Universiteit Leiden
The Netherlands

Artificial Intelligence

The Generation of Evolving Drum Sequences Using Artificial Intelligence

Sven Paqué

Supervisors:
Edwin van der Heide & Peter van der Putten

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

18/01/2023

Abstract

In the field of artificial music generation, not a lot of work has been done on drum sequences and especially the song structure in these drum sequences. In this exploratory study, we investigated the capabilities and limitations of Magenta's MusicVAE and DrumsRNN for generating drum sequences that contain phrasing and musical structure. Both models have different architectures and mechanisms for handling long-term dependencies in drum sequences. Our results showed that both models were able to generate realistic, changing drum sequences but struggled to incorporate specific song structures such as breakdowns and verses. We found that MusicVAE was superior in terms of generating different genres and incorporating more originality while DrumsRNN more seemingly random sequences with less proper reflection of musical structure concepts. The study provides new insights into the capabilities and limitations of these state-of-the-art models in drum sequence generation and suggests areas for future research.

Contents

1 Introduction

Music has the power to move us, and for centuries, humans have been creating and enjoying musical compositions. With the advances in artificial intelligence (AI), we are now able to explore the potential for computers to create music as well. AI creativity, in general, has been an interesting direction for many researchers around the world. We have already seen major progress in generating images, videos and text. For example, Google recently presented Imagen, a text-to-image diffusion model which transforms written text into an image [Saharia et al., 2022]. It has a deep level of language understanding.

Similar attempts have also been made to generate music, but a system which truly understands music on a deep level is yet to be discovered [Dhariwal et al., 2020]. One of the latest models, OpenAI's Jukebox, has trouble with understanding and verses and choruses. The model is built with audio only so the quality of the generated pieces is not really at a professional level. Generating music artificially is a complex task where a lot of things come together ranging from melody and chords composition to rhythm and lyrics to harmony and groove.

In this thesis, we focus on the rhythmic and drum aspects of a musical composition. And especially how AI can be used to generate drum sequences that consider the higher-level structure of music, such as phrasing (changes in the drums that happen over time) and the interplay of intensity and relaxation. By introducing these into the drum generation process, we can create drum sequences that reflect the structure and flow of a song. We are self-training Magenta's DrumsRNN and MusicVAE model with different datasets to examine the capabilities and limitations in embodying a musical structure. Both models' architectures are designed by the Magenta team [Magenta, 2016]. Magenta is a project from Google Brain, which is a deep learning artificial intelligence research division at Google [Helms et al., 2018].

The remainder of this thesis is organized as follows: Section 2 covers the musical background and related work in drum sequence generation; Section 3 explains our research question and research approach; Section 4 explains the musical tools and libraries used in this thesis; Section 5 discusses the architecture of both models, differences and limitations; Section 6 describes the acquisition and preprocessing of training data and the training process; Section 7 shows the results and convergences of the models; Section 8 discusses the results in relation to our models and answers our research questions; Section 9 concludes and suggests areas to improve for further research.

2 Background

Within this thesis, multiple musical concepts will be referred to and in order to understand them all the definitions will be given here. We also discuss related work to further understand the context of this thesis and identify gaps in current research.

2.1 Music theory

Drum percussion sequences are time-based. It contains structured sounds which have their own set of special features. Whenever such a sound or instrument is played at a time step it is called an event. Quantization is the process of aligning every "event" exactly to the regular grid of time. If the quantization is set to for example the 16th note, every event will be "snapped" to the nearest 16th

note position. Unquantized means that the event can also be slightly "off-beat", this is sometimes also preferred because it gives a humanized sound, because in general humans are also always slightly off-beat. This can create swing and groove.

In our models, we only work with quantized drum sequences. A bar is a segment of time defined by a given number of beats. Bars are used to organize and structure music, they also provide a framework for these models to follow.

2.2 Phrasing

Within this thesis, the term phrasing will be used several times, therefore it is important to know what that means. Drums are repetitive, so to keep the listener engaged, small things change in drum sequences. This is called phrasing in electronic drum sequences. There are two different kinds of phrasing; short-term and long-term. Short-term phrasing can be achieved by removing or adding a percussive element at start of a new phrase, or changing the position of percussive elements. Long-term phrasing focus on emphasizing a concept in the drum sequence. It can be considered as the thematic development of a drum piece [Dostál, 2012]. These are changes over time, for example adding tension by continuously adding more percussive elements. Phrasing can also contribute to the groove of a percussive sequence.

2.3 Related work in drum generation

Over the years, several machine learning models have been developed for generating drum sequences. One of the first music pieces generated by AI is Illiac Suite, a stochastich generated piece with Markov chains [Hiller and Isaacson, 1958]. However, a big limitation of Markov models is that they have trouble representing long-term temporal dependencies because they typically only consider the previous n states, where n is a small number, in determining the probability of the current state. Few decades later, the RNN was introduced which was great at the time [Hopfield, 1982]. However, however it turned out that RNNs also failed to learn long-term dependencies due to vanishing gradient problem [Informatik et al., 2003]. Recurrent neural networks in combination with LSTM (Long Short-Term Memory) units were introduced to mitigate this problem, as LSTMs are able to save information about past events at each cell. Choi used these LSTM models effectively for creating drum patterns [Choi et al., 2016]. Dimos Makris et al. [Makris et al., 2017] expand upon this work by considering constraints, for instance, composing drum patterns that take external information (bass, melody) into account.

The previous mentioned models are creating drum patterns from prior trained data. The approach shown in this paper [Vogl and Knees, 2017] is a system which is creating drum patterns based of a seed pattern. The algorithm is generating variations of your seed drum pattern.

These researchers have demonstrated the effectiveness of using LSTMs for drum pattern generation, however music is time-based. Music has a hierarchical structure, with songs made up of higher-level building blocks (phrases) and smaller repetitive patterns (bars). To capture this structure, a model needs a mechanism for taking into account the higher-level structure of a song. Instead of generating smaller repetitive patterns (bars), the drum generation should also focus on the temporal structure, that means introducing phrasing into electronic drum generation.

Herremans and Chew created MorpheuS, which is a framework which introduces phrasing in harmonic pieces of music, i.e. keys [Herremans and Chew, 2017]. It is capable of generating pieces

