GENERATING COHERENT DRUM ACCOMPANIMENT WITH FILLS AND IMPROVISATIONS

Rishabh Dahale [1] **Vaibhav Talwadker**[1] **Preeti Rao**[1] **Prateek Verma** [2]
[1] Department of Electrical Engineering, Indian Institute of Technology Bombay, India
[2] Stanford University
dahalerishabh1@iitb.ac.in, talwadkerv@gmail.com, prao@ee.iitb.ac.in, prateekv@stanford.edu

ABSTRACT

Creating a complex work of art like music necessitates profound creativity. With recent advancements in deep learning and powerful models such as transformers, there has been huge progress in automatic music generation. In an accompaniment generation context, creating a coherent drum pattern with apposite fills and improvisations at proper locations in a song is a challenging task even for an experienced drummer. Drum beats tend to follow a repetitive pattern through stanzas with fills/improvisation at section boundaries. In this work, we tackle the task of drum pattern generation conditioned on the accompanying music played by four melodic instruments – Piano, Guitar, Bass, and Strings. We use the transformer sequence to sequence model to generate a basic drum pattern conditioned on the melodic accompaniment to find that improvisation is largely absent, attributed possibly to its expectedly relatively low representation in the training data. We propose a novelty function to capture the extent of improvisation in a bar relative to its neighbors. We train a model to predict improvisation locations from the melodic accompaniment tracks. Finally, we use a novel BERT-inspired in-filling architecture, to learn the structure of both the drums and melody to in-fill elements of improvised music.

1. INTRODUCTION

Songs in popular music genres like rock are typically split into different sections such as the verse, bridge, and chorus. While the primary task of a drummer is to play in time, it is also important for the drummer to be consistent with the song structure. Traditionally fills, or short groups of notes, are played as the song transitions from one section to another (say, verse to chorus). Thus, it can serve as an indicator to the audience as well as the band, of an upcoming transition in the song. The duration of fills generally tends to be only a few beats long, no more than the length of a bar. Even though they are rare events in a drum track, fills are an important part of the overall aesthetics. Beyond signaling transitions, drum fills can also be played in sections where the accompanying instrumentation is sparse.

Motivated by the above, we improve the quality of the generated drum tracks from seq-to-seq models by incorporating fills/improvisations towards our overall goal of accompaniment generation. This is achieved via the following three distinct stages:

1. **Basic Drum Pattern Generation:** Using a seq-to-seq model to generate a drumbeat as the accompaniment to given melodic instrument tracks of guitar, bass, strings, and piano (i.e. the Melodic Accompaniment - MA).

2. **Improvisation Location Detection:** Detecting explicitly the position of improvisation from the MA using a self-similarity function and mini BERT model.

3. **Generating Improvised Bars:** Generating the fills in the previously detected bars.

Our main contributions are: (i) We show that traditional attention-based transformer architectures fail to capture the "improvisation" due to implicit data imbalance. (ii) We also show that the sampling-based approaches fail to produce a variation of pattern at the right location. To mitigate this, we learn to predict where to improvise directly from the melody tracks using powerful self-attention-based architectures. (iii) We propose a novel in-filling approach, inspired by BERT that can look at the context of drums and the context of melody and use it to generate the improvised bars. (iv) We demonstrate an MLP-based synthesis module for drum improvisation generation from a latent code. For simplicity we have ignored the dynamic (velocities) in the generated drum patterns of this work.

2. RELATED WORKS

Since the introduction of the Transformer architecture [1], there has been a growing interest in this model for sequential task modeling like in NLP and music generation. The architecture uses an attention mechanism to learn long-term patterns and can easily surpass dilated convolutional-based methods such as WaveNet [2]. They achieve state-of-the-art performance in a variety of natural language problems [3], music/audio understanding [4] and generation tasks [5, 6]. However, for generative models, the decoding strategy still remains an open question. Even though high-quality models can be obtained by the use of likelihood as the training objective, likelihood maximiza-

© Rishabh Dahale, Vaibhav Talwadker, Preeti Rao and Prateek Verma. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Rishabh Dahale, Vaibhav Talwadker, Preeti Rao and Prateek Verma, "Generating Coherent Drum Accompaniment with Fills and Improvisations", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

