Text Conditioned Symbolic Drumbeat Generation using Latent Diffusion Models

Pushkar Jajoria (0009-0006-3789-5372)[1,2], **Dietrich Klakow**[1], and **James McDermott** (0000-0002-1402-6995)[2]

[1]*Spoken Language Systems (LSV)*, **Saarland University**, Saarbrücken, Germany
[2]*Department of Computer Science*, **University of Galway**, Galway, Ireland

ABSTRACT

The challenge of generating coherent and novel MIDI drumbeats conditioned on text prompts remains largely unsolved, primarily due to the scarcity of well-annotated datasets linking text and MIDI drumbeats. Existing models have made strides in AI-generated music, yet they often fall short in producing high-quality drum beats that also align well with textual prompts.

This study introduces a text-conditioned approach to generating drumbeats with Latent Diffusion Models (LDMs). We use informative conditioning text extracted from training data filenames. By pretraining a text and drumbeat encoder through contrastive learning within a multimodal network we align the modalities of text and music closely. Additionally, we examine an alternative text encoder based on multi-hot text encodings. Inspired by music's multi-resolution nature, we train the MIDI autoencoder using a novel LSTM variant, MultiResolutionLSTM (MRLSTM), designed to operate at various resolutions independently. In common with recent LDMs for image generation, we also speed up the generation process and bring down the generation time for a single drumbeat to **1.1 seconds**[1] by running diffusion in the autoencoder latent space.

We demonstrate the originality and variety of the generated drumbeats by measuring distance (both over binary pianorolls and in the latent space) versus the training dataset and among the generated drumbeats. We also assess the generated drumbeats through a listening test focused on questions of quality, aptness for the text prompt, and novelty. BERT model achieved quality and aptness scores comparable to the dataset drumbeats (differences of -1.99% and +5.64% respectively), while exhibiting a **22.26%** improvement in novelty.

1. INTRODUCTION

Research in AI-generated music has seen fast progress in recent years [1–6]. Some recent deep learning models have successfully generated realistic-sounding music [7] by training on large datasets like *The Lakh MIDI Dataset* [8]. Although researchers have created models that can generate drum accompaniments (see Section 2), creating coherent and novel MIDI drumbeats conditioned on text prompts remains a challenge due to a lack of a well annotated dataset. In this research we show that such a model can be trained by taking inspiration from state of the art text-conditioned *image* generation models [9–12] and extracting descriptive tags from the folder structure.

Contrastive Language Image Pretraining (CLIP) jointly trains a text encoder and an image encoder by aligning both modalities in a common latent space. This latent space can then be used for various downstream tasks, including zero-shot and one-shot classification, as well as text-conditioned image generation [9]. In the context of diffusion models, a text-conditioned Denoising Diffusion Probabilistic Model (DDPM) utilizes text embeddings derived from the CLIP-trained text encoder to guide the image generation process. Specifically, the text prompt is first encoded into a latent representation, which is subsequently integrated into the denoising steps of the DDPM. This integration helps to steer the reverse diffusion process, ensuring that the generated images are semantically aligned with the input text [12].

Recent advances in Text-Conditioned Image Generation using diffusion models have laid the groundwork for generative systems, particularly when applied within latent spaces as opposed to the more natural pixel space [12] in the case of images. This shift to latent space has the benefit of dealing with a more compressed representation of data, which can lead to more efficient processing and the pretrained encoder-decoder can lead to potentially superior generation outcomes using the compressed representations. The adoption of diffusion techniques in latent space, along with conditioning diffusion on extraneous variables like text, not only improves the model's stability during training but also enhances the quality and relevance of outputs. This paper builds upon these advantages for the case of midi drumbeat generation.

Conditioning the deep learning generative model on text in the case of music or drumbeat generation is not trivial. As opposed to the case of images, which have rich datasets linking text and images, music lacks such datasets. Commonly used MIDI datasets such as *The Lakh MIDI Dataset* [8] and the *Magenta Groove MIDI Dataset* [13] do not offer the rich text required to train such a model. To circumvent this issue, we work with the Groove Monkee dataset which provides descriptive filenames for its MIDI drumbeats (see Section 3). To deal with this text we in-

[1] We used Intel(R) Core(TM) i7-7700 CPU @ 3.60/4.20GHz - GeForce GTX 1050 Ti for generating a drumbeat.