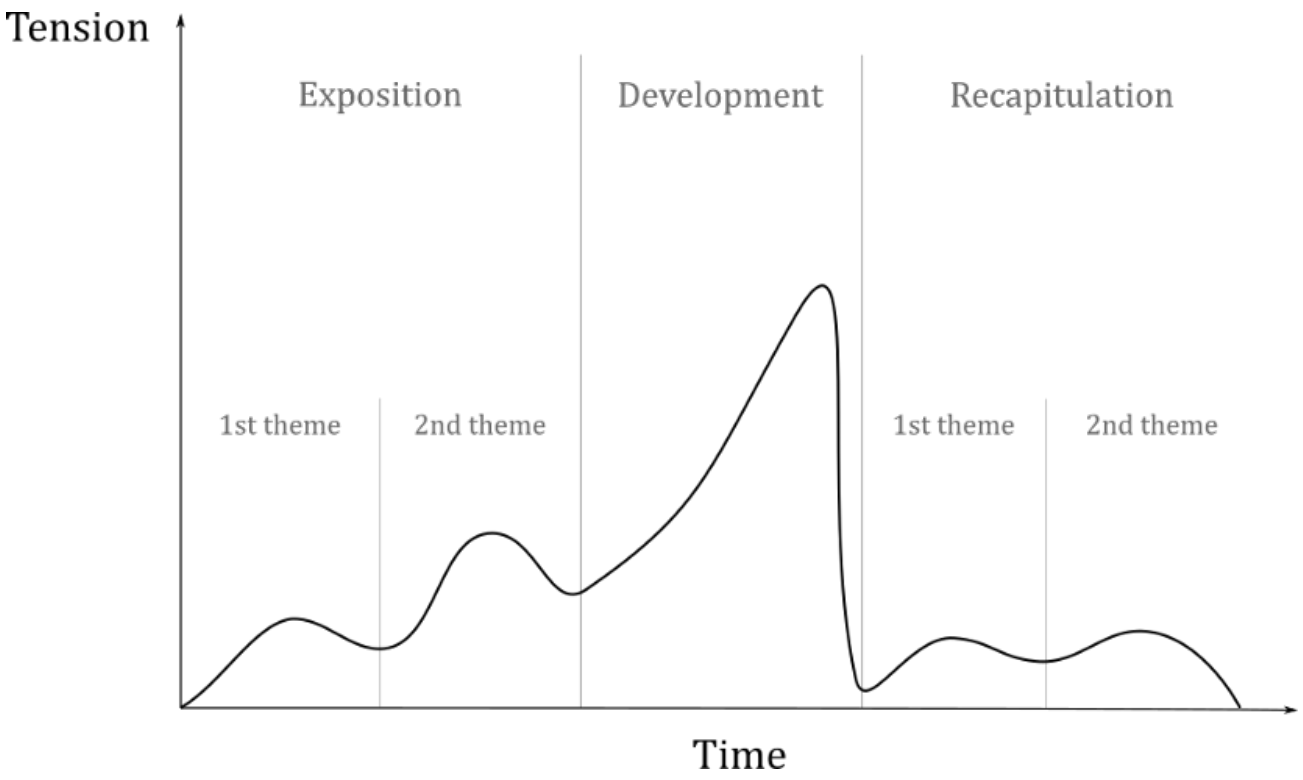


Figure 1: Visualization of Tension; Source: [Herremans and Chew, 2017]

with long- and short-term repeated pattern structures. Tension reflects the interplay of intensity with relaxation and resolution. We are looking for similar behaviour except in drums that are generated rather than morphing an original sound.

To the best of our knowledge, there are limited works that attempt the learning and generation of phrasing in percussion (drum) sequences. The architecture proposed in [Makris et al., 2019] is creating drum sequences that take conditional information into account, like for example extracted features from accompanying tracks, but also information about metrical structure and grouping information. The last part is interesting for our research. The so-called CNSL architecture added grouping indicators during the conversion of actual songs to training data. Selected parts of a song were marked as different phrases. The use of these grouping indicators allowed the architecture to be aware of changes in a song, and could therefore learn the structure of phrases within a song. Whilst a lot of researchers focus on the short-term consideration of accompanying tracks [Makris et al., 2022] [Lattner and Grachten, 2019]. We are interested in the resulting behaviour of models similar to these, and what is needed for them to model a certain concept in just the drums, as seen in Figure 1.

One model that has been successful in generating just drum sequences with a focus on phrasing aswell is DrumsRNN, a deep learning model developed by the Magenta team at Google [Magenta, 2016]. DrumsRNN uses LSTM units to learn patterns of drum sequences and is using an attention mechanism to process the input more efficiently and effectively. It can selectively focus on certain parts of the input when processing it. Another model that has been successful in generation drum sequences is MusicVAE [Roberts et al., 2018]. It is based on the idea of a

variational autoencoder (VAE), which is a type of generative model that can learn to reconstruct an input by encoding it into a lower-dimensional latent space and then decoding it back to the original space.

3 Research question

Since DrumsRNN and MusicVAE have different architectures and ways of capturing long-term dependencies. It is interesting to identify how these models compare when they generate longer sequences with phrasing and structural information. This raises us to the following research question:

> *RQ1: How do MusicVAE and DrumsRNN compare in their ability to generate 2-minute long, realistic and evolving drum sequences, and how does this performance vary with the size of the training data and the genre of the drums?*

It's also important to determine the limitations of both models by evaluating their capability in genres and originality. The following subquestions help us understand the output of the models in musical context and therefore provide directions for further research.

> *RQ1a: How does the size of the training data influence the output of the models?*
>
> *RQ1b: How do the models perform when trained on different genres of music?*
>
> *RQ1c: How do the models compare in terms of the variety or diversity of drum sequences that they are able to generate?*

3.1 Research approach and evaluation

This exploratory study will involve training both models with the same datasets containing distinct genres. We will compare the output of MusicVAE and DrumsRNN on two datasets: a techno-house dataset and a jazz-soul dataset. The generated drums will be analyzed qualitatively, utilizing knowledge on musical structure and the various characteristics of drums across different genres. It is important to note that using a more systematic approach like a metric is beyond the scope of this research. The goal of this study is to first identify the capabilities and limitations in the current models, and to provide directions for future research in this area. So therefore it isn't necessary to work with quantitative approaches yet. Additionally, the research will examine the influence of the size of the training data and the genre of music on the performance of the models. By interpreting the results, we will discuss the ability of MusicVAE and DrumsRNN to generate realistic and evolving drum sequences and identify the limitations of both models. The findings will provide directions for further research.

The various characteristics in which the drums will be qualitively analyzed are explained below. These are four important criteria which will help us understand the musical context of the drum percussions.

1. Phrasing

2. Originality
3. Realism
4. Genre discrimination

The first criterion, phrasing, can be defined as long-term and short-term phrasing. Long-term phrasing evaluates whether the drums in the output demonstrate an understanding of musical structure, such as sections like verses and breakdowns. Short-term phrasing, looks at the ability of the drums to evolve in a way that keeps the listener interested. There should be enough changes in a drum on a short time period.

The second criterion, originality, evaluates how different the output from each model is. We will be analyzing how creatively the models are able to produce drum sequences that have their own 'character'.

The third criterion, realism, is an evaluation of whether the drums are realistic and adhere to correct rhythm and stylistic conventions. For example, a kick that is played on every one beat in techno and house.

The last criterion, genre discrimination, refers to a models' ability to distinguish and generate outputs corresponding to different genres. It can be defined as the capability of a model to distinguish specific characteristics of certain genres, all when generating output using the same dataset.

In conclusion, these four criteria provide a comprehensive evaluation of the outputs of the two models, allowing us to understand their strengths and weaknesses in terms of understanding and producing music.

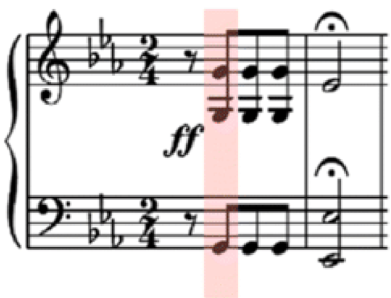
4 Methods

The methods section of this study explains the use of MIDI formats and various musical attributes for training our DrumsRNN and MusicVAE models. The specific libraries and programs utilized in this process, such as the Magenta library, will also be discussed in this section.

4.1 MIDI

We need to use some sort of data structure that can hold information about features and/or events, for instance where the instrument is played on the time axis. But also what instrument is played because we will be using more than one instrument. One way of confining these informative details into one structure is to use sheet music. It is a form of symbolic music (music stored in a notation-based format). Another way is to use MIDI (Musical Instrument Digital Interface)[Mirbeygi et al., 2022], where information about the music is stored as a tabular sequence, more precisely than sheet music. It also contains information about velocity (the loudness of a note) for instance. The difference is displayed in Figure 2.

MIDI has been the standard for almost any representation of either instruments, or drum tracks in music, therefore we will also be using MIDI for our drum sequences. There are two types of MIDI; MIDI-0 which is just one track of MIDI (e.g just drums) and MIDI-1 that contains multiple tracks (e.g bass , melody, piano and drums). For drums, there exists a standard drum mapping called the General MIDI Drum Map Format. Each MIDI note and also it's pitch has been assigned to certain drum instruments. This is shown in Figure 3.