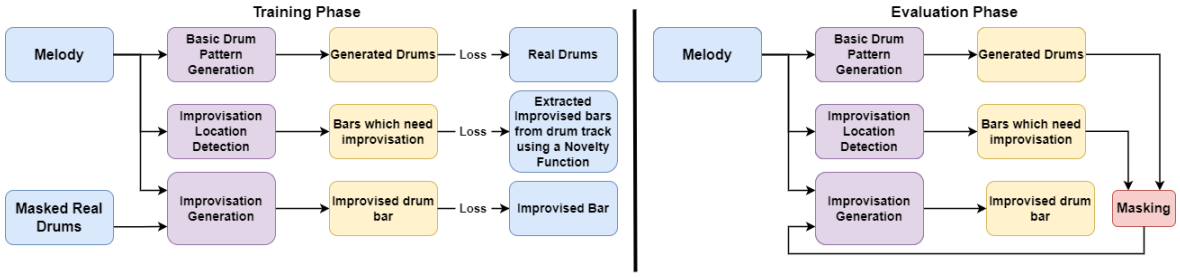

Figure 1. System Overview - Pipeline used for training (left half) of each individual module; combined pipeline (right half) for the evaluation phase.

tion methods like beam search lead to degeneration [7]. To solve this issue, many researchers use sampling-based methods like temperature sampling, top-k sampling, and nucleus sampling [7–9]. Despite all of the success of sequential modeling recently, there still exist many issues that are relevant to our current work such as understanding rare words or sparsely occurring events of interest [10]. Another major problem is the presence of biases in the generated output, as they mimic the distribution present in the training datasets [11]. These issues are ubiquitous across datasets and modalities and are implicit in our task due to the low representation of improvised bars. We here show how these constraints and biases affect the quality of drum generation, and the steps we take to mitigate them.

Conditioned drum beat generation is an important subtask of music generation. Wei et al. [12] observed that the polyphonic melody self-similarity matrix (SSM) is structurally similar to the drum SSM. They used this to predict the drum SSM from the melody SSM and used this intermediate representation to generate the drumbeat. While they used the audio form of the melody as the input, the symbolic domain also offers opportunities for similar research. An example is the work of [13] using symbolic representation of the drum track to predict locations and generate improvisations for the drum track. In this work, we address drum generation conditioned on a melodic accompaniment track, bringing considerably more complexity to the problem.

In the symbolic domain, systems use a discretized representation such as MIDI tag and pianoroll representations that capture the essential information at the semantic level. Pianoroll is a score-like matrix representing a piece of music. Note pitch and time are represented by the vertical and horizontal axes, respectively. The velocities of the notes are represented by the values. The time is generally quantized on a sub-beat level and each instrument track is represented by a separate pianoroll matrix. Dong et al. [14] used this representation along with CNN-based GAN model for music generation. Another popular symbolic domain representation is the MIDI tag representation which we refer to as the “serialized grid representation". In this method, the input is represented by a sequence of MIDI-like tags. Huang et al. [6] used this MIDI tag representation with modified transformer architecture to generate long-duration piano music. Several modifications have also been proposed to this representation. Huang et al. [15] proposed a revamped MIDI (REMI) which introduced `DURATION`, `BAR` and `POSITION` tags to improve the quaity of generated music. Ren et al. [16] further modified this representation to include multi-track representation by introducing `TRACK` tag and used the Transformer-XL model to generate multitrack songs. Nuttall et al. [17] modified the MIDI tags of nine percussion instruments to represent notes being played by a triplet of pitch, velocity, and start time, and used it with the Transformer-XL model to sequentially generate the drum pattern. Thorn et al. [18] demonstrated three experiments with the Transformer-XL model with varying input and output representation and control. In two of the experiments, they tried to control the drum machine while in the third experiment they tried to generate the drum pattern directly.

While the Transformer-XL model facilitates the generation of longer duration (musical) sequences, they still suffer from the same issues of biases in dealing with the implicit data-imbalance that exists in the training dataset [19, 20]. As the specific focus of this work is to find ways to capture the rare events in the generated output, i.e., fills and improvisations, we consider only the necessary context for an improvised bar in the form of 11 bar segments of the songs.

