Unsupervised Neural Transcription of Percussive Audio

Revanth Reddy Pasula[1]

[1]Department of Computer Science Wichita State University, Wichita, United States

Publication Date: 2025/07/14

Abstract: Transcription of percussive audio without human-labeled data is still a challenging area of research for music information retrieval. This work presents a deep learning solution that learns to transcribe drums autonomously without requiring human-annotated datasets. The strategy uses a neural transcription model alongside a fixed synthesizer module that, collectively, iteratively improve the drum transcription by maximizing the accuracy of reconstructed audio—without any human-annotated datasets. The experimental results indicate that the unsupervised system provides performance that is on par with fully supervised models, with added scalability. These results indicate that self-supervised learning has the potential to actually improve the accuracy of transcription for drums, opening the door for its wider application to automatic music analysis and generative sound modeling.

Keywords: *Drum Transcription; Unsupervised Learning; Self-Supervised Audio; Music Information Retrieval; Neural Networks.*

How to Cite: Revanth Reddy Pasula (2025). Unsupervised Neural Transcription of Percussive Audio. *International Journal of Innovative Science and Research Technology,* (RISEM–2025), 79-86.
https://doi.org/10.38124/ijisrt/25jun168

I. INTRODUCTION

Music transcription is the problem of approximating a symbolic music score y from an audio waveform x. A supervised learning regime is used where one learns a model (an analysis function F<sub>a</sub>) which estimates a score $\hat{y} = F_{a}(x) $ by optimizing a loss between the estimated and true scores on a collection of labeled examples. Supervised methods have predominated drum transcription since early times: earlier methods used hand-crafted time-domain and spectral features, with subsequent classifiers (e.g., support vector machines or random forests) for identifying percussive events. More recent methods have used non-negative matrix factorization to represent spectrograms as decompositions into drum-specific spectral templates and activation patterns, and hidden Markov models to enforce temporal continuity on sequences of drum onsets. Deep neural network models have recently come into use on this problem: hierarchical feature representations have been learned directly from spectrogram inputs using convolutional neural networks, and recurrent neural networks have captured temporal relationships within percussion activity to refine onset and offset detection.

One of the biggest limitations of supervised drum transcription is that large annotated datasets are hard to come by, limiting the generalization performance of trained models. Methods for synthesizing training data or using teacher–student learning models have tried to overcome the lack of annotations, but each still depends on some manner of supervision. Unsupervised methods on the other hand present an annotation-independent solution that is scalable and has the potential to generalize further from past seen data. Taking cues from how humans learn musical transcription – by listening, playing, detecting mistakes, and practicing their craft – we want to create a system that is able to critique its own transcription performance, compute reconstruction error, and correct its own mistakes. In this paper, we present Drummer Net, an unsupervised transcription system for drums that is free of any external ground-truth annotations. Drummer Net learns from its own mistakes by improving its transcriptions iteratively based on how well it is able to reconstruct the original audio from its own predictions of the drum notation.

II. SYSTEM DESIGN PRINCIPLES

We introduce a deeply integrated neural network setup aimed at performing unsupervised drum transcription. This method intentionally avoids relying on predefined or manually crafted rules for identifying drum sounds. Instead, it uses a self-supervised learning approach to label the data. Our system is built on key design principles, and we back these up with performance benchmarks and component-by-component tests to verify the value of each part. We also acknowledge certain limitations we encountered and outline future paths for developing truly unsupervised systems and expanding into more general music signal processing tasks.

III. DRUMMERNET: PROPOSED METHODOLOGY

➢ *Architectural Framework*

Our transcription system, which we call DrummerNet, is built from multiple modular blocks. Each block has specific configurations, such as the number of channels, kernel size, and stride, which are described in detail later. We use Exponential Linear Unit (ELU) activations in every convolutional and recurrent layer to help the network train quickly and stably. An overview of the full architecture can be found in Table II (see Section III-B).

➢ *Analysis Module (Fa)*

This module handles the raw audio input and generates an activation sequence for each type of drum sound. It starts with modified U-Net blocks to extract audio features, followed by layers that model how these features change over time (using recurrent neural networks). Finally, it applies a special two-part Sparsemax activation function. When in use, this module outputs smooth activation curves for each drum, which are then converted into actual drum hit times using a peak-picking method (explained in Section IV-C).