Time (Ticks)	Message	Channel	Note Number	Velocity
60	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	63	100
0	NOTE ON	2	51	100
0	NOTE ON	2	39	100
240	NOTE OFF	1	63	0
0	NOTE OFF	2	51	0
0	NOTE OFF	2	39	0

(a) (b)

Figure 2: Representation of (a) Sheet Music (b) MIDI Source: [Mirbeygi et al., 2022]

The drum instruments discussed in this work, incorporate a typical 909 drum set that comprises 9 instruments, including a kick drum, snare drum, closed and open hi-hat, three toms, and a crash and ride cymbal. These percussive elements are the most commonly used in music that contains drums. This drum set is able to roughly reproduce the majority of popular percussive sequences. We are using a drum rack of 9 instruments (percussive elements) within 4/4 bar measures of 16th notes. The 4/4 is the so-called time signature. It specifies how many notes are counted as a beat by the denominator and the number of beats in each bar by the numerator. The most common one is time signature is 4/4 and for the sake of this thesis, we won't get into complex time signatures. The drum set explained represents one of the most used drum patterns dynamics for drum sequences.

We use Ableton to create and play MIDI files. It is a Digital Audio Interface (DAW). It also has a digital TR-909 drum kit, sampled from the original 909 physical drum machine. We have used this drumset and more jazz-like instruments to create genre-specific drums with the generated MIDI files.

4.2 Python

Due to the fact that a lot of models are implemented in Python, it was chosen as the main language for using these models for better interoperability. Python is a flexible, easy-to-use, interpreted programming language, which caused a lot of scientific researchers to use it. There are also many libraries and packages available for ML-related computing. It simplified development in these areas immensely. We have also used Python to create scripts for data pre-processing.

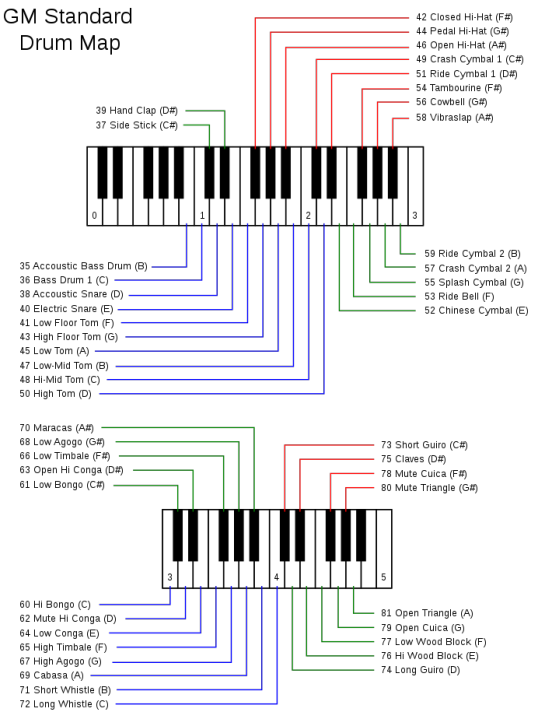


Figure 3: General MIDI Drum Maps; Source: [Staff,]

4.3 Magenta

Magenta is a powerful library for manipulating source data into usable trainable data. It is built on Tensorflow, which is an end-to-end open source platform for machine learning [Smilkov et al., 2019]. There is also an Javascript API so-called MagentaJS, but we will stick to the Python variant as explained in 4.2 Python. NoteSequence is a data representation created by Magenta similar to MIDI, but it offers many great other utilities. NoteSequence is essentially a Protocol Buffer, which is a platform-neutral mechanism for serializing structured data. Protocol Buffers can also be extended with new information without having to update code or making existing data invalid. NoteSequences can easily convert to representations that are useful for model training (e.g one shot tensors). It also includes other utilities– like slicing, quantizing, extracting musical components– that are useful when preprocessing data for training. Another useful feature is that NoteSequence are programmed so that it starts with the first note, and ends with the last note. So if your MIDI file starts with silence, it won't be taking that into account.

The Magenta library provides access to pre-trained models like MusicVAE or DrumsRNN. However, it also allows us to train our own models. We will be using the Magenta library to create a self-trained version of MusicVAE and DrumsRNN with our own datasets.

5 Models and architectures

In this section, we begin by outlining the fundamental concepts of Recurrent Neural Network (RNN) layers, which form the basis of the two models under consideration. We then provide an

in-depth examination of both models and their architectural design. Finally, we highlight the key distinctions between the two models.

5.1 RNN layers

First off, both models use RNN layers. The use of RNN layers in these models is crucial due to their capability of dependently processing data. They are able to maintain a "memory" of the previous input, allowing it to use information from the past to inform its current computations. For example, an RNN layer processes input vectors by combining them with a state vector to produce an output vector. This output vector is then used to update the state vector in the next step, allowing the model to take into account previous events. Multiple RNN layers can also be stacked by using the output of one RNN layer as input for the next.

Drums can exactly be described as a sequence of events which is essentially a long vector. It is also very important to know what has happened in the past because you need to keep repetitiveness in drums with some degree. This is why RNN layers are used in both models as starting point. The main differences between both models is the architecture they use.

5.2 DrumsRNN explained

DrumsRNN is built using an LSTM architecture, which is a type of model that utilizes RNN layers with LSTM units. RNNs suffer from the vanishing gradient problem, which occurs when dealing with long sequences of data and bigger layers. The error gradients that are used to update the network and therefore learn, become so small that they even stop changing over time. This can lead to poor performance on tasks like longer sequence drum generation. LSTM units are designed to better preserve and control the long-term memory of the RNN and therefore mitigate the vanishing gradient problem. It can make better predictions on longer sequences of data.

In general, LSTM units are the cells that are responsible for maintaining this "memory" of the previous input and output, different gates are used to control the flow of information into and out of the cell. The input gate determines what information from the current input will be added to the cell state. The forget gate determines what information from the previous cell state will be retained in the updated cell state. And the output gate determines what information from the cell state will be used to compute the LSTM unit's output. The specific mathematical details of this model are beyond the scope of this thesis.

DrumsRNN is processing drum hits as events and predicts the next drum hit based on conditional probabilities. The model is only suitable for quantized sequences. The model takes in MIDI files, classifies them, and then transforms them into one hot vectors. It automatically reduces the amount of drum instruments to nine. The model uses these one hot vectors for training. This process is shown in Figure 4.

The model is already pre-trained on thousands of MIDI files, however they have done very little curation of this training data set. The following questions arise:

1. What has it been trained on exactly?
2. What is the length of the MIDI files?

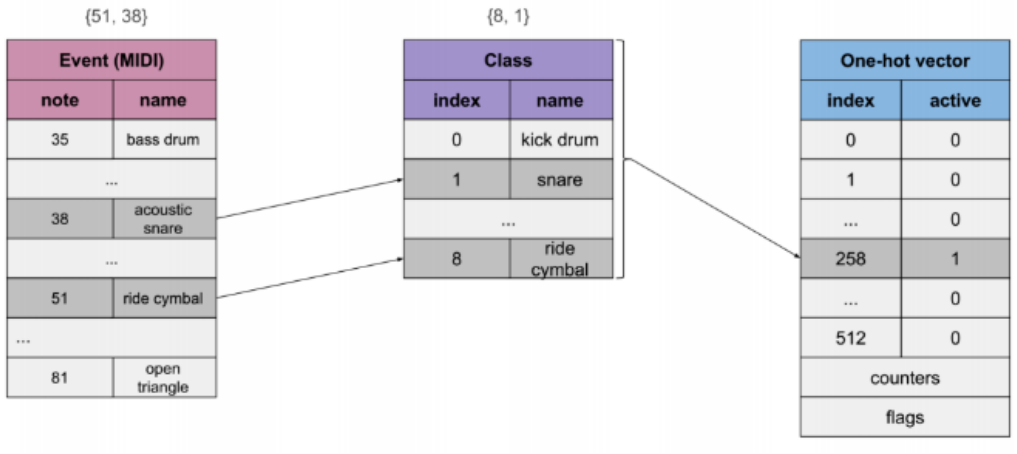


Figure 4: MIDI to One Hot Vectors; Source: [Herremans and Chew, 2017]

3. Are the MIDI files self-made loops or are they transcribed from real songs?

4. What is the genre of the drum MIDI files?

In order to answer these questions, we have decided that we train both models with the same training sets. These MIDI files will have multiple properties that are needed in order for the models to include phrasing in its generation. This will be thoroughly explained in Section 6 Training.