3. DATASET

In this work, we use the Lakh Pianoroll Dataset (LPD-5 cleansed) [14], derived from the Lakh Midi Dataset [21] which is a collection of 21,425 multitrack pianorolls files consisting of following tracks: Piano, Guitar, Bass, Strings and Percussion. All the songs in this dataset are of 4/4-time signature i.e., all the songs contain 4 beats in a bar and the dimensionality of each bar in this dataset is 128 (pitch) x 96 (time steps) i.e. each beat is divided into 24 parts. Our input, which we call melodic accompaniment (MA), consists of notes played by the 4 melodic instruments and output is a percussion instrument pattern conditioned on the input which we call the percussion accompaniment (PA).

Evaluation of generative music systems faces harder challenges than that of image generation systems [22]. We expect our system to replicate the rhythmic consistency and diversity of the dataset. Any drum beat generation system must have correct onset locations in a beat. If the onsets are not properly matched, it appears as if the drums are lagging/leading the melody. We show that our model is able learn this by capturing the onset location distribution.

To compare generated samples with original drums in the dataset, we use the following metric: Instrument Count - Number of distinct percussion instruments used in a bar. To evaluate our outputs in terms of rhythmic consistency, we use the following objective metric: Percussion pattern consistency in consecutive bars.

4. METHOD

The overview of the proposed system is shown in Figure 1. Details of each of the models are provided in subsequent sections. Our models are trained for 300 epochs with Adam optimization [23] starting with a learning rate of 1e-4 and decaying it till 1e-6. All the setup was carried out using the Tensorflow [24] framework. The following two modules are being used in all the models:

1. **Embedding Module:** As the pianoroll matrices are highly sparse, we pass them through 2 layers with 1024 and 128 ReLU [25] activated dense layers to capture the inter instrument and inter pitch dependencies. For the decoder branch of Basic Drum Pattern Generation model, we use 128 dimensional token embedding layer.

2. **Position Encoder:** We concatenate the sinusoidal positional representations [1] with the 128 dimensional vectors and use a dense layer to project them back in 128 dimension space.

4.1 Data Processing & Representation

To compress the input and output representation in our work, we perform the following preprocessing steps: (i) trim the track for start and end silence bars; (ii) resample the beats to 8 parts per beat, making each bar 32 timesteps long; (iii) keep only active MIDI pitches in all melodic instruments, i.e. MIDI 21 to MIDI 83 (notes A0 to B5); (iv) combine similar instruments in the MIDI representation of the percussion track. For example, under the snare drum, MIDI 38, which corresponds to acoustic snare, and MIDI 40, which corresponds to electric snare, are clubbed together; (v) choose only 16 percussion instruments as they capture 85.3% of all the percussion instrument strokes: snare drum, open hi-hat, close hi-hat, kick drum, ride cymbal, crash cymbal, low-floor tom, high-floor tom, high tom, hi-mid tom, low tom, cowbell, pedal hi-hat, tambourine, cabasa, and maracas; (vi) to decrease the amount of training parameters, we binarize the percussion track for seq-2-seq models and exclude velocities; (vii) split the song in non-overlapping contiguous 11 bar samples. The finer grids are superior for representing drum audio and fills [26]. We however opt for a quantized representation, capturing most of the significant musical events, allowing us to model longer duration dependencies by the compressed representation. It will be interesting to compare various representations as a future work.

We use a modified version of pianoroll representation for MA representation. We concatenate the pianorolls (Figure 2a) of different instruments instead of adding them to different channels [14]. As our input is quantized to 8 parts per beat, we represent each $\frac{1}{8}^{th}$ part of the beat by a 256-dimensional vector split amongst the 4 melodic instruments. The first dimension of this 64-dimension vector is the silence state. This is a binary state representing if the instrument under consideration is silent. The rest of the 63 dimensions contain the velocities of the MIDI notes 21-83 being played.

