Completing Audio Drum Loops with Symbolic Drum Suggestions

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

Teresa Pelinski [*]
Centre for Digital Music
Queen Mary University of London, UK
t.pelinskiramos@qmul.ac.uk

Marina Nieto
Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
marinanietogimenez@gmail.com

Sergi Jorda
Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
sergi.jorda@upf.edu

ABSTRACT

Sampled drums can be used as an affordable way of creating human-like drum tracks, or perhaps more interestingly, can be used as a mean of experimentation with rhythm and groove. Similarly, AI-based drum generation tools can focus on creating human-like drum patterns, or alternatively, focus on providing producers/musicians with means of experimentation with rhythm. In this work, we aimed to explore the latter approach. To this end, we present a suite of Transformer-based models aimed at completing audio drum loops with stylistically consistent symbolic drum events. Our proposed models rely on a reduced spectral representation of the drum loop, striking a balance between a raw audio recording and an exact symbolic transcription. Using a number of objective evaluations, we explore the validity of our approach and identify several challenges that need to be further studied in future iterations of this work. Lastly, we provide a real-time VST plugin that allows musicians/producers to utilize the models in real-time production settings.

Author Keywords

Generative AI, Drums, Real-time, Transformer, VST Plugin

CCS Concepts

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

1. INTRODUCTION

In many contemporary sample-based styles of music, producers often use drum parts that are re-imaginations of previously recorded/produced patterns. These drum loops can be re-utilized with little to no modification, or alternatively, can be temporally sliced and re-arranged, even to the extreme that the resulting pattern would no longer resemble the original pattern. Additionally, a producer can choose to modify a given pattern by adding extra percussive parts on top of the sampled pattern. The latter approach is the target of this work. Specifically, we develop an automatic drum generation system that provides stylistically consistent symbolic drum suggestions that can complete (or rather infill/inpaint[1]) an existing audio loop.

One objective for this work was to develop the system such that it could affordably be used in production settings. Moreover, having developed a number of drum generation tools in our previous works, we also wanted to explore whether our symbolic-to-symbolic generative systems could be efficiently adapted to work with audio inputs.

Hawthorne et al. [13] demonstrate that challenges involving music performance generation using raw audio, ranging from structural incoherence [26] to low audio fidelity [6], can be significantly improved by separating the timbral/acoustic aspects of the generation process from the compositional elements. In particular, in their model (called Wave2Midi2Wave), the authors use an existing piano transcription system [12] to transcribe the raw audio recordings of piano performances. Subsequently, new performances are generated using the transcribed MIDI scores in conjunction with the Music Transformer model [15]. The resulting symbolic performances are then synthesized into audio using a conditional WaveNet model [26]. For this work, we take a similar approach by factorizing the system into two separate stages (audio to symbolic to symbolic), however, with several significant differences described below.

In Wave2Midi2Wave, musical events in the audio recordings need to be accurately detected and classified. However, we suspected that in our case, we might not need to classify the drum events in a given recording perfectly; instead, having an understanding of the spectral structure of

[*]Equal contribution

[1]Different terminologies such as **Infilling** and **Inpainting** can be found in the literature referring to the task of completing partial information. In the music domain, the term **infilling** can refer to the task of filling in suggestions between two parts detached by a missing gap [17], or in other works [9], can refer to overlaying a score with consistent suggestions. In the image domain, the term **Inpainting** [7], is also used to refer to painting in corrupted or missing segments of an image. In the context our work, we use infilling and inpainting interchangeably to refer to the task at hand.

the recording would be enough to identify potential spaces within the recording that can be infilled. Exploring the reliability of a reduced representation of the audio, as opposed to an accurate transcription, was indeed a major motivation behind this work. Secondly, drum loop samples are often used in a grid-based environment as they are primarily utilized in a hardware sampler, a groove-box or a digital audio workstation. Moreover, as opposed to long-term piano performances in which tempo can be intentionally varied for expressivity, in the case of sampled drum loops utilized in a grid-based environment, the loop's tempo is most likely constant and fixed to the tempo of a project. For these reasons, inspired by [9], we decided to use a tempo agnostic representation in which only drum hits, their corresponding velocity/loudness values, and their timing relative to a fixed grid are registered (Section 2.1.1).

1.1 Related Work

There are a number of works which address infilling of drum patterns with symbolic suggestions. Infilling in these works can refer to either generating patterns for fully masked segments of a score or generating additional events superimposed on top of an existing score. For example, Dahale et al. [5] provide a three-stage symbolic-to-symbolic drum accompaniment generation system in which, during the final stage, a pre-generated drum pattern is fully masked within a single bar, and subsequently, the masked bar is completed with a drum fill. On the other hand, Lee et al. [19] present PocketVAE which aims to complete a user provided drum template by by removing and/or infilling extra events. Moreover, Gillick et al. [9] present a suite of sequence-to-sequence variational auto-encoder models to generate humanized drum performances, including an infilling model capable of generating hi-hat patterns for a MIDI drum performance lacking hi-hat parts. Moreover, they demonstrate the effectiveness of modelling symbolic drum performances using a non-tokenized tempo-agnostic grid-relative representation. Along with this work, Gillick et al. have released a sizable dataset of professional symbolic drum performances. This dataset and the proposed data representation methodology serve as the main inspiration for our work.