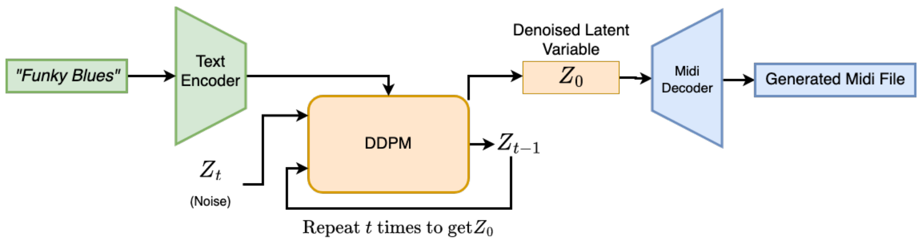

Figure 1. Text conditioned MIDI file generation flow incorporating all elements of the model. The overall flow involves converting text prompts to text embeddings. These text embeddings along with noise (Z_0) are passed into a Latent Diffusion Model, and decoded to produce the final drumbeat.

vestigate both a large language model (LLM) with CLIP-like pretraining, and a multi-hot keyword-based approach (see Section 4). Either approach gives a text encoding that can be used to guide drumbeat generation. Figure 1 shows the overall architecture of our system. The provided text prompt is converted into text embeddings using the text encoder described in Section 4.1. The random noise, denoted as $Z_{t_{\max}}$, along with these text encodings are autoregressively passed into the LDM that iteratively generates $Z_{t_{\max}} \ldots Z_0$. Z_0 is then passed into a pianoroll Decoder that generates the final pianoroll, a real-valued matrix, representing the drumbeat. This pianoroll is converted back into a MIDI file.

2. RELATED WORK

LDMs have been used in the image domain with applications in the real world. Rombach et al. [12] have demonstrated the generation of high-resolution images using a similar architecture.

The translation of these methods into music is not trivial. Firstly, music lacks such a rich dataset of text and symbolic music. The Lakh MIDI dataset lacks textual content which is required to train such models. The *Expanded Groove MIDI Dataset* [14] does offer annotations for each drumbeat in the form of genre tags and bpm, it lacks more detailed information about the parts of the drumset that were used for each drumbeat.We address this by providing a novel mechanism to take advantage of an implicit form of labeling present in a previously unused dataset (see Section 3), which we have found to be of higher quality than the others.

Secondly, the data distribution of images and music is very different. Symbolic music in the form of a pianoroll is very sparse, in contrast to image data. The relevant features in music span multiple bars, containing both local rhythmic patterns and long-term cross-bar dependencies. For instance, while note onsets and durations within a bar represent local temporal features, musical phrases also evolve across bars to give long-term dependencies. In contrast, computer vision is primarily driven by local features, motivating architectures such as CNN [15]. We address these issues by designing an LDM model based on RNNs as opposed to CNNs in order to better capture temporal features in music. We take this a step further by creating a novel feature-extracting LSTM layer which works at multiple temporal resolutions. Thus, we demonstrate that LDMs can work for a different data distribution other than images, i.e. drumbeats.

While research has shown that a good lower-dimensional latent representation of drumbeats can be learned by a Variational Autoencoder (VAE) [16–19] and used for drumbeat generation by interpolation in the latent space, we aim to go beyond this by introducing diffusion, sequence modelling, and text-conditioning. In contrast to Beat Blender [19] we also include drum velocity which greatly expands expressivity.

Researchers have also used methods other than AEs to generate novel drumbeats and drum accompaniments. Both Kaliakatsos-Papakostas [20] and Hoover and Stanley [21] use evolutionary algorithms to generate drums: in one case to achieve conceptual blending, in the other for interactive control taking a "scaffold" from existing instrumental tracks. Dahale et al. [6] generate drumbeats as accompaniments by conditioning them on other instruments like *string, bass, etc* using the Lakh MIDI dataset. Their model does not have a text conditioning element.

Makris et al. [22] address the challenge of text-conditioned drumbeat generation using a novel text encoding scheme and a sequence-to-sequence architecture that utilizes a Bidirectional Long Short-Term Memory (BiLSTM) Encoder and a Transformer-based Decoder. Other methods like MuseCoco [23] generate symbolic music by extracting music attributes from text. We diverge from MuseCoco by also training a LLM based text encoder which works with natural language rather than just keywords and musical attributes. While their output contains more than just drums, their generation time is much high at 50 seconds for a 43 second music.

Other systems such as JukeDrummer [24] work with audio signals instead of symbolic music, often using VQ-VAE. We diverge from this research both from the perspective of output format and the underlying generation method.

Yet other existing research focuses on pitched music as opposed to drumbeats, e.g. [25]. These authors have shown that the text encoder, learned by joint contrastive training of a music encoder and text encoder, is meaningful for downstream tasks. These results are consistent with the image-text case [9]. We take advantage of these findings for the drumbeat generation case.

Sample Paths of MIDI files from the dataset	**Extracted Keywords**
Retro Funk GM/116 Say It/Fills/116 Say It Ride Fill 11.mid	'Fills', 'Funk', 'Retro', 'Ride', 'Fill', 116BPM
World Beats GM/Layered Beats/4-4 Layered Beats/145 Latin Rock 02.mid	'4-4', 'Latin', 'Rock', 145BPM
Progressive GM/5-4 Grooves/180 5-4 02 F1.mid	'Progressive', 180BPM

Table 1. Example MIDI filepaths from the dataset along with their extracted keywords.

3. DATASET

For the development and evaluation of our model, we use the Groove Monkee [2] dataset, a collection of MIDI drum loops. This dataset has a wide range of styles and genres like Rock, Blues, Latin, African, Electronic, etc., and song parts such as Verse, Chorus, and Fill. The dataset includes a total of 37,523 MIDI drum loops, including a variety of time signatures. The dataset is offered in a nested folder structure with each folder along with the MIDI file labeled accordingly. We use the 11,340 samples which are in simple time signatures as opposed to compound time signatures.