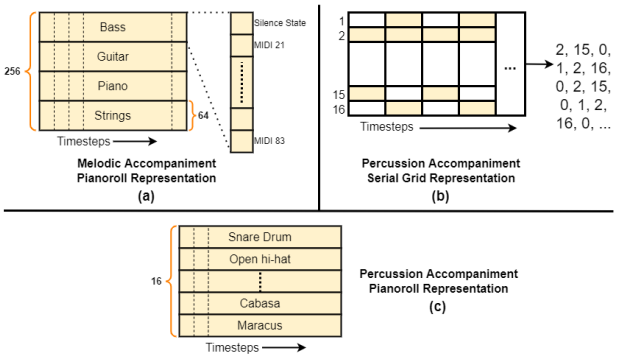

Figure 2. Data representation methods used in this work

For the PA, we adopt a mixed representation. For our Basic Drum Pattern Generation model, which is a transformer seq-2-seq model, we take advantage of the language modeling tasks and use a serialized grid representation (Figure 2b). In this representation, only the active percussion instruments which are being played are unfolded into a sequence of tokens. We add a silence state token and shift by one token making a total of 18 tokens for the percussion track. For the final model, the improvisation generation model, we give the MA and masked basic PA pattern as the inputs. We use the pianoroll representation for the PA representation for this model (Figure 2c) as only a fixed number of timesteps needs to be masked in this representation.

4.2 Train-Test Splits and Data Augmentation

We split the 21,425 songs in LPD-5 into 16,832 songs for training and 4,593 songs for the validation set. As the number of songs is limited, we use the validation set as the test set. We apply the following data augmentation strategies (inspired by sensor dropout methods in robotics [27]) to all of our models' inputs to increase the robustness in the training process:

1. **Random instrument masking:** We randomly mask one of the instruments in MA for 40% of the samples in every epoch. This 40% is equally split between the four melodic instruments. Musically this implies that one of the instruments has stopped playing. This encourages the model to consider all the instruments in the MA while making a prediction.

2. **Random timestep masking:** We randomly mask 20% of the timesteps in every sample. Musically this leads to a small disruption in the rhythm of the song which helps in better generalization of the model.

For the improvised bar generation model, which takes in the MA and masked PA as inputs, (section 4.5), we use the following additional augmentation methods:

1. **Input masking:** We randomly drop one of the inputs (MA or PA) to the model in 20% of the input samples. This

ensures that the model is not dependent on only one input for improvisation generation.

2. **Drum noise:** In the evaluation phase, the drum inputs are taken from the output of the basic drum pattern generation model. As this process could be prone to errors, we simulate this by adding the random noise to the drum samples while training. The random noise can either add new drum strokes or remove some old strokes. Our analysis showed that the PA has a very low density in the pianoroll format (average $\approx 5\%$). Hence we perturb the PA density by a maximum of 1%

4.3 Basic Drum Pattern Generation

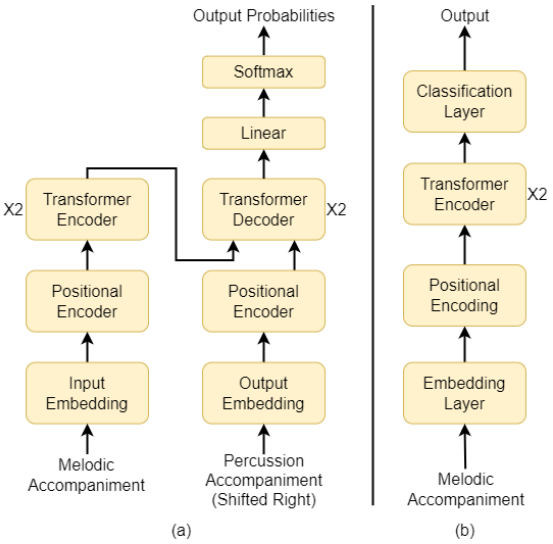

Figure 3. (a) Sequence to sequence model, used for basic drum pattern generation; (b) Improvisation Location Detection Model

We use the Transformer encoder-decoder model [1] to generate a basic drum pattern (Figure 3a). This is done by giving the MA as the input to the encoder and the shifted PA tokens to the decoder branch. Both the inputs are first passed through the embedding module followed by the position encoder. The embedded inputs are passed through 2 layers of encoder/decoder module with 128 dimensional latent space and 8 attention heads. At the output, we have 18 neurons corresponding to the 16 drum instruments, silence token, and shift token.