Although not focused on drums, there are a few other notable works on infilling [16, 4, 11, 21]. Huang et al. [16] introduce a convolutional neural network (CNN) model trained to reconstruct partial scores. Similarly, Chang et al. [4] propose an XLnet-based [31] infilling model capable of generating infilling suggestions of variable length. Moreover, Guo et al. [11] present a multi-track transformer-based infilling system in which the generations can be modified using several user-controllable parameters.

In the following sections, we will discuss our methodology and provide the results of our findings, demonstrating the potential and practicality of this approach and the directions needed to further explore in the subsequent iterations of the work.

2. METHODOLOGY

Figure 1 illustrates an overview of the methodology used in this work. The model used for this work, Transformer Groove Infilling (TGI), consists of a stack of Transformer encoders [27] and a custom output layer. We trained several TGI models on various experiments defined in Table 1. Table 2 summarizes the hyperparameters for the TGI models. In Section 2.1, we discuss the data used for training and the input/output representations used in TGI experiments, and in Section 2.2, we briefly review the training process.

Exp.	Task
IH/IHS	Infilling Closed Hi-Hats: predicting the closed hi-hat part of an input audio drum loop that lacks the closed hi-hat part. Additionally, we trained an identical version of the model that receives a symbolic drum score instead (IHS).
IKS	Infilling Kicks and Snares: predicting the kick and/or snare for an input audio drum loop that lacks the kick and/or snare part.
IRL/IRH	Infilling Random: predicting events (hits) for an incomplete input audio drum loop. We trained two instances of the TGI for this task: the Infilling Random Low (IRL) and the Infilling Random High (IRH) models, in which we respectively mask 10-30% and 40-70% of the hits present in the original drum loop

Table 1: Infilling experiments

2.1 Dataset and Data Representation

The data used for training the TGI models was obtained from the Groove MIDI Dataset (GMD)[2] [9]. From GMD, we used 20,270 2-bar MIDI drum loops with a time signature of 4/4 and a vocabulary of 9 instruments.

During the training process, we mask[3] the drum loops partially according to the infilling tasks in Table 1. We use a direct symbolic representation of the masked elements as output targets (Section 2.1.1). Then, we synthesize the partially masked versions of the samples using several drum kit SoundFonts (Section 2.1.3) and subsequently extract a reduced representation from the resulting raw audio loops (Section 2.1.2); we use these representations[4] of audio loops as inputs of the TGI models.

	Model	
Hyperparameter	IH/IHS	IKS/IRL/IRH
Model embedding dimension	32	256
Feedforward dimension	512	512
# of attention heads	16	2
# of encoder blocks	6	6
Training epochs	110/70	60/160/150
Relative Size	1	13.6

Table 2: Model hyperparameters[5]

2.1.1 Symbolic Representation (HVO)

Drum loop sequences can be symbolically represented by three T × M matrices, where T corresponds to the number of time-steps, in this case, 32 (2 bars with 16 sub-divisions each), and M corresponds to the number of instruments, in this case, 9. Each of the matrices contains information about the hits (H), velocities (V), and offsets (O). This representation is similar to the one used in [9].

2.1.2 Reduced Audio Representation (MSO)

We extract a set of multiband synthesized onsets (MSO) from the audio loops. MSO maps an onset spectrogram to 32 time steps and eight frequency bands; as such, MSO is a reduced representation in time and frequency. MSO is obtained as follows:

1. **Differential spectrogram**[6] We compute the Short-Time

[2]`https://magenta.tensorflow.org/datasets/groove`.
[3]In this context, masking refers to removing information
[4]Except IHS model that is a symbolic-to-symbolic generator
[5]Tuned using the Weights & Biases [1] random sweep tool.
[6]Differential spectrogram calculations are adapted from [3].

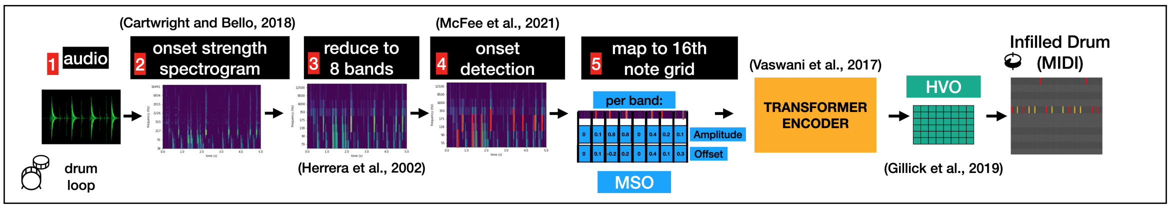


Figure 1: An overview of the methodology used in this work

Fourier Transform (STFT) using the librosa[7] [20] implementation. Then, the spectrogram resulting from the dot product of the STFT with a triangular filterbank with logarithmically spaced center frequencies is obtained. The difference between each frame and the previous 22 frames is computed to obtain a differential spectrogram (i.e., the moving mean is subtracted). The spectrogram is then half-wave rectified, logarithmically scaled, and clipped.