3.1 MIDI Preprocessing

To use MIDI in a typical deep learning setting, we typically convert it to a pianoroll format as follows. We follow a previously described lossy procedure [16]. We extract metadata from the Groove Monkee MIDI files, such as the file's resolution, BPM, time signature, and track length in both beats and ticks. We assume that 128 time-slices are sufficient to represent 4 bars. Loops are tiled to give a standard length of 4 bars. We assume 9 drum channels (kick, snare, closed hi-hat, open hi-hat, ride, crash, low-tom, mid-tom, high-tom) are sufficient to represent the large majority of tracks. All drum types (multiple kicks, snares, hi-hats, bongos, etc.) are mapped into one of the 9 channels. We create an array of 128×9 float values. A value of zero indicates no event, and a non-zero value indicates a note-on event. Note-off events are not represented but are rarely needed in drumbeats.

3.2 Text Processing

The textual metadata for our model was extracted from the hierarchical folder structure of the Groove Monkee dataset, which organizes MIDI files into folders and subfolders based on genre and other descriptive characteristics. Each MIDI file's filepath includes information indicative of its genre as well as specific attributes of the drumbeat it contains (see Table 1 for examples). From this path, common identifiers such as *"Groove Monkee"*, *"GM"*, *"Bonus"*, etc., were removed, resulting in a unique string for each MIDI file. This approach allowed us to utilize the full path names as a proxy for the musical genre and characteristics of the drumbeats, under the assumption that the structured naming convention and folder organization provide a representative context for each MIDI file.

[2] See Ethics Statement.

4. METHOD

We first train an AE model on drumbeat data, as described in Section 4.2. The encoder maps from the data space to a latent space, and the decoder later maps back. A large language model (LLM) text encoder which embeds the text information into text embeddings is described in Section 4.1. We also hypothesized that due to the keyword-type text in the dataset, an alternative, training-free text encoder based on keyword multi-hot encoding could work as an alternative to the LLM, and this is described also. The DDPM model which runs the denoising process, guided by text, is described in Section 4.4. The training algorithm is as outlined in Algorithm 1.

Algorithm 1 Training Algorithm: Text-Conditioned MIDI Drumbeat Generation

```
1: Construct the text representation $T(w)$ for each input text $w$ using either:
       1: A pretrained BERT model to extract contextual embeddings.
       2: A multihot encoding indicating the presence of predefined keywords in $w$.
2: Train a pianoroll encoder and decoder $E$ and $D$ using reconstruction loss.
3: for each (pianoroll $m_i$, text $w_i$) in Dataset do
4:     Get text embeddings $T(w_i)$.
5:     Get the latent vector $Z_0$ for $m_i$ using $E(m_i)$.
6:     Add noise to $Z_0$ to give $Z_1 \dots Z_{t_{\max}}$.
7:     Train the DDPM model to predict $\epsilon_s := (Z_t - Z_{t-1})$, given $Z_t$, $T(w_i)$, and timestep $t$.
8: end for
```

4.1 Text Encoding

The text information corresponding to each drumbeat is extracted by converting the path (including the filename) for each MIDI file into a string that represents that particular MIDI file. Given that the text is more oriented towards keywords rather than natural language, we chose to explore and compare two alternative methods for generating text embeddings.

4.1.1 Contrastive Language-MIDI Pretraining

As is the case with CLIP [9], we learn a multi-modal embedding space by jointly training a MIDI encoder and a text encoder. The model is trained to minimize the cosine similarity between $N^2 - N$ contrasting pairs m_i and t_j ($i \neq j$), while maximizing the cosine similarity between N matching pairs m_i and t_i. The loss function is a symmetric cross entropy loss over the similarity scores, i.e. both over N MIDI embeddings given the text and N text embeddings given the MIDI file. Figure 2 contains the overall architecture of this pretraining which is similar to the original CLIP approach except for a few key differences,

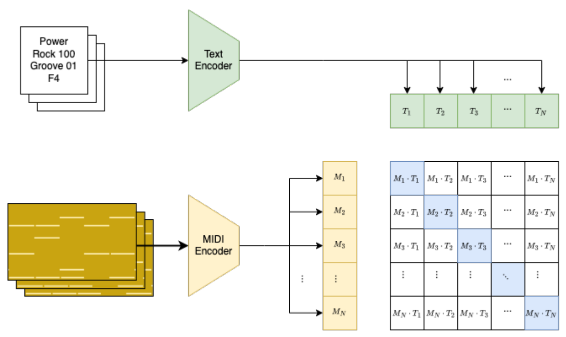
Power
Rock 100
Groove 01
F4
Text Encoder
T_1 T_2 T_3 ... T_N
MIDI Encoder
M_1 M_2 M_3 ⋮ M_N
$M_1 \cdot T_1$ $M_1 \cdot T_2$ $M_1 \cdot T_3$ ... $M_1 \cdot T_N$
$M_2 \cdot T_1$ $M_2 \cdot T_2$ $M_2 \cdot T_3$ ... $M_2 \cdot T_N$
$M_3 \cdot T_1$ $M_3 \cdot T_2$ $M_3 \cdot T_3$ ... $M_3 \cdot T_N$
$M_N \cdot T_1$ $M_N \cdot T_2$ $M_N \cdot T_3$ ... $M_N \cdot T_N$