4.3.1 Sub-module Evaluation

We evaluate the above model with the negative log-likelihood (NLL) values over the train and the validation set. As the model outputs a distribution over the 18 output tokens, there are multiple ways to decode it. We test the greedy method of decoding, where the token with the maximum probability is selected at every step and simple sampling method. The model is trained using categorical cross-entropy loss, achieved a NLL of 0.108 and 0.112 on the train and validation split, respectively.

4.4 Improvisation Location Detection

4.4.1 Novelty Function

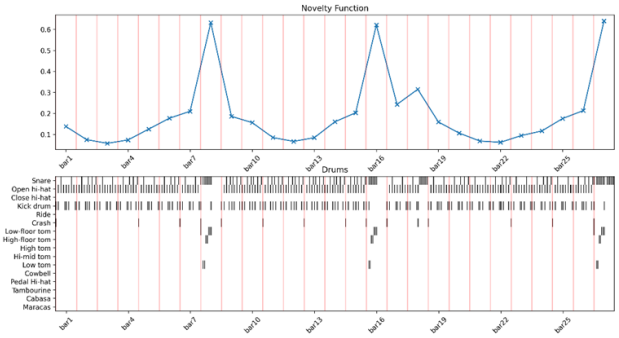

Figure 4. Novelty Function plot for a given drum track

In order to extract locations in the MA that warrant a fill, we propose the following method:

1. We use a 11-bar drum sample to calculate the Novelty value of center bar. The 5 bars on it's left and right are the context bar. The novelty value of a bar is calculated as the average weighted dissimilarity over all the context bars. We use the following equation to calculate the dissimilarity between 2 bars:

$$\frac{||bar_i - bar_j||_1 \times k}{||bar_i||_1 + ||bar_j||_1} \quad (1)$$

We utilize the hanning window to represent the weighting parameter k.

2. This calculation is done for all the bars across a track, except the first and the last 5 bars due to the lack of context bars. From Figure 4 it can be seen that the novelty function peaks at the bar with a drum fill. These bars are then extracted using a peak picking mechanism.

3. To generate the dataset for our task, we pick the bars with a local maxima as the positive samples. To filter out minor deviations, e.g., bar 18 in Figure 4, we put a threshold of 0.1 on the peaks height difference from its neighbours. Maximum of 10% of total bars in a song with these characteristics are selected as the positive samples. Same number of bars from the rest of the non peak regions are selected as negative samples.

4.4.2 Model Architecture

We use a 2 layer BERT [28] (Figure 3b) to detect the location of improvisations. The input to this model is an 11-bar MA and the prediction is done for the middle bar. The MA is passed through an embedding layer followed by the positional encoder. The embedded inputs are then passed through 2 layers of transformer encoder with 64 dimension latent space and 12 attention heads, followed by 1024 neurons. These are finally passed to a 2 dimensional softmax activated dense layer which acts as a classification module. The above model is trained using Huber loss [29], as it is robust to outliers and less sensitive to noise.

4.4.3 Sub-module Evaluation

We monitored the accuracy, precision, and recall of the model in terms of detecting the improvised bars where the

target is the original drum track. The final results can be found in Table 1. As the dataset is split equally between positive and negative samples, we have balanced precision and recall values.

	Precision	Recall	Accuracy	F1 Score
Train	92.3	92.3	92.3	92.3
Val	79.1	79.4	79.3	79.2

Table 1. Performance of the Improvisation Location Detection model. All the values are in %.

