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.

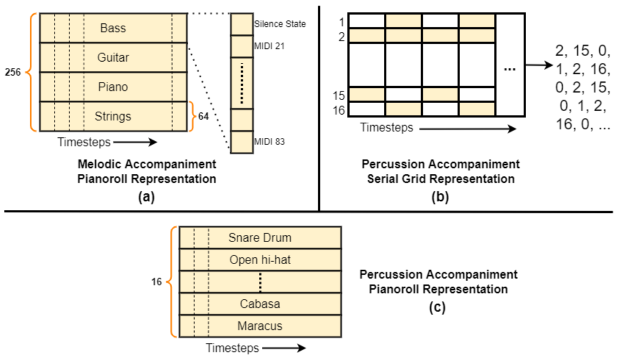

Figure 2. Data representation methods used in this work

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

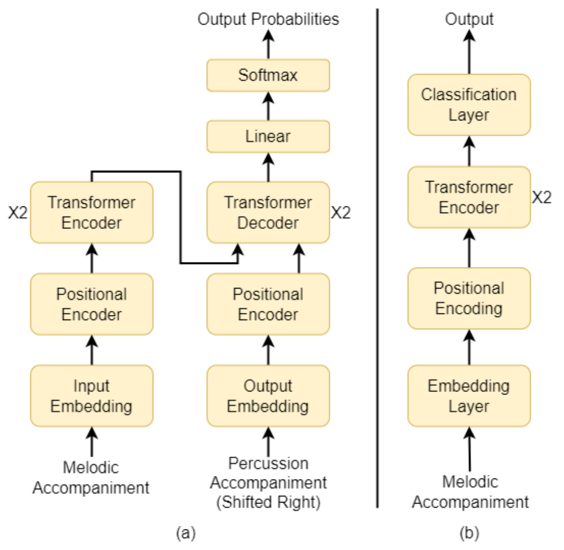

Figure 3. (a) Sequence to sequence model, used for basic drum pattern generation; (b) Improvisation Location Detection Model

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*

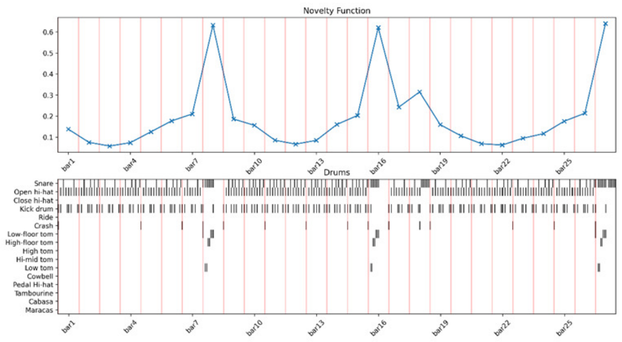

Figure 4. Novelty Function plot for a given drum track

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:

$$\frac{||bar_i - bar_j||_1 \times k}{||bar_i||_1 + ||bar_j||_1} \tag{1}$$

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

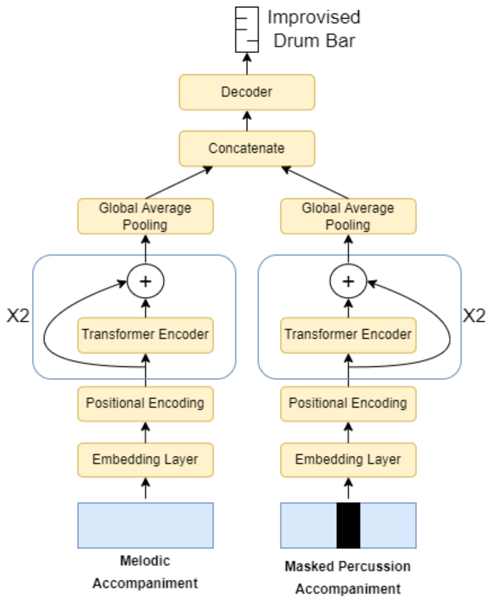

Figure 5. Improvised Bar Generation Model

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`

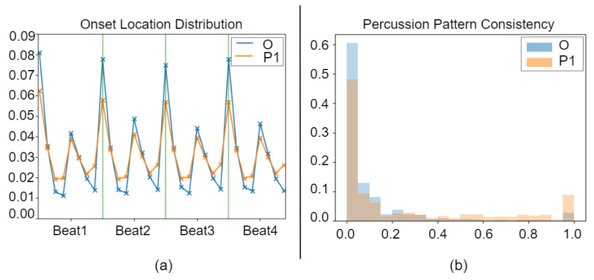

Figure 6. (a) Distribution of onset position in a bar (b) Distribution of Percussion Pattern Consistency

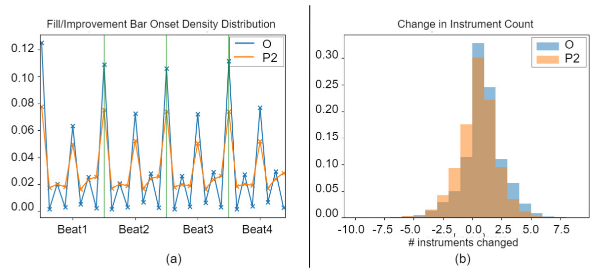

Figure 7. Improvised bars: Distribution of (a) onset location (b) change in instrument count (w.r.t. previous bar)

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.