➢ *U-Net Architecture*

We use a one-dimensional version of the U-Net to pull out both short-term and long-term patterns from the audio. The encoder starts with a convolutional layer (128 channels, kernel size 3, stride 1) and is followed by ten similar layers, each having 50 channels. Max-pooling layers reduce the time dimension between these layers. This process produces a compact representation of the input that captures about 0.192 seconds of audio history at a 16 kHz sample rate. The decoder mirrors this structure, gradually restoring the original time scale using six convolutional layers and upsampling steps. The final output is a time-compressed version of the original waveform, reduced by a factor of 16.

➢ *Recurrent Layer Configuration*

After feature extraction, we pass the data through three gated recurrent unit (GRU) layers. The first runs both forward and backward through time with 100 hidden units, the second runs forward with 50 units, and the third spreads across drum types with K units, where K is the number of drum classes. This structure helps the model learn from past and future information, stay consistent over time, and link different drum sounds together. The output from the last GRU layer aligns with the number of drum types in the system.

➢ *Activation Function (Sparsemax)*

Drum hits don't happen constantly or all at once, so we want the model's outputs to be sparse—both over time and across drum types. Sparsemax helps with this by normalizing the outputs like Softmax but allowing some values to be exactly zero. We apply it in two ways: across the different drum types, and across small 64-sample windows of time (about 64 milliseconds). This ensures only a few drums are active at a time, and that only a few moments in each window carry information. These two outputs are multiplied together to get the final activation signal.

➢ *Upsampling Strategy*

Because our model works on a compressed version of the audio timeline, we need to scale the output back to full resolution. We do this by inserting 15 zeros between each activation value, effectively increasing the sequence's length by 16 times. This method avoids blurring the exact timing of drum hits while restoring the full 16 kHz resolution.

➢ *Synthesis Module (Fs)*

This part of the system takes the predicted activations and tries to rebuild the audio waveform. It uses K fixed convolutional filters—one for each drum type—that mimic how each drum would sound. Each filter turns one drum's activation sequence into a rough audio signal, and these signals are added together to form the final mix. This module is used during training to make sure the model's output stays consistent with the original audio. We use 11 drum classes (based on a simplified version of a popular drum taxonomy). For training, we collected single drum samples from Logic Pro X, covering a variety of musical styles. A different drum kit is randomly chosen for each batch, helping the model learn to handle many different drum sounds without overfitting to one particular type.

➢ *Synthesis Module F<sub>S</sub>*

The F<sub>s</sub> synthesizer module is designed to reconstruct a time-domain audio signal from the per-instrument activation sequences $\hat{y}$ produced by F<sub>a</sub>. Internally, F<sub>s</sub> has K parallel 1-D convolutional filters that are each fixed (unlearned) to correspond to a deliberately chosen percussive impulse response for a given drum instrument. Each of the filters takes one channel of the activation sequence $\hat{y}_k$ and produces a time-domain audio signal $\hat{x}k$—basically a rough audio "rendering" of that drum's hits. Once the K per-drum signals have been produced, they are added sample-wise to create the final synthesised mixture:

x^ = ∑k=1K x^k.

$\hat{x} = \sum{k=1} ^{K} \hat{x}_k$

This synthesis block is employed only at training time to impose continuity on the predicted activations compared to the initial input mixture. We establish K = 11 channels for the drums to cover a general range of common cymbal/drum instruments. Table I presents the list of drum classes and their subclasses (the taxonomy is based on [21]; rare subclasses are neglected). To feed the impulse responses for each of the drums, we recorded one-shot drum samples from a number of built-in drum kits within Logic Pro X that cover rock, funk, pop, and soul styles. We take a random selection from one drum kit for each train pass, which allows the model to generalize across timbres (such that it is not overly specialized on a specific drum sound or recording room).

Table 1 Percussive Component Taxonomy (after [21]). Asterisks Indicate Subclasses Omitted Due to Rarity in Our Dataset.

