducing its dimensionality, allowing the real-time retrieval and generation of rhythmic patterns. However, these systems are not easily integrated into digital music production workflows (Magenta Studio and RhythmVAE being notable exceptions), they are not really designed for live performance contexts, and they are hard to train, resulting in a lack of customization capabilities for professional use (Gómez Marín, 2018; Roberts et al., 2019). Our goal is to design a generative rhythm system that can be trained with minimal training data, to better enable artists without extensive computational resources to build and explore bespoke rhythm models. To use such a system in contemporary music production and live performance settings, it has to be capable of performing real-time inference, close to real-time training, and be capable of being interconnected to other systems using standard communication protocols such as MIDI or Open Sound Control (Wright, 2005).

2.2 Meter and music genres

We are interested on modeling and playing with contemporary music genre rhythms such as *footwork*, *trap*, *2-step*, *gqom*, *drum and bass*, and *dembow*. The appeal of these rhythms comes from their *tempo octaves*—that is, their actual tempo is ambiguous and can be perceived or embodied at one specific tempo, or half or double that tempo (Schreiber & Müller, 2014). For example, when dancing to a *drum and bass* track, people can embody the music as if the tempo is 160 beats per minute (bpm) or as if it is 80 bpm. In addition, triplets are an intrinsic part of the rhythms, as is easily perceived in *dubstep* or *trap*, for example (Duinker, 2019; Phillips, 2021). This stylistic signature is sometimes so strong, that the *meter* of those rhythms is perceived differently, not just the individual rhythmic elements in triplets.

[3] `https://experiments.withgoogle.com/ai/beat-blender/view/`
[4] `https://vibertthio.com/drum-vae-client/`

The meter is the metrical structure or timing framework by which a series of beats are organized into repeating patterns (Large, 2008). Listeners mentally fit rhythms to meter based on the position of accents, i.e., instances where a note is salient relative to its neighbors (Samuels, 2018). We therefore can understand meter as "a mode of attending" when experiencing rhythms (Gjerdingen, 1989). Meter in which each beat has subdivisions as powers of two, such as in 2/4, 3/4, or 4/4 is called *simple meter*. Meter in which each beat has three or multiples of three subdivisions, such as 6/8, 9/8, or 12/8, is referred to as *compound meter*. London (2012) states that—although there are differences in motion and expression between simple and compound meter—the basic distinction between "twoness" and "threeness" is the essential part of the character of each meter.

The combination of simple and compound meter is the foundation of many African and Latin American rhythms such as *tresillo* and *bembé* (Brandel, 1959; Peñalosa & Greenwood, 2009), but is also an intrinsic part of much contemporary music. Depending on the genre, it is common to hear a simple meter core pattern with one or more instruments doing rhythmic phrases in triplets on top of it. For example, in *trap* music this effect happens usually in the hi-hat patterns—this genre is greatly defined by this effect—and sometimes in the vocal flow. This technique used in filter-modulated bass lines is also a key characteristic of the *dubstep* genre.

Although compound meter and triplets are widely used in different music genres and cultures, several studies on the modeling and generation of rhythm using machine learning have supposed simple meter suffices to represent music. Some researchers have assumed that "the very large majority" of Western music "follows dyadic patterns" and can be partitioned in powers of two (Paiement, Bengio, Grandvalet, & Eck, 2008), and others have taken this binary-normative view of music—even when studying the perception of beats and the complexity of rhythms (Bouwer, Burgoyne, Odijk, Honing, & Grahn, 2018). There are some notable exceptions. The early study by Ellis and Arroyo (2004) found evidence of compound meter is one of the main principal component bases when analyzing a dataset of rhythms encoded in MIDI files of 10 different genres. Later on, Mauch and Dixon (2012) implemented a data representation with a grid granularity of 12 equally spaced divisions per beat span, thus allowing the representation of binary and ternary subdivisions with the same grid. More recently, Oore, Simon, Dieleman, Eck, and Simonyan (2020) made no assumptions about any underlying musical grid or meter and described rhythms solely in terms of their durations and time differences. Colombo (2021) applied domain knowledge to design a meter-normalized rhythm representation to encode 32 different combinations of durations and timeshifts.

Considering all recent advances in the modeling of rhythm, it is unclear why most drum generation and beat making systems based on machine learning have assumed a simple meter grid that is not able to encode drum patterns and rhythms from a large musical repertoire. In the next subsections, we will review some of these systems as well as the datasets that have been used to train them.

2.3 Rhythm and latent spaces

Several recent research projects have aimed at modeling and encoding rhythmic drum patterns using machine learning. Given a set of user-provided examples, their regularities are learned and encoded into a model. Then, the user can sample patterns from the original distribution or generate new, unheard rhythms. For instance, Choi, Fazekas, and Sandler (2016) created a model trained on drum patterns from songs by Metallica to generate new rhythmic sequences. Their approach was based on a text-based LSTM (Long Short Term Memory) network, so they had to adapt and encode the rhythm onsets to fit the data representation. They limited the number of events in a bar to 16 by quantizing the drum tracks to 16th notes in a simple meter binary grid. They trained a model using MIDI files of drum tracks of songs by Metallica, and created drum sequences that were "reasonable rock drum patterns." However, since the data representation they used did not encode the microtiming of onsets nor how hard they were struck—known as *velocity* in the MIDI domain—the resulting patterns were very rigid. Instead of focusing on a specific musical artist or style, Nikolov (2016) trained another LSTM network on drum patterns from a wide range of music genres. This work used professionally produced MIDI loops, increased the quantization grid to a 32nd note, but used only rhythms in simple meter with a 4/4 time signature.

To learn longer-term musical structure, researchers from the Google Magenta team experimented with language modeling and LSTMs to encode and generate melodies and drum patterns. They released network architectures designed to learn representations of melodies[5] and rhythms[6] encoded in a symbolic format, as well as models trained on large datasets containing "thousands of MIDI files" of undisclosed origin. The data representation did not encode microtimings or velocities. Using the same data representations implemented in those projects, and with the goal of modeling sequences with even longer-term structure, the Magenta team released MusicVAE (Roberts, Engel, Raffel, Hawthorne, & Eck, 2018). This network used a hierarchical variational autoencoder (VAE) architecture to encode and generate melodies, drums, and "trios" consisting of a drum part, a bass line, and a melodic line. For these three categories, the Magenta team released models that were trained on a large dataset of MIDI files collected from the web. Roberts et al. reported that MusicVAE was able to generate 2-bar drum sequences reliably, but failed when trying to reconstruct 16-bar sequences.

In the aforementioned approaches to the modeling of drum patterns, only the position of the drum onsets was encoded, not their microtiming or velocity. These two characteristics are important for giving the drum loops a human feel, or groove. To overcome those flaws, Gillick, Roberts, Engel, Eck, and Bamman (2019) and the Magenta team released GrooVAE, a set of models trained on real drum performances that encoded the velocities and microtimings for each onset. The data representation of GrooVAE used a binary grid with a resolution

[5] `https://github.com/magenta/magenta/tree/master/magenta/models/melody_rnn`

[6] `https://github.com/magenta/magenta/tree/master/magenta/models/drums_rnn`

of 16th notes and relied on the microtiming factor to shift the onsets. However, this factor was only able to encode small deviations from the grid resulting in the inability to encode rhythms in compound meter or with rhythmic elements in triplets.

