DEEP UNSUPERVISED DRUM TRANSCRIPTION

Keunwoo Choi
Spotify
keunwooc@spotify.com

Kyunghyun Cho
New York University, Facebook AI Research
kyunghyun.cho@nyu.edu

ABSTRACT

We introduce DrummerNet, a drum transcription system that is trained in an unsupervised manner. DrummerNet does not require any ground-truth transcription and, with the data-scalability of deep neural networks, learns from a large unlabeled dataset. In DrummerNet, the target drum signal is first passed to a (trainable) transcriber, then reconstructed in a (fixed) synthesizer according to the transcription estimate. By training the system to minimize the distance between the input and the output audio signals, the transcriber learns to transcribe without ground truth transcription. Our experiment shows that DrummerNet performs favorably compared to many other recent drum transcription systems, both supervised and unsupervised.

1 Introduction

Transcription is a music information retrieval task with the goal of estimating the score y when input audio x is given. The majority of the recent transcription systems is based on supervised learning, where the transcriber is an *analysis* system $\hat{y} = F_a(x)$ that is trained with annotated pairs $\{(x_m, y_m)\}_{m=1}^{M}$ to minimize the distance between y and $\hat{y}$ [6, 7, 27, 31, 33, 34, 37, 38].

The trend is similar in drum transcription on which we focus in this paper. Supervised learning approaches may incorporate models based on frame-based feature extraction and classification [15], non-negative matrix factorization (NMF) for pattern matching [10], or hidden-Markov model [25]. More attention has been given recently to deep learning based models such as convolutional neural networks (CNNs, [13, 34]) and recurrent neural networks (RNNs, [33, 37, 38]), all of which have greatly improved transcription systems.

However, the lack of a large-scale annotated dataset is one of the most frequently mentioned obstacles that hinder further improvement. In practice, this limits the generalizeability of supervised learning systems, as will be discussed in Section 4, and using synthetic data is one way to address this issue [7, 39]. Although there have been proposals to use unlabeled data [42, 43], the issue remains as they still rely on supervised learning combined with teacher-student learning [16]. Parallel to those approaches, an annotation-free and, therefore, a more scalable and generalizable alternative would be unsupervised learning.

Unsurprisingly, one of the humans' music learning procedures, self-taught by trial-and-error, is very similar to unsupervised learning. For example, musicians learn to transcribe by (a) listening, (b) playing an instrument, (c) identifying differences, and (d) making adjustments. *Can this be done without any supervision? Yes*, if the person can spot the pitch difference (e.g., the pitch should be higher or lower). Consistent with this logic, developing a transcription system based on unsupervised learning would be feasible if the system can test the estimation, measure the error, and correct itself accordingly.

To implement such an unsupervised transcription system, we need a *synthesis* system, $\hat{x} = F_s(\hat{y})$, making the overall system $\hat{x} = F_s(F_a(x))$. During its training, the system is given $\{x\}_{m=1}^{M}$ and trained to minimize the distance between x and $\hat{x}$. There have been few systems relying on unsupervised learning as explained above. In MIR, the system in [1] utilized sparse coding to learn a dictionary of time-frequency templates of piano and harpsicord, assuming a (matrix-)multiplication model with additive noise, $F_s(\mathbf{y}) = \mathbf{A}\mathbf{y} + \mathbf{e}$. Yoshii et al. proposed to use sparse coding in a jointly-learned chord recognition and transcription system [44]. Berg et al. designed a probabilistic graphical model that parameterizes the spectral and temporal envelopes, note events, and note activations, in order to transcribe piano by inferring their parameters [2]. In drum transcription, many systems have used NMF to decompose a drum track spectrum into spectral templates and their temporal activations (or transcription) [26, 41]. Several variants of NMF were proposed to address the limits of the fixed spectrum template of NMF [19, 20, 29]. Lastly, a similar system can be found in computer vision, where the parameters of input images are estimated by reconstruction using an image renderer [18].

In this paper, we introduce DrummerNet, a deep neural network based drum transcription system that is trained by unsupervised learning. With a more end-to-end approach, DrummerNet is distinguished from previous research [1, 2, 44], which has strong priors on the target sounds. In §2, we present the system design principle behind DrummerNet, followed by its details in §3. In §4, the evaluation results are discussed along with the ablation study. We present our conclusion, the problems of our system, and the future direction towards fully unsupervised learning in transcription/MIR in §5.