4.5 Improvisation Generation

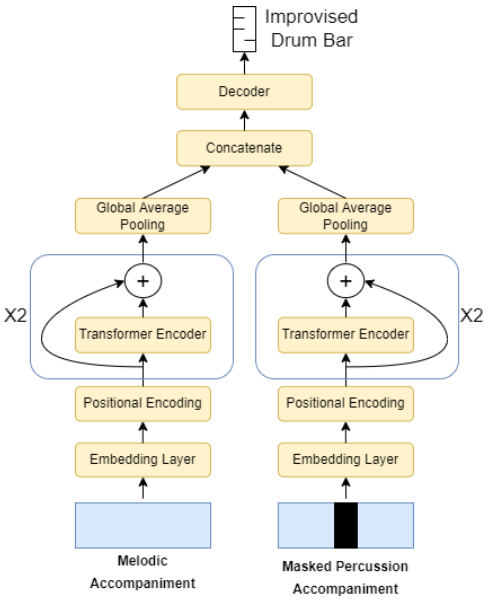

Figure 5. Improvised Bar Generation Model

The final step in our system is the generation of improvised bars. To achieve this we use the architecture shown in Figure 5. We provide 11 bar MA and PA as the input to the model. During the training phase, the PA is the original percussion track, whereas, during the evaluation/generation phase, the percussion track generated by our first stage model (section 4.3) is used. The 6th bar in the percussion sample (middle bar) is the target bar and is masked while giving as the input.

We generate a summary vector of both the MA and the PA inputs. Both are first passed through an embedding layer followed by a position encoder module. This is then passed through 2 layers of transformer encoders with 128 dimensional latent space and 8 attention heads. Even though larger/bigger models could potentially lead to better results, we have used the resources at our disposal. We note that any further improved performance will only apply to in-fill detection and synthesis.

We add skip connections to ease the flow of gradients [30]. To generate the summary vector for each input, we use a global averaging technique. These two vectors are concatenated and passed through a decoder structure which looks at the concatenated vector to generate the improvised drum bar. We test the following decoder architectures with our model:

1. **MLP:** A 3-layer dense network with 2048-2048-512 neurons is used. The final layer is sigmoid activated. The outputs are reshaped to 32 (timesteps) $\times$ 16 (percussion instruments) to get the improvised bar.

2. **MLP mixer:** MLP mixers [31] are simple alternatives to convolution and self-attention. They are based on multi-layered perceptrons applied across either temporal dimension or feature dimension.

3. **Conv1d:** A simple conv1d architecture with blocks of 2 layers of conv1d followed by upsampling.

We did not opt for an auto-regressive architecture, as this work does not assume causality, and we incorporate the right context as well as melody for the fill synthesis. There have been other works for improvisation synthesis, e.g. [32], also using the left and right context, even if only using drums

4.5.1 Sub-module Evaluation

We treat the prediction of the improvised bars as a regression problem. We similarly train it with Huber loss as it is less sensitive to outliers. We do not use cross-entropy loss for generation firstly purely as a design choice, and intuitively each of the time step token in the prediction within a bar lack probabilistic interpretation. We monitor and report the accuracy, precision, and recall of the models. As the distribution of 0s and 1s is not uniform in the predicted sample, F1 score provides a better insight in the performance of the models. From Table 2 it can be seen that a simple 3 layered MLP decoder is able to perform better than the complex MLP mixer and Conv1D architecture.

		Precision	Recall	Accuracy	F1 Score
MLP	Train	98.8	93.2	86.3	95.9
	Val	82.9	**70.3**	79.0	**76.0**
MLP Mixer	Train	97.5	45.0	88.7	61.6
	Val	**83.5**	42.1	**86.0**	56.0
Conv1D	Train	55.2	70.6	86.2	61.7
	Val	53.1	70.3	86.1	60.5

Table 2. Results for various decoders used in the Improvised Bar Generation model (all the values are in %)

5. EVALUATION

To evaluate the quality of the generated PA pattern of the proposed system, we conduct both objective and subjective tests [1] with: **O:** Original MIDI drum patterns from the dataset; **P1:** The basic drum pattern generated by our Basic Drum Pattern Generation model (section 4.3); **P2:** The final drum pattern with fills and improvisations generated by the complete system.

We screen the generated samples to eliminate those with more than 4 silent bars and those where the variation of bar density is high as measured by the standard deviation of the bar density. After applying the mentioned filtering to 8192 P1 drum samples generated by simple sampling method, we are left with 3762 samples for further evaluation.