BOTTLENECKS AND SOLUTIONS FOR AUDIO TO SCORE ALIGNMENT RESEARCH

Alia Morsi, Xavier Serra
Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
alia.morsi@upf.edu, xavier.serra@upf.edu

ABSTRACT

Although audio to score alignment is a classic Music Information Retrieval problem, it has not been defined uniquely with the scope of musical scenarios representing its core. The absence of a unified vision makes it difficult to pinpoint its state-of-the-art and determine directions for improvement. To get past this bottleneck, it is necessary to consolidate datasets and evaluation methodologies to allow comprehensive benchmarking. In our review of prior work, we demonstrate the extent of variation in problem scope, datasets, and evaluation practices across audio to score alignment research. To circumvent the high cost of creating large-scale datasets with various instruments, styles, performance conditions, and musician proficiency from scratch, the research community could generate ground truth approximations from non-audio to score alignment datasets which include a temporal mapping between a music score and its corresponding audio. We show a methodology for adapting the Aligned Scores and Performances dataset, created originally for beat tracking and music transcription. We filter the dataset semi-automatically by applying a set of Dynamic Time Warping based Audio to Score Alignment methods using out-of-the-box Chroma and Constant-Q Transform extraction algorithms, suitable for the characteristics of the piano performances of the dataset. We use the results to discuss the limitations of the generated ground truths and data adaptation method. While the adapted dataset does not provide the necessary diversity for solving the initial problem, we conclude with ideas for expansion, and identify future directions for curating more comprehensive datasets through data adaptation, or synthesis.

1. INTRODUCTION

Audio to Score Alignment (ASA) is a longstanding Music Information Retrieval (MIR) problem which aims to synchronize a musical score with its audio performance to map between each instant in a recording and a position in the score. When conducted in an online (real-time) fashion, it is often referred to as Score Following, in which the music stream to be aligned (MIDI or audio) must be processed as it is received. When conducted in an offline (non-realtime) fashion, it is often referred to as simply ASA, in which the music stream can be processed after it is fully received. This allows the advantage of looking forward and backward into the input stream [1] before returning the alignment result of each fragment.

Often, ASA problems are solved with methods previously used in audio to audio alignment problems, where the task becomes aligning the performance audio to a synthesized version of the music score [2–5]. In such methods, the alignment is conducted by comparing the same features computed from the synthesized score and the performance audio. Nevertheless, alignment is sometimes performed in the symbolic modality by first transcribing the audio then aligning it with the MIDI score [6–8], and recently alignments were conducted between audio and sheet music images directly through devising intermediate representations allowing both modalities to be compared [9, 10]. The most prevalent approaches through which ASA has been addressed are Dynamic Time Warping (DTW) [1, 2, 4, 7], and Hidden Markov Models (HMM) [6, 8, 11–13], with the former being a popular choice for alignments conducted audio to audio.

ASA research usually begins by defining the scope of the problem and accordingly proposing a system. The characteristics of the target music (i.e. the instruments, recording conditions, musician proficiency, and performance conventions) affect the scope, so ideally, researchers should find annotated data representative of such music. Very often this is not possible, as the datasets available are small and do not cover a variety of musical scenarios. This complicates benchmarking and evaluation as researchers tend to vary in their choice of data, which we believe complicates the ability to move ASA research beyond the typical use-cases. Researchers have sometimes relied on synthetic data as a low-cost and practical way to generate relevant alignment data with properties not found in other datasets. This was done to curate evaluation data [2, 10], to create ground truth for HMM training [8], or to induce temporal mismatches and file corruptions [14].