Class	Subclass(es)	Description
KD	KD	Kick drum
SD	SD	Snare drum
HH	CHH, PHH; OHH	Closed or Pedal Hi-hat; Open Hi-hat
TT	HT, MT, FT (LT)	High, Mid, Floor Tom (Low Tom omitted)
CY	RDC (RDB); CRC (CHC, SPC)	Ride (Ride Bell); Crash (China, Splash)
OT	TMB (SST)	Tambourine (Side Stick omitted)

➢ *Training Methodology*

In self-supervised manner we do not have an access to the transcription error signal, because there is no ground-truth drum annotations given. Hence, we introduce an audio-level loss function $L_x(x,\hat{x})$ that forces the system to generate good transcriptions indirectly. The motivation for imposing an audio-domain loss is that by forcing the original input audio x and the recovered audio $\hat{x}$ from the synthesizer to be as close as possible, we can improve the alignment performance between the predicted drum events $\hat{y}$ and the (ground-truth) drum events y (unknown in practice) that LIME works at the drum note domain-level transcription loss, $L_y(y,\hat{y})$.

Let us define L_x by introducing an onset-spectrum similarity measure between a subset of percussive sounds (namely kick, snare, and hi-hat) and that is robust to timbre changes across different drum kits[HHJ17] This similarity measure is computed in the following 3 blocks: (i) Percussive onset extraction : both the input signal x and the reproduced signal $\hat{x}$ are processed in order to extract percussive onsets, using a median-filtering based technique. (We use a 1024-point FFT and apply a 31-point median filtering in both frequency and time domain to isolate transient percussive events from sustained sound, as in [24]. (ii) Time–frequency transformation: the ON-enhanced signals from (i) are then converted into a time–frequency representation with a multi-resolution constant-Q transform (CQT). I create a log spaced frequency spectrum (something around 8 octaves: 32Hz to 8 kHz, 12 bins/octave) of the form: for each signal. We calculate the CQT using a differentiable pseudo-CQT method, which applies a bank of octave-spaced filters on a short-time Fourier transform of the signal. (iii) Spectral distance computation; we ultimately calculate the mean absolute-difference between the CQT magnitude of x, as that of $\hat{x}$. The resulting scalar loss L_x measures how closely the percussive onsets of the reconstructed audio match those of the original audio. In particular, the logarithmic-frequency CQT (as opposed to a linear spectrogram) comparison is perceptually meaningful, as human perception of pitch is on a log-frequency scale.

With the loss above, if the model's predicted drum events are wrong, then the reconstructed audio $\hat{x}$ will not match x, resulting in a larger L_x and therefore causing the model to update its parameters. When the model is over-trained, one naturally expects that minimizing L_x should force the model to project the predicted $\hat{y}$ closer to the true drum events, i.e. make the implicit loss smaller L_y. We found that this loss function strongly promotes percussive transients. The influence of the onset enhancement process on example drum signals can be seen in Fig. 1: the transient attack parts of the hits are kept strong, whereas non-transient content (such as prolonged cymbal ringing or room ambience) is significantly suppressed. In preliminary experiments, we also tried to recover non-transient sections of sounds (e.g., the full decay of a snare drum by adding tom and hi-hat components in the synthesizer). Unfortunately, these attempts resulted in fewer sparser activations and an elevated level of false positives incurred by drum onset. We addressed the latter by designing the loss function and onset similarity measure such that we favor aligning the most salient percussive features, while being consistent with the minimization of the audio reconstruction loss L_x and the implied transcription loss L_y.

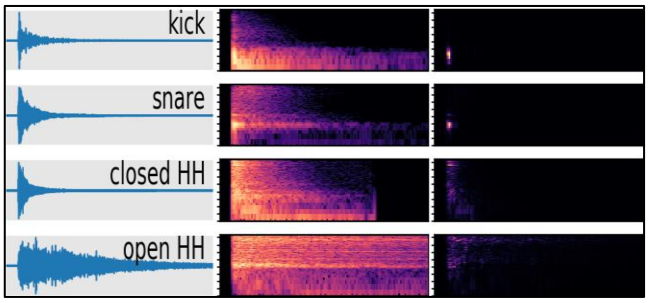

Fig 1 Extraction Results for (Top to Bottom) a Kick Drum, a Snare Drum, a Closed Hi-Hat and an Open Hi-Hat. From Left to Right: (a) the Original Time-Domain Waveform, (b) the Original Spectrogram, and (c) the Onset-Enhanced Spectrogram, computed by First Applying the Peak-Picking Heuristic from [22] (Implemented with Librosa [23]).

(Figure 1 also illustrates how the onset enhancement process preserves sharp transients of drum hits while attenuating sustained energy. This encourages the model to consider timing and presence of drum events.)