Figure 2. Pretraining both Text and MIDI encoders to create a shared latent space for text and MIDI pianorolls, similar to CLIP [9]. The MIDI encoder is discarded after training and only the text encoder is used later.

i.e. we replace the image encoder with a MIDI encoder. While the MIDI encoder is trained from scratch, the text encoder consists of a single projection head over the pre-trained *'bert_uncased_L-4_H-512_A-8'* embeddings. The weights for the BERT model are frozen at the time of training and only the projection head is trained.

4.1.2 Multi-hot text embedding

Since the nature of our text data is somewhat keyword-like rather than complete sentences in natural language, we created an alternative keyword-based method. We created a curated list of 57 keywords by taking the 95% percentile of musically-relevant keywords in the text data. For each text input, we created a multi-hot binary vector, where a position in this vector is hot/active if the corresponding keyword is present in the text. We also append the bpm as an integer if present in the text prompt in the form of either a 3 digit number or the 3 digit number immediately succeeded by "*bpm*".

Similar to the text encoder trained in the previous section 4.1.1, the denoising model (discussed in 4.4) ingests the textual information in the form of a Multihot Text Context Vector. This technique allows for the encoding of multiple textual descriptors simultaneously, affording the model a simple view of the textual context. Such an encoding scheme makes it easier for the model to extract textual clues required to guide the denoising process.

The limitations of this approach are that firstly, there is a limited number of keywords that can be provided as text to the generation model; secondly, the multi-hot vector does not represent natural language, so for example, a text input like "*No Ride Cymbals*" would yield a vector where the "*Ride Cymbal*" position is hot.

4.2 Autoencoder

For LDM, we train a MIDI Auto-Encoder (AE) which can create a compact representation of a drumbeat. Just like a typical AE, the encoder transforms pianoroll drumbeats into a latent variable Z, which is then subsequently decoded back into the pianoroll space. The model is trained on reconstruction loss. Note that the MIDI encoder in discussion is different from the MIDI encoder in section 4.1.1

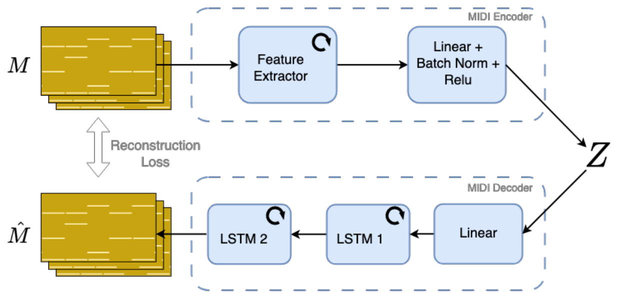

Figure 3. We train an Autoencoder (AE) for pianoroll drumbeats using reconstruction loss. The MIDI encoder feature extractor consists of a novel 3-stacked MRLSTM which looks at the MIDI file at different resolutions. While the decoder uses 2 LSTMs with a linear layer.

which was not trained on reconstruction loss but rather to link text and MIDI onto a common latent spaces using contrastive loss. The AE operates on a 128×9 pianoroll representation, where each entry is a continuous value between 0 and 1, indicating the velocity of a predicted *note-on* event. During MIDI file generation, outputs below a threshold of 5/128 are set to zero to remove low-intensity activations. Figure 3 describes the architecture of our AE.

4.3 MultiResolutionLSTM

Motivated by the multi-resolution nature of music, we use Multi-Resolution LSTM (MRLSTM) component as part of the encoder. It is designed to analyze pianorolls at multiple temporal resolutions. In our implementation, the MRLSTM works at resolutions of 1:1, 1:2, and 1:4. The lowest resolution (1:4) focuses on every fourth time-slice, which in a typical simple-time rhythm will be of higher metric weight. By using three LSTM networks to analyze the midi files at different resolutions, the MRLSTM divides the musical information into segments which are more closely aligned with musicology. The first LSTM assesses the entire sequence of 128 timesteps to understand the finer musical structure. Subsequently, the second LSTM analyses every other beat capturing the intermediate rhythmic patterns. Lastly, the third LSTM focuses on every fourth beat aligning with significant metric positions according to music theory. Both 1:2 and 1:4 LSTMs work with a shifting start position to ensure all beat positions are covered. This approach allows our model to grasp both the macro and micro rhythmic and melodic nuances, ensuring a richer reconstruction of midi drumbeats. We found that the Encoder performed better both in training and test reconstruction loss when using the MRLSTM. The generated midi files had a tighter metric structure in terms of bass drums and high hats using the MRLSTM as compared to a normal LSTM. We provide the PyTorch module for MRLSTM as a direct replacement of LSTM as part of our code [3].