Several applications have been released based on the Magenta binary view of music, its data representations, and pre-trained models. For example, the Neural Drum Machine[7] is a web-based application in which the user seeds the system with a short rhythmic sequence, and the model "imagines" the continuation. Beat Blender packages MusicVAE pre-trained models in a web application in which the user plays back patterns and creates paths in a latent space filled with rhythms. The Drum Beats Latent Space Explorer[8] is another web application that uses a VAE architecture trained on 33K MIDI drum files to learn a bi-dimensional latent space representation that can be explored in a browser. Additionally, the Magenta team bundled several of the applications developed in their research group into Magenta Studio, a suite of applications and models to be used in standalone mode or as an Ableton Live plugin that works directly with MIDI data (Roberts et al., 2019). Although using Magenta Studio directly in Ableton is useful for music production the lack of direct training within the application does not allow musicians to generate bespoke models, which could avert its use in professional and creative contexts. R-VAE, the system we present in this article, permits musicians to train custom models of varying-size datasets.

DrumNet is a project developed at Sony CSL aiming at the conditional generation of drum patterns based on the rhythmic relationships learned from different instruments in multi-track recordings and encoded into a 16-dimensional space (Lattner & Grachten, 2019). The system is trained with audio data but outputs MIDI or audio. Each drum instrument is visualized in a two-dimensional latent space where a heat map depicts the density of training data points in it. All the examples shown in the accompanying site are in simple meter,[9] and there is no explanation of the training data beyond saying that comes from *pop/rock/electro* songs. The application is still being tested and is not open to the public. Finally, M4L.RhythmVAE (Tokui, 2020) is a rhythm generation system that encodes onsets, velocities, and microtimings. It is based on GrooVAE but has a much simpler network architecture for faster training. The author implemented it as an Ableton plugin for easy integration into musical production contexts. We tested the device and realized it had most of the features we were looking for. However, it wasn't able to encode triplets and there was no way of knowing how the rhythmic patterns were distributed in the latent space.

All in all, the assumption that the structure of most music can be expressed as powers of two has resulted in the inability to represent, model, and generate computationally music from significant existing genres.

[7] `https://bit.ly/nikolov-neuralbeats`
[8] `http://altsoph.com/pp/dsp/map.html`
[9] `https://sites.google.com/view/drum-generation`

2.4 Datasets of rhythms

The exclusive use of binary rhythmic representations is also common within publicly available rhythm datasets. For instance, Battenberg and Wessel (2012) collected a dataset of rhythms using an electronic drum. They recorded a large number of rhythmic sequences aiming at having a sizeable collection of popular rhythmic styles. However, the dataset had a strong bias toward rock and the authors quantized all collected data to exact 16th note subdivisions. Mauch and Dixon (2012) performed a corpus study using almost five million one-bar-length files extracted from more than 70 thousand symbolic music files from the web. The authors reported the dataset seems to mostly include *pop/rock*, *jazz*, *classical*, *film music*, and *country/folk*. There is no reference to new genres or traditional music and the authors report that 90 percent of the data was in 4/4, and only 1.5 percent was in compound meter. Since in the research presented in this article we want to model drum patterns and rhythms in simple and compound meter, using datasets of genres with a heavy bias towards simple meter in datasets is not useful.

In terms of dataset size and processing power, most of the architectures, models, and applications we have reviewed have been trained using very large datasets of rhythms. For example, MusicVAE models were trained on about 1.5 million unique MIDI files (Roberts et al., 2018), GrooVAE was trained on the 13 hours of MIDI and audio drumming of the Groove MIDI dataset (Gillick et al., 2019), and the Expanded Groove MIDI Dataset models were trained on 444 hours of music (Callender, Hawthorne, & Engel, 2020). Such approaches to modeling rhythm within very large-scale collections entail practical challenges because they attempt to learn commonalities from very disparate data. For example, learning a single rhythmic model from *rock*, *jazz*, *house*, and *reggaeton* can be challenging because the data may be essentially different, and may require large datasets and processing power to converge to something useful. As a result, creating generic models from large and diverse data can be interesting and reasonable from computational or music-theoretic points of view but this can also hinder customization. One of the costs of small-sample, low-dimension machine learning models is that the resulting latent space is often extremely specific to the source data and may differ dramatically from a more general latent rhythmic space. However, the ability to train models from smaller datasets can enable the modeling of niche genres or even a personal style, while requiring fewer resources for training. This is can be particularly useful for individual creative practitioners that usually do not have access to large datasets or the processing power to train very large models.

Some prior work has enabled musicians to create bespoke supervised learning systems with small training sets. For instance, Wekinator (Fiebrink, Trueman, & Cook, 2009) uses shallow multi-layer perceptron neural networks to learn bespoke mapping functions (e.g., from a performer's gestures to sound synthesis parameters) using small datasets generated by musicians in real time. Dinculescu, Engel, and Roberts (2019) designed a system to control a large and generic latent variable model with a smaller one that is trained quickly. Their approach learns a compressed representation of the latent vectors of MusicVAE and generates musical melodies from only portions of its latent space. The user can steer this process by seeding melodies using a web-based app.

2.5 Comments on biases

There are two big issues in previous work on the modeling of rhythms using latent space representations: (i) all previous approaches were not designed to encode music in compound meter, and (ii) to be trained, most systems have been designed for and trained with massive datasets and large processing power—so training usually takes a very long time and cannot be done at home. "The more data the better," a typical approach in machine learning systems makes sense for some contexts, but when designing systems that provide human creative agency, a small-scale approach can be more meaningful.

These two issues are problematic because they introduce biases. The first issue introduces biases toward particular genres and cultures because only certain rhythm types can be encoded and modeled by the neural network. The second issue introduces biases towards particular types of users because large architectures and expensive datasets only allow some people (or corporations) to train these models, hindering access to customization. We think that choosing a model architecture large enough to sufficiently learn a reasonable representation of small-scale and more manageable datasets is more inclusive of people working in home environments. A smaller scale may lead to better chances of having trainable systems working in real time, which are needed for musical contexts where customization is important.

As a result, in our approach to contemporary beat making using machine learning we want to try to overcome these biases by allowing the musician to (i) use a single data representation to encode rhythms in simple and compound meter, (ii) train models with small datasets and low processing power. In the next section, we present how these ideas are integrated into our system.

3 Development

In this section, we present the data representation we chose to encode rhythms in different meters, the network architecture and software stack we selected to train models and generate rhythmic sequences, the interaction metaphor and parameters that are exposed in the user interfaces, and a novel solution to observe the rhythmic patterns in the latent space.