2. **Frequency resolution reduction** Subsequently, we reduce the number of frequency bins to 8 bands. The center frequencies of these bands have been obtained from [14], where they determine the frequency bands that perform well for drum kit instrument classification. To reduce the initial frequency bins to these eight bands, we select each band's highest onset spectrogram value and time-frame.

3. **Time resolution reduction** Next, we treat each band in the frequency-reduced differential spectrogram as an onset strength envelope and pass it through the librosa [20] onset detection function. This function returns the time-frames at which the hit onsets occur. Since all our input loops have a 2-bar measure, we split each bar into 16 divisions, resulting in a 32 time-step grid for each 2-bar loop. We create an offset matrix and an onset strength matrix with 32 columns (time-steps) and eight rows (frequency bands). The offset matrix values correspond to the difference between the time frame at which a hit onset occurs and its nearest time-step (column) in the grid. The values in the onset strength matrix correspond to the differential spectrogram value at the time-frame at which the onset occurs. The stacking of these two matrices results in the final audio input representation, which has a shape of 32×16.

2.1.3 Preparation of Training Sets

We used one SoundFont for synthesizing the input audio loops in the IH experiment (refer to Table 1). For the IKS, IRL, and IRH experiments, we use a pool of 25 SoundFonts to synthesize the audio loops to emulate the variability of real drum kits, as it is typically done in automatic drum transcription tasks [29]. In the case of the IKS, we also use different combinations of removed instruments (kicks, snares, or kicks and snares) with SoundFonts. Similarly, we combine different random samplings of the original GMD sequences with various SoundFonts in the IRL/IRH datasets. The resulting dataset sizes are shown in Table 3.

2.2 Training

The models were optimised using the stochastic gradient descent algorithm, and we used early stopping regularization. The epochs at which we early stopped the models can be found in Table 2. The loss $\mathcal{L}$ for each voice i at time-step j was defined as follows:

$$\mathcal{L}_{i,j} = \begin{cases} w\,(\mathcal{L}_{hi,j} + \mathcal{L}_{vi,j} + \mathcal{L}_{oi,j}) & \text{if } h_{i,j} = 0. \\ (\mathcal{L}_{hi,j} + \mathcal{L}_{vi,j} + \mathcal{L}_{oi,j}) & \text{if } h_{i,j} = 1. \end{cases} \quad (1)$$

where $\mathcal{L}_h$ corresponds to the hit loss (binary cross-entropy), $\mathcal{L}_v$ to the velocity loss (mean squared error), and $\mathcal{L}_o$ to the offset loss (mean squared error). The weighting factor $w \in (0, 1]$ is a hyperparameter that allows for balancing the importance of silences and hits, as silences significantly outnumber hits. The final loss for each sequence corresponds to the average of losses per voice and time-step ($\mathcal{L}_{i,j}$).

[7]`https://librosa.org/`.

	Experiment			
Split	IH/IHS	IKS	IRL	IRH
Train (80%)	15036	63299	64766	64235
Test (10%)	1921	8181	8215	8153
Validation (10%)	1681	7755	8081	8008
Total	18638	79235	81062	80396
Relative size	1	4.25	4.35	4.31

Table 3: Processed datasets sizes by experiment. Sizes refer to number of 2-Bar Beats in 4/4.

3. RESULTS

For evaluating the TGI models, we use the methodology proposed by Yang et al. [30]. In this approach, for each set of data (ground truth or generated), a set of features are extracted. Subsequently, two sets of analyses are conducted on the collected features: (1) **Absolute Analysis:** statistical distribution of the features within each set, and (2) **Relative Analysis:** pair-wise comparison of feature values from within a set (intra-set) and pair-wise comparison of feature values from one set with all values of the same features corresponding to another set (inter-set). Sections 3.1 and 3.2 respectively provide the absolute and relative analyses of the generations conducted using a set of drum-related features summarized in Table 5 [10, 2].

Feature	Description
Total Step Density	Ratio of steps with at least one hit over the total number of steps [10]
Average Voice Density	Per step voice densities averaged across all steps. Density refers to the proportion of voices active at step
Vel Similarity Score	Difference between velocities of the second bar minus the first one
Weak to Strong Ratio	Ratio of the number of onsets not on downbeats to the ones on downbeats
Swingness	Measure of the amount of swing [2]
Hisync	Syncopation of Hats, Ride and Crash [10]
Midsync	Syncopation of Snare and Toms [10]
Lowsync	Syncopation of Kick [10]

Table 4: List of features used for evaluation

For the evaluations, we used the validation subset of GMD; this subset was not used during the training or hyperparameter tuning processes. The resulting examples were processed as discussed in Section 2.1.