[3] https://github.com/pushkarjajoria/Text-Conditioned-Drumbeat-Generation/blob/main/Midi_Encoder/model.py#L10

4.4 Diffusion in Latent Space

To improve the stability and speed [12] of our diffusion process, we implement diffusion within the latent space created by our AE described above. The LDM is trained to learn the conditional probability distribution

$$p_\theta(\epsilon|Z_t, t, w) \tag{1}$$

where ϵ is the noise present in the noised latent variable Z_t after t timestep of adding noise, with this process being conditioned upon textual information in the form of text embeddings w. This enables the iterative and autoregressive denoising of a randomly sampled Z_t, progressively estimating Z_{t-1} through to Z_0, thus refining the generated output with each step.

While at sampling time, Z_t is sampled from a Normal Distribution, at training, we use the MIDI encoder discussed in Section 4.2 to generate Z. This latent embedding is then noised as per the noising schedule and by sampling a t between 1-1000 to create Z_t. The loss for each step is computed using the mean squared error between ϵ and the predicted noise, $\hat{\epsilon}$.

4.5 Model & Training Details

The denoising model consists of 3 linear layers with batch normalization and ReLU activation. The input to the first linear layer is the concatenation of Z along with sinusoidal positional encoding of the timestep t and the text embeddings (either using the multi-hot text embeddings or the text embeddings as per Section 4.1.1). The autoencoder latent encoding after t timesteps of noising, Z_t, is of 128 dimensions. We experimented with 64, 128 and 256 dimensional latent vectors and found 128 dimensions suitable. We also pass an empty text embedding for 5% of training steps to encourage the model to predict the noise without the text embeddings. This is especially useful when there are no active musical tags in the text prompt as per our curated list.

5. EXPERIMENTS AND RESULTS

In this section, we describe how we tested our drumbeat generation deep learning model for creating MIDI drumbeats.

Firstly, following [26] we have compared distributions of inter-set and intra-set distances in different setups, visualised as probability density functions with kernel smoothing. We measure how dissimilar the generated drumbeats are, both from other generated drumbeats for the same text prompt (intra-set) and also from the most similar elements of the dataset (inter-set).

Lastly, we evaluated the quality of the generated music through a listening test. For this case, we avoid the problems associated with "musical Turing tests" [26] by focusing on questions of quality, suitability for the text, and novelty, rather than questions of artificiality.

While our model explicitly predicts note velocities rather than binary activations, we do not evaluate them in isolation. Instead, velocity variations are implicitly assessed through the listening test and as part of latent space comparisons.

5.1 Experiments

We create 8 text prompts – namely, *latin triplet*, *4-4 electronic*, *funky 16th*, *rock fill 8th*, *blues shuffle*, *pop ride*, *funky blues*, and *latin rock*—designed to capture a diverse range of genres and rhythmic elements present in our dataset. These prompts were selected based on preliminary analyses that identified them as representative of the most common rhythmic patterns and styles. For each text prompt, 10 drumbeats are generated using our model. We then compute the $\binom{10}{2} = 45$ pairwise distances using two metrics: Hamming Distance on the binarized pianorolls and Euclidean Distance in the latent space of the autoencoder (AE). The Hamming distance is applied to the binary representations of the pianorolls to capture differences in rhythmic onsets and pattern structures, thereby providing a measure of similarity or dissimilarity for pianorolls. In contrast, the Euclidean distance in the AE latent space quantifies similarity in a compressed feature space that captures higher-level, abstract structural variations between drumbeats. It provides evidence that the encoder is learning meaningful representations and forming clusters that reflect the inherent structural variations in the drumbeats, rather than collapsing them irrespective of the text input.

We also compute the distance of the generated drumbeats from the closest 1% of the drumbeats in the dataset for each text-prompt variant. The Hamming Distance when computed with the dataset, helps in detecting overfitting; if the model were simply reproducing training examples, the pairwise distances would be very low. Also, the pairwise distances show that the generated drumbeats have variety, i.e. the model does not always generate the same beat for the same prompt. While the Euclidean distance in the AE latent space quantifies the similarity in high-level structural features. These measures enable the reader to assess whether the generated outputs are similar to or deviate from the training data, thereby offering insights into the balance between fidelity towards the text prompt and novelty from the dataset. The results, summarized in the third set of columns denoted by "Generated vs Dataset" in Figure 5 indicate that the generated drumbeats are not mere replications of the training data.

We also study the effects of text-conditioning by analyzing the Hamming distances between drumbeats generated from the same text prompts compared to those generated from different text prompts. Notably, the final column of Figure 5 reveals a crucial insight: even when utilizing identical text prompts to generate multiple drumbeats, the resulting drumbeats exhibit differences. This variation highlights the variability inherent in the text-conditioned drumbeat generation model.