3.1 Data structure

In this research, we want to encode repeating drum rhythms. A fixed-length representation is sufficient to encode such patterns, but the underlying data must be capable of appropriately accommodating different meters within a single structure. As we discussed in Section 2, most previous research for the modeling

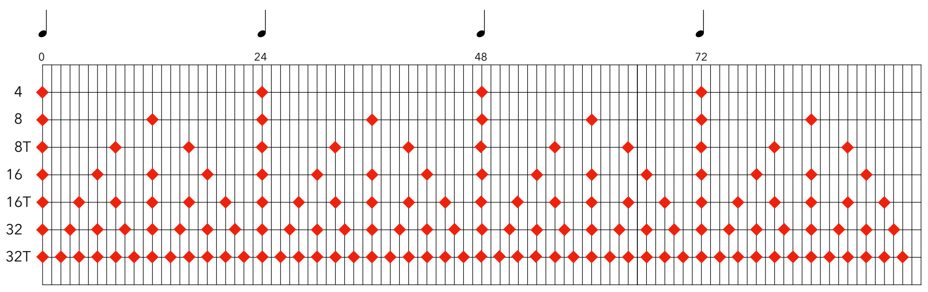

Fig. 1. Data structure chosen to allocate simple and compound meter rhythms. The horizontal axis shows all 96 ticks in one 4/4 bar with 24 ppqn (pulses per quarter note) resolution. The y axis show standard subdivisions, where 4 indicates quarter notes, 8 indicates eight notes, 8T is used for eight triplets, and so on. The maximum resolution is a 32^{nd} triplet note.

of rhythm used a grid of sixteen 16^{th} notes per bar of 4/4 time, corresponding to a resolution of 4 ppqn (i.e., four pulses—or subdivisions—per quarter note). To properly encode onsets in simple and compound meter and finer grids, we implemented the DIN Sync-24 data structure introduced by Roland in their drum machines and sequencers of the early eighties (e.g., TR606, TR808, TR909, TB303, MC4), and later incorporated in the MIDI 1.0 standard (Graves-Brown, 2014). This structure has a data resolution of 24 ppqn which is optimal for our purposes because a quarter note is divided into 24 pulses, hence allowing divisions by two and three. As can be see in Figure 1, this representation allows to encode up to 32^{nd} triplet notes.

3.2 Neural network architecture

Autoencoders (Kingma & Welling, 2014) are a popular neural network architecture for generative systems. These neural networks are trained to copy their input to their output by encoding the training data into a lower-dimensional latent representation and then decoding this latent representation back to output data. Autoencoders learn to compress the data while minimizing the reconstruction error. Autoencoders include an encoder component capable of mapping real-world phenomena (here, a rhythm) onto this compact representation (i.e., a vector in latent space), as well as a decoder component capable of generating realistic new content (here, new rhythms) from any new latent vector. Nearby points in the latent space tend to represent similar phenomena (here, musically similar rhythms). Consequently, a sequence of points along a continuous curve in latent space should be decodable into a sequence of rhythms that are perceived as gradually changing over time.

Variational autoencoders (VAEs) employ additional assumptions and constraints to explicitly attempt to create a latent space that affords smooth interpolation. Specifically, the encoder component of a VAE maps an input to

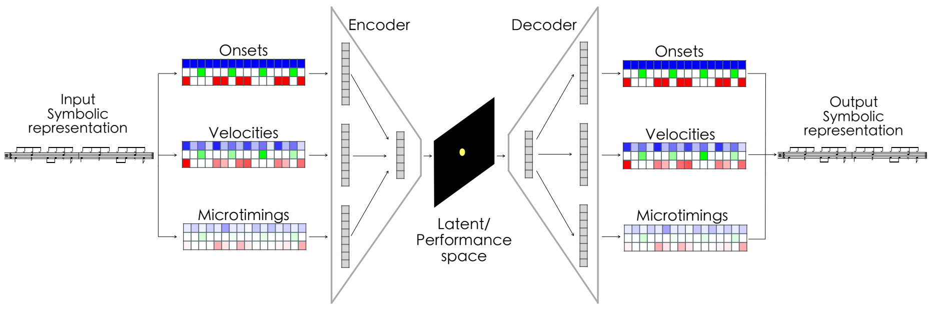

Fig. 2. At training time, rhythmic data in symbolic music format is described in terms of their onsets, velocities, and microtimings, and encoded into a latent space. At generation time, a musician samples this space directly using a performance interface. The rhythmic patterns are then retrieved and decoded into a symbolic music format.

parameters of a probability distribution, rather than to a single latent vector. To generate new content, samples are drawn from this probability distribution, and the decoder then maps from these samples to new content. Given the generative possibilities of VAEs, we chose this architecture as the basis to implement a system we called R-VAE. (Vigliensoni, McCallum, & Fiebrink, 2020).

R-VAE is based on the Tensorflow.js VAE implementation tfjs-vae[10] and M4L.RhythmVAE (Tokui, 2020). While the former provides the Tensorflow backend for the VAE, the latter provides a data structure based on the one by Gillick et al. (2019) that encodes the onsets of rhythms, their velocities, and microtimings. We adapted this system to accommodate up to 32nd triplets and created a dynamic visualizer that reveals all the patterns in a given latent space at a time. To determine the dimension of the vectors to be processed by the network, we examined commercial MIDI drum loops libraries[11,12] and sample packs[13,14] of contemporary music rhythms and found that most encoded four instruments (i.e., kick, snare, closed hi-hat, open hi-hat). Several added two more instruments (usually mapped to cymbals), and a few incorporated three more instruments (in general mapped to toms, cowbell, and percussion), adding up to a total of nine instruments. This number of instruments is the same found in previous research on drum sound classification (Herrera, Yeterian, & Gouyon, 2002), and used in implementations by Tokui (2020) and Gillick et al. (2019). As a result, our chosen data representation for the encoding of one bar of 4/4 time comprises three vectors (for onsets, velocities, and microtimings) of size 864. This vector size accommodates 24 pulses $\times$ 4 quarter notes $\times$ 9 drum instruments. The values in the encoding and decoding vectors depend on their data. The onset vectors

[10] `https://github.com/songer1993/tfjs-vae`
[11] `https://groovemonkee.com/collections/midi-loops`
[12] `https://www.thelooploft.com/collections/midi-loops`
[13] `https://cymatics.fm/collections/drum-kits`
[14] `https://www.ghostsyndicate.net/`

are encoded with 0 or 1 according to the onset position. Velocities are encoded as a float number between $]0, 1]$ according to the normalized MIDI velocities of each onset. Microtimings are encoded with a float between $] - 1, 1[$ according to their time-shift from the grid, where 0 is the exact, nominal position in the grid, and ± 1 is one tick before or ahead of the nominal onset position.

Our neural network configuration thus consists of a VAE architecture with 864 dimensions for the input, 512 for one intermediate layer, and two dimensions for the latent space. The batch size is set to 64, the optimization algorithm to Adam, and the activation function to LeakyReLU. The favoring of fully connected feedforward layers instead of the bidirectional LSTMs, as in Gillick et al. (2019), permits faster training using CPUs. Figure 2 shows a schematic diagram of the R-VAE architecture, showing the encoding of onsets, velocities, and microtimings of the rhythmic data.

For model training and playback, R-VAE is implemented as a Max for Live device. [15] This implementation has the advantage of straightforward interconnection with Ableton, offering fast and simple MIDI and audio routing for changing and processing drum sounds, easy mapping using MIDI or OSC to play with the latent space using a gestural controller, and device automation capabilities for fine control during recording or mixdown.

To evaluate the performance of the model being learned and to prevent overfitting, at training time the data is split randomly into training and validation sets using a 5:1 proportion for training:validation. The training and validation losses are computed as the mean of the onset, velocity, and microtiming losses, and are displayed and plotted dynamically versus the epoch number at training time. These curves allow the user to decide when there is no further significant reduction in the loss to halt the training process at any moment.

We informally compared the computing needs of our implementation with other architectures such as Magenta's MusicVAE and GrooVAE. These network architectures are more complex and larger than our implementation and we noted that the overall training time—even with a GPU—is in the order of hours when processing large datasets. In comparison, our implementation is meant to be used with little data and can be trained on a CPU in shorter periods. For example, musically useful and smooth models with a limited scope in variability, but that can be used in creative practice are learned from sets of dozen data points in about 45 seconds. We found that the reduction of training time and the required amount of data enables a greater ability to experiment with data and model training, and, ultimately, facilitates interactive musical exploration with the system.