[1] Note:Additional objective evaluation methods are reported in the supplementary document `https://bit.ly/2022ismirsupp`

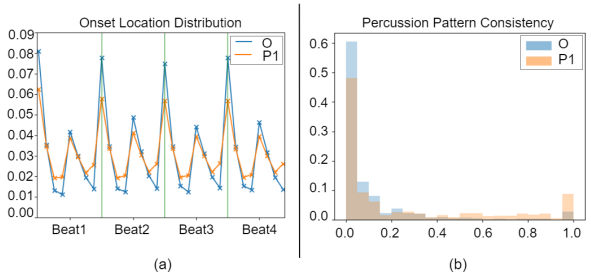

Figure 6. (a) Distribution of onset position in a bar (b) Distribution of Percussion Pattern Consistency

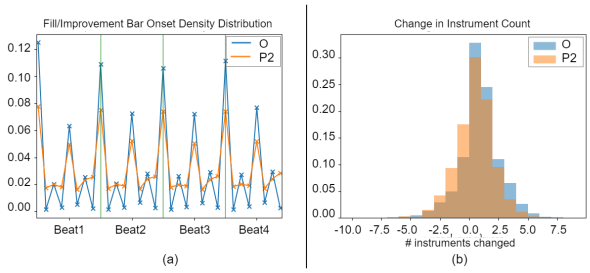

Figure 7. Improvised bars: Distribution of (a) onset location (b) change in instrument count (w.r.t. previous bar)

A basic requirement that any drum pattern generation system must fulfill is the rhythmic positioning of onsets. Generally, in a bar of rock music, the first beat (downbeat) and the third beat have similar instrumentation, featuring a hi-hat and kick drum onset. Beats 2 and 4, commonly referred to as the backbeat, include a hi-hat and snare drum onset. While the quarter notes are accented across the bar, 8^{th} notes are accented within the beat interval. Figure 6a shows the onset distribution present in the bar of both O and P1 samples. We can observe that most of the drum patterns are 8^{th} note patterns and we find a larger proportion of onsets at the accented locations within a beat interval. This is particularly intriguing because the model is given no explicit downbeat or subdivision information yet is still able to emphasis the subdivisions required for an 8^{th} note pattern.

We also do a one-to-one comparison of the P1 outputs against their target drum pattern to understand how closely the patterns match with the original drum sample based on the following metric:

Instrument Count (IC) is defined as the total number of distinct instruments used in a bar. To see whether our model is able to replicate the behavior of multi-instrument dependency, we calculate the deviation of IC in the generated sample with reference to the original target drum track for the same MA. We observe that in 75.8% of the drum bars, we are able to replicate the IC, while in 99.3% of the bars our model was off by at most 1 instrument.

Another important aspect that needs to be considered while generating drum patterns is to have a rhythmic (pattern) consistency across bars. We evaluate this aspects of the generated drum pattern using the following metric:

Pattern Consistency: For consecutive bar pair, we calculate the distance between the drum patterns using (1) keeping $k = 1$. The distribution of the bar distances is shown in Figure 6b. We can see that the generated drum bars are more or less similar to each other with some minor deviations due to the sampling decoding method. The overlapping area of the two distributions is 80.4%.

Next, on the improvised O and P2 bars, we used the following evaluation methods to see how well our models captured the fills/improvisations:

Onset Position: Figure 7a shows the distribution of onset location of percussion instruments across the improvised bars. We observe a slightly higher proportion of 16^{th} note patterns in the improvised O bars as compared to onset distribution across the non-improvised bars seen in Figure 6a. We can see that P2 system is largely able to capture this behavior as well.

Instrument Count (IC) Change: Generally during a fill/improvisation, some additional instrument are being introduced. IC change is measures as the change in IC of improvised bar compared to its previous bar. Figure 7b shows the distribution of IC change. The overlapping area of the two distributions is 87.9%, This shows that our model was able to capture the general trend of IC change.

Additionally, to evaluate the perceptual quality of the generated outputs, we present the generated samples to trained musicians. We created 3 pairs i.e. O & P1; P1 & P2; O & P2 for each MA-PA track and presented them to two guitarists and a multi-instrumentalist with experience ranging from 5 to 10 years. They were asked to provide detailed comments on the drum pattern in terms of timing, appropriateness of fills and coherence of the PA with MA. A common comment from the musicians was regarding the monotonicity in P1 track. As a result when the O & P1 pair was presented, majority of the times O was preferred, but when P1 & P2 were presented, musicians were found to appreciate the fills as it provided a lively feel to the PA.