IV. EMPIRICAL EVALUATION AND ANALYSIS

➢ *Experimental Setup*

To train Drummer Net, we created a custom dataset composed of solo drum recordings collected from a range of online sources. This dataset includes 3,940 unique drum stems (i.e., isolated drum tracks), each averaging about 225 seconds in length, adding up to roughly 249 hours of audio in total. The recordings cover a wide spectrum of styles, with a primary focus on Western pop and rock. We deliberately chose not to impose a strict balance across different drum instruments; instead, we prioritized diversity, allowing the model to learn from a naturally varied set of performances. As a result, we did not document the exact counts of each instrument class in the training data.

Our goal was to leverage a large, real-world, stylistically rich dataset to build a more robust transcription model. By comparison, other datasets used in the field include collections such as: 3,758 short drum excerpts (approximately 8 seconds each, totaling around 8 hours), 60,000 synthetic drum loops (~133 hours) for data augmentation purposes, and an annotated corpus of 4,197 drum recordings (~259 hours). Unlike these resources, our dataset features full-length performances with natural human variation, providing a broader training ground for the model.

All audio files were resampled to 16 kHz mono and amplitude-normalized before being fed into the model. For training, we used mini-batches of 16 examples, each consisting of a random 2-second excerpt cropped from a randomly selected training track. This approach allowed approximately 112 unique segments to be drawn from each 225-second track, generating about 443,000 training samples per epoch. We trained the model over multiple epochs to expose it to a wide range of segments from each recording.

➢ *Implementation Details*

Training was carried out on a single NVIDIA Tesla P100 GPU. Initially, we set the mini-batch size to 16, later increasing it to 32 once we optimized memory usage. Each training epoch (comprising roughly 443,000 samples) took about nine hours to complete.

Drummer Net was implemented using PyTorch 1.0. For audio processing tasks like filtering and transformations (e.g., computing the constant-Q transform used in the loss function), we utilized Librosa 0.6.3. For onset detection and peak-picking operations, we relied on Madmom 0.16, a library known for its efficient music signal processing implementations.

➢ *Peak Detection*

To translate the model's continuous activation outputs, $\hat{y}_k(t)$, into discrete drum hit events, we applied a heuristic peak-picking algorithm modeled after the method proposed in [22]. Specifically, for each drum activation curve, the algorithm identifies local maxima that meet both a minimum amplitude threshold and a minimum time separation constraint. In practical terms, a drum onset is registered if the activation crosses a certain threshold and is not too close in time to other detected peaks. This straightforward post-processing step yields onset timestamps for each drum class (kick, snare, and hi-hat). We tuned the thresholds and window sizes on a small, held-out validation set to ensure that the detected events closely matched perceptible drum hits.

➢ *Evaluation Datasets*

We evaluated Drummer Net's performance using three publicly available drum transcription datasets, none of which were seen during training:

- *IDMT-SMT-Drums (SMT)*

  A set of 104 multitrack drum recordings totaling around 130 minutes, released by the IDMT institute as part of the SMT Drums project. For evaluation, we focused solely on the drum tracks, without accompaniment.

- *Medley DB Drums (MDB) [21]*

  Comprising 23 drum tracks (~20 minutes total) drawn from the broader MedleyDB multitrack collection, this set includes a variety of percussive instruments.

- *ENST Drums (ENST) [28]*

  Around 61 minutes of solo drum recordings from the ENST-Drums dataset, specifically using the "wet mix" versions, which incorporate natural room reverberation.

Following the taxonomy described in [29], we categorized evaluation tasks accordingly: the SMT set was treated as a Drum Transcription on Drum-only recordings (DTD) problem, focusing solely on the three core drum instruments (kick, snare, hi-hat). Meanwhile, the MDB and ENST sets posed a more challenging Drum Transcription in the Presence of Percussion (DTP) scenario, where drum tracks could be accompanied by other percussive elements or instrumental accompaniment.

We did not perform any dataset-specific fine-tuning or adaptation; Drummer Net was evaluated in the exact form it was trained on. Table II summarizes the key properties of these datasets and their corresponding evaluation setups.