3.1 Absolute Analysis

We start the evaluations by looking into the distribution[8] of the generated hits and their associated velocity and offset values. Subsequently, we use the relevant features in Table 4 to compare, in absolute terms, the generations against the expected ground truth targets.

In the HVO representation used for this work, silent events (non-hits) significantly outnumber onset events (hits). To evaluate whether the models generate patterns with the same ratio of hits to non-hits, we reviewed the box-plot distribution of the number of hits per each of the generated samples obtained from each TGI model. Moreover, to evaluate whether the models generate hits at correct positions, we reviewed the *Predictive Positive Value (PPV)* of generations - i.e. the proportion of true hits (hits matching ground truth) to false hits (see Figure 2). Comparing the number of predicted hits (middle plot in Figure 2) to the ground truth hits (left plot in Figure 2), we can see that in the case of IHS and IH, while similar to the ground truth data, the interquartile ranges of the number of generated hits are positively skewed, and the medians are slightly higher than that of the ground truth data. Moreover, the skewness and the median of the same distributions for IKS, IRL and IRH are on par with the ground truth distributions. However, a closer look at the *PPV* values shows that a noticeable number of the generated hits occur at locations where silence was expected. Nevertheless, in all models, at least half of the hits in over half of the generated samples are predicted at their desired locations.

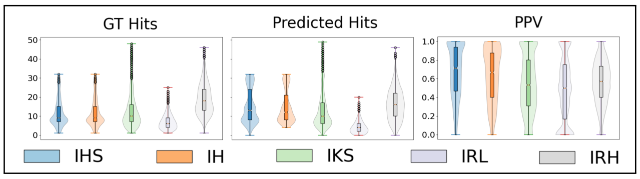


Figure 2: Distributions of hits per sample

The distribution of the per-sample average velocities and offsets of the hits are shown in Figure 3. This figure shows that in all cases, the models struggle to replicate the velocity and offset variations existing in the training data.

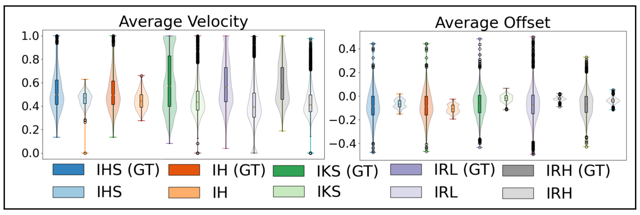


Figure 3: Distributions of velocities and offsets per sample

The features presented in Table 4 can be used to better understand the structural quality of the generations. Table 5 summarizes the mean and standard deviation of the distribution of the features extracted from the generations as well as the ground truth samples.

Total step density values can be interpreted as the ratio of time steps containing at least one drum hit to entirely silent time steps. Moreover, *Average Voice Density* is a similar measure with the distinction that it also considers the number of onsets occurring at a time step. In the case of IHS and IH models, the density of generated onsets is higher than the targets. This observation is consistent with the distributions of hits in Figure 2. On the contrary, the IKS and IRH generate patterns with the same ratio of non-silent to silent time steps while also ensuring that the number of onsets in non-silent time steps is consistent with the ground truth data. Finally, the IRL model generates patterns that contain more silent time steps than the ground truth; however, as the *Average Voice Density* values are on par with the ground truth, we suspect that the model generates more hits at non-silent time steps. Moreover, similar to ground truth data, the velocity similarity scores show that the models generate highly symmetrical patterns in all cases, except for IRL. Nevertheless, in the case of IHS, IH and IKS, the generations are more symmetrical than expected.

Swingness is a measure of delayed second eight notes [2]. The swingness values in all infilling tasks are significantly smaller than the ground truth data. This observation is expected as the amount of swing in a given pattern is highly dependent on the offsets of events, and as previously discussed, all models struggle to generalize the offset patterns available in the training set. Finally, the only model capable of generating patterns syncopated similarly to the ground truth patterns is the IRH model.

3.2 Relative Analysis

Similar to [30], for each of the feature sets obtained from the generations and the ground truth patterns, we create two sets of relative distances: (1) Ground truth intra-distances: distances of each feature value within the ground truth set from every other value within the same set, and (2) Inter-distance of generations from ground truth: distances of each feature value in the generated sets from every corresponding values in the ground truth set. Subsequently, we use a Gaussian kernel (using Scott's bandwidth selection method) [24, 25] to estimate a probability density function (pdf) to describe the distribution of the values in the inter/intra sets. Finally, for each feature, the Kullback–Leibler (KL) divergence distance and the overlapping area (OA) of the inter-distance pdf and the intra-distance ground truth pdf are measured to establish the closeness and similarity between the distributions. Plotting KL and OA for two sets of generations allows for evaluating the performance of two generative models against one another (see Figure 4).