5.3 MusicVAE explained

The MusicVAE model introduced by Roberts et al. is able to learn a latent space of musical sequences that allows it to model long-term structure in drum percussions [Roberts et al., 2018]. A latent space is simply a representation of compressed data. It learns features from data and simplifies them in order to find patterns. Unlike DrumsRNN, the model uses a Variational Autoencoder (VAE) architecture. Autoencoders have these latent vectors encoded through RNN layers with LSTM units, where a lot of holistic and informative features are contained. These latent vectors can then be decoded back into, for instance, musical sequences, that emphasize those features. RNN language or LSTMs models mostly fail to include these informative latent encodings [Ok et al., 2022]. A hierarchical decoder is added to avoid the 'posterior collapse' problem, which is common amongst recurrent VAEs. It is able to utilize the latent code more efficiently and therefore provide better interpolation, reconstruction and phrasing performance.

Several studies have reported success on short-term sequences including such an autoregressive encoder like the CNSL architecture [Makris et al., 2019]. Our interest in MusicVAE comes from the fact that they use a VAE architecture with hierarchical decoder, which addresses the aforementioned issue of modeling long-term structure. The MusicVAE architecture and its implementation provide a powerful basis for exploring the long-term generation of phrasing in drum percussions.

5.4 LSTM architecture vs VAE architecture

Both models are created to capture temporal and structural dependencies in the data, in order to generate drum sequences that contain some sort of phrasing and 'knowledge' about the concept

represented in drums (tension, release). However, both models use different architectures to achieve these goals. VAE's are more leaning towards the idea of compressing the data into an compact representation, and then decompressing the data from this compact representation to generate sequences from. It is more of a clustering process that find features and capture these in latent vectors. On the other hand, RNN's work sequentially and use a loop structure to allow the network to maintain a state that depends on previous states. This allows the model to capture dependencies in data. It doesn't build a representation of the data, but it uses their internal LSTM state to model capture the temporal and structural dependencies in the data.

Both architectures have their limitations and benefits, for example an VAE architecture has more trouble with complex, non-structured data [Wild, 2018] but is said to better construct sequences with structure. An LSTM architecture is way slower to train because it must process an entire sequence before it can update the internal state. However, it can always produce a high quality sample, because it doesn't have to sample from a latent space (which might be not large enough for example).

Comparing these two models is really interesting because it allows us to see if one model is superior in generating longer drum sequences, or if they have specific strengths and weaknesses. Or what happens when the models are asked to generate genre-specific drums. It might also be that one model somehow still isn't even capable of generating coherent drums, and we can examine why that might be. This brings us back to our research question 'How do MusicVAE and DrumsRNN compare in their ability to generate 2-minute long, realistic and evolving drum sequences?' which is the main focus of this study. In order to make a good comparison between these models, we will not only examine the models themselves, but also the input data that they use. We will do this by varying the length and genres of the input data and observing how the models perform under different conditions.

6 Experiments

The important thing of both models is to capture long-range structure on a variety of data. Perceiver AR is a model which can generate long solo piano pieces with clear long-term coherence and structure [Hawthorne et al., 2022]. However, the input data also plays a huge role. Their training set is a bit different than ours because it is not based on MIDI files but on audio files, however they stated that it is important to have long enough training data, because short training data (5-10 seconds) is not likely to contain any long-term structure. They also stated that using too long training data was actually harmful for their algorithm.

Based on this information we will be using MIDI files that exhibit stylistic and structural coherence that spans 2 minutes. Another important thing is that the model is able to actually understand longer MIDI files. That means that it can infer the phrasing and structure contained in the longer training data. We will be looking at both models to see how they achieve this. This section will provide details on the data collection, preprocessing and model training.

6.1 Training Data

Our first instinct was that in order for the models to generate drums with phrasing and enough changes, it should be trained on data that contain phrasing. Since we had trouble finding full-length,

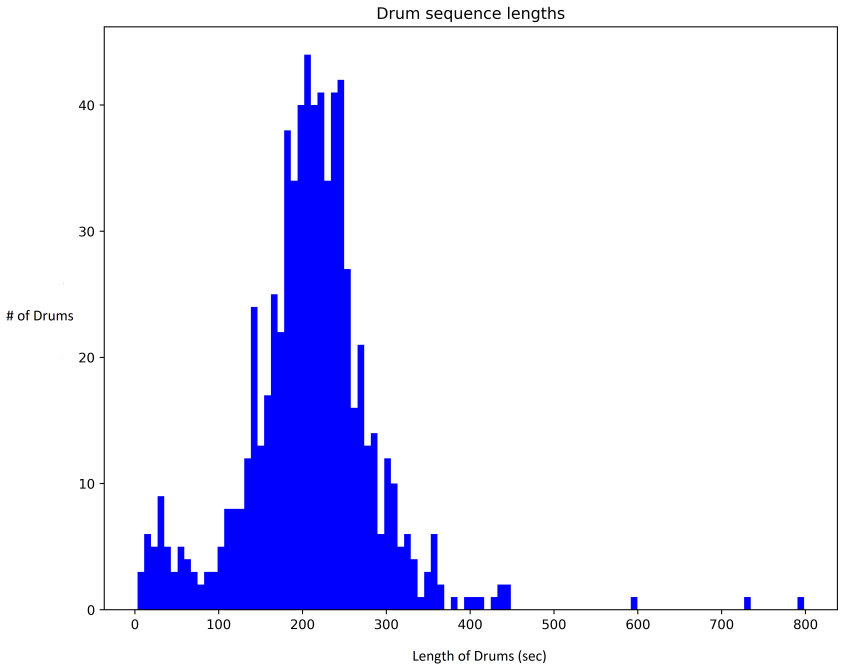


Figure 5: Histogram frequency of data vs drum length in seconds

multiple minutes MIDI files just containing 909 drums, we started by ten different MIDI files ourselves containing drums that span 2 minutes. They are made and played back on 130BPM and can be found at Github. These MIDI files have long-term coherence and structure by containing different sections and alternating between different percussive elements. The ten MIDI files will be used for training both algorithms. We created these ten MIDI files by downloading the Drum Machines sample pack from Ableton. It included some MIDI files of one bar length. We used these MIDI files as a basis to create 2 minute drum sequences. We have taken inspiration from techno tracks from the 90's E.G: The Martian - Stardancer and Jeff Mills - The Bells.

We carried out explorative experiments with these drum tracks, however it seemed that just twenty tracks wasn't enough for training. The model was underfitting. We have used the Lakh MIDI Full dataset created by Colin Raffel [Raffel, 2016] to extend our data. It is a full collection of 176,581 deduped MIDI files. They contain contain, drums, chords, melodies and also full songs. Since no attempt was made to curate the dataset, many MIDI files were also likely to corrupt. We are in need of only drum MIDI files that have a dance 4/4 beat and MIDI files that are classified as jazz/soul songs. As a result, we need to prepare the Lakh dataset before feeding it into the model.

The Python script takes the full dataset library and loops through the data checking if the MIDI files are valid and if they contain a drum track which can be used. As explained in 4 Methods, MIDI's can have multiple tracks each for their own purpose: One for drums, one for piano chords, one for main melodies etc. It is important to deconstruct the MIDI's and obtain the drum track(s). PrettyMIDI is a Python library which offers tools to manipulate and change MIDI's.

