BaDumTss: Multi-task Learning for Beatbox Transcription

Priya Mehta(✉), Meet Maheshwari, Brihi Joshi, and Tanmoy Chakraborty

Department of CSE, IIIT-Delhi, Delhi, India
{priya20033,meet20012,brihi16142,tanmoy}@iiitd.ac.in

Abstract. The challenge of transcribing audio into symbolic notations is a well-known problem in music information retrieval. In this work, we explore a novel task – *automatic music transcription for Beatbox sounds*, also known as *Vocal Percussions*. As Beatbox sounds cannot be created in a synthetic manner, they inherently vary within the same speaker as well as across different speakers. To address this, we propose BaDumTss, which makes use of a pretraining strategy over a novel sequence traversal method, thereby ensuring robustness and efficiency against new Beatbox sequences. Furthermore, BaDumTss is agnostic to time-based stretches and warps, as well as amplitude changes in the Beatbox sequence. It predicts both onsets and frame-set in a multi-task manner while gaining a whopping 56% and 326% relative improvement frame-set and onset-level F1 scores over the best performing baseline respectively. We also release an annotated dataset of monophonic Beatbox sequences along with their corresponding MIDI labels, the first of its kind comprising Beatbox samples with different variations such as time-stretches, pitch shifts, and added noise.

Keywords: Automatic music transcription · Multi-task learning · Sequence modelling

1 Introduction

Vocal percussion, *aka* Beatboxing, is perhaps the most intuitive way to express a rhythm without the use of an actual instrument. A Beatbox makes use of several techniques to replicate different instruments in an actual drum. For example, both Beatboxing and Drumming have instruments like kicks, Hi-hats, etc. that have distinct sounds. However, like any other musical piece, transcribing it is not only tedious but also an immensely challenging task. It is difficult even for humans to identify which beat was played when and for how long. Thus, the need for automatic transcription methods is dire. Along with these challenges, Beatboxing brings forth additional constraints like biometric factors, since each person has a different voice, style and volume. One way to transcribe a Beatbox

P. Mehta and M. Maheshwari—Equal contribution.

J. Gama et al. (Eds.): PAKDD 2022, LNAI 13282, pp. 173–186, 2022.
https://doi.org/10.1007/978-3-031-05981-0_14

sequence into valid symbolic representation is to convert it into MIDI (Musical Instrument Digital Interface) representations. A MIDI sequence is a lossless way of encoding information of the audio as well as the metadata.

In our work, we focus our attention on the problem of **Beatbox transcription** and address the following two problems:

1. How can we create Beatbox transcription datasets that adhere to natural constraints like biometric factors that Beatboxing suffers from, as well as maintain the general percussive patterns of the Beatbox sequences?
2. Can we also design an efficient yet robust system that improves previous transcription results but with lesser additional resources?

In this work, we study the problem of Beatbox transcription in detail – from developing a novel, fully annotated dataset to designing a new method for the task. Our primary contributions in this work can be summarised as follows:

1. We present a novel dataset[1] consisting of Beatbox sequences, accompanied by corresponding MIDI annotations. We also conduct a user-study that validates that our generated Beatbox sequences are *musically cohesive*, and distinct from random sounding Beatbox instruments that are stitched together.
2. Inspired by the recent success of pretraining in other domains, we propose BaDumTss, a U-Net based unsupervised pretraining unit, followed by a novel Bottled Sequence Traversal Unit and a multi-task autoencoder to strengthen BaDumTss towards high-level changes like new sequences, and low-level changes like time and amplitude shifts, as well as jointly predicting onsets and frame-sets.

2 Related Work

2.1 Automatic Music Transcription

Automatic Music transcription (AMT) has long been studied, and various techniques have been proposed for the ultimate goal of representing any kind of instrument in a human/machine-readable file format. Early machine learning techniques for AMT relied on hand-crafted feature extraction. One such work on polyphonic piano transcription [20] proposes a discriminative model based on frame-level SVM for pitch classification and note-level Hidden Markov Models [1]. However, such methods have become outdated as machine learning based techniques generalize poorly on a large corpus of multiple instruments.