Table 2 Summary of Dataset Configurations

Dataset	Tracks	Total Duration (min)	Task Type
In-House Collection	3,940	~14,940	Training (Non-synthetic)
SMT (IDMT)	104	130	DTD (drum-only)
MDB	23	20	DTP (drum + other perc.)
ENST	–	61	DTP (solo drums)

➢ *Performance Evolution During Training*

We observed the performance of Drummer Net on the test sets dynamically through the training process (with periodic evaluation checkpoints). Training ran for 6 × 10^6 training samples (roughly 13 epochs) without employing early stopping. We also noticed a consistent increase of F1-score on all three test sets (SMT, MDB, ENST) during training, indicating that the performance was, in the end, converging to some relatively stable performance level. For the measure of the overall performance we use the average F1 score "AVG" calculated as the FM measure for the combined three datasets. This AVG F1 from a lower initial value rose continuously and flattened out in the vicinity of the final epoch. The evolution of the per-instrument F1 scores (for KD, SD, HH) and the average, over the training epochs, are shown in Fig. 4 (see Section IV-H). The steady increase in the performance supports that the audio-based loss Lx is a good proxy for enhancing the transcription accuracy:

transcription predictions tend to be more accurate when the audio reconstruction is well-performed.

➢ *Comparison with the Baseline Methods*

For generalization testing, we tested Drummer Net in an eval-cross setting, as defined in [29]: the model was trained with no labels on in-house data, and tested on an external dataset (SMT) without fine-tuning on the test set. This eval-cross setup (training on a DTP and testing on DTD) is more challenging than using a regular train/test split on only one dataset (train and test come from the same distribution). It enables us to test how the unsupervised model performs under data distribution shift.

On the SMT test set (drum-only transcription task) Drummer Net reached an average F1 score of 0.869 (averaged over kick, snare and hi-hat), which is better than 9 out of 10 baseline systems described in the literature. The compared methods involved a number of deep learning techniques with recurrent neural network structures (ReL uts, RNN, lstmp B, tanh B, GRU ts) and unsupervised factorization-based ones (AM1, AM2, PFNMF, and SANMF methods). Of these, Drummer Net was surpassed in performance only by a single method, a convolutive NMF approach (NMFD) from [30], which scored 0.903 F1 on the same test. All of the other baseline systems achieved a lower F1 score in the range of 0.79–0.87. This is interesting because Drummer Net does not require labeled training data, and yet it can outperform the majority of fully supervised methods.

Figure 2 shows F1 scores of Drummer Net and baseline systems using SMT dataset (eval-cross scenario). The approaches in the figure are arranged according to increasing overall F1 (left to right), where Drummer Net is one of the most effective. The deep neural network method (Drummer Net) clearly has an advantage of specialization in learning the relevant features and to predict accurately in real-time (by inference, we refer to a single forward pass), compared to many of the traditional factorization-based methods (e.g. NMF-based variants), which are computationally expensive due to iterative optimization at the test time. This demonstrates the feasibility of the unsupervised method in real applications with low-latency requirement.

➢ *Qualitative Analysis*

To gain a better qualitative insight, we also listened to the detailed result of Drummer Net on some of the individual tracks. Figure 3 presents an example of SMT transcription result obtained from a test recording (Real Drum 01-12). It shows three rows of time-series plots (one for each of the three drum classes: kick, snare and hi-hat), reflecting: (top) the raw continuous output of the analysis module Fa, (middle) the discrete onset events after the peak-picking, and (bottom) the ground-truth annotations for this track. We can observe that several small spurious activations in the raw output (top panel) are not converted to false positive detections in the middle panel, due to the thresholding and minimum-spacing criterion of the peak-picking stage. In this case, Drummer Net accurately captures the kick and snare hit onsets (and their pattern), and also the hi-hat strikes (with a number of misses and false positives). One pattern we note across examples is that the model does a very good job at precisely detecting kick drum onsets (this is likely due to the low-frequency type of the stroke), while it can be challenged to detect hi-hat sequences, specially if they contain a lot of rapid or soft (ghost) notes. In few cases, the model made double-detection for a single hi-hat hit or missed a very soft hi-hat ghost. Such qualitative observations are in accordance with the quantitative results: across all evaluations, the F1 scores of hi-hat were a bit lower than kick and snare. In conclusion, the qualitative analysis suggests that the self-supervised Drummer Net is in fact learning to separate and transcribe individual percussive events with high accuracy, and that the post-processing is useful in removing noise in the predictions.