Name	Description	Note
n, N	The temporal index/length of audio input	
k, K	The index/total number of drum components	K=11
x, y	Mixture and transcription	$\in \mathbb{R}^N$
$\hat{x}, \hat{y}$	Estimations of mixture/transcription	$\in \mathbb{R}^N$

Table 1: Symbols used in this paper

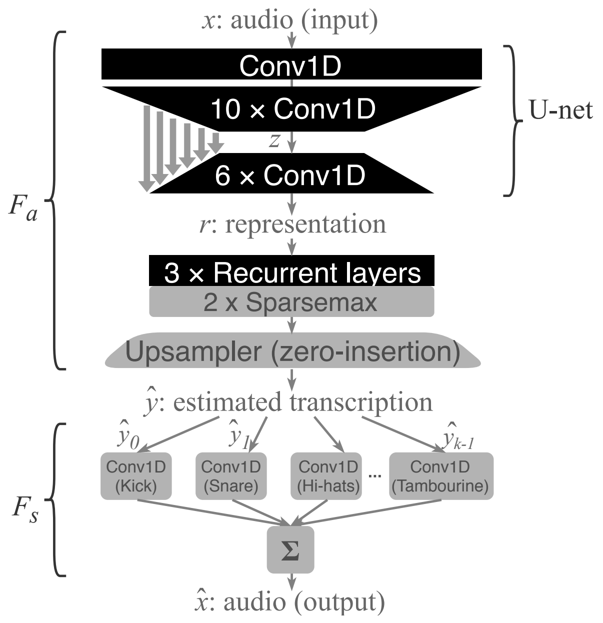

Figure 1: Block diagrams of DrummerNet structure. Trainable modules are illustrated as black boxes and fixed modules as rounded grey boxes.

2 System Design Principles

Training the proposed DrummerNet is similar to the previous unsupervised learning approaches in music [1, 2, 44], as they all train a system to output $\hat{x}$ that reconstructs the original signal x. The difference between $\hat{x}$ and x works as a proxy of the difference between $\hat{y}$ to y.

There are three conditions under which unsupervised learning of a transcriber can be achieved successfully. First, the output of the analysis module F_a must be in the form of transcription – a set of discrete events representing the timing and intensity of the notes. Second, the synthesis module F_s must synthesize the audio signal given the transcription input $\hat{y}$. Third, all the components in the network must be differentiable as we rely on backpropagation to train it.

3 DrummerNet

In this section, we introduce the proposed system structure. We specify the number of channels, kernel size, and stride as `(channel, kernel, stride)`. All the convolutional and recurrent layers use an exponential linear unit as an activation function [9]. [1]

[1] The implementation of DrummerNet is available on `https://github.com/keunwoochoi/DrummerNet`

3.1 Analysis module F_a

The analysis module F_a, as illustrated in the top half of Figure 1, takes the audio signal x as an input and processes it through a series of U-net variant [30], recurrent layers, and gated Sparsemax activation [21]. After training, this module is used as a transcriber (with peak-picking).

U-net The U-net consists of 1D convolutional layers, max-pooling layers, and concatenations between the encoder and the decoder. The encoder consists of a convolutional layer `(128, 3, 1)` followed by 10 convolutional layers `(50, 3, 1)` interleaved with max-pooling of size 2. As a result, it outputs $z \in \mathbb{R}^{N/1024}$ which has a receptive field size of 3,072 time steps.

The decoder has only 6 convolutional layers `(50, 3, 1)` interleaved with a concatenation with the feature map at the same depth as in the encoder and a $\times 2$ bi-linear interpolation. We call the output of decoder $r \in \mathbb{R}^{N/16}$, the representation based on which the transcription is estimated. The asymmetry between the encoder and the decoder makes the length r to be shorter by a factor of $4^2 = 16$ compared to that of input x. Assuming the input audio is sampled at 16 kHz, [2] r would have a sampling rate of 1,000 Hz.

Recurrent layers We use three recurrent layers: (GRUs [8]) $\{$along time-axis, bi-directional, 100-channel$\}$, $\{$along time-axis, uni-directional, 50-channel$\}$, and $\{$along channel-axis, uni-directional, K-channel$\}$. These three recurrent layers have properties of i) being bi-directional so that the onset at n can be determined by the vicinity of n (both the past and the future), ii) enforcing temporal dependency, and iii) enforcing component-wise dependency, respectively. The width (or the hidden vector size) of the third recurrent layer is equal to K, the number of drum components in the synthesizer, to map each channel to each drum component.

Sparsemax In an ideal case of transcription, there would be local sparsity along both the time and channel-axes because the drum events would not repeat with a rate of 1,000 Hz (which is faster than 16-beat on 240 BPM), nor would all the K drum components be activated simultaneously. Although sparsity is one of the properties that *can* be achieved by the autoregressive nature of the recurrent layers, we add Sparsemax [21] activation to encourage it explicitly. The output of Sparsemax has two important properties: i) it always sums to 1 (same as Softmax) and ii) it is highly likely to be sparse with actual zeros (unlike Softmax). In DrummerNet, two Sparsemax layers are applied in parallel, one along channel-axis (=instrument-axis) and the other time-axis within a non-overlapping window size of 64. This design choice is based on the assumption that there are only a few onsets among notes (channel-axis sparsity) and within 64 samples at $\hat{y}$, or 64 ms (temporal sparsity). The outputs from these two Sparsemax layers are then multiplied element-wise.