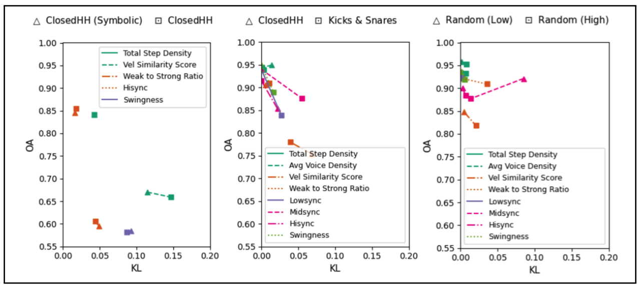


Figure 4: Inter-distances of generations from ground truth samples compared against intra-distances within ground truth samples. (left plot: the analysis is conducted only on the hi-hats. Middle and Right plots: analysis of the complete infilled pattern, i.e. generations superimposed on inputs

The first relative comparison was conducted to investigate whether there is any merit to using the reduced audio representation (MSO) instead of a symbolic version. To this

[8]Audio samples: `TransformerGrooveInfilling.github.io` code: `github.com/pelinski/TransformerGrooveInfilling`

	IH			IKS		IRL		IRH	
	GT	Pred (Symbolic)	Pred	GT	Pred	GT	Pred	GT	Pred
Total Step Density	(0.37, 0.22)	(0.48, 0.28)	(0.47, 0.27)	(0.38, 0.24)	(0.39, 0.26)	(0.20, 0.11)	(0.12, 0.09)	(0.48, 0.18)	(0.42, 0.20)
Avg Voice Density	N/R	N/R	N/R	(0.04, 0.03)	(0.05, 0.03)	(0.02, 0.01)	(0.02, 0.01)	(0.07, 0.03)	(0.06, 0.03)
Vel. Sim. Score	(0.77, 0.31)	(0.94, 0.11)	(0.95, 0.03)	(0.83, 0.28)	(0.91, 0.18)	(0.33, 0.44)	(0.23, 0.40)	(0.80, 0.30)	(0.83, 0.30)
Weak to Strong Ratio	(2.77, 8.30)	(1.12, 1.49)	(1.71, 2.68)	(2.16, 6.33)	(1.07, 2.12)	(1.84, 3.05)	(1.13, 2.50)	(1.83, 2.18)	(1.43, 1.98)
Swingness	(0.21, 0.32)	(0.00, 0.01)	(0.00, 0.00)	(0.23, 0.29)	(0.03, 0.04)	(0.19, 0.29)	(0.00, 0.01)	(0.29, 0.30)	(0.01, 0.02)
Hisync	(0.19, 0.17)	(0.11, 0.12)	(0.12, 0.11)	N/R	N/R	(0.07, 0.08)	(0.04, 0.06)	(0.15, 0.14)	(0.12, 0.12)
Midsync	N/R	N/R	N/R	(0.15, 0.20)	(0.17, 0.21)	(0.08, 0.10)	(0.06, 0.10)	(0.17, 0.16)	(0.15, 0.18)
Lowsync	N/R	N/R	N/R	(0.11, 0.13)	(0.03, 0.06)	(0.04, 0.06)	(0.02, 0.04)	(0.10, 0.09)	(0.06, 0.07)

Table 5: Mean and standard deviation (μ, σ) of features extracted from all samples within each set. Note: In some cases, certain features are not relevant (N/R). e.g., in the case of IH, samples only contain closed hi-hats, consequently, Midsync and Lowsync are irrelevant. Also, in this case, average voice density will be equal to the total step density divided by a factor of 9

end, we studied the relative performance of the IHS and IH models (left plot in Figure 4). For this comparison, we only compared the generated closed hi-hats on their own. The resulting KL/OA plot shows that the pdf of the intra-distances from the two versions are on par with each other, with negligible KL difference values in the case of the *Velocity Similarity Score*.

To establish the relevance of the infilling suggestions in the context of the partially masked inputs, we used the same relative evaluation except that we compared the generations superimposed with the partially masked targets and compared them against the complete unmasked ground-truth targets (middle and right plots in Figure 4). The results ($KL < 0.1, OA > 0.7$) show that, in all cases, the infilled patterns statistically hold high level of similarity to the target patterns.

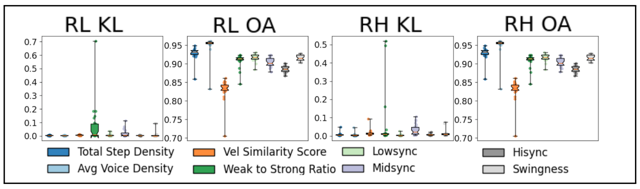


Figure 5: Impact of SoundFonts on KL/OA distributions

Finally, the relative analysis done here is computationally intensive. As a result, we had to limit the relative analysis to a subset of examples in the validation set. The 2-bar samples in the validation set come from a limited number of long performances split into smaller 2-bar segments. These long performances vary in duration, and certain performances are over-represented in the validation set. We used three evenly separated segments from each long performance to ensure balanced representation, resulting in 116 samples. Depending on the target infilling task, these samples were partially masked and then synthesized using a single drum kit SoundFont. That being said, we were aware that, given the nature of the task at hand, the selection of the drum samples might have highly impacted the performance. As a result, we repeated the analysis for all of the available 25 SoundFonts and looked into the distribution of the KL/OA values (see Figure 5). These results indicate that, while occasionally there are outliers in the distributions, all distributions have minimal inter-quartile ranges, attesting that the analysis conducted here is often consistent regardless of the drum-kit used for synthesizing the input patterns.