Drum Transcription. Vocal percussion sounds are an imitation of drum sound patterns, and parallels can be drawn between these two transcription tasks. Most datasets for Automatic Drum Transcription (ADT) are still of limited size, which has led to DrummerNet [3], an unsupervised method and the state-of-the-art system for ADT. It contains a series of U-net [13] variants, which

[1] The source code and dataset are available at https://github.com/LCS2-IIITD/BaDumTss-PAKDD22.

in turn consist of 1-D convolution layers, recurrent layers, and gated Sparsemax activation. Pedersoli and Yi [14] explored the idea of pre-stacking a U-net, originally proposed to improve biomedical image segmentation. For drum-based sequences containing rhytmic patterns, RNNs showed competitive results [21]. This has also led to similar variants like the usage of GRU in a solo drum context as well as in polyphonic drum context. Furthermore, the application of a bidirectional-RNN with soft attention mechanism [4] has also been explored.

Beatbox Transcription. As compared to other instruments like piano, Beatbox transcription has been sparsely explored. A few techniques have been proposed for vocal percussion sound recognition; however, they do not introduce Beatbox sounds. One of the early studies [9] presented an Autonomous Classifier Engine (ACE) based on C4.5 decision trees for classifying Beatboxing sounds. It experimented on five type of instruments where each clip has only one label/instrument. Other studies [10] included a Hidden Markov Model (HMM) based model for the Beatbox instrument classification, in addition to pitch analysis and onset detection for feature extraction. A variant of this work [11] trained a fusion model based on HMM and Gaussian mixture model on a larger vocabulary.

All the aforementioned models work well with (and are applied only on) isolated samples. There is no evidence for good performance when applied to full-length rhythmic sequences of vocal percussion. Moreover, none of these models deal with onset detection of a beat in the music, which is necessary for a complete transcription pipeline in order to generate the corresponding MIDI sequences.

2.2 Input Audio Representation

Earlier studies focused on feature extraction from raw audio signals, but the same is now obsolete with the advent of deep learning models. Cheuk et al. [16] showed that a significant improvement in transcription accuracy is possible by choosing the right input representation. They concluded that the Mel spectrogram ensures high transcription results even though it is a compact representation of the original audio itself. Log-Mel Spectrogram is a variant of a mel spectrogram with log-scaled amplitude values and is a popular choice used by current transcription models for multiple instruments like piano [5] and drum [18].

2.3 Datasets for AMT

Standard datasets are available for popular instruments such as piano performance and their corresponding MIDI files [8], and for drum performances and aligned MIDI files [7,24]. However, Beatboxing itself does not include one specific instrument; rather it is a human imitating different percussion instrument sounds. The most related work released a dataset [19], which was proposed for vocal percussion analysis with the aim of improving algorithms for drum pattern query by vocal imitation. Samples in these dataset are recorded by amateurs with little or no experience in the art of Beatboxing and contain four different labels,

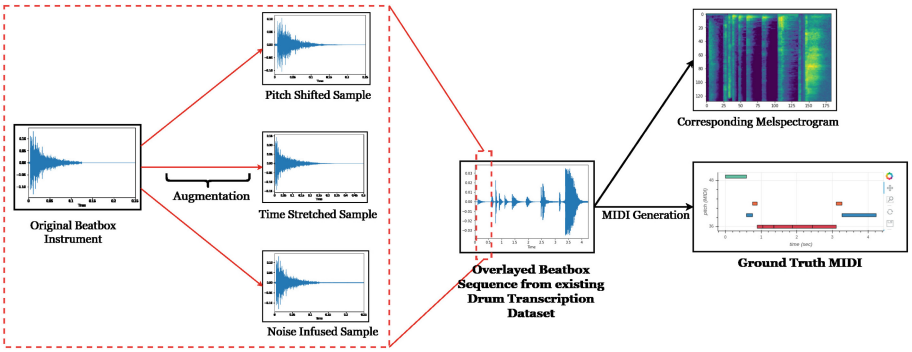

Fig. 1. Data curation pipeline: on each individually extracted Beatbox sample, we arbitrarily apply augmentations of three different types, which are then sewn together into a sequence. This sequence is constructed by overlaying the samples over existing drum-based transcription datasets. These Beatbox sequences are then used to generate the ground-truth MIDI samples, and their corresponding Mel Spectrograms are used as input to BaDumTss.

namely *kick*, *snare*, *hithat open*, and *hihat close*. This work concluded that onset detection for Beatboxing is not a trivial task and invites a novel approach.