3.3 Web-based model player

In addition to the Ableton device, we also released R-VAE as a web-based application that can be used as a rhythm model player, enabling performers to

[15] `https://github.com/vigliensoni/R-VAE`

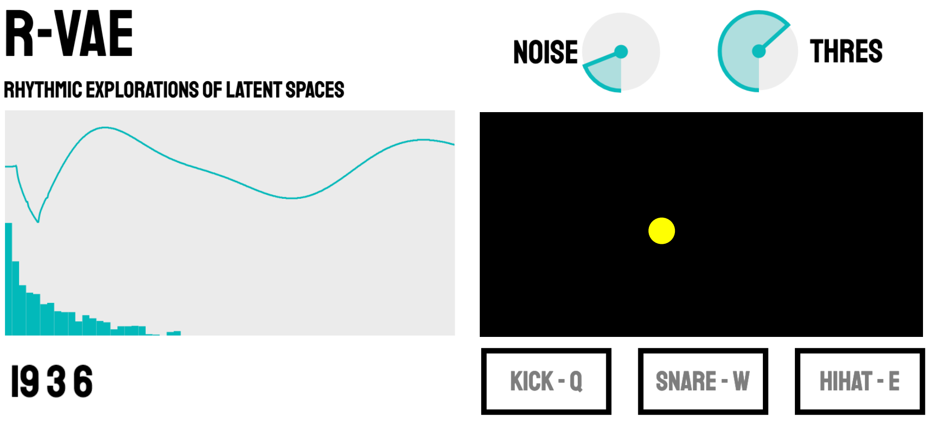

Fig. 3. The user interface of R-VAE-JS web application. The latent space on the right can be explored by moving the yellow circle. Additional knobs for *threshold* and *noise*, as well as mute buttons for "kick," "snare," and "hi-hat," allow the performer to interact with and control the web-based instrument. Three web audio interfaces (i.e., audio spectrum, oscilloscope, and timing) are displayed on the left for visual feedback.

explore rhythmic latent spaces and make music directly in the browser. In Figure 3 we show the user interface of the web-based version of R-VAE. We can see the performance space delimited by the black zone on the right.

To provide performers with a meaningful metaphor for their interaction with the system, we say that they are in control of a playback head that can be placed and moved in zones of the performance space to retrieve and decode rhythmic patterns. As such, performers can explore and play directly with the performance space through two basic gestures. First, *clicking* on any given point of the latent space will retrieve and decode a rhythm sequence. Second, *dragging* between points in the space enables performers to interpolate between rhythms. The user interface exposes additional controls to add variability to the network decoding. The *noise* knob adds a random value taken from a normal distribution to the position in the latent space where points are being decoded. From the user perspective, this variable controls the precision of the mapping between the performance space to the latent space. Higher noise values will retrieve rhythms that are further away from the chosen position, and lower values will retrieve closer rhythms. *Threshold* controls the complexity of the patterns decoded from the latent space by setting a number above which the onset probabilities retrieved from the latent space will trigger an onset. This results in higher threshold values triggering simpler sequences, and lower threshold values triggering more complex ones. In addition, *mute buttons* per instrument allow the performer to silence instruments at any given moment.

3.4 Dynamic visualizer

The performance space shown in the R-VAE interface enables the performer to retrieve rhythmic patterns from the latent space. However, from a human-computer interaction point of view, this poses challenges because the latent space is a continuous function where salient characteristics of the original distribution are encoded but the space axes have no clear labels. In other words, there are no visual cues about how the rhythmic pattern onsets are distributed in the space, and so the performer is "blind" to the characteristics of the space. To alleviate the lack of visual feedback and improve the playability of the system, we designed and developed a dynamic visualizer.

Most of the approaches to learning and playing with rhythmic latent spaces reviewed in Section 2 provide a static visualization of the latent space. This visualization is usually a static piano roll representation that displays the onset patterns for one point in the space at a time. This means the performer does not have cues about how the rhythmic patterns are organized in the rest of the space. To uncover the structure of the latent space we implemented a visualization that relies on the temporal, dynamic nature of the musical events (Vigliensoni, McCallum, Maestre, & Fiebrink, 2020).

The dynamic visualization tool is based on mapping the onset probability values retrieved from the drum instruments for each point in the space to the brightness of their representation in the user interface canvas. To achieve this, we sample the original, continuous two-dimensional latent space at discrete points over time and retrieve the onset probabilities for the drum instruments. Then, we scale the instruments' probability values to the range $[0, 255]$ and fill square matrices with these scaled probability values. Cells within each matrix correspond to a specific drum instrument. These instruments are rendered in the user interface canvas using a single color per instrument and 8-bit RGBA values.

Figure 4 illustrates the mapping from the latent space to the dynamic visualization. On the left of the figure, we see four matrices of order 3, corresponding to four discrete points (i.e., four rhythms) in the latent space. Using a clocking system sync to a specific tempo, an imaginary playback head traverses all the matrices from time $t = 0$ to $t = T - 1$. Each instrument in a rhythmic pattern will trigger a specific matrix cell with a single color. For example, a kick will only trigger *red* pixels in the position $[0, 0]$ of the matrices, a snare will only trigger *green* pixels in position $[0, 1]$, and hi-hats will only highlight pixels in *blue* in position $[0, 2]$. The time dimension t shows how the colors within each of the matrices' cells change according to the onset probabilities retrieved from the latent space for the corresponding instruments. Each pixel of the visualization canvas is mapped to the latent space and updated accordingly over time. The data representation of R-VAE encodes nine drum instruments that can be displayed in the visualizer, but in the current prototype we opted to display only the three most common instruments (i.e., kick, snare, and hi-hat), which form the core of most rhythms in popular music (Mauch & Dixon, 2012). This results in a natural vertical spacing of the patterns on the screen.

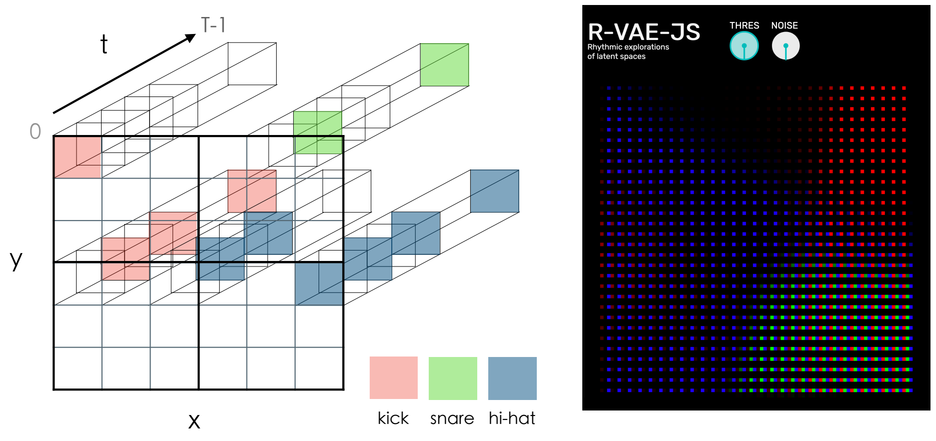

Fig. 4. Diagram illustrating the mapping from the latent space to the performance space canvas. On the left, four discrete points (i.e., four rhythms) of the latent space are sampled over time, each represented in the figure as a 3-by-3 matrix. Each instrument in a rhythmic pattern will trigger a specific matrix cell with a single color. On the right, the full performance space, made of 900 points sampled from the latent space, is shown. Here, the visualization shows how the activation of different drum instruments (i.e., kick, snare, and hi-hat) differs across the latent space for this particular moment in time, *t*. A full measure consists of 96 of these images, played in sequence to produce a dynamic animation.