6. CONCLUSION AND FUTURE WORK

We have successfully shown a method to produce coherent drums accompaniment with improvised bars by conditioning on a given melodic accompaniment. A novel BERT inspired infilling architecture is proposed, along with self-supervised improvisation locator. By learning, where and how to improvise, our evaluations indicate improved generation quality. Thus with a two step approach, we mitigate the biases intrinsic with data-imbalance, and shortcomings that exists with current machine learning architectures. The system can further be improved by learning optimal sampling techniques, which still remains an open problem. As a future work, we could improve the detection performance by employing larger and deeper architectures. This work highlights a serious drawback of traditional language-based generators, which have shown promise in a lot of different fields, yet they fail to capture subtle musical signals, where they are often sparsely occurring in otherwise repetitive and common patterns.

7. REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[2] P. Verma and C. Chafe, "A generative model for raw audio using transformer architectures," *arXiv preprint arXiv:2106.16036*, 2021.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[4] P. Verma and J. Berger, "Audio transformers: Transformer architectures for large scale audio understanding. adieu convolutions," *arXiv preprint arXiv:2105.00335*, 2021.

[5] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[6] 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," *arXiv preprint arXiv:1809.04281*, 2018.

[7] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," *arXiv preprint arXiv:1904.09751*, 2019.

[8] J. Ficler and Y. Goldberg, "Controlling linguistic style aspects in neural language generation," *arXiv preprint arXiv:1707.02633*, 2017.

[9] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical neural story generation," *arXiv preprint arXiv:1805.04833*, 2018.

[10] T. Schick and H. Schütze, "Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 8766–8774.

[11] E. Sheng, K.-W. Chang, P. Natarajan, and N. Peng, "The woman worked as a babysitter: On biases in language generation," *arXiv preprint arXiv:1909.01326*, 2019.

[12] I.-C. Wei, C.-W. Wu, and L. Su, "Generating structured drum pattern using variational autoencoder and self-similarity matrix." in *ISMIR*, 2019, pp. 847–854.

[13] F. Tamagnan and Y.-H. Yang, "Drum fills detection and generation," in *International Symposium on Computer Music Multidisciplinary Research*. Springer, 2019, pp. 91–99.

[14] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[15] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1180–1188.

[16] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "Popmag: Pop music accompaniment generation," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1198–1206.

[17] T. Nuttall, B. Haki, and S. Jorda, "Transformer neural networks for automated rhythm generation," 2021.

[18] O. Thörn, "Ai drummer-using learning to enhancearti cial drummer creativity," 2020.

[19] A. Caliskan, J. J. Bryson, and A. Narayanan, "Semantics derived automatically from language corpora contain human-like biases," *Science*, vol. 356, no. 6334, pp. 183–186, 2017.

[20] P.-S. Huang, H. Zhang, R. Jiang, R. Stanforth, J. Welbl, J. Rae, V. Maini, D. Yogatama, and P. Kohli, "Reducing sentiment bias in language models via counterfactual evaluation," *arXiv preprint arXiv:1911.03064*, 2019.

[21] C. Raffel, *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University, 2016.

[22] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, "Deep learning techniques for music generation–a survey," *arXiv preprint arXiv:1709.01620*, 2017.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[24] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.

[25] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[26] J. Gillick, J. Yang, C.-E. Cella, and D. Bamman, "Drumroll please: Modeling multi-scale rhythmic gestures with flexible grids," *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, 2021.

[27] G.-H. Liu, A. Siravuru, S. Prabhakar, M. Veloso, and G. Kantor, "Multi-modal deep reinforcement learning with a novel sensor-based dropout."

[28] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[29] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.

[30] L. Pepino, P. Riera, and L. Ferrer, "Emotion recognition from speech using wav2vec 2.0 embeddings," *arXiv preprint arXiv:2104.03502*, 2021.

[31] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[32] J. Gillick, A. Roberts, J. Engel, D. Eck, and D. Bamman, "Learning to groove with inverse sequence transformations," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2269–2279.

[33] 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.