3 Proposed Dataset

Beatboxing is the art of generating percussive sounds from the human body. As it is with any other human-centered dataset, the collection and annotation of Beatbox samples and sequences is a very tedious task. Limited data is a common problem for any automatic music transcription task, and Beatbox datasets impose several additional constraints. Firstly, we cannot prepare a completely synthetic dataset using software and other tools as human voice is required for the music to sound natural, which distinguishes a Beatbox sequence from a regular Drum sequence. Secondly, since Beatbox samples need to be recorded using some recording equipment, it adds other ungovernable factors affecting sound quality while recording such as environmental noise. Lastly, for utilizing any supervised learning algorithm, annotations in the form of a regular textual symbolic representation like the MIDI file of the corresponding musical file is essential; yet the MIDI-level annotation for Beatbox sequences is not only tedious, but also prone to hearing errors. In this section, we detail our data curation strategy – individual sample collection, data augmentation, Beatbox sequence generation, and ground-truth MIDI generation. Figure 1 demonstrates the architecture of the data collection process.

3.1 Beatbox Sample Collection

Given that Beatbox sequences are difficult to annotate, we begin our data collection at each sample level. We first begin by collecting individual Beatbox audio

samples for four instruments - *kick*, *snare*, *clap*, *hihat*. These are samples that are clipped from online tutorials for Beatboxing and our own samples. The ratio of male to female Beatboxers in our dataset is 6:4, ensuring enough diversity with respect to the fundamental frequencies ($F0$). With this, we end up with a total of 176 unique Beatbox samples. The breakdown of samples at the instrument level is provided in Table 1.

3.2 Data Augmentation

Using 176 collected samples, we create a semi-synthetic dataset by leveraging data augmentation methods as described in [15] and implemented in [12]. Augmented samples are generated by applying a random series of transformations for heterogeneity. The augmentations that are applied are provided in detail below:

1. **Pitch Shift:** Keeping in mind the biometric factors responsible for each human voice to be different from the other, pitch shifts are introduced in the original audio samples. We generate a random fractional number, which is then used to shift the pitch of an audio by a factor that ranges between 10 half-steps in both the directions – increasing or decreasing the pitch.
2. **Time Stretching:** In order to produce a factor of temporal variation, we also introduce a random factor for time-stretch based augmentations to our Beatbox samples. In the final dataset, the length of individual Beatbox samples ranges between 85%–115% of the original duration.
3. **Noise Injection:** A tolerable amount of noise is injected in varying proportions to the audio clips for different sound qualities. In order to inject realistic levels of noise to mimic real-world recordings that do not overshadow the actual content of the sample, we add random noise whose amplitude reaches a peak of 0.5% of the maximum amplitude of the sample.
4. **Amplitude Shift:** Keeping in mind the rhythmic sequence as well as the distance of a recorder from the microphone, we shift the amplitude by 75%–125% of the original sample.

A total of $5,457$ augmented sample from original Beatbox samples are created. Table 1 shows the breakdown of samples at the instrument level.

3.3 Audio Sequence Generation

One of the ways to create a Beatbox sequence comprising the above generated samples is to randomly stitch the samples together. However, such sequences cannot be termed as Beatbox as they do not contain any percussive nature or musicality, and end up sounding like noise.

In order to combine Beatbox samples, we make generate Beatbox sequences from already existing drum-based transcription datasets. *Extended Groove* [7] is one such large dataset with varying number of drum styles (like Jazz, Rock, etc.). It is a polyphonic dataset wherein there are multiple overlapping beats at the same time. For this work, we focus only on monophonic sequences; thus,

Table 1. For each of the four classes, Audio Samples (Original) denotes the total number of raw vocal percussion utterances, and Audio Samples (Augmented) denotes total number of vocal percussion utterances generated with controlled synthetic augmentation.

Instrument	#Original samples	#Augmented samples
Kick	54	1681
Snare	48	1467
HiHat	53	1653
Clap	21	656
Total	176	5457

we convert the drum sequences from polyphonic to monophonic by selecting the instrument with the highest intensity at any given time-frame. We then overlay the Beatbox samples, that are selected randomly within each instrument class, as per the drum sequences present in the dataset.