Despite this variability, our further analysis, illustrated in Figure 4, demonstrates a discernible impact of text-conditioning on drumbeat generation while using the Hamming distance. In the Euclidean space, while the same-text drumbeats do not cluster as closely as one might antici-

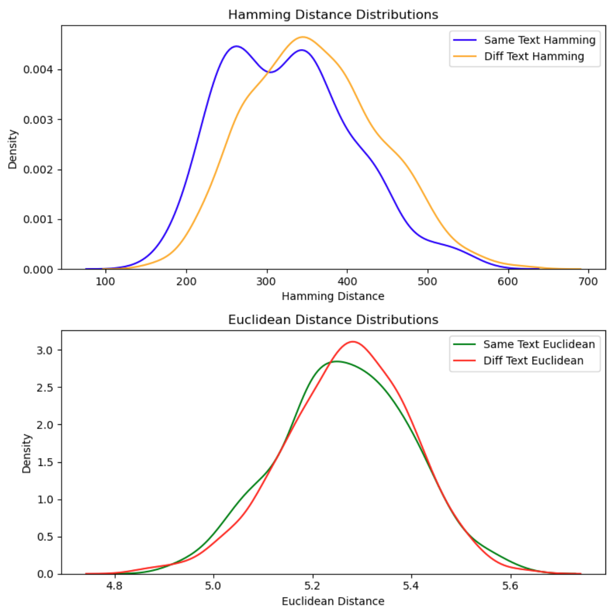

Figure 4. Density plots comparing Hamming (top) and Euclidean (bottom) distances of drumbeats generated from identical versus different text prompts.

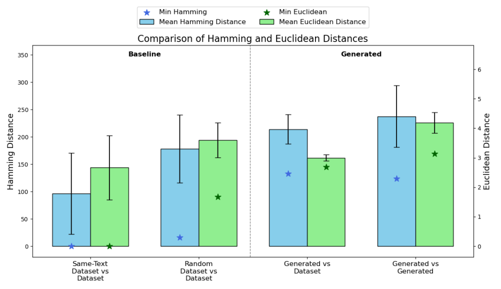

Figure 5. Comparison of Hamming and Euclidean distances across dataset and generated drumbeats. The first two sets of columns are to provide a sense of scale. The first set of column values are for drumbeats from the dataset with the same text multi-hot vector (even though the original full text may be different). For the second set of column drumbeats are generated by randomly sampling 2 texts from the dataset. The third set of columns denoted by *"Generated vs Dataset"* focuses on novelty or generalization, by comparing generated drumbeats with the closest 1% of the dataset drumbeats. The last set of column measures *variety* by comparing multiple drumbeats generated from the same text encoding.

pate, given the autoencoder's design and training objective focused on just reconstruction loss, there remains a substantial difference between the distances achieved in the same-text and different-text conditions. This outcome suggests that, although the autoencoder does not explicitly encode genre-specific characteristics or ensure close clustering of similar genres in the latent space, text-conditioning nonetheless exerts a subtle but significant influence on the generated drumbeats.

5.2 Listening Test

While our empirical evaluation demonstrates that the generated drumbeats exhibit substantial variation and are not mere replications of the training dataset, these quantitative metrics alone are insufficient if the outputs do not exhibit the qualities of authentic drumbeats. After all, even a randomly constructed pattern could deviate from the dataset while failing to sound musically convincing. To address this aspect, we conducted a listening test to assess the quality and musical validity of the generated drumbeats. We conducted a survey comprising 40 drumbeats created using 10 randomly selected text prompts from our dataset, resulting in four variants for each prompt: (1) the original dataset drumbeat which matches the prompt, (2) a drumbeat generated by LDM via our multi-hot text encoding, (3) a drumbeat generated by LDM via our BERT text encoding, and (4) a drumbeat generated by LDM with an empty text encoding[4], serving as a control example.

Each variant was randomly assigned an order between 1 and 4, with assignments varying for each text prompt to prevent subjects learning the assignment. This ordering was consistent between all participants. Participants were allowed to go back to change their responses.

[4] For empty text encoding, we use a zero tensor.

Drumbeats ranged from 8 to 15 seconds in length, leading to an estimated survey completion time of 30 to 60 minutes. We recruited subjects both in-person and online. Each participant was selected based on their experience in at least one of the following areas: music performance, music composition, or musicology. Subjects were blind to the specifics of the drumbeat generation methods along with the nature of each variant. There were three evaluation criteria: quality, aptness to the given text prompt, and novelty. These were each assessed using a Likert scale. To mitigate subjective bias in our listening test, we provide detailed descriptors for each Likert scale point, ensuring a consistent evaluation framework. For quality, the ratings range from *Poor: Unfit for musical composition*, to *Excellent: Perfectly suited for a musical composition as is*. For aptness, ratings range from *Not at all: Completely irrelevant to the text*, to *Perfectly: Fully relevant and accurately reflects the text*. For novelty, the scale ranges from *Not novel at all: Very common and I have heard many similar beats*, to *Extremely novel: Unique and I have never heard anything similar*. These descriptors help anchor participant judgments, reducing variability in subjective interpretation. We received a total of 14 responses to our survey, out of which 12 were fully completed. This makes a total number of questions answered across all categories to be **480**. These 480 answers (i.e. 120 answers for each of the four categories) were used for all subsequent statistical evaluations.