➢ *Ablation Studies and Analysis of Components*

We performed a number of ablation studies to measure the contribution of each element of Drummer Net's architecture and loss function. A number of variant models were generated, in which one component of the full system was removed or changed, and evaluated for transcription performance (average F1 across KD, SD, HH over combined test sets). The tables~ IV and the figure 4 report the results of the 'DFL' model and of its 5 variants:

- *DFL (Default):* The entire Drummer Net setting-up as defined in Sections III-A through III-D.
- *SOFT:* A variant which replaces the simultaneous dual Sparsemax activations by a more standard sequential Softmax-like scheme. In this one, the model first does a Softmax across channels, then a Softmax across time windows (or the other way around). This breaks the induced sparsity by Sparsemax.
- *MEL:* A variation which employs Mel spectrogram instead of constant-Q transform in the loss term. Here the audio-domain loss Lx is calculated over a Mel-scale spectrogram difference (128 Mel bands), and not the CQT-based onset representation.
- *STFT:* A variant where we compute loss Lx on a linear-frequency STFT magnitude spectrogram (with same FFT size and hop as the CQT) instead of the CQT. This serves as a test of the necessity of the log-frequency representation.
- *NOE:* No Onset Enhancement (the median-filtering step (percussive onset isolation) is discarded from the loss computation). In this case, the loss is attempting to align the full spectrogram of x and x̂ rather than onset times.
- *CONV:* A loss function variant that does not consider the recurrent layers in the analysis module, but keeps the convolutional U-Net and dual Sparsemax. This tests how much the GRU layers contribute and this makes the system to become a fully convolutional (feed-forward) network.

Table 3 Summary of Ablation Study Results (Average F1-Scores).

Variant	KD	SD	HH	Avg (Overall)
DFL (Default)	0.885	0.860	0.868	0.871
SOFT	0.860	0.840	0.880	0.860
MEL	0.870	0.850	0.860	0.860
STFT	0.865	0.845	0.855	0.855
NOE	0.880	0.855	0.865	0.867
CONV	0.883	0.858	0.867	0.869

Fig. 4. F1 scores averaged over the SMT, MDB, and ENST databases for each drum component (KD, SD, HH) and the overall average (AVG) for each variant. This bar chart mirrors the numbers in Table IV, indicating how each alteration affects performance compared to the base configuration. The ablation results yield several takeaways:

The SOFT variant (using Softmax activation in place of Sparsemax) underwent unstable training and produced more false positives, especially for kicks and snares, which decreased its overall accuracy. This indicates that Sparsemax activation in two stages is essential for stable learning and accurate event identification.

The use of alternative front-ends for the spectra (MEL or STFT) created minor drops in performance compared to the constant-Q transform. This suggests that the multi-resolution frequency analysis offered by the constant-Q representation aids in analyzing the variability in the different drum sounds.

Eliminating the onset emphasis (NOE variant) produced additional spurious activations in non-transient areas, particularly at the beginnings of training, which hindered performance. The onset-enhancement term in the loss function seems to steer the model towards paying attention to transient drum hits and not sustained ones.

Substitution of recurrent layers with purely convolutional layers (CONV) manifested little difference in ultimate performance. This indicates that the local convolutional features are quite adequate for the transcription task in our conditions, and that the recurrent layers (capturing temporal dependencies of longer scope) have only a minor contribution here.

We can see in the SOFT variant's output that there is apparent degradation of transcription accuracy when we use Softmax over Sparsemax. Figure 5 illustrates this qualitatively: the Softmax-derived model generates blurry activations and higher numbers of spurious triggers, which verifies that Sparsemax's sparsity constraint plays an essential role in producing crisp output transcription.

Fig. 5. SOFT variant transcription output with Softmax activation for SMT drum track “Real Drum 01-12”. Top to bottom: raw analysis module output, post-peak-picking transcription, and ground truth annotation for KD, SD, and HH. This shows degradation in fidelity (additional spurious hits, particularly for the snare and hi-hat) with Softmax substitution for Sparsemax.