3.4 User Study on the Generated Audio Sequences

We conduct a user study to determine the qualitative musicality of our dataset. We curate a set of 10 randomly picked generated Beatbox sequences. Furthermore, to this set, we add 10 Beatbox sequences with randomly stitched samples, i.e., the generation process includes randomly picking the Beatbox samples and lining them one after another. We then randomly order these 20 sequences, and ask our annotators to distinguish which of the sequences are randomly generated and which of the sequences sound distinctly percussive.

For our study, we employ 27 annotators, out of which 17 have professional music training. The inter-annotator agreement was seen to be substantially high, for categorising the sequences into random and musically cohesive (Cohen's Kappa (κ) is 0.87). This validated that our generated dataset in fact was musically cohesive, and hence, appropriate to be used for a transcription task.

4 Proposed Methodology

Our proposed model architecture, BaDumTss, comprises of three primary components - **U-Net based pretraining** followed by **Bottled Sequence Traversal** and lastly, **Multi-task Autoencoders**. In this section, we explain our efforts in designing these three components as well as their relevance in the overall task of Beatbox transcription.

4.1 Pretraining Unit

A standard U-Net [13] architecture, designed originally for the task of semantic segmentation in images, consists of three components – (i) *a contracting*

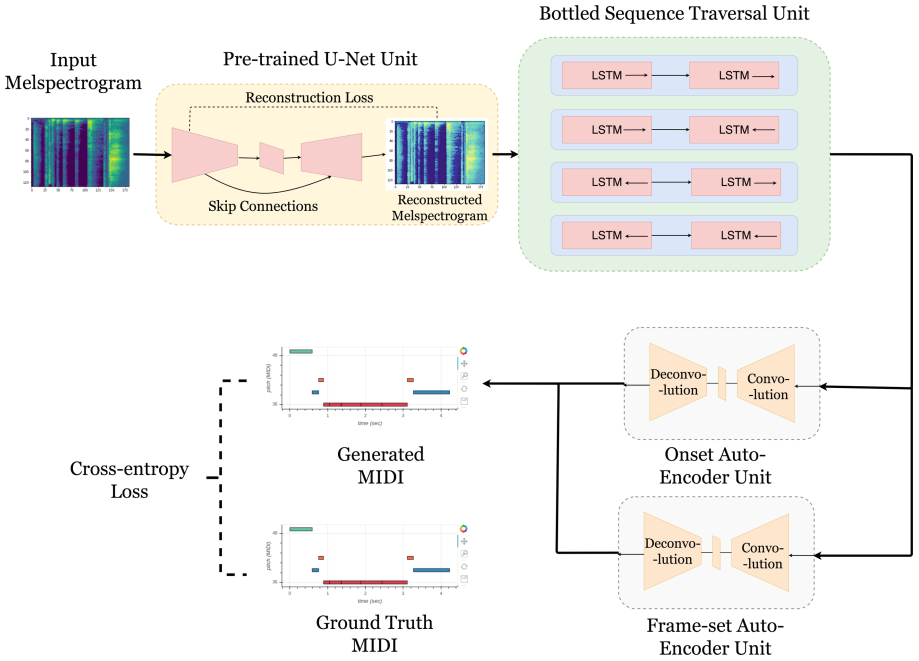

Fig. 2. A schematic diagram of our proposed model, BaDumTss.

path, which is a vanilla convolutional neural network, morphing the input to a bottle-necked representation of a certain dimension; (ii) *an expansive path*, which performs deconvolutions to generate a feature map of the same size as the input image, from the bottle-necked representations; and (iii) *skip connections*, which is the stacking of corresponding layers of the convolutional stack with the deconvolutional stack, used to prevent the catastrophic forgetting of information learnt during the expansive path. One difference between the final feature map and the input image is that the depth of the final feature map is same as the number of classes that would be used for the final pixel classification.