On the right of the figure, we see an image capture of the full performance space at time *t*. At this specific moment, hi-hats (in blue) are happening in most of the space, except for the upper right part, where kicks (in red) are happening. Snares (in green) are not triggered alone. It must be noted that this representation and distribution of onsets change rapidly, at a pace of 96 images per bar of music.

Both implementations of R-VAE, for Ableton Live[16] and the R-VAE-JS browser-based model player,[17] are available for public use. Several models trained on rhythms in different meters are also available.[18] A video demonstrating a web-based performance with R-VAE-JS at the Network Music Festival 2020 is available at `https://youtu.be/lnRe5vZ6Sps`. The video shows the user interface in action, the dynamic visualization of the latent space, ane the control parameters. The drum sounds in the video come from an external drum synthesizer triggered via MIDI. The external triggering of sounds allows further variability, such as real-time control of the drum sounds, compression, and additional effects.

[16] `https://github.com/vigliensoni/R-VAE`
[17] `https://github.com/vigliensoni/R-VAE-JS`
[18] `https://github.com/vigliensoni/R-VAE-models`

4 R-VAE in Musical Practice

In this section, we provide details and insights on using R-VAE in a full cycle of professional music-making—–from data collection and model training to composing, performing, and premiering an album in a music festival. We spent several months collecting and transcribing rhythms; training models; adding and removing datapoints; improvising with the generated latent spaces; and retraining the models if needed. We repeated this process several times for different datasets until we got to a point where we had rhythmic spaces that were smooth and had enough variability for using them in real-life music production. We then composed and recorded a set of nine musical pieces and premiered them live using R-VAE on stage.

4.1 Data collection

Given the bias of existing machine learning rhythmic datasets towards a narrow range of mainstream genres, we compiled a series of our own featuring the rhythms we are interested in from modern music genres. As we also found provisions lacking in commercial MIDI libraries, we decided to create the symbolic files by transcribing them manually from audio, or by correcting the output of audio to MIDI converters. For this very reason, our datasets are small, in the range of a dozen of MIDI clips each.

4.2 Feasibility of training from small datasets

Given the restricted size of our datasets, our first experiments explored the feasibility of using R-VAE to learn useful representations in this context. The variables and parameters that, at this point, we had to control the learning process were the data—both in quantity and type—and the number of training epochs. We started by learning models from a small but controlled amount of data, that is four MIDI clips (two in simple and two in compound meter), then we gradually increased the number of training epochs from 10 to 1000 epochs and we assessed the subjective qualities of the resulting latent spaces by visualizing, playing, and comparing them with the original training data.

In Figure 5 we show the visualization of the latent spaces generated by this small dataset for 10, 100, and 1000 training epochs. In this static visualization, and for the sake of a good comparison, we show the same pulse per measure for the three spaces (a dynamic version of this visualization can be watched at `https://youtu.be/zOnPZGYJ6Fs`). It can be seen that after 10 epochs, the model is undertrained resulting in half of the space being mostly noise. In these non-ideal conditions, the outputs of the network are random onsets, velocities, and microtimings in large portions of the latent space. At 100 epochs, the amount of noise is reduced greatly, and it completely disappears at 1,000 epochs. When we increased the number of training epochs to values higher than 250 the training

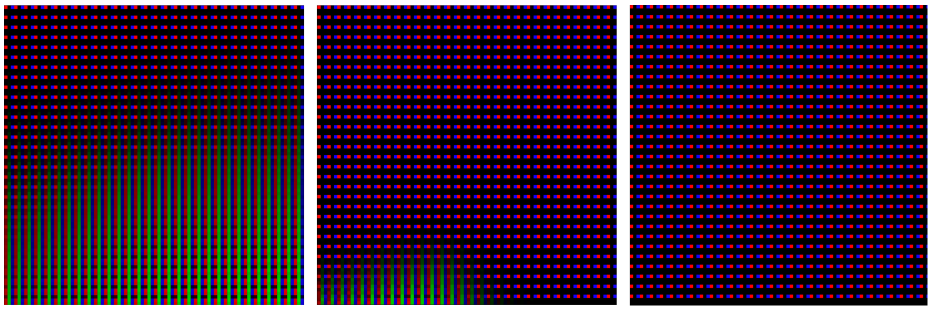

Fig. 5. Visualization of the latent space configuration of a small training set consisting of four MIDI clips after different three training cycles (i.e., 10, 100, and 1000 training epochs). The static plot shows the first pulse of the measure. The visualizer helps the performer to easily detect noisy zones.

loss was usually higher than the validation loss. This resulted in overfitted models, where the spaces were clean and the rhythmic zones were clearly delimited, but the transitions were more abrupt.

We then incrementally increased the number of MIDI clips, varied the number of training epochs, and observed the resulting spaces using the visualizer to assess the training process qualitatively. We found that in around a dozen clips, the latent spaces started to be smoother and could create better interpolations. This ability, however, was not only dependent on the number of clips but also on their rhythmic characteristics. We found that smoother transitions were learned when providing examples of different but related patterns. Using very small datasets as were using may create some artifacts in the latent space, such as zones where the training process did not converge and there were random onsets, especially when using low thresholds. In such cases, we added more datapoints to the training data or we used higher thresholds when we were performing. All in all, we found that a sweet spot between stability and variability for learning usable latent spaces happened in between 100 and 250 epochs for one or two dozen MIDI clips. The time required to train the spaces was linear to the number of epochs and training data. In the context of small-scale datasets, and 100 to 150 epochs, this time varied between 36 and 150 seconds (running on a 2.8 GHz Intel Core i7 computer). This amount of time was acceptable for trying and assessing different configurations of data and the system, and testing the results in musical practice.

In Figure 6 we show the interpolation of onsets between two points of a latent space learned from 12 clips of footwork, with a threshold value set at 0.3. Kick, snare, and hi-hat onsets are red, yellow, and blue. Whisker lines on top of each onset show the corresponding velocity decoded from the latent space. Whisker lines touching the upper tip of each onset diamond indicate maximum velocity. We can see that the model can interpolate from patterns in a very binary feel (i.e, patterns one to ten), to patterns with elements in compound meter (i.e., patterns