The results, depicted in Figure 6, indicate that participants generally perceived the Quality of each category of drumbeats as comparable and satisfactory. Thus, generated drumbeats were *as good as* professionally-recorded ones. Concerning Aptness, as hypothesized the generated drumbeats were much more apt to the text, versus the control drumbeats which were generated by ignoring the text. The

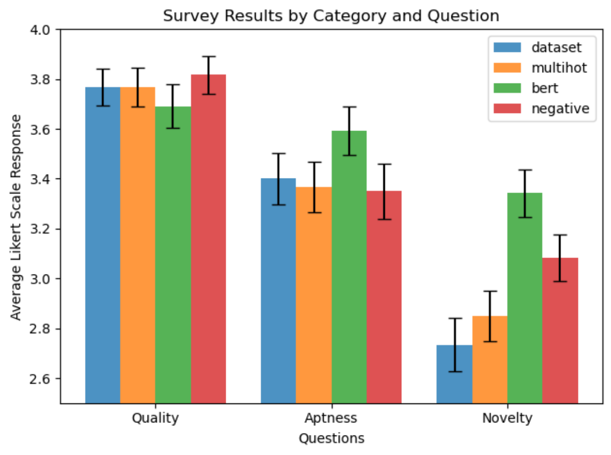

Figure 6. The aggregated responses from N=12 participants of the survey rating each category of drumbeat on Quality, Aptness to the Text Prompt and Novelty. The y-axis shows the mean score based on the Likert scale of 1-5, magnified between 2.5-4. The error bars denote ±1 SEM. The results show that the Quality across all categories is roughly the same, showing that the model is able to generate high-quality drumbeats. The BERT model values for Aptness to text and Novelty are slightly better than those for drumbeats directly from the dataset. The multi-hot text model gives worse Aptness and Novelty.

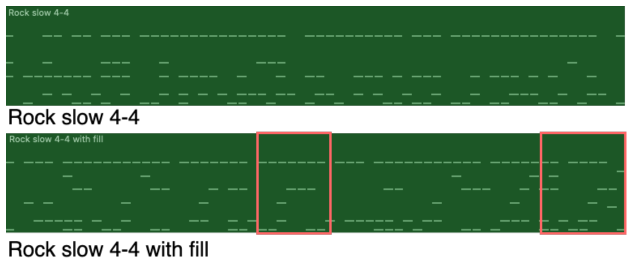

Figure 7. Two drumbeats generated with different text prompts *"Rock slow 4-4"* and *"Rock slow 4-4 with fill"*, as shown: this illustrates that the adding the term *"with fill"* has the expected effect. These drumbeats can be heard on `https://soundcloud.com/user-32049071/rock-slow-4-4` and `https://soundcloud.com/user-32049071/rock-slow-4-4-with-fills`

BERT model's outputs also received higher Aptness scores than the drumbeats from the datatset, which was not expected, and is a strongly positive result for our model. The BERT model, in particular, was noted for its ability to generate drumbeats that were perceived as high in Novelty. It is also worth mentioning that the drumbeats originating from the dataset, which were created by human musicians, received the lowest scores for Novelty, along with the multi-hot text encoder.

6. CONCLUSION AND FUTURE WORK

In this study, we developed a system for generating drumbeats conditioned on textual prompts using Latent Diffusion Models. We also presented MRLSTM, a new variant for LSTM that works at different resolutions which is better suited for symbolic music. The results demonstrate the system's capability to produce high-quality, coherent drumbeats that align with human expectations given a text prompt. Analysis from both the Listening Test and the experimental data in Figure 5 confirms the system's capability to generate new drumbeats. Figures 4, 7, and 6 illustrate that both experimentally and through human evaluation, the generated drum beats are significantly impacted by the input text and compete in aptness with professional drumbeats. Participants in the listening test rated the quality of these drumbeats as comparable to those in the *Groove Monkee* dataset, underscoring our model's ability to create drumbeats that are as good as human-generated ones across three criteria. The use of diffusion in the latent space enhances the quality and generation speed and opens possibilities for real-time integration. We have made the code along with some cherry-picked[5] generated samples available on GitHub[6] and SoundCloud[7].

We suggest in the future the text prompt generation could be improved by employing prompt augmentation techniques to achieve better embeddings of the keyword-like text in a musical context. LLMs models like BERT are better suited for natural language and hence such augmentation may also result in better text embeddings. This could be done in a controlled manner using current LLMs to convert the keyword like text into more free flow natural language. Conducting a larger-scale study in a more controlled environment will provide more reliable data, providing more insight into the human perceived capabilities of the system. The MRLSTM shows great promise and could benefit from an ablation study on different tasks in Music Generation and Music Information Retrieval for symbolic music to further comment on its efficacy. The observed differences in the Latent space between the Same-Text and Different-Text conditions warrants further investigation into how text prompts shape the musical output. Investigating more by isolating the dimensions with high variance might yield some meaningful insights about the latent space and hence on the impact on text conditioning in the latent space.