[2] This is the sampling rate of input audio in our experiment.

Class	Subclass	Description
KD	KD	Kick drum
SD	SD	Snare drum
HH	CHH, PHH	Closed/pedalled hi-hat
	OHH	Open hi-hat
TT	HIT, MHT,	High/high-mid/
	HFT, LFT*	high-floor/low-floor tom
CY	RDC, RDB*	Ride cymbal, ride cymbal bell
	CRC, CHC*,	Crash/china cymbal
	SPC*	splash cymbal
OT	SST*, TMB	side stick, tambourine

Table 2: A drum component hierarchy [36]. The synthesizer F_s consists of 11 classes, following Subclass of the table with omitting ones marked with asterisks *.

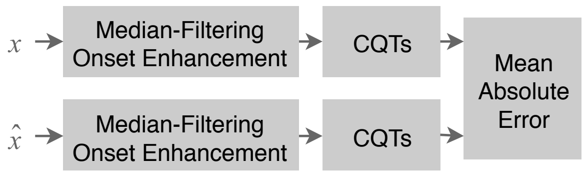
x
$\hat{x}$
Median-Filtering Onset Enhancement
Median-Filtering Onset Enhancement
CQTs
CQTs
Mean Absolute Error

Figure 2: The block diagrams of loss calculation

Upsampler Finally, the low temporal resolution of the Sparsemax output is addressed by zero-insertion upsampling by the factor of 16. According to this, we modify the temporal quantization rate of events, unlike the upsampling of a digital signal.

3.2 Synthesis module F_s

The synthesis module F_s consists of K parallel 1D convolutional layers and a channel-wise summing operator. The kernel of each layer is not trained but fixed to the known waveform of each drum component to convert a transcription of a component $\hat{y}_k$ into a track $\hat{x}_k$. The tracks are summed to generate the final output $\hat{x}$ ($= \sum_{k=1}^{K} \hat{x}_k$), the synthesized audio signal. This module is only used during training.

In the implementation, we use $K = 11$, using Subclass in Table 2, following [36]. Ones marked with asterisks were excluded due to their scarcities in our source of isolated drum recordings, which consisted of 12 virtual drum instruments provided by Logic Pro X. Multiple drum kits, including rock, pop, funk, and soul [3], were used to prevent the network from overfitting to a specific drum kit. During training, a drum kit was randomly assigned for every batch.

3.3 Learning

Unable to directly compute the transcription loss during unsupervised learning, we carefully designed a loss function at the audio level, $L_x(x, \hat{x})$, as minimizing it would also minimize the transcription loss, $L_y(y, \hat{y})$. To do so,

[3] Brooklyn, Heavy, Liverpool, Neo Soul, Detroit Garage, Motown Revisited, Portland, Sunset, Speakeasy, SoCal, Smash, and Slow Jam. All with velocity=98.

Module	Input (size)	Output (size)
U-net encoder		
Conv1D	x:$(1, N)$	$(C*, N)$
10× Conv1D	$(C*, N)$	$(C*, N/1024)$
U-net decoder		
6× Conv1D	$(C*, N/1024)$	r:$(C*, N/16)$
Recurrent layers	$(C*, N/16)$	$(K, N/16)$
Sparsemax	$(K, N/16)$	$(K, N/16)$
Upsampler	$(K, N/16)$	$\hat{y}$:(K, N)
Synthesis module		
Channel splitter	$\hat{y}$:(K, N)	$K \times \hat{y}_k$:$(1, N)$
Each Conv1D	$\hat{y}_k$:$(1, N)$	$\hat{x}_k$:$(1, N)$
Sum (mixer)	$K \times \hat{x}_k$:$(1, N)$	$\hat{x}$:(K, N)

Table 3: The shapes of inputs/outputs of the module in DrummerNet. $C*$ indicates the number of channels but unspecified.