To design our model, we make use of a U-Net for pretraining our entire network structure. We begin by extracting the Mel spectrogram (M) of the audio sequence, that is of the dimension $N \times T$, where N is a pre-specified number of Mels and T is the number of time frames in our Mel spectrogram. We treat M as an input image which is passed through a standard U-Net architecture, denoted by $f_\Theta(\cdot)$. However, we attempt to use U-Net to reconstruct the input Mel spectrogram in the output feature map, $Z = f_\Theta(M)$. Therefore, the number of channels in our output feature space is the same as that of the input. We make two major changes to the vanilla U-Net architecture – firstly, the number of channels in the output feature map is the same as the input, thus deviating from its intended usage for semantic segmentation; and secondly, we make use of a reconstruction loss between the output feature map and the input image,

as given in Eq. 1. Therefore, our pretraining step is entirely unsupervised and does not require any labels such as the MIDI sequence.

$$\text{Reconstruction Loss} = \sum_{i=1}^{n} (Z_i - M_i)^2 \tag{1}$$

This pretrained U-Net is different from an Autoencoder because unlike an Autoencoder where the bottle-necked representation is used for downstream tasks, we make use of the output feature space for the downstream transcription task. Furthermore, we maintain the contracting and expansive paths as well as skip connections as that of a U-Net, which are not present in a standard Autoencoder.

4.2 Bottled Sequence Traversal Unit

The BST comprises four composite LSTM units - each of them taking in the output of the U-Net as the input. However, each LSTM unit processes the sequence differently. As it can be seen in Fig. 2, each LSTM unit processes its input either in the forward direction or the reverse direction. We permute all such combinations of two LSTMs together, resulting in four permutations. The final output of the BST is the average of the output representations of these four LSTM permutation units. Thus inherently making the model robust to a variety of samples, without actually requiring these samples.

4.3 Multi-task Autoencoder Unit

The final unit of our model is a **Multi-task Autoencoder Unit**. The architecture of an Autoencoder is exactly the same as that of the U-Net, however, an Autoencoder does not have skip connections. However, we branch out two Autoencoders, both which take in the output provided by the BST as input. One of the Autoencoders, called the **Frame-set Auto-Encoder Unit** is used to predict the frame-sets, i.e., the Beatbox instrument corresponding to every time-frame. The output of this Autoencoder is then compared with the ground-truth frame-set matrix, with the help of a Cross-entropy Loss.

The second autoencoder, the *onset autoencoder unit*, is used to predict the onsets, i.e., the start times of every note. The onset prediction is carried out as follows - for each frame, a binary prediction is made, which characterises whether an onset occurs in the given time-frame, which is then used to compare with the ground truth onset matrix using the binary cross-entropy loss.

Thus, our model learns both the onsets and the frame-sets in a multi-task manner, by sharing the Pretrained U-Net and the BST to learn generic sequence-level information. The Multi-task autoencoder unit helps in learning frame-set and onset level information separately, which are then trained in an end-to-end fashion.

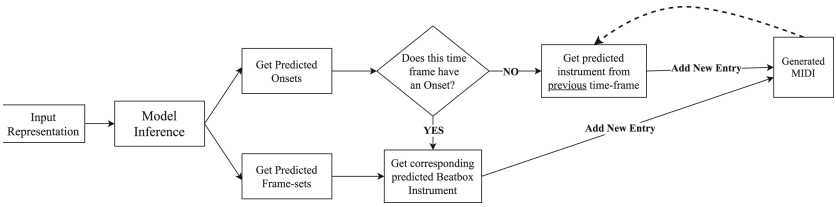

Fig. 3. The flowchart detailing the process of generating the MIDI transcription for an input representation using BaDumTss. Based on the predicted onsets, the inference pipeline decides whether a new Beatbox instrument is played, or the previous instrument is held, for the given time-frame.

Table 2. Performance of BaDumTss and two baselines – **AB** and **NE2E**.

Frame-level metrics	AB						NE2E		
	XGB	ADB	RF	SVM	LR	MLP	PRESTK	RNNDRUM	BaDumTss
Frame-set F1	0.21	0.11	0.14	0.27	0.41	0.38	0.63	0.62	**0.97**
Frame-set accuracy	0.43	0.31	0.37	0.40	0.57	0.40	0.59	0.57	**0.98**
Onset F1	-	-	-	-	-	-	0.08	0.23	**0.98**

4.4 Inference-Time MIDI Generation

After BaDumTss has been trained end-to-end, we can generate the transcribed MIDI in the following manner:

1. We convert the test audio into a melspectrogram and pass it through the model. We receive two outputs from the model - a frame-set matrix and an onset matrix.
2. Given the onset matrix, for every onset in a time frame, we search for the corresponding time-frame in the frame-set matrix and extract the predicted Beatbox instrument. For any frame in the onset matrix where there are no onsets found, we assume that the instrument is 'held', and thus, the Beatbox instrument in the previous time-frame is used.
3. Thus, we now have time-dependent Beatbox instrument predicted, that can be easily converted into the MIDI format.