7. ACKNOWLEDGMENTS

This work was carried out at the University of Galway with funding from the Polifonia Project of the European Union's Horizon 2020 research and innovation programme under grant agreement N. 101004746. The final part of this project was completed at Saarland University.

8. ETHICS STATEMENT

The Groove Monkee Mega-Pack dataset is a commercial product available on `https://groovemonkee.com/products/mega-pack`. We train on it for research purposes only. We do not offer this dataset for download, nor do we offer our trained model for download as it could be seen as competing with Groove Monkee's commercial

[5] Cherry-pick factor of 5. Note that no cherry-picking was used for the listening evaluation.
[6] https://github.com/pushkarjajoria/Text-Conditioned-Drumbeat-Generation
[7] https://soundcloud.com/user-32049071/sets/generated-drumbeats

product. A free demo version of the dataset is available from Groove Monkee, enabling the reader to test our code.

9. REFERENCES

[1] J. A. Biles, "GenJam: Evolution of a jazz improviser," in *Creative evolutionary systems*. Elsevier, 2002, pp. 165–187.

[2] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *International conference on machine learning*. PMLR, 2018, pp. 4364–4373.

[3] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, "Deep learning techniques for music generation – a survey," 2019.

[4] D. Herremans, C.-H. Chuan, and E. Chew, "A functional taxonomy of music generation systems," *ACM Computing Surveys*, vol. 50, no. 5, p. 1–30, Sep. 2017. [Online]. Available: http://dx.doi.org/10.1145/3108242

[5] D. Herremans and E. Chew, "Morpheus: Generating structured music with constrained patterns and tension," *IEEE Transactions on Affective Computing*, vol. 10, no. 4, p. 510–523, Oct. 2019. [Online]. Available: http://dx.doi.org/10.1109/TAFFC.2017.2737984

[6] R. Dahale, V. Talwadker, P. Rao, and P. Verma, "Generating coherent drum accompaniment with fills and improvisations," 2022.

[7] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," 2018.

[8] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, 2016. [Online]. Available: https://colinraffel.com/projects/lmd/

[9] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021.

[10] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020.

[11] A. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," 2021.

[12] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," 2022.

[13] J. Gillick, A. Roberts, J. Engel, D. Eck, and D. Bamman, "Learning to groove with inverse sequence transformations," in *International Conference on Machine Learning (ICML)*, 2019.

[14] L. Callender, C. Hawthorne, and J. Engel, "Improving perceptual quality of drum transcription with the expanded groove midi dataset," 2020.

[15] O. A. Montesinos López, A. Montesinos López, and J. Crossa, *Convolutional Neural Networks*. Cham: Springer International Publishing, 2022, pp. 533–577. [Online]. Available: https://doi.org/10.1007/978-3-030-89010-0_13

[16] J. McDermott, *Representation Learning for the Arts: A Case Study Using Variational Autoencoders for Drum Loops*. Cham: Springer International Publishing, 2021, pp. 139–161. [Online]. Available: https://doi.org/10.1007/978-3-030-59475-6_6

[17] A. Tikhonov and I. Yamshchikov, "Drum beats and where to find them: Sampling drum patterns from a latent space," 07 2020.

[18] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," 2019.

[19] T. Blankensmith and K. Phillips, "Beat Blender," 2018. [Online]. Available: https://experiments.withgoogle.com/ai/beat-blender/view/

[20] M. Kaliakatsos-Papakostas, "Generating drum rhythms through data-driven conceptual blending of features and genetic algorithms," in *Computational Intelligence in Music, Sound, Art and Design: 7th International Conference, EvoMUSART 2018, Parma, Italy, April 4-6, 2018, Proceedings*. Springer, 2018, pp. 145–160.

[21] A. K. Hoover, M. P. Rosario, and K. O. Stanley, "Scaffolding for interactively evolving novel drum tracks for existing songs," in *Applications of Evolutionary Computing: EvoWorkshops 2008: EvoCOMNET, EvoFIN, EvoHOT, EvoIASP, EvoMUSART, EvoNUM, EvoSTOC, and EvoTransLog, Naples, Italy, March 26-28, 2008. Proceedings*. Springer, 2008, pp. 412–422.

[22] D. Makris, G. Zixun, M. Kaliakatsos-Papakostas, and D. Herremans, "Conditional drums generation using compound word representations," 2022.

[23] P. Lu, X. Xu, C. Kang, B. Yu, C. Xing, X. Tan, and J. Bian, "Musecoco: Generating symbolic music from text," *arXiv preprint arXiv:2306.00110*, 2023.

[24] Y.-K. Wu, C.-Y. Chiu, and Y.-H. Yang, "JukeDrummer: Conditional beat-aware audio-domain drum accompaniment generation via transformer VQ-VAE," 2022.

[25] S. Wu, D. Yu, X. Tan, and M. Sun, "CLaMP: Contrastive language-music pre-training for cross-modal symbolic music information retrieval," 2023.

[26] L.-C. Yang and A. Lerch, "On the evaluation of generative models in music," *Neural Computing and Applications*, vol. 32, no. 9, pp. 4773–4784, 2020.