In Section 2, we cover several variants of the family of ASA methods comprised of audio representation plus DTW, showing how each alignment scenario is often addressed in an isolated manner. We believe it would be useful to unify such scenarios under one vision, allowing us to expand the definition of ASA and set clear benchmark-

© A, Morsi, X. Serra. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** A, Morsi, X. Serra, “Bottlenecks and Solutions for Audio to Score Alignment Research”, in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

ing and evaluation strategies representing each of the scenarios formerly addressed in isolation. This would enable researchers to compare between systems and identify directions for improvement, but cannot be achieved without enough varied data to support the development and evaluation of such ASA systems. Since creating datasets is costly, especially at a larger scale, we believe that first we must exhaust the ability to leverage datasets created for tasks other than ASA whenever possible and then synthesize more data as a compliment. This paper demonstrates an example by reusing the Aligned Scores and Performances (ASAP) dataset [15] to generate approximated ground truths for ASA since it provides 520 classical solo piano performances (audio and MIDI) beat aligned with their symbolic MIDI scores. We thoroughly describe this process in Section 3. In Section 4, we describe the methodology for validating the generated data and discuss their potential problems and aptness for ASA, which involves the application of several DTW-based systems to conduct offline ASA (which, from now onward, we refer to as just ASA). In Section 5, we explain how we use the obtained results to filter data for problems and highlight some limitations of our methods. Although the adapted dataset alone is not a solution to the aforementioned bottleneck, especially since it does not possess the necessary diversity to cover a variety of ASA scenarios, we conclude in Section 6 by explaining how it can be expanded to support the creation of a unified benchmark and the development of new ideas for ASA research.

2. RELATED WORK

Although this paper concerns ASA, contextualizing its developments requires references to Score Following, which has received more attention over the years. Dixon [16] and Arzt et al. [17] use online versions of DTW suitable for the real-time nature of Score Following. In later years we find the work of Duan et al. [11] and Nakamura et al. [8], both of which propose HMM systems for the same task. Henkel [27] et al. introduce a different paradigm for Score Following, where audio is aligned to score images end-to-end using reinforcement learning. For ASA, recent DTW-based approaches include [2–4, 7]. Table 1 summarizes the dataset choices in the works above, highlighting the variation among researchers in their choice of datasets. The same datasets can be used for the training and evaluation of Score Following and ASA. Although currently inactive, there was a Music Information Retrieval Exchange (MIREX) [1] entry for Score Following which can be used for benchmarking, and it includes several of the datasets shown in Table 1. But the MIREX datasets are small and do not represent a wide variety of scenarios. Moreover, there is no MIREX challenge for ASA.

[1] https://www.music-ir.org/mirex/wiki/2021:Real-time_Audio_to_Score_Alignment_(a.k.a_Score_Following)

2.1 DTW-based Methods

Given two time series $U = u_1, ..., u_n$ and $V = v_1, ..., v_m$, the goal of DTW is to find a minimum cost path $W = w_1, ..., w_n$ where every element in W is an ordered pair (i, j) indicating that the elements u_i and v_j have been aligned. Over the years, researchers have introduced different variants of DTW depending on the specifics of their target problem. For example, FastDTW [28] is a popular, more efficient variant of the algorithm. Moreover, there is the Memory Restricted Multi-Scale DTW (mrmsDTW) [29], which caps the memory requirements of the DTW algorithms for large audio files, for which a python implementation was recently made available in [30]. To the best of our knowledge, there has not been a thorough comparison of all the DTW methods for ASA, although Agrawal et al. [3] compare the results of their proposed system with JumpDTW [31], NWTW [32], and MATCH [16] based on their ability to handle structural variations in the audio compared to the score it is to be aligned with. Moreover, Shan and Tsai [10] compare the alignment results of their proposed Hierarchical DTW with those of JumpDTW and subsequenceDTW [33], where they use intermediate representations [9, 10] allowing the computation of distances between audio and score images. In addition to the variants described above, it is important to note that even within single DTW variant, performance can vary based on system choices such as normalization, the chosen time scales of the feature sequences, and the use of penalties and path constraints [14].

2.2 Audio to Score Alignment Features

Classic features used for ASA are Semitone Energy based features such as Constant-Q Transforms (CQTs) and Pitch Class Profile based features, more commonly known as Chroma representations. In a parameter search by Raffel and Ellis [14] the best alignments were attained with a log-magnitude based CQT. Ewert et al. [21] develop the DLNCO representation, which balances the tradeoff between chroma robustness and time resolution. More recent approaches explore the realm of using learned features [2, 9], or learned distance measures [3]. In [2], the authors explore the use of transposition invariant features learned in an unsupervised way on ASA, thus diverging from pitch based features. They conduct their experiments on piano and orchestral data, and report a result improvement in both. However, in [4], the authors claim that such pitch invariant features underperform in conditions of large tempo variations. So, in their approach, they use features learnt directly from music at the frame level by using a twin Siamese network each containing a Convolutional Neural Network (CNN), and in addition explore the use of salience representations proposed by [34]. In recent work, Automatic Music Transcription (AMT) has been used to first transcribe the target audio before aligning it to the score notes [6, 7].