Here, we note that the onsets prove to be very important, as they help us determine when an instrument is being played newly, or carried over from the previous time-frame. This technique is often used in polyphonic transcription tasks, following [5]. Figure 3 details the process with the help of a flowchart.

5 Experimental Results

In our study, we focus only on frame-level and onset-level results, and not note-level results. This is primarily because note-level results in a monophonic audio is equivalent to developing an instrument classification pipeline for all the samples

present in the sequence; whereas, frame-level and onset-level predictions directly address the transcription task and is more like a localisation task - we need to predict *which sample* occurs at *what time*.

We compare our method with two distinct categories of baselines –

1. **Algorithmic Baselines (AB):** Certain algorithms do not explicitly predict onsets; rather, it requires an algorithmic onset detection, followed by frame-set prediction that is learnt by a model. Therefore, we tweak such algorithms for a monophonic usecase - we begin by identifying onsets (the time at which a sample starts playing) of each sample by first calculating the onset strength envelope (a technique which determines where possible onsets occur) and then further, selecting the peaks from this envelope. The audio is then chunked on the basis of these detected onsets, which is then fed into a classification pipeline, used for instrument classification. These two-component baselines are then assembled together to create a transcription pipeline for monophonic Beatbox sequences. For these baselines, since onset detection is done algorithmically and not learnt, onset-level F1 scores are not available.
   For these baselines, we make use of the Librosa [12] implementation for onset detection, followed by 7 state-of-the-art machine learning algorithms - XGBoost (**XGB**), AdaBoost (**ADB**), Random Forests (**RF**), Support Vector Machines (**SVM**), Logistic Regression (**LR**), Multi-layer perceptron (**MLP**) and 2-Nearest Neighbours (**KNN**).
2. **Neural End-to-End Baseline (NE2E):** For these baselines, we focus on those methods which are end-to-end – they do the onset as well as frame-set prediction, together, and produce the final MIDI matrix during inference. The baselines that we use are:
   (a) Pre-stacked U-Net for Transcription (**PRESTK**) [14] - This model makes use of a pre-stacked U-Net followed by CNNs for the task of polyphonic transcription.
   (b) RNN-based drum transcription (**RNNDRUM**) [21] - This model makes use of an RNN-based architecture for the task of drum transcription.

Table 2 shows the performance of the competing methods in terms of Frame-set accuracy and F1 score, as well as Onset F1 scores, for those cases where it can be calculated. As we can observe, the algorithmic baselines fail to perform competitively, when placed in comparison to neural end-to-end methods. This is because for the latter, the onset prediction and instrument classification go hand-in-hand with the end-to-end transcription process, as it has been shown earlier in [5] for polyphonic transcription. BaDumTss beats the best baseline (PRESTK), adding large relative improvements of 56% and 326% in Frame-set- and onset-level F1 scores.

Ablation Study. Components of our architecture and conduct an ablation study with different components separately. The architectures used in the ablation study are shown below:

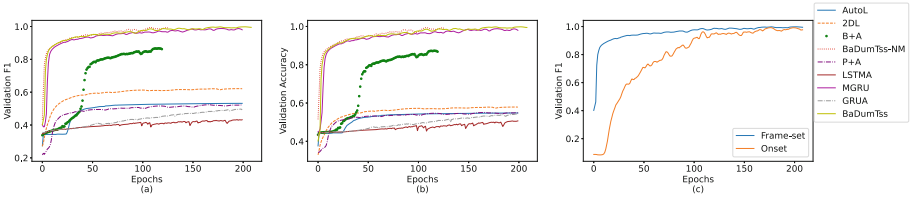

Fig. 4. Ablation study and analysis: we observe that our proposed model not only performs better, but also converges faster, when compared to other models used in the ablation study. (a) and (b) demonstrate the Validation F1 and Accuracy scores respectively with respect to training epochs. (c) demonstrates the Validation F1 scores of Frame-set and Onset prediction for BaDumTss.