We extracted 715 MIDI files (7%) out of 10000 MIDI tracks. As seen in Figure 5, the drum tracks

are centered around 200 seconds duration on average and therefore are usable. Since this dataset consisted of mostly songs in MIDI format, the drum tracks also contained phrasing and enough changes to be viable in our research.

Since we also want to compare the behaviour of the models when generating drums from different genre dataset, we have extracted 739 MIDI files that are labeled as soul and jazz. Soul and jazz are both genres that has its roots in blues, and they contain significant other drums than house and techno. We have used the Lakh LMD-matched dataset for this, since they have been matched with MSD (Million Song Database). When the song is correctly matched with the MSD tag, we can obtain the genre and other specifications of that song using Last.fm API. All the MIDI files, scripts and references can be found at Github.

6.2 Training DrumsRNN

Magenta has two DrumsRNN models trained with different configurations; a single drum and a drum kit. The drum kit configuration allows for polyphony which means that multiple drums can hit at the same time. This configuration has room for 9 drum instruments, which is enough for generating drums as explained in 4 Methods.

DrumsRNN has an attention mechanism that allows the model to focus on specific parts of the input sequence when making predictions. By adjusting the *attn_length* hyperparameter, the model controls how much of the input sequence the model takes into account when making its predictions. 32 is two bars, 64 is four bars respectively. A larger attention length will allow the model to consider more of the input sequence, potentially leading to more accurate predictions.

When training the DrumsRNN model, we have set the attn_length on 64. We are trying to convey phrasing into our generation process, and by allowing the model to focus on a larger part of the input sequence, it can better recognize and incorporate these changes. We have also lowered the model size to RNN layers of [64,64]. This way it could better resonate with the size of our datasets and have less static output.

DrumsRNN has been trained on the techno-house dataset, and the jazz-soul dataset with batch size of one. In order to prevent overfitting, we experimented with different batch sizes. However, when we used batch sizes higher than one, we found that the model tended to overfit, likely due to the small size of the dataset. As a result, we chose a batch size of one for the final model. For the remaining hyperparameters, we used the default settings, including a clip norm of 3 and a drop out probability of 0.5.

We use Tensorboard to monitor our training process [Abadi et al., 2015]. Tensorboard is used to visualize the model's accuracy and loss per timestep, with graphs for the training set and evaluation set. The DrumsRNN model includes a function that automatically splits the dataset into a training and evaluation set, which are used during the training process. We have set the split on 10%, that means that 10% is evaluation data, and 90% is training data. We have chosen for this because it is the default and common way for splitting the dataset in this model.

6.3 Training MusicVAE

MusicVAE has multiple configurations curated by the Magenta team. These include configurations for melodies, drums and also multiple instruments combined. The basic configurations for drums are as following:

- *cat-2bar_drums* (2 bar sequence with 9 drum instruments)
- *nade-2bar_full* (2 bar sequence with 61 drum instruments)
- *groovae-4bar* (4 bar sequence trained on real drummers)

Some of these configurations have extensions, for example *cat-2bar_drums* also has a big model version. And *groovae-4bar* has some 2-bar extensions which allows it to "humanize" a drum sequence.

The issue with previous configurations is that they have a limited maximum sequence length of only four bars, or roughly 10 seconds. We have developed a new MusicVAE configuration using the *cat-2bar_drums* model. This new configuration allows for training and generating drum sequences up to 2 minutes long, as we have increased the maximum number of bars to 60, rather than the previous limitations of two or four. This MusicVAE configuration is also specifically designed to only take in 9 drum instruments, to be in line with the DrumsRNN model which also has the same amount of drum instruments.

When generating drum sequences with default configuration *cat-2bar_drums* configuration. It only generated important features, like verses, breaks and buildups. Examples of these are; snare rolls (every 16th note a snare so this creates tension); parts without any kick (only percussion/high hats are playing). This is due to the nature of MusicVAE of encoding musical components into features, and later combining and decoding it back. This way it learns the different aspects of a drum, and with the LSTM encoder it can construct some kind of long-term structure in the drums. So when we generate up to two minutes of drums, it should contain some changes over time.

The hyperparameters of this configuration are chosen carefully as well. Since we are training DrumsRNN with [64,64] RNN layers, batch size of 1 and learning rate of 0.001. We have also configured the MusicVAE model configuration to these settings, however after training it wasn't able to generate a long sequence. Only the first few seconds were filled with correct drum information. This is probably because for the MusicVAE model, it had too little RNN layers. It can't process the amount of data correctly, therefore only the first few seconds were filled with holistic information. When using [256,256] layers, The model required around three times the training time of using [128,128] layers. However the output can be considered of the same quality. The behaviour doesn't really improve when you make the model size bigger. Therefore, we extended the RNN layers to [128,128] which could fill the entire drum sequence with drum information and being reasonable fast. It is in fact also the default layer size for example *cat-2bar_drums*.

MusicVAE has been also been trained on the techno-house dataset, and the jazz-soul dataset with batch size of one. In order to prevent overfitting, we experimented with different batch sizes. This model also tended to overfit when using higher batch sizes, probably because the dataset is too small for higher batch sizes. For the remaining hyperparameters, we used the default settings, including free_bits of 48, z_size of 128, max_beta of 0.2 and a sampling schedule that uses an inverse_sigmoid function with a rate of 1000.

MusicVAE employs the same monitoring library as DrumsRNN, which is using Tensorboard [Abadi et al., 2015]. However MusicVAE did not include with a function to automatically split the dataset into a training and evaluation set. So therefore we have manually divided the dataset, using a 90% training set and 10% evaluation set, the same way as DrumsRNN.

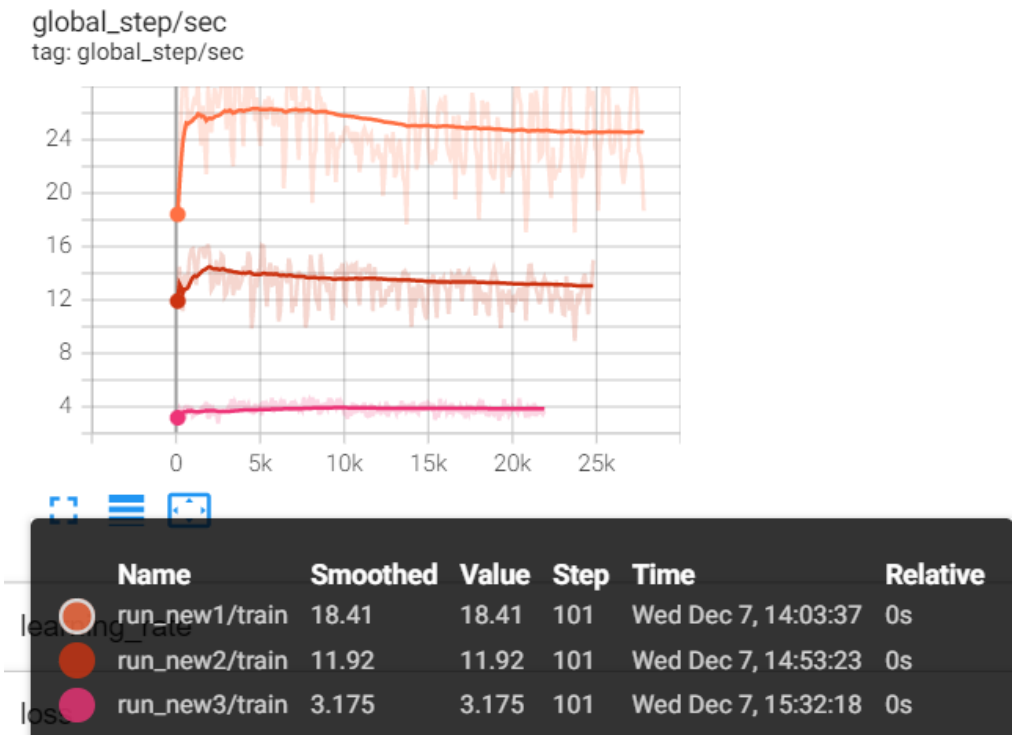


Figure 6: Training times of model size comparison where run_new1 is [64,64, run_new2 is [128,128], run_new3 is [256, 256]; Source: Tensorboard [Abadi et al., 2015]

7 Evaluation

In this section, we begin by demonstrating the convergences of both models. We then provide an in-depth evaluation of both models' generated drum sequences regarding our research topics.