4. DISCUSSION

The absolute and relative evaluations presented in the previous section are an effective way of validating the final versions of our models. Moreover, they allow us to identify the short-comings of our approach so as to better design the next iterations of our work. While feature-based statistical analysis of the generations allows for gaining an overview of the performance of a model, promptly interpreting the results of such analysis is not possible as the obtained per feature statistics need to be actively and thoroughly studied, not only on their own but also relative to one another. As such, this validation process can not be effectively utilized during different stages/epochs of the training process. In this section, we present a simple global representation of generations, called *velocity heatmap*. This representation proved highly effective in identifying invalid models during our training process. Also, we demonstrate how this representation can be used to gain a better understanding of the results of the statistical feature-based evaluation during the final validation process of our work.

A *velocity heatmap* is a 2-dimensional histogram visualizing all the onsets corresponding to a single drum voice. To obtain the velocity heatmap for each drum voice, we create a scatter plot in which each point represents an onset at a given location within a 2-bar duration (x-axis) and its corresponding velocity value (y-axis). We iterate through all the samples within a given set and place all the onsets for the desired drum voice within the scatter plot. Finally, a 2-dimensional histogram (heatmap) is created based on the compiled scatter plot. In the resulting heatmap (e.g. Figure 6), the 'warmer' regions indicate a higher density of onsets while 'colder' regions indicate a lower density. We divide the heatmaps by genre to visualize the common patterns in each genre.

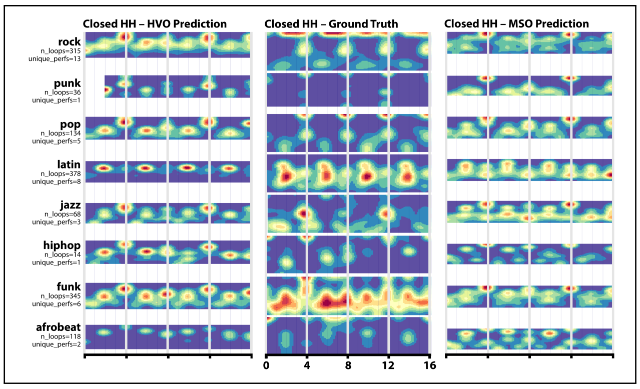


Figure 6: Velocity heatmaps per genre for IHS generations (left), validation set ground truth (center), and IH generations (right). Only the first four quarter-note beats are shown as the plots are highly symmetrical

Figure 6 compares the velocity heatmaps of hi-hat patterns obtained from the IHS and IH models against the hi-hat patterns in the target set. Consistent with the findings in the previous section, the generations from the symbolic model (left plot) hold a noticeable similarity to the audio-based model (right plot). Moreover, consistent with the results in Section 3, the plots indicate that the generations in both cases have lower velocity ranges and are also compactly populated around the gridlines. Additionally, the generations corresponding to different genres hold some degree of similarity to each other, while this similarity does not always exist in the ground truth samples. In other words, the generations are biased towards the over-represented genres[9] in the dataset. The velocity heatmaps of the generations from other models[10] illustrate that these shortcomings exist in all of the models developed for this work (see Figure 7 for IKS velocity heatmaps). In the future iterations of the work, we will need to improve the velocity and offset performance. Moreover, we will need to investigate methods of dealing with the imbalances in the dataset.