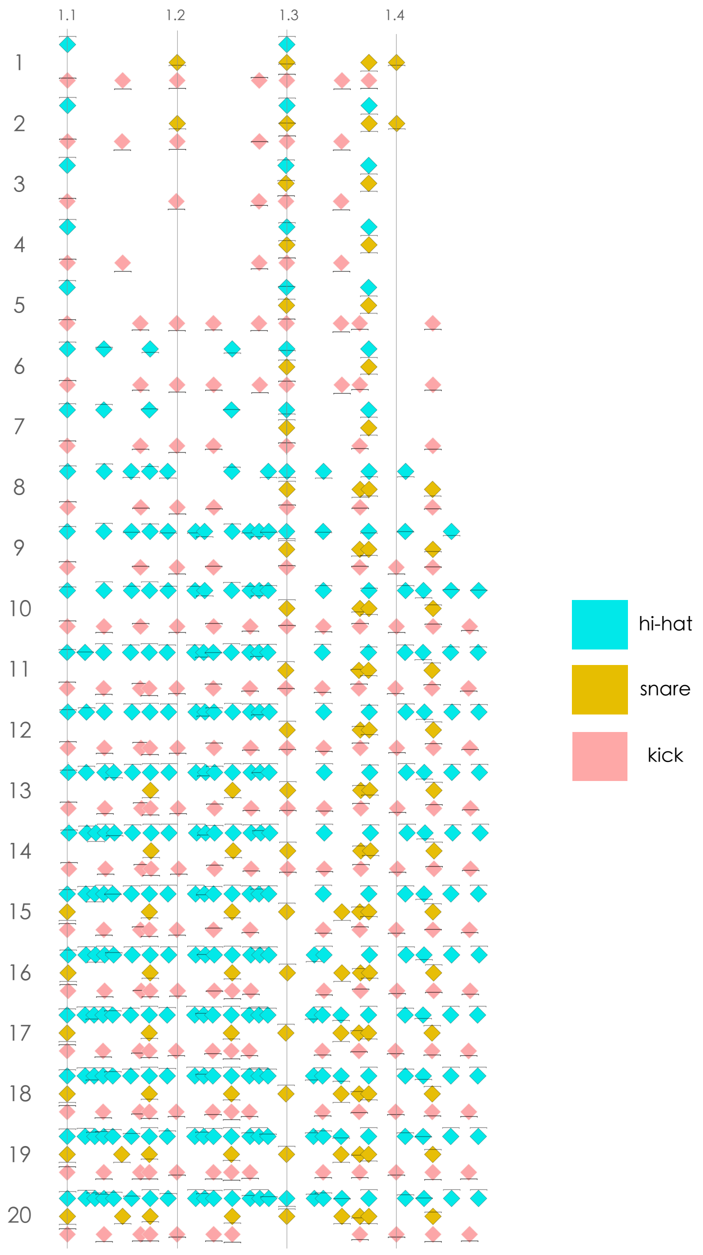

Fig. 6. Interpolation between two points of a latent space learned from 12 MIDI clips of *footwork* using a threshold value of 0.3. Each row shows the kick, snare, and hi-hat onset for one beat. All rows except 17 and 18, exhibit a unique onset pattern of kick, snare, and hi-hat. Velocities for each onset are shown using a whisker line.

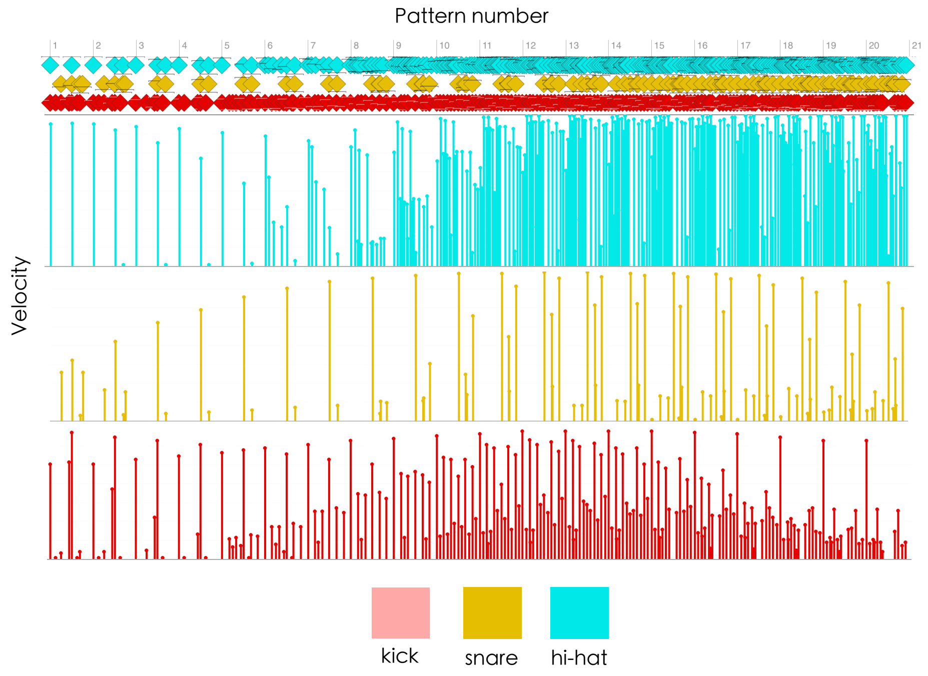

Fig. 7. Interpolation between points in the latent space displaying the velocities for each onset. The lower rows show the varying MIDI velocities for the three drum instruments across the decoded patterns. Figure 6 showed that patterns 17 and 18 exhibited the same onset structure. Here we see that their velocities are different and so are those patterns.

11 to 20). This change can be easily visualized in the kick and hi-hat patterns. It is worth mentioning that sampling the space with different threshold values will retrieve a different set of patterns from the space. The lower the threshold value, the higher the number of onsets and the complexity of the decoded patterns. The onsets in this particular interpolation correspond to a small subset of all available rhythms in the continuous space and were retrieved by sampling the space at regular intervals. It can be seen that all the patterns are different, except for patterns 17 and 18, which exhibit the same onset structure. However, when analyzing their velocities, we can see that they are different, as can be seen in Figure 7.

All in all, a dozen MIDI clips can generate a space that encodes many more patterns than the original ones.

4.3 Composition

With the set of R-VAE models we trained from the small-scale datasets, one of the authors of this article composed a series of nine musical works grouped under the name of “Clastic Music.” *Clast* is a term used in geology that refers to rocks made of fragments or sediments of pre-existing rocks. We found this akin to how we are using pre-existing drum patterns to create new fragments and rhythmic structures.

We not only tried one specific model for each one of the pieces, but we also explored playing with spaces in parallel. With this approach, we had two instances of R-VAE running in sync. While we sampled the latent space at a specific point and decoded a rhythm using the first instance, we dynamically changed the position of the playback head in the second instance and layered a varying pattern that added variability to the first R-VAE instance output. Another parallel approach was setting up two low frequency oscillators to the playback head position in one instance—so that the position of the playback head was changing gradually in a controlled manner—and a second layer was added on top.

In terms of the sound output, we first tried triggering samples from the MIDI output of R-VAE, but we soon realized we could take advantage of the realtime possibilities of sound synthesis to add expressivity to the rhythmic patterns. As a result, we chose to couple R-VAE with the Dave Smith Instruments Tempest, an analog drum synthesizer that exposes many of its synthesis parameters in a complex gestural interface. Having access to control the actual generation of sound allowed us not to depend on constantly changing the latent space sampling point to add variability to the patterns, but to control the sound generation in realtime (e.g., by changing the oscillator frequencies, lowpass or highpass filter frequencies and resonance, amplitude and filter envelopes, etc.) A second instrument we used to trigger sound was the Applied Acoustic Systems Chromaphone, a physical modeling synthesizer built around the combination of acoustic resonators. The physical modeling approach to sound generation allowed us to generate a wide range of sounds due to user-chosen signal flows and different resonator types that contributed an additional layer of expressivity to drum sounds.

Due to the intrinsic nature of R-VAE, and our interest in exploring a specific set of rhythms, our compositions were essentially rhythmic pieces. We decided *Clastic Music* would have an increase in tempo throughout the album, where the first tracks would start pretty slow, and the tempo would gradually increase until reaching a faster tempo at the end. However, we took advantage of the tempo octave features of the rhythms we were modeling, and we modulated the perceived tempo throughout the tracks. This is especially noticeable in the first pieces—where there are zones in 85 bpm and others at 170 bpm—and in the last one—with tempo zones perceived at 160 bpm and others at 80 bpm.