7.1 Generation

Both models have a temperature parameter when generating drum sequences. The higher the temperature, the more 'çreative' and 'randomized' the output gets. The lower the temperature, the more 'safe' and 'basic' the output gets. In general we will be outputting sequences with a default temperature of 1.0 which is the perfect middle ground for our research.

7.2 Results

We trained and tested two different models for drum generation: DrumsRNN and MusicVAE. We evaluated the models based on their ability to generate realistic drum patterns that contain phrasing, as well as their ability to generate patterns in a specific style. After the written evaluations, the results are presented in a table for better understanding. It is important to note that these results are based on the limited scope of the study and may not be generalizable beyond the conditions tested.

7.2.1 Convergence

To demonstrate the convergence of the models, we present Figures 7 and 8 displaying the training and evaluation loss over time. The training loss is the error of the model on the training dataset, and the evaluation loss is the error of the model on the validation dataset. In Figure 7 is DrumsRNN trained with the jazz-soul dataset and in Figure 8 is MusicVAE trained with the techno-house dataset. While the convergence on other datasets is not shown here, the graphs presented provide a solid basis for assuming similar convergence behavior on other unseen datasets.

Both models' training and evaluation loss decrease over time with the validation loss slightly higher than the training loss. This indicates that the model is improving and generalizing well to the validation dataset. Since the DrumsRNN takes almost 5 times as long to train as MusicVAE, we failed the resources to train DrumsRNN with as many steps as MusicVAE. However, both models are not underfitting nor overfitting, so we can state that the models are fitted well enough for the generation of our drum sequences.

7.2.2 Size of training data

The size of the training data can have a significant influence on the behavior of a trained model. Generally, more data allows the model to learn more about the patterns and structures in data, however too much data can cause noise and increase time and resources required to train the model. We compared the performance of DrumsRNN trained on a 1500-sample techno-house dataset to the performance of a 700-sample techno-house dataset. We found that the output of DrumsRNN trained on the 1500-sample dataset contained more 'noisy' drums than the output of the model trained on the 700-sample dataset. The drums become less structured and more random, resulting in the instruments being played at seemingly random times. One example of this is the amount of

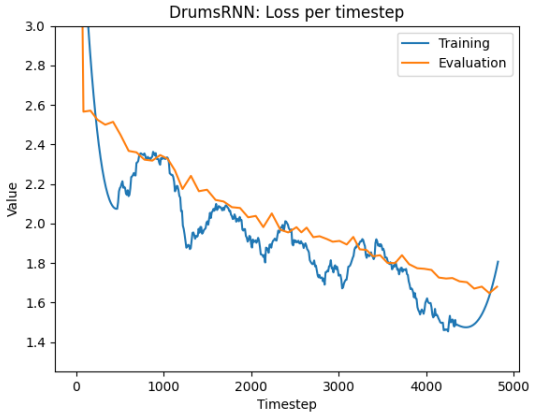


Figure 7: DrumRNN: Jazz-soul convergence graph

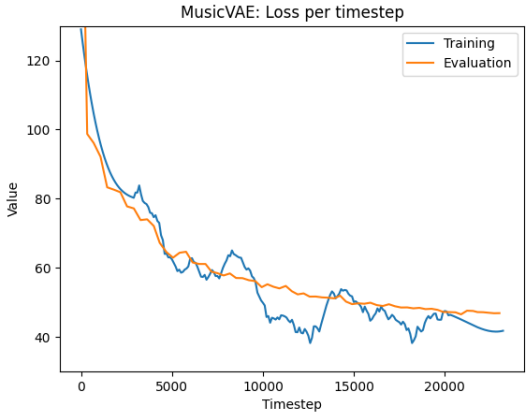


Figure 8: MusicVAE: Techno-house convergence graph

snare rolls (which are snares on every 16th beat) used in the drum sequence. Normally they are a great tool for buildups but not if they are used randomly throughout the song. It makes the drum sequence not pleasant to listen to.

`Reference: DrumsRNN/techno-house_1500/drum_seq_snarerolls`

However, another drum sequence generated has a consistent house beat characterized by a hat on every off beat, a snare on the second and fourth beat, and a kick on every one beat. In addition to this structural foundation, the sequence also displays a sense of musical phrasing, as demonstrated by the addition of high percussions (e.g. hats and rides) at 20 seconds and later on the addition of more snares at 36 seconds to create tension and build up to a cohesive drum groove.

`Reference: DrumsRNN/techno-house_1500/drum_seq_full`

The output of the trained DrumsRNN model on a 700-sample dataset contained some more minimalistic drums, that means less instruments playing at the same time. But every drum sequence generated is pleasant to listen to.

`Reference: DrumsRNN/techno-house_700/drum_seq_groove`

We have also tested MusicVAE with twice as much data but we did not observe any difference between both datasets. In general, the 700-sample dataset trained DrumsRNN contained a more balanced distribution output, with some minimalistic drums and some fuller drums. Due to these reasons we have decided to continue with the 700 curated dataset for training both models.

7.2.3 Genre discrimination and originality

We compared the performance of MusicVAE and DrumsRNN on a jazz-soul dataset and a techno-house dataset, which are two distinct genres in drum characteristics. The main difference between jazz-soul and techno-house is the way that in jazz-soul the drums are typically characterized by their use of rhythms in the high-hat, as well as a focus on ride cymbal patterns [Gottlieb, 2011]. The

hi-hat cymbals are often used to provide a more distinct randomized feeling or groove, sometimes also even played without any kick drum or snare. In contrast, techno-house drums are essentially focused on a steady four-on-the-floor kick pattern, and a snare on every second kick, which is the driving factor [Hydlide, 2018]. The hi-hat cymbals are often used to provide a consistent, driving pulse and are almost always played with a consistent pattern.

MusicVAE could grasp the concept of the jazz-soul drums way better than DrumsRNN, particularly in terms of capturing the high hat grooves, also the kick drum was less prominent in the jazz-soul drums which is more common in jazz.

`Reference: MusicVAE/longbar_jazz-soul/highhatgroove`

DrumsRNN struggled to capture the distinct drums used in jazz and soul and operated more randomly. DrumsRNN did not produce different drums when generating jazz-soul versus techno-house.

`Reference: DrumsRNN/jazz-soul_700/decentdrumsbutnotjazz`

In this example, DrumsRNN is limiting itself to a four-on-the-floor kick pattern, the drums are correct although it doesn't really capture any jazz structure. Both models were successful in producing techno-house music, but MusicVAE generated more creative and varied pieces with more diverse rhythms. It used more toms in this arrangement, for instance.

`Reference: MusicVAE/longbar_techno-house\thriving hhs and toms`

DrumsRNN is less 'diverse' and more 'simplistic'. Overall, these results suggest that MusicVAE is more capable of capturing the nuances of jazz and soul music compared to DrumsRNN. It also could generate more diverse drums where DrumsRNN appeared to generate more consistent.

7.2.4 Phrasing

We also compared the phrasing capabilities of MusicVAE and DrumsRNN in the generation of techno-house drums. The models were pretty good at changing elements and keeping the listener entertained. MusicVAE generated a sort of breakdown: the kick changed for few bars to only playing on every second beat, and then going back to every beat.

`Reference: MusicVAE/longbar_techno-house\little-breakdown`

DrumsRNN also provided a bit structure, some of the generated pieces were developing their elements, starting with only the kick drum, then the snare, and then high hats, and after one minute it added ride cymbals.