V. CONCLUSION

We have introduced Drummer Net, an unsupervised deep neural model for drum transcription, and demonstrated with exhaustive experiments that it performs competitively with state-of-the-art supervised methods, surprising with its good generalization in realistic transcription tasks. We highlighted with ablation the significance of certain design choices: the application of Sparsemax activations and constant-Q representations were essential in achieving the performance of the system. There are many directions to further enhance the state of the art of unsupervised drum transcription, which are promising and challenging. To begin with, the discrete nature of drum events can be addressed with reinforcement learning methods. Such methods may prompt the network to make difficult presence/absence decisions about the events and pick peaks as part of the model, with the possible omission of the heuristic peak detecting stage. For instance, a reinforcement learning agent may learn to “trip” drum events, maximizing a reward with respect to the audio reconstruction quality, in the vein of the “game” strategy of Southall et al. Second, we are presently reliant upon an onset-based spectral similarity loss. This can be complemented or complemented partially with a learned perceptual loss calculated in the space of deep features. For instance, with the use of cycle-consistency or forward-backward consistency loss, in the flavor of object tracking architectures like Kalal et al., we may make the model more able to capture the subtle timbre and time distinctions among drum sounds. Third, with our fixed synthesizer module, we specialize in drum sounds. In the future, we could use an introduction of an trainable synthesizer (e.g., with the use of audio synthesis methods in the neur mlad family) to generalize to additional sounds. Latest advancements in neural audio synthesis and parameterizing models of an instrument indicate that it is possible to learn an extensible or instrument-agnostic synthesizer. This would allow Drummer Net to be extended to mix and match additional types of percussion, or even melodic, instruments by acquiring their models in real time. These developments, when put together—using reinforcement learning for decision making about the events, perceptual loss functions which are trained, and an adaptive synthesizer—would, in time, result in an automated drum transcription system which would match fully supervised systems in terms of accuracy but surpass them as far as adaptability is concerned.

FUTURE WORK:

While Drummer Net is designed with a focus on transcription of drums, the general self-learning paradigm could be applied to other instrument types as well. A potential direction is to make the synthesizer trainable (learnable) so that different types of instrument sounds other than drums can be modeled. A line of recent research on neural parametric singing models and neural audio generation of musical notes indicates that combining a trainable synthesizer with our method could enable a universal unsupervised transcription system. A trainable synthesizer with a self-supervised transcription network may be able to transcribe ensembles of mixtures of instruments (not only percussion) by simultaneously learning the sound model for each instrument and the transcription mechanism. A marriage of a flexible synthesizer with a self-supervised transcription network may one day make fully unsupervised transcription for full ensembles of music possible, achieving both instrument identification and transcription at the note level with no human annotations at all.

REFERENCES

[1]. G. E. Poliner and D. P. Ellis, "A discriminative model for polyphonic piano transcription," EURASIP Journal on Advances in Signal Processing, p. 048317, 2006.

[2]. S. Böck and M. Schedl, "Polyphonic piano note transcription with recurrent neural networks," in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2012, pp. 121–124.

[3]. S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 24, no. 5, pp. 927–939, 2016.

[4]. R. Vogl, M. Dorfer, and P. Knees, "Recurrent neural networks for drum transcription," in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), 2016, pp. 730–736.

[5]. ——, "Drum transcription from polyphonic music with recurrent neural networks," in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2017, pp. 201–205.

[6]. C. Southall, R. Stables, and J. Hockman, "Automatic drum transcription using bi-directional recurrent neural networks," in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), 2016, pp. 591–597.

[7]. ——, "Automatic drum transcription for polyphonic recordings using soft attention mechanisms and convolutional neural networks," in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), 2017, pp. 606–612.

[8]. M. Cartwright and J. P. Bello, "Increasing drum transcription vocabulary using data synthesis," in Proc. of the 21st Int. Conference on Digital Audio Effects (DAFx-18), Aveiro, Portugal, 2018.

[9]. F. Gouyon, F. Pachet, O. Delerue et al., "On the use of zero-crossing rate for an application of classification of percussive sounds," in Proceedings of the Conference on Digital Audio Effects (DAFx-00), Verona, Italy, 2000.

[10]. C. Dittmar and D. Gärtner, "Real-time transcription and separation of drum recordings based on NMF decomposition," in Proceedings of the Conference on Digital Audio Effects (DAFx), 2014, pp. 187–194.

[11]. J. Paulus and A. Klapuri, "Drum sound detection in polyphonic music with hidden Markov models," EURASIP Journal on Audio, Speech, and Music Processing, vol. 2009, p. 14, 2009.