Clastic Music can be listened at `https://bit.ly/3yo6rm1`

4.4 Performance

To examine further the use of R-VAE in a musical performance context, we premiered Clastic Music live at the 21st International Festival for Digital Creativity and Electronic Music, MUTEK. For this performance context, instead of interacting with the latent space using a computer trackpad or mouse, we opted to

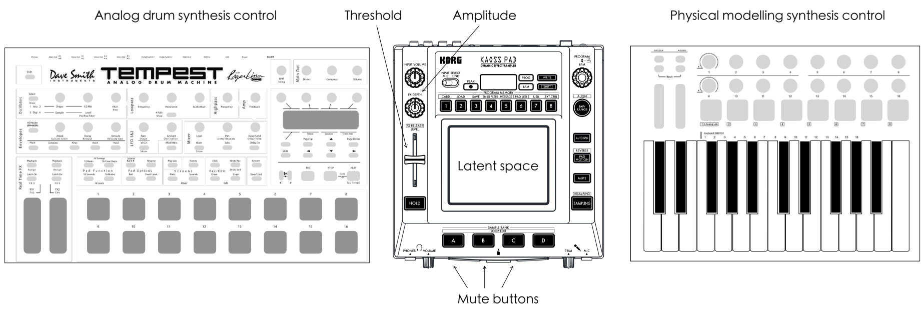
Analog drum synthesis control
Threshold
Amplitude
Physical modelling synthesis control
Latent space
Mute buttons

Fig. 8. The mapping of parameters to control R-VAE and the sound synthesis generation was implemented through three interfaces. The R-VAE decoding from the latent space was controlled through a Kaoss Pad XY grid, potentiometers, and buttons (the device in the middle). The interface of a Tempest synthesizer was used to control the drum sound parameters. Knobs and sliders in a third controller were mapped to the parameters of a physical model synthesizer.

map the R-VAE parameters to a KORG Kaoss Pad KP3. This device has a big touchpad that can be used as a MIDI controller, and we mapped it to the latent space in R-VAE. We also mapped the controller's slide potentiometer to the R-VAE threshold, three buttons to mute or unmute the main drum instruments, and a rotary potentiometer to the overall amplitude level. The knobs and modulation wheel of a second MIDI controller were used to control parameters of physical models in Chromaphone, and the synthesis parameters of drum sounds were controlled directly with the Tempest interface. In Figure 8 we show the configuration we used to perform with R-VAE on stage.

This way of interacting with the R-VAE and the sound synthesis engines gave us a different perspective on how to play with the full system. That is, instead of tweaking one thing at a time, as you would do with a mouse or trackpad, we were able to do several things at the same time. For example, we were able to interpolate between points in the space while changing the threshold or changing the amplitude envelopes of drum sounds. This was a big change because it gave us the power of not only having a simple two-dimensional mapping from the performance to the latent space, but a three-dimensional one, where we could also control the complexity of the patterns retrieved. Even further, considering the control we had on the sound generation, we ended up with a multi-dimensional mapping, where we were able to control the sequences of patterns to be played as well as the sonic characteristics of the drum sounds.

Having a tactile interface also provided us with a degree of proprioceptive feedback that was missing when interacting with the computer through the trackpad. Long rehearsals with pre-made models, the new type of feedback, and the fact that live performance is different from composition or exploration mode when in the studio, resulted in not needing to look at the dynamic visualizer.

Snippets of the MUTEK live performance can be accessed at `https://youtu.be/0y69ANhUlvM?list=PL2ySFw5-RHDaA4mKaCeGAzl2Kh_uklSoR`

5 Conclusions and Future work

In this article, we have presented R-VAE, a system designed for the modeling and exploration of latent spaces learned from rhythms encoded in MIDI clips. We investigated the viability of the system by encoding rhythms from music genres such as *footwork*, *gqom*, and *trap*, and using the models in real-life musical production and live performance.

We empirically confirmed our system was able to encode rhythms from those genres and learn models with small training datasets. As a result, the main contributions of this research are twofold: (i) a machine learning device for the generation of rhythms that incorporates a data representation that broadens the type of rhythms that can be encoded towards modern music genres, and (ii) a dynamic visualizer that displays all rhythmic patterns in the latent space at once. Although the first contribution could be seen as a modest technical extension to previous implementations, we think it entails a great musical contribution since, for the first time in the modeling of rhythms using VAEs, we can now encode rhythms in compound meter, thus reducing the biases (e.g., toward particular genres and cultures) produced by data representations that only encode certain types of rhythms. In regards to the second contribution, our visualization advances research in latent spaces for music by moving beyond conventional mapping toward a more integrated visual expression of the sonic material.

When experimenting with data selection and model training, we found that the system was able to learn useful and playable models with as low as one dozen MIDI clips. However, this approach is not without problems. If the learning process overfits the training datapoints, the model may only retrieve from the latent space the same original examples. An undertrained model, on the contrary, may not converge to a meaningful representation. As such, decoded rhythms will not bear resemblance to the training set. Another drawback of using small training sets is that the topology of the learned space may be uneven. That is, the space exhibits zones where a single pattern is decoded and transitions to adjacent zones are abrupt. On the contrary, when we increased the number of clips to a few dozen, the learned latent spaces were richer and exhibited a more even topology, helping the performer to create smoother interpolations when moving the playback head between rhythms in different zones of the performance space. Even if not perfect, individually crafted models from smaller datasets could prove to be useful and inspiring to the creative practitioner. The amount of data needed to generate creatively interesting models will vary with context and intention, but small data may be more suited to generate custom models that are good for exploring a specific idea or creative concept.

We investigated the expressive possibilities of generating and playing with rhythmic latent spaces learned from data in real-life musical production. We observed that the dynamic visualizer captures nicely how the different patterns are distributed in the latent space, and provides much-needed visual feedback when interacting with the models. Since patterns are synced in time, similar, neighboring zones flash synchronously, exposing previously hidden rhythmic clusters in the space. On the contrary, adjacent zones with elements in different meter flash asynchronously, giving the performer a natural visual cue to discriminate these zones and their boundaries.

We investigated the viability of R-VAE in live performance contexts at the MUTEK International Festival of Digital Creativity and Electronic Music and the Network Music Festival 2020. We found that additional R-VAE controls such as *threshold* are very important in live contexts because it adds another dimension to control how the rhythmic patterns are sampled from the latent space. In other words, by limiting the number of onsets retrieved from the latent space at any given time, the *threshold* parameter helps the performer control the complexity of the decoded rhythmic patterns. As a result, just using the threshold in a single point of the latent space can lead to many interesting rhythmic variations and perceptually different patterns.

Interacting and performing with latent spaces encoding rhythms pose excellent questions, from both a computational and musical point of view. For example, how can we characterize the latent space in terms of the *smoothness* of the interpolations? What is a good metric to measure the *richness* of the encoded space? Musical performance contains some pertinent differences from other fields unaddressed by current technical literature. For example, when measuring rhythmic similarity, the apparently small change of moving a few onsets from a 16th grid to its closest triplets is small in terms of edit distance, but it can have a large perceptual impact. Furthermore, there are some instrumental hierarchies when working with rhythms. For example, changing a few hi-hats in a rhythmic sequence is likely to have a subtler perceptual influence than changing a drum kick pattern.