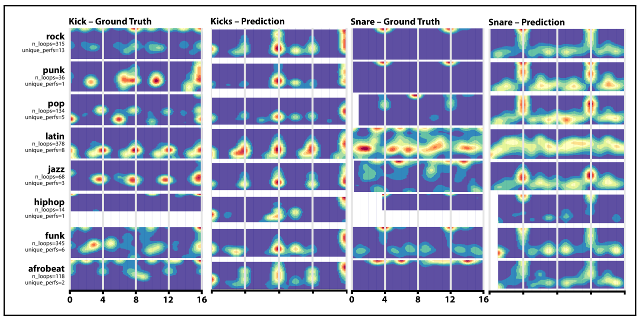


Figure 7: Velocity heatmaps per genre for IKS

5. DEPLOYMENT IN REAL PRODUCTION SETTINGS

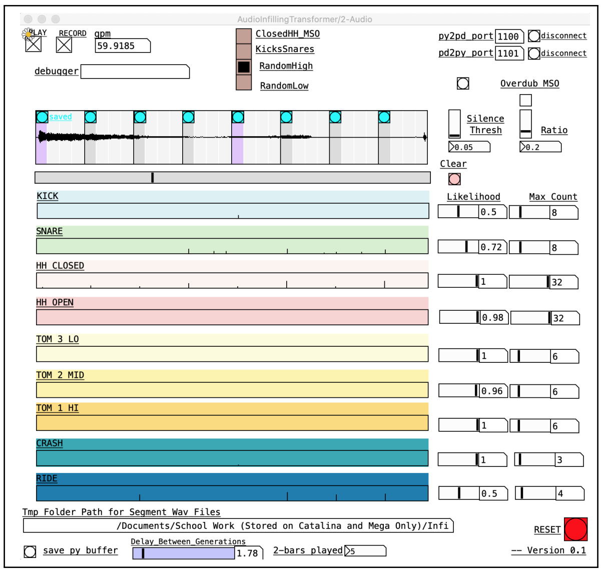
A major benefit of the representation used in this work is that the models are computationally light, allowing the inference to be carried out on the CPU. To take full advantage of this potential, we have prepared a VST plugin consisting of a Pure Data [23] graphical frontend and a command-line based python backend. In this context, the frontend is responsible for the visualization of the inputs and the generations and also allowing for interaction with the model. On the other hand, the backend is in charge of extracting representations from the audio inputs and subsequently running inference on the models. Moreover, a file-based communication is used for transferring audio data from the frontend to the backend, while an OSC [28] connection is used for transferring the generations from the python backend to the frontend so as to be visualized and played through the plugin host. Figure 8 illustrates the interface of the developed plugin.

Given that file-based communication can be very slow, we limit the inter-process transfer of audio to a fixed length equivalent to the duration of an eight note. Moreover, the delay of the communication pipeline along with the additional delays caused by input preparation and model inference result in a total latency of less than 100ms for 0.5 seconds of incoming audio. To improve the performance and to allow for a seamless interaction with the models, we are currently working on developing a self-contained VST3 plugin of the system using the JUCE framework [18]. This plugin will be used for conducting user studies of the system.

Lastly, to showcase the potentials of the system, a number of demos have been prepared. These demos were prepared

[9]More than 10 genres/styles are available in GMD, but four styles dominate the training data. Rock: 33%, Latin: 17%, Jazz: 13%, Funk: 10%)

[10]Rest of the heatmaps available here:
wandb.ai/mmil_infilling



gpm 59.9185

py2pd_port 1100

pd2py_port 1101

Silence Thresh 0.05

Ratio 0.2

	Likelihood	Max Count
KICK	0.5	8
SNARE	0.72	8
HH CLOSED	1	32
HH OPEN	0.98	32
TOM 3 LO	1	6
TOM 2 MID	0.96	6
TOM 1 HI	1	6
CRASH	1	3
RIDE	0.5	4

Tmp Folder Path for Segment Wav Files
/Documents/School Work (Stored on Catalina and Mega Only)/Infi

Delay_Between_Generations 1.78

2-bars played 5

Figure 8: Interface for the VST plugin deploying TGI models

using samples collected from `freesound.org` [8]. The first set of samples used consist of primarily percussive instruments to investigate the performance of the system using input types on which it was trained. That said, to demonstrate the potential of the system in contexts outside of the training task, we have also prepared a set of input loops that are not primarily percussive. These demos, a tutorial on how to use the system as well as all our source codes are publicly available at:

`transformergrooveinfilling.github.io`

6. CONCLUSION

This paper presented a suite of models capable of infilling audio drum loops with additional complementary symbolic drum suggestions. Our models generate the infilling suggestions using a simplified representation derived from audio drum loop patterns. Our evaluations confirm that infilled drum patterns are statistically similar to the training data. However, the evaluations also indicate that while the generated patterns are structurally correlated with the training data, they underperform in replicating the velocity and timing nuances available in the training set. Finally, our models are relatively lightweight and computationally affordable to deploy in practical applications involving users outside the research community.

7. ETHICAL STANDARDS

During this work, we only used publicly available data as well as publicly available tools. For researching/developing the methods involved, we had access to a high-performance computing cluster; we acknowledge that these tools are not publicly available, specifically, for independent researchers. As a result, we have done our utmost best to publicly share not only the source code of our work but also any trained versions of our models, as well as, scripts to easily load, use, and study the trained models without requiring any specific high-performance hardware.

This work was carried out in compliance with NIME Principles & Code of Practice on Ethical Research [22]. Moreover, throughout this work, we were not in any situation that gave rise to a conflict of interest.

8. REFERENCES

[1] L. Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[2] F. Bruford, O. Lartillot, S. McDonald, and M. B. Sandler. Multidimensional similarity modelling of complex drum loops using the GrooveToolbox. In *Proc. of the 21st Int. Society for Music Information Retrieval Conf.*, pages 263–270, Montreal, Canada, 2020.

[3] M. Cartwright and J. P. Bello. Increasing drum transcription vocabulary using data synthesis. In *DAFx 2018 - Proceedings: 21st International Conference on Digital Audio Effects*, pages 72–79, Aviero, Portugal, 2018.