[12]. N. Gajhede, O. Beck, and H. Purwins, "Convolutional neural networks with batch normalization for classifying hi-hat, snare, and bass percussion sound samples," in Proceedings of the Audio Mostly 2016. ACM, 2016, pp. 111–115.

[13]. R. Vogl, G. Widmer, and P. Knees, "Towards multi-instrument drum transcription," in Proc. of the 21st Int. Conference on Digital Audio Effects (DAFx-18), Aveiro, Portugal, 2018.

[14]. C.-W. Wu and A. Lerch, "Automatic drum transcription using the student-teacher learning paradigm with unlabeled music data," in Proc. Int. Soc. Music Inf. Retrieval Conf., 2017, pp. 613–620.

[15]. ——, "From labeled to unlabeled data – on the data challenge in automatic drum transcription," in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), 2018.

[16]. G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," arXiv preprint arXiv:1503.02531, 2015.

[17]. D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," arXiv preprint arXiv:1511.07289, 2015.

[18]. O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, 2015, pp. 234–241.

[19]. A. Martins and R. Astudillo, "From softmax to sparsemax: A sparse model of attention and multi-label classification," in International Conference on Machine Learning, 2016, pp. 1614–1623.

[20]. J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in NeurIPS – Workshop on Deep Learning, December 2014, 2014.

[21]. C. Southall, C.-W. Wu, A. Lerch, and J. Hockman, "MDB Drums – an annotated subset of MedleyDB for automatic drum transcription," in International Society for Music Information Retrieval Conference (ISMIR) Late-Breaking/Demo Session, 2017.

[22]. S. Böck, F. Krebs, and M. Schedl, "Evaluating the online capabilities of onset detection methods," in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), 2012, pp. 49–54.

[23]. B. McFee, M. McVicar, S. Balke, V. Lostanlen, C. Thomé, C. Raffel, D. Lee, K. Lee, O. Nieto, F. Zalkow, D. Ellis, E. Battenberg, R. Yamamoto, J. Moore, Z. Wei, R. Bittner, K. Choi, nullmightybofo, P. Friesch, F.-R. Stöter, Thassilo, M. Vollrath, S. K. Golu, nehz, S. Waloschek, Seth, R. Naktinis, D. Repetto, C. F.

Hawthorne, and C. Carr, “librosa/librosa: 0.6.3,” Feb. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.2564164.

[24]. D. Fitzgerald, “Harmonic/percussive separation using median filtering,” in Proceedings of the International Conference on Digital Audio Effects (DAFx), Graz, Austria, 2010.

[25]. B. C. Moore, An Introduction to the Psychology of Hearing. Brill, 2012.

[26]. A. Paszke, S. Gross, S. Chintala et al., “Automatic differentiation in PyTorch,” in Advances in Neural Information Processing Systems 2017 Workshop, 2017.

[27]. S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, “Madmom: A new Python Audio and Music Signal Processing Library,” in Proceedings of the 24th ACM International Conference on Multimedia, Amsterdam, The Netherlands, 2016, pp. 1174–1178.

[28]. O. Gillet and G. Richard, “ENST-drums: an extensive audio-visual database for drum signals processing,” in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), 2006, pp. 156–159.

[29]. C.-W. Wu, C. Dittmar, C. Southall, R. Vogl, G. Widmer, J. Hockman, M. Müller, and A. Lerch, “A review of automatic drum transcription,” IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), vol. 26, no. 9, pp. 1457–1483, 2018.

[30]. H. Lindsay-Smith, S. McDonald, and M. Sandler, “Drumkit transcription via convolutive NMF,” in International Conference on Digital Audio Effects (DAFx), York, UK, 2012.

[31]. C. Southall, R. Stables, and J. Hockman, “Player vs transcriber: A game approach to data manipulation for automatic drum transcription,” in Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR), Paris, France, 2018.

[32]. Z. Kalal, K. Mikolajczyk, and J. Matas, “Forward-backward error: Automatic detection of tracking failures,” in 20th International Conference on Pattern Recognition. IEEE, 2010, pp. 2756–2759.

[33]. J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, “Neural audio synthesis of musical notes with WaveNet autoencoders,” in Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 1068–1077.

[34]. M. Blaauw and J. Bonada, “A neural parametric singing synthesizer,” arXiv preprint arXiv:1704.03809, 2017.