Although the design and usage of R-VAE described in this paper have been informed by the professional experience of the research team, including substantial professional music performance work by the first author using the system, further development will also benefit from evaluation and use by others. As a result, future planned work includes a user study where professional beat makers outside the research team interact with the system. We aim to gain insights into the usefulness of the visualizer, the perceived qualities of models trained from very small datasets, and the adequacy of R-VAE in different musical contexts (i.e., improvisation, performance, and production). We expect this study will uncover problems and design opportunities, and provide insights to inform the design process and further research in this area.

Our experience with R-VAE has reinforced the idea that a system for the exploration of latent spaces of musical rhythms is worth pursuing further. For example, systems like this could be also used for browsing through libraries of rhythms, common in contemporary music production, or for the generation of non-linear music soundtracks, common in videogames. Finally, the visualizer presented in this work was implemented for models created with a VAE network but its design is easily generalizable and has the potential to be used with other network architectures.

6 Acknowledgments

This research has been supported by Fonds de recherche du Québec – Société et culture (FRQSC) through a research creation grant (Ref. 2020-B5-272998). Important parts of this project used ComputeCanada's High Performance Computing resources.

References

Battenberg, E., & Wessel, D. (2012). Analyzing Drum Patterns Using Conditional Deep Belief Networks. In *Proceedings of the 13th International Society for Music Information Retrieval Conference* (pp. 37–42).

Bouwer, F. L., Burgoyne, J. A., Odijk, D., Honing, H., & Grahn, J. A. (2018, January). What makes a rhythm complex? The influence of musical training and accent type on beat perception. *PLOS ONE*, *13*(1).

Brandel, R. (1959). The African hemiola style. *Ethnomusicology*, *3*(3), 106–117.

Callender, L., Hawthorne, C., & Engel, J. (2020). Improving perceptual quality of drum transcription with the expanded groove MIDI dataset. *arXiv:2004.00188*.

Choi, K., Fazekas, G., & Sandler, M. (2016). Text-based LSTM networks for automatic music composition. In *Proceedings of the 1st Conference on Computer Simulation of Musical Creativity.*

Colombo, F. F. (2021). *Learning music composition with recurrent neural networks* (Doctoral dissertation, EPFL, Lausanne). doi: 10.5075/epfl-thesis-10406

Dinculescu, M., Engel, J., & Roberts, A. (2019). MidiMe: Personalizing a MusicVAE model with user data. In *Proceedings of the 33rd Conference on Neural Information Processing Systems.*

Duinker, B. (2019). Good things come in threes: Triplet flow in recent hip-hop music. *Popular Music*, *38*(3), 423–456. (Publisher: Cambridge University Press)

Ellis, D. P. W., & Arroyo, J. (2004). Eigenrhythms: Drum pattern basis sets for classification and generation. In *Proceedings of the 5th International Conference on Music Information Retrieval* (pp. 101–106).

Fiebrink, R., Trueman, D., & Cook, P. R. (2009). A meta-instrument for interactive, on-the-fly machine learning. In *Proceedings of the 9th International Conference on New Interfaces for Musical Expression* (pp. 280–285).

Gillick, J., Roberts, A., Engel, J., Eck, D., & Bamman, D. (2019). Learning to groove with inverse sequence transformations. In *Proceedings of the 36th International Conference on Machine Learning.*

Gjerdingen, R. O. (1989). Meter as a mode of attending: A network simulation of attentional rhythmicity in music. *Integral*, *3*, 67–92.

Graves-Brown, P. (2014). Plugging in: A brief history of some audio connectors. *World Archaeology*, *46*(3), 448–461.

Gómez Marín, D. (2018). *Similarity and style in electronic dance music drum rhythms* (Ph.D. Thesis, Universitat Pompeu Fabra). Retrieved 2021-03-12, from `http://www.tdx.cat/handle/10803/543841` (Accepted: 2018-05-10T10:44:33Z Publication Title: TDX (Tesis Doctorals en Xarxa))

Herrera, P., Yeterian, A., & Gouyon, F. (2002). Automatic classification of drum sounds: A comparison of feature selection methods and classification techniques. In C. Anagnostopoulou, M. Ferrand, & A. Smaill (Eds.), *Music and Artificial Intelligence* (pp. 69–80). Berlin, Heidelberg: Springer. doi: 10.1007/3-540-45722-4_8

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations.*

Large, E. W. (2008). Resonating to musical rhythm: Theory and experiment. In *The psychology of time* (pp. 189–232). Emerald Group Publishing Limited.

Lattner, S., & Grachten, M. (2019). High-level control of drum track generation using learned patterns of rhythmic interaction. In *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2019).*

London, J. (2012). *Hearing in Time: Psychological Aspects of Musical Meter.* Oxford University Press.

Mauch, M., & Dixon, S. (2012). A corpus-based study of rhythm patterns. In *Proceedings of the 13th International Society for Music Information Retrieval Conference.*

Nikolov, S. (2016, April). *NeuralBeats: generative techno with recurrent neural networks.* Retrieved 2021-03-01, from `https://medium.com/@snikolov/neuralbeats-generative-techno-with-recurrent-neural-networks-3824d7ba7972`

Oore, S., Simon, I., Dieleman, S., Eck, D., & Simonyan, K. (2020). This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, *32*(4), 955–967.

Paiement, J.-F., Bengio, S., Grandvalet, Y., & Eck, D. (2008). A generative model for rhythms. In *Neural Information Processing Systems, Workshop on Brain, Music and Cognition.*

Peñalosa, D., & Greenwood, P. (2009). *The clave matrix: Afro-Cuban rhythm: Its principles and African origins.* Redway, CA: Bembe Books.

Phillips, M. T. (2021). Soundcloud rap and alien creativity: Transforming rap and popular music through mumble rap. *Journal of Popular Music Studies*, *33*(3), 125–144.

Roberts, A., Engel, J., Mann, Y., Gillick, J., Kayacik, C., Nørly, S., ... Eck, D. (2019). Magenta Studio: Augmenting creativity with deep learning in Ableton Live. In *Proceedings of the International Workshop on Musical*

Metacreation (MUME).

Roberts, A., Engel, J., Raffel, C., Hawthorne, C., & Eck, D. (2018). A hierarchical latent vector model for learning long-term structure in music. In *Proceedings of the 35th International Conference on Machine Learning.*

Samuels, B. (2018). *Comparative study of beat and temporal pattern perception in a songbird* (Unpublished doctoral dissertation). Western Ontario.

Schreiber, H., & Müller, M. (2014). Exploiting global features for tempo octave correction. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 639–643).

Tokui, N. (2020). Towards democratizing music production with AI-design of variational autoencoder-based rhythm generator as a DAW plugin. *arXiv preprint arXiv:2004.01525*.

Vigliensoni, G., McCallum, L., & Fiebrink, R. (2020). Creating latent spaces for modern music genre rhythms using minimal training data. In *Proceedings of the 11th International Conference on Computational Creativity (ICCC'20).* Coimbra, Portugal (Online).

Vigliensoni, G., McCallum, L., Maestre, E., & Fiebrink, R. (2020). Generation and visualization of rhythmic latent spaces. In *Proceedings of the 2020 Joint Conference on AI Music Creativity.* Stockholm, Sweden (Online).

Wright, M. (2005). Open Sound Control: An enabling technology for musical networking. *Organised Sound*, *10*(3), 193–200.