[4] C.-J. Chang, C.-Y. Lee, and Y.-H. Yang. Variable-length music score infilling via xlnet and musically specialized positional encoding. In *Proc. of the 22nd Intl. Society for Music Information Retrieval Conf.*, page 97–104, Online, 2021.

[5] R. A. Dahale, V. V. Talwadker, P. Rao, and P. Verma. Generating Coherent Drum Accompaniment with Fills and Improvisations. In *Proceedings of the 23rd International Society for Music Information Retrieval Conference*, pages 264–271, Bengaluru, India, Dec. 2022. ISMIR.

[6] S. Dieleman, A. van den Oord, and K. Simonyan. The challenge of realistic music generation: modelling raw audio at scale. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[7] O. Elharrouss, N. Almaadeed, S. Al-Maadeed, and Y. Akbari. Image inpainting: A review. *Neural Processing Letters*, 51(2):2007–2028, 2020.

[8] E. Fonseca, J. Pons Puig, X. Favory, F. Font Corbera, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra. Freesound datasets: a platform for the creation of open audio datasets. In *Hu X, Cunningham SJ, Turnbull D, Duan Z, editors. Proceedings of the 18th ISMIR Conference; 2017 oct 23-27; Suzhou, China.[Canada]: International Society for Music Information Retrieval; 2017. p. 486-93.* International Society for Music Information Retrieval (ISMIR), 2017.

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

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

[11] R. Guo, I. Simpson, C. Kiefer, T. Magnusson, and D. Herremans. Musiac: An extensible generative framework for music infilling applications with multi-level control. In *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*, pages 341–356, Cham, 2022.

[12] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. H. Engel, S. Oore, and D. Eck. Onsets and frames: Dual-objective piano transcription. In *Proc. of the 19th International Society for Music Information Retrieval Conf.*, pages 50–57, Paris, France, 2018.

[13] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C. A. Huang, S. Dieleman, E. Elsen, J. H. Engel, and D. Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019.

[14] P. Herrera, A. Yeterian, and F. Gouyon. Automatic classification of drum sounds: A comparison of feature selection methods and classification techniques. In *Music and Artificial Intelligence. ICMAI 2002*, volume 2445, pages 69–80, Scotland, UK, 2002. Springer Verlag.

[15] C. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck. Music transformer: Generating music with long-term structure. In *7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019.

[16] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck. Counterpoint by Convolution. In *Proc. of the 18th Intl. Society for Music Information Retrieval Conf.*, pages 211–218, Paris, France, 2017.

[17] D. Ippolito, A. Huang, C. Hawthorne, and D. Eck. Infilling Piano Performances. In *Workshop on Creativity and Design, NeurIPS 2018*, page 4, 2018.

[18] JUCE. Jules' utility class extensions (version 7). `https://juce.com/`, 2023.

[19] K. Lee, W. Kim, and J. Nam. Pocketvae: A two-step model for groove generation and control. *arXiv preprint arXiv:2107.05009*, 2021.

[20] B. McFee, A. Metsai, M. McVicar, S. Balke, C. Thomé, C. Raffel, F. Zalkow, A. Malek, Dana, K. Lee, O. Nieto, D. Ellis, J. Mason, E. Battenberg, S. Seyfarth, R. Yamamoto, viktorandreevichmorozov, K. Choi, J. Moore, R. Bittner, S. Hidaka, Z. Wei, nullmightybofo, D. Hereñú, F.-R. Stöter, P. Friesch, A. Weiss, M. Vollrath, T. Kim, and Thassilo. Librosa/librosa: 0.8.1rc2, May 2021.

[21] G. Mittal, J. Engel, C. Hawthorne, and I. Simon. Symbolic music generation with diffusion models. In *Proc. of the 22nd Intl. Society for Music Information Retrieval Conf.*, page 468–475, Online, 2021.

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

[23] M. S. Puckette et al. Pure data. In *ICMC*, 1997.

[24] D. W. Scott and S. R. Sain. Multidimensional density estimation. *Handbook of statistics*, 24:229–261, 2005.

[25] B. A. Turlach. Bandwidth selection in kernel density estimation: A review. In *CORE and Institut de Statistique*, 1993.

[26] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. In *9th ISCA Speech Synthesis Workshop*, page 125, Sunnyvale, CA, USA, 2016.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5999–6009, 2017.

[28] M. Wright, A. Freed, and A. Momeni. 2003: Opensound control: State of the art 2003. *A NIME Reader: Fifteen Years of New Interfaces for Musical Expression*, pages 125–145, 2017.

[29] C. W. Wu, C. Dittmar, C. Southall, R. Vogl, G. Widmer, J. Hockman, M. Muller, and A. Lerch. A

Review of Automatic Drum Transcription. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 26(9):1457–1483, 2018.

[30] L.-C. Yang and A. Lerch. On the evaluation of generative models in music. *Neural Computing and Applications*, 32(9):4773–4784, 2020.

[31] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, volume 32, Red Hook, NY, USA, 2019.