1. Autoencoder + Linear (**AutoL**): This makes use of the Autoencoder unit followed by a linear layer.
2. BST + Autoencoder Unit (**B+A**): This architecture is the same as our proposed model, minus the pretraining unit.
3. Pretraining Unit + Autoencoder Unit (**P+A**): This architecture is the same as our proposed model, minus the BST.
4. BaDumTss with GRU (**MGRU**): This is the same as our proposed model, but we replace LSTMs in the BST with GRUs.
5. Single forward LSTM + Autoencoder Unit (**LSTMA**: Here, instead of using our LSTM permutations as described in the BST, we make use of a single LSTM, followed by an autoencoder Unit.
6. Single forward GRU + Autoencoder Unit (**GRUA**): Same as above, but with a single forward GRU rather than a single forward LSTM
7. 2D Convolutions + Linear Layers (**2DL**): We tweak the CNNs being used so far, and instead make use of a series of 2D CNNs, followed by a linear layer.
8. BaDumTsswith only one autoencoder Unit (**BaDumTss-NM**): This is the non-multitask version of BaDumTss, wherein, we only make use of frame-set level predictions for final MIDI generation. This does not contain an onset autoencoder Unit.

Figure 4 shows the validation F1 and accuracy scores with respect to the training epochs. We can clearly notice that BaDumTss not only converges faster as compared to other configurations of its units, but also performs better by a large margin. Another model that closely reaches the same F1 and accuracy is MGRU and BaDumTss-NM, which is the same architecture as that of BaDumTss; however, the LSTM units are replaced by that of GRU in the BST and BaDumTss-NM only predicts the frame-set. However, BaDumTss converges faster than MGRU. Even though BaDumTss-NM seems to perform competitively with BaDumTss, it encounters severe issues during inference time, which are discussed in Sect. 6.

6 Error Analysis: BaDumTss vs. BaDumTss-NM

As we can see in Fig. 4, the ablation study suggests that BaDumTss and BaDumTss-NM are competitively close to each other. Thus, one might argue why a multi-task method is needed at all. This distinction is made during MIDI generation. For BaDumTss, Sect. 4.4 detailed our inference process for generating MIDI transcriptions. However, for BaDumTss-NM, there are no predicted onsets. Therefore, the predicted frame-sets are directly converted to MIDI. This process has a caveat. It is observed that for sequences longer than 5 seconds, even though BaDumTss-NM is able to correctly predict frames, their placement throughout the sequence duration is haphazard. Most samples are off by a couple of frames, thereby rendering the generated MIDIs useless, as it is not a direct transcription of the original audio. BaDumTss on the other hand, is able to accurately predict onsets and frames at the same time, thus, ensuring that the frames are *correctly* placed throughout the sequence duration. This observation validates the usage of a multi-task model for jointly predicting onsets and frames at the same time.

Another important observation is found in the Validation curves for Onset and Frame-set prediction in Fig. 4. One can notice that Onset prediction takes a *significant* amount of time to reach the same level of performance as frame-set prediction. This can sometimes hamper the transcription performance, if the training is stopped midway. Thus, even though BaDumTssis better at the overall transcription task than BaDumTss-NM, it loses out on the training speed, as onset predictions take time to reach the ideal performance. This can be improved by employing pretraining strategies for learning onset behaviour, that can save time while training the actual transcription model. We conclude that this is out of scope for our current work.

7 Conclusion

In this work, we firstly developed a novel dataset consisting of Beatbox sequences, along with their corresponding transcribed MIDI sequences as ground-truth. We attempted to capture biometric differences and natural noise by modifying recorded Beatbox samples with semi-synthetic augmentations. On this collected dataset, we also proposed BaDumTss, a new method that is demonstratively robust to new Beatbox sequences, as well as agnostic to time and amplitude-based variations. It shows a 56% and 326% relative improvement on frame-set and onset-level F1 scores over the best performing baseline. For future work, we aim to further explore transformer-based pretraining strategies that would also help us learn coherent musical structure present in musical sequences.

Acknowledgement. The authors would like to acknowledge the support of the Ramanujan Fellowship (SERB, India), Infosys Centre for AI (CAI) at IIIT-Delhi, and ihub-Anubhuti-iiitd Foundation set up under the NM-ICPS scheme of the Department of Science and Technology, India.