`Reference: DrumsRNN/techno-house_700/some-phrasing`

In the following piece, MusicVAE also generated a nice interpolation between the crash cymbal playing at 20 seconds and the toms which continues throughout the drum sequence.

`Reference: MusicVAE/longbar_techno-house\nice phrasing-toms-rides`

However, overall it lacks patterns that repeat and are changed within the arrangement. Drum patterns are typically divided into bars and phrases, but it appears that MusicVAE and DrumsRNN did not correctly identify these bars and phrases. Instead of inserting or modifying elements at the end of a bar and continuing this until the end of the phrase or bar, they appeared to make changes at arbitrary points.

	DrumsRNN	MusicVAE
Short-term phrasing	+	++
Long-term phrasing	-	-
Originality	-	++
Genre discrimination	- -	+
Realism	+	++

Table 1: Visualization of the evaluation of both models

7.2.5 Realism

In terms of the realism and correctness of the generated drum patterns, MusicVAE outperformed DrumsRNN. MusicVAE generated more diverse and cohesive drum patterns, while DrumsRNN's patterns were less dynamic and featured less interaction between the various drum elements. For example, MusicVAE incorporated both ride and high hat cymbals in a way that contributed to the overall groove, whilst in DrumsRNN most of the time only one cymbal was playing.

`Reference: MusicVAE/longbar_techno-house\thriving hhs and toms`

Here, as explained above, DrumsRNN output is more static than MusicVAE.

`Reference: DrumsRNN/techno-house_700/static drums`

Despite these differences, both models were able to produce drum sequences that were on beat and had a consistent rhythm. Table 7.2.5 shows a compact visualization of how well the models were able to incorporate these concepts into their drum generation.

8 Discussion

To understand the different results of both models, we can refer back to the main research question that we formulated at the beginning of this thesis: "How do MusicVAE and DrumsRNN compare in their ability to generate 2-minute long, realistic and evolving drum sequences, and how does this performance vary with the size of the training data, the genre of the music, and the use of multiple genres for training?". In order to answer this question we first look at our sub-research questions regarding data size and genres.

RQ1a: How does the size of the training data influence the output of the models?

Our experiments showed that increasing the data size did not generate better results. In fact, with DrumsRNN it worsened the output because the output got messier the more phrasing features it found. MusicVAE's output didn't get better when using twice as much training data. This finding is not in line with the general consensus of larger training datasets can lead to a improved model performance. A possible explanation for the lack of improvement in the models' performance despite

an increase in the amount of training data is that the model may not have been trained for long enough or with enough data to show significant changes. It is also possible that the model is not complex enough to effectively learn from the additional data. DrumsRNN took longer to train than MusicVAE. It is certainly possible that if we had more resources to train DrumsRNN for a longer period, the output could have improved. This way it could be explained that MusicVAE stayed flat and DrumsRNN detoriated.

Another reason might be that the data does not contain enough structure to improve the both models. Although the datasets were randomly sorted, the smaller dataset can contain better structured drums by chance. Musical data is very complex because every song is structured differently. In order to process this complex data, the models need to be improved because otherwise it is limited to only using curated 2-bar datasets for example. An improvement could be for example adding time-markers that indicate different sections like [Makris et al., 2022] has proposed. This way, the model can better understand sections because it is reducing the data's complexity. In general, we found that other factors such as the characteristics and complexity of the data influences the performance more than purely the training data size.

RQ1b: How do the models perform when trained on different genres of music?

When trained on different genres, MusicVAE was able to generate realistic drum sequences, understanding the differences to some degree in drums between techno-house and jazz-soul. DrumsRNN appeared more randomly, there wasn't really a difference in drums in both genres. This might be the reason of different architectures. MusicVAE learns a compact concept of the drums, and then samples from it. It did a better job of capturing the differences between genres. Whereas DrumsRNN is learning sequentially, it has more trouble learning these distinct features, and therefore is behaving more 'randomly'. Another explanation could be that DrumsRNN needs more training data in order to make full use of the model, when DrumsRNN was trained with twice as much data, it did capture some distinct features, for example snare rolls (which are a common concept in the techno-house genre) in the techno-house dataset. However, when generating it didn't fully understand when snare rolls were used so it placed them randomly again, and therefore got way messier output than when using less data.

RQ1c: How do the models compare in terms of the variety or diversity of drum sequences that they are able to generate?

We found that MusicVAE and DrumsRNN differ in terms of the variety or diversity of drum sequences that they are able to generate. MusicVAE generated a wider range of different grooves, high hat patterns, and tom patterns in its output compared to DrumsRNN, which tended to produce more "static" drum sequences. However, the drum sequences generated by DrumsRNN were still technically accurate. This is because MusicVAE uses the VAE architecture to learn a continuous latent space that represents the full structure of the input music. This allows the model to be more versatile when generating drum sequences, because it samples from this latent space. In contrast, DrumsRNN uses a conditional probability at every time step to decide what the next drum event/events should be. This means that the output of this model is more reserved as it is dependent on the previous events and is not able to explore a wider range of possibilities. Despite this, DrumsRNN consistently produced high-quality samples when generating, whilst MusicVAE sometimes generated samples that were lacking in drum content, as it was not able to effectively fill and use the latent space.

RQ1: How do MusicVAE and DrumsRNN compare in their ability to generate 2-minute long, realistic and evolving drum sequences, and how does this performance vary with the size of the training data and the genre of the drums?

Our analysis in the previous research questions showed that both models were able to generate drum sequences that were realistic and evolving to some degree, for example it included snare movement and some parts were higher intensity than others (tension and release). But in spite of that, neither model seemed to fully understand a complete song structure, such as breakdowns and phrases. While DrumsRNN was able to structure the songs more like its input data to some extent, it is debatable whether this is actually beneficial, as both models did not have a strong understanding of the concept of bars and phrases, so it is essentially just copying the input data rather than inferring the underlying structure.

Overall, MusicVAE and DrumsRNN are both capable of generating 2-minute long, realistic drums, but they are not so much 'evolving' in a way that drums do in an actual song. However, MusicVAE demonstrates a superior understanding of genre and originality, resulting in the ability of generating genre-specific drums and more diverse drums. DrumsRNN, on the other hand could work better in live settings since it is generating sequentially, it always produces a sample with enough drum content.

9 Conclusions and Further Research

When talking about music, not many work has been done on drum sequences. Most musical models are designed for synths and melodies, that whilst drums are one of the most important aspects in modern dance music for example. This inspired us to work only with drums and especially their ability to understand song structure like verses, choruses and short term phrasing (changes in drums over time). It also inspired us to investigate the general understanding of music in these models, specifically in regards to genre and diversity.

We wanted to determine whether these models could understand and replicate the structure and development of a song in the same way that a human drummer might. Given the limited number of drum sequence generation models currently available, we sought to provide new insights into the capabilities and limitations of MusicVAE and DrumsRNN.

By comparing the performance of these two models, we hope to gain a deeper understanding of the factors that contribute to the generation of realistic and structured drum sequences. Another important goal of our study was to examine the influence of various factors on the generated drum sequences, including the size of the training data, the genre of the drums, and the level of originality contained in the generated patterns. That led us to the following research question: "How do MusicVAE and DrumsRNN compare in their ability to generate 2-minute long, realistic and evolving drum sequences, and how does this performance vary with the size of the training data and the genre of the drums."

We have chosen MusicVAE and DrumsRNN, as they are the only models existing for just drum generation. Other than that is it interesting because both models have a different architectures and mechanisms to keep in account long-term dependencies in the drums. MusicVAE is a type of variational autoencoder (VAE). It uses a VAE architecture to learn the underlying structure of the music in the dataset, and can then be used to generate new, unique music based on that structure. DrumRNN, on the other hand, is a type of recurrent neural network (RNN) with LSTM cells. It

uses an LSTM architecture to learn and generate drum sequences, and can be used to generate new, unique drum patterns based on the patterns it has learned. It also uses an attention mechanism to process the input more effectively.