References

1. Cazau, D., Wang, Y., Adam, O., Wang, Q., Nuel, G.: Improving note segmentation in automatic piano music transcription systems with a two-state pitch-wise HMM method. In: ISMIR (2017)
2. Ishizuka, R., Nishikimi, R., Nakamura, E., Yoshii, K.: Tatum-level drum transcription based on a convolutional recurrent neural network with language model-based regularized training (2020)
3. Choi, K., Cho, K.: Deep unsupervised drum transcription (2019)
4. Southall, C., Stables, R., Hockman, S.: Automatic drum transcription for polyphonic recordings using soft attention mechanisms and convolutional neural networks. In: Proceedings of the 18th International Society for Music Information Retrieval Conference (2017). https://doi.org/10.5281/zenodo.1415616
5. Hawthorne, C., et al.: Onsets and frames: dual-objective piano transcription (2018)
6. Wang, Y., Salamon, J., Cartwright, M., Bryan, N.J., Pablo Bello, J.: Few-shot drum transcription in polyphonic music. In: Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR 2020, Montreal, Canada, 11–16 October 2020
7. Callender, L., Hawthorne, C., Engel, J.: Improving perceptual quality of drum transcription with the expanded groove MIDI Dataset (2020)
8. Hawthorne, C., et al.: Enabling factorized piano music modeling and generation with the MAESTRO dataset. In: International Conference on Learning Representations (2019)
9. Sinyor, E., McKay, C., Fiebrink, R., McEnnis, D., Fujinaga, F.: Beatbox classification using ACE. In: ISMIR (2005)
10. Picart, B., Brognaux, B., Dupont, S.: Analysis and automatic recognition of Human BeatBox sounds: a comparative study. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)
11. Evain, S., et al.: Beatbox sounds recognition using a speech-dedicated HMM-GMM based system. In: Models and Analisys of Vocal Emission for Biomedical Applications Firenze, Italy (2019)
12. librosa/librosa: 0.8.0. https://doi.org/10.5281/zenodo.3955228
13. Weng, W., et al.: U-Net: convolutional networks for biomedical image segmentation. IEEE Access **9**, 16591–16603 (2015)
14. Pedersoli, F., Tzanetakis, G., Yi, K.M.: Improving music transcription by pre-stacking AU-Net. In: ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)
15. Cartwright, M., Bello, J.P.: Increasing drum transcription vocabulary using data synthesis. In: Proceedings of the International Conference on Digital Audio Effects (DAFx) (2018)
16. Cheuk, K.W., Agres, K., Herremans, D.: The impact of Audio input representations on neural network based music transcription. In: 2020 International Joint Conference on Neural Networks (IJCNN) (2020)
17. Kong, Q., Li, B., Song, X., Wan, Y., Wang, Y.: High-resolution Piano transcription with pedals by regressing onsets and offsets times (2020)
18. Jacques, C., Roebel, A.: Data augmentation for drum transcription with convolutional neural networks. In: 2019 27th European Signal Processing Conference (EUSIPCO) (2019)
19. Delgado, A., McDonald, S., Xu, N., Sandler, M.: A new dataset for amateur vocal percussion analysis. In: Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound (2019)

20. Poliner, G.E., Ellis, D.P.W.: A Discriminative Model for Polyphonic Piano Transcription. EURASIP J. Adv. Signal Process. **2007**(1), 1–9 (2007). https://doi.org/10.1155/2007/48317
21. Poliner, G.E., Ellis, D.P.W.: A Discriminative Model for Polyphonic Piano Transcription. EURASIP Journal on Advances in Signal Processing **2007**(1), 1–9 (2007). https://doi.org/10.1155/2007/48317
22. Vogl, R., Dorfer, M., Knees,P.: Drum transcription from polyphonic music with recurrent neural networks. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2017). https://doi.org/10.1109/ICASSP.2017.7952146
23. Vogl, R., Widmer, G., Knees, P.: Towards multi-instrument rum transcription. In: Proceedings of the 21st International Conference on Digital Audio Effects (DAFx 2018), 4–8 September 2018, Aveiro, Portugal (2018)
24. Gillet, O., Richard, G.: ENST-drums: an extensive audio-visual database for drum signals processing. In: Proceedings of the 7th International Conference on Music Information Retrieval (2006). https://doi.org/10.5281/zenodo.1415902
25. Emiya, V., Bertin, N., David, B., Badeau, R.: MAPS - a piano database for multipitch estimation and automatic transcription of music. Res. Rep. **11.**, 00544155 (2010)