The differences is that MusicVAE uses an encoder-decoder mechanism to find all features of the drums and compresses them into a huge latent space, whilst DrumsRNN uses the LSTM cells to save the information and then learn the features. When generating, MusicVAE samples drums from this latent space, whilst DrumsRNN generates sequentially where each drum hit is based on a learned conditional probability.

Overall, our results showed that both models were able to generate realistic drum sequences spanned up to two minutes without losing it's correctness in the drums. Both models also achieved some phrasing and development, for example there were high hat movements, and clap movements in both models. There were periods with less intensity and periods with higher intensity (less instruments, more instruments).

However both models did fail to really model a song structure in their generations. We found that both models had difficulty incorporating specific song structures such as breakdowns or sections with verses and choruses into their generated drum sequences. In terms of performance of both models, MusicVAE seemed superior when it comes to generating different genres and generating with more original pieces. DrumsRNN had less variation in grooves, less variation in the instruments and less cohesive sections. It seemed to generate drums more randomly, without having a proper knowledge of the concept 'drums'. That means that its only concern is to generate drums that are correct, and not so much worry about thematic development and grooves. This is in line with the original design concept of DrumsRNN [Magenta, 2016].

Another advantage of MusicVAE is that it could better discriminate between genres. MusicVAE was capable of keeping the difference in kicks and high hats between genres in mind whilst generating drums. One of the main reasons for this difference in behaviour is because MusicVAE builds a structured repertoire of which it samples instead of the sequential generation of DrumsRNN. This allowed for more variation because it has a latent space full of variations, and it allowed for better understanding of drums in general.

9.1 Further research

Further research could be conducted to improve the understanding of song structure and genre in drum sequence generation models. The biggest problem we found is that the models had a weak understanding of actual music theory. So the models could be improved by incorporating additional music theory concepts into the models, such as the use of musical phrases, breakdowns, tension and release.

Another way to improve the current models is to train the models on larger and more diverse datasets that include a wider range of song structures and genres. However, it is important to use techniques such as data augmentation to reduce the complexity in the training data and include enough variation to help the generalisability of both models.

Additionally, research could be conducted to optimize the hyperparameters and architecture of the models to better handle the complexity of music and improve the quality of the generated drum sequences. The models could also be improved by exploring the use of transfer learning and fine-tuning to adapt existing drum generation models to new musical styles or genres. It might better understand different genres and their structures. An example of this is training the model

on a large dataset of drum tracks in one musical style, and then fine-tuning the model on a smaller dataset of drum tracks in a different style, to create more versatile and adaptable drum generation models. The model can therefore understand the general concept of drums and then tune to a specific genre.

We have also found that with 9 drums, it can be a bit too limited to create a drum sequences which evolves and 'lives' on its own. So a final suggestion for further research could be to use a wider range of instruments in the drum generation model to create more diverse and rich sounds. This could include incorporating a variety of percussion instruments, such as conga's and shakers. The DrumsRNN model needs to be adjusted cause it can currently only take 9 drum instruments.

References

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Choi et al., 2016] Choi, K., Fazekas, G., and Sandler, M. (2016). Text-based lstm networks for automatic music composition.

[Dhariwal et al., 2020] Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., and Sutskever, I. (2020). Jukebox: A generative model for music.

[Dostál, 2012] Dostál, M. (2012). Musically meaningful fitness and mutation for autonomous evolution of rhythm accompaniment. *Soft Computing*, 16.

[Gottlieb, 2011] Gottlieb, D. (2011). *The Evolution of Jazz Drumming: A Workbook for Applied Drumset Students*. Hudson Music, LLC, pap/com edition.

[Hawthorne et al., 2022] Hawthorne, C., Jaegle, A., Cangea, C., Borgeaud, S., Nash, C., Malinowski, M., Dieleman, S., Vinyals, O., Botvinick, M., Simon, I., Sheahan, H., Zeghidour, N., Alayrac, J.-B., Carreira, J., and Engel, J. (2022). General-purpose, long-context autoregressive modeling with perceiver ar.

[Helms et al., 2018] Helms, M., Ault, S. V., Mao, G., and Wang, J. (2018). An overview of google brain and its applications. In *Proceedings of the 2018 International Conference on Big Data and Education*, ICBDE '18, page 72–75, New York, NY, USA. Association for Computing Machinery.

[Herremans and Chew, 2017] Herremans, D. and Chew, E. (2017). Morpheus: Generating structured music with constrained patterns and tension. *IEEE Transactions on Affective Computing*, PP:1–1.

[Hiller and Isaacson, 1958] Hiller, jr., l. a. and Isaacson, l. m. (1958). musical composition with a high-speed digital computer. *journal of the audio engineering society*, 6(3):154–160.

[Hopfield, 1982] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558.

[Hydlide, 2018] Hydlide (2018). Basic Elements House Music.

[Informatik et al., 2003] Informatik, F., Bengio, Y., Frasconi, P., and Schmidhuber, J. (2003). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *A Field Guide to Dynamical Recurrent Neural Networks*.

[Lattner and Grachten, 2019] Lattner, S. and Grachten, M. (2019). High-level control of drum track generation using learned patterns of rhythmic interaction.

[Magenta, 2016] Magenta (2016). Drumsrnn: Generating drum sequences with recurrent neural networks. https://magenta.tensorflow.org/drums-rnn.

[Makris et al., 2017] Makris, D., Kaliakatsos-Papakostas, M., Karydis, I., and Kermanidis, K. L. (2017). Combining lstm and feed forward neural networks for conditional rhythm composition. In Boracchi, G., Iliadis, L., Jayne, C., and Likas, A., editors, *Engineering Applications of Neural Networks*, pages 570–582, Cham. Springer International Publishing.

[Makris et al., 2019] Makris, D., Kaliakatsos-Papakostas, M., Karydis, I., and Kermanidis, K. L. (2019). Conditional neural sequence learners for generating drums' rhythms. *Neural Comput. Appl.*, 31(6):1793–1804.

[Makris et al., 2022] Makris, D., Zixun, G., Kaliakatsos-Papakostas, M., and Herremans, D. (2022). Conditional drums generation using compound word representations.

[Mirbeygi et al., 2022] Mirbeygi, M., Mahabadi, A., and Ranjbar, A. (2022). Speech and music separation approaches - a survey. *Multimedia Tools and Applications*, 81.

[Ok et al., 2022] Ok, C., Lee, G., and Lee, K. (2022). Informative language encoding by variational autoencoders using transformer. *Applied Sciences*, 12(16).

[Raffel, 2016] Raffel, C. (2016). Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching.

[Roberts et al., 2018] Roberts, A., Engel, J., Raffel, C., Hawthorne, C., and Eck, D. (2018). A hierarchical latent vector model for learning long-term structure in music.

[Saharia et al., 2022] Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., Salimans, T., Ho, J., Fleet, D. J., and Norouzi, M. (2022). Photorealistic text-to-image diffusion models with deep language understanding.

[Smilkov et al., 2019] Smilkov, D., Thorat, N., Assogba, Y., Nicholson, C., Kreeger, N., Yu, P., Cai, S., Nielsen, E., Soegel, D., Bileschi, S., Terry, M., Yuan, A., Zhang, K., Gupta, S., Sirajuddin, S., Sculley, D., Monga, R., Corrado, G., Viegas, F., and Wattenberg, M. M. (2019). Tensorflow.js: Machine learning for the web and beyond. In Talwalkar, A., Smith, V., and Zaharia, M., editors, *Proceedings of Machine Learning and Systems*, volume 1, pages 309–321.

[Staff,] Staff, M.

[Vogl and Knees, 2017] Vogl, R. and Knees, P. (2017). An intelligent drum machine for electronic dance music production and performance.

[Wild, 2018] Wild, C. M. (2018). With Great Power Comes Poor Latent Codes: Representation Learning in VAEs (Pt. 2).