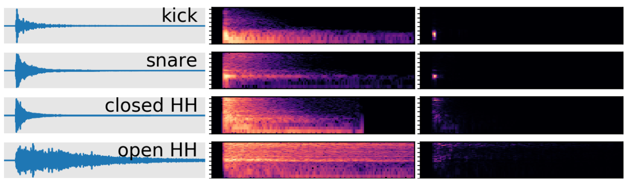

Figure 3: The effect of drum extraction for kick, snare, close hi-hat, and open hi-hat, from top to bottom. Columns are from left to right: original waveform, original spectrum, and onset-enhanced spectrum

L_x should be able to differentiate the drum components – kick drum (KD), snare drum (SD), and hi-hat (HH) – while being invariant to the varying drum kits. Perceptually, there are clear differences between KD, SD, and HH. Although both impulsive, KD is in the low-frequency band while SD is in the mid-frequency band. SD is also relatively tonal and has a longer envelope. HH is more complicated to describe due to its variation from its playing technique. For example, closed and pedalled-HH's are in the high-frequency band, impulsive, and with relatively low energy, while open-HH's are similar except louder with a longer noisy envelope.

We thus define and use *onset spectrum similarity*, which is designed to represent the similarity based on the onset part of sounds in the spectrum domain. As illustrated in 2, it is measured by i) applying median-filtering based drum extraction [12] which enhances onsets (with a FFT size of 1024 and median filter length of 31 on both axes), ii) converting to multi-resolution CQTs (constant-Q transform) for both x and $\hat{x}$, and then iii) calculating the mean absolute difference between them.

Among many spectral magnitude representations, we use (log-magnitude) CQT since the logarithmic frequency scale is known to match well to human auditory perception [23]. We followed the implementation of Pseudo-CQT [4] which multiplies linear-to-octave filterbanks to an STFT. As a result, the CQT covered nearly 8-octave bands

[4] `http://librosa.github.io/librosa/`

from 32.07 Hz (C1) to 8 kHz (the Nyquist frequency of our experiment) with a 12-band/octave resolution. This implementation is differentiable.

Figure 3 shows the effect of onset enhancement. It preserves the characteristics of the drum components in the transient part while removing the after-onset components. This process makes L_x and L_y more similar, as the non-transient parts vary more among drum kits due to their random and noisy nature. In a preliminary experiment, for example, the network tried to reconstruct all the non-transient components of SD using tom-toms and HHs, resulting in non-sparse and severe false-positive detection of onsets.

4 Experiments and Analysis

4.1 Setup

For the training of DrummerNet, we used an in-house dataset of drum stems that are crawled from many websites. The dataset consisted of 3,940 unique tracks averaging 225 seconds each for a total of 249 hours. Since the dataset was crawled from various websites, some details, such as the distribution of drum components, are hard to identify. The tracks were mostly popular western rock/pop music. Alternatives to this in-house dataset can be found in [7] (3,758 drum sample recordings (×8 second = over 8 hours) or 60,000 synthesized drum loops (×8 second = over 133 hours)) and [39] (4,197 drum tracks (259 hours)). We opted for the in-house dataset because it provided more diversity as it was not synthesized.

Each audio file was resampled to 16 kHz and down-mixed to mono. The training batch size was 16, and for each audio file, we randomly selected a 2-second segment. On average, there were 112.5 segments in a track, and therefore training with 443,250 (=3,940 × 112.5) items would be approximately one epoch. With a Nvidia Tesla P100 and a batch size of 32, it took about 9 hours to train a single epoch. We implemented DrummerNet using Pytorch 1.0 [24] and used Librosa 0.6.3 [22] and Madmom 0.16 [4] for audio processing and peak-picking.

We used a heuristic peak-picking method introduced in [5]. This method selects a peak $\hat{y}[n]$ at n if it satisfies the three conditions in Eq. (1),

$$\begin{aligned}\hat{y}[n] &= \max(x[n-w_m], ..., x[n+w_m]) \\ \hat{y}[n] &\geq \text{average}(x[n-w_a], ..., x[n+w_a]) + \delta \\ n &> n_{lp} + w_w,\end{aligned} \quad (1)$$

where the max window w_m=50 ms, average window w_a=100 ms, threshold δ=0.2, waiting window w_w=50 ms, and n_{lp} is the last detected peak. We mainly use F1 score along with Precision and Recall using mir_eval [28]. The tolerance window is 50 ms.

After training, we test the system on three public datasets: IDMT-SMT-Drums (SMT, 104 drum tracks, total 130 minutes [10]), Medley-DB Drums (MDB, 23 tracks, total 20 minutes [36]), and ENST-drums (ENST, 61 minutes [14], drum-only tracks known as 'wet-mix' of 'minus-one' subset). According to [40], a task is DTD[5] if tracks are drum-only, more precisely KD/SD/HH-only, and the system annotates KD/SD/HH events. This is the case for the SMT dataset. A task with the system annotating KD/SD/HH but with drum tracks consisting of more than those three components, e.g., tom-toms and cymbals, is named DTP[6] in [40]. Following this convention, we evaluate DTD with SMT (Section 4.3), and DTP with MDB/ENST. We did not fine-tune for any dataset in any experiment and used the whole datasets for evaluation only.

[5] DTD: drum transcription of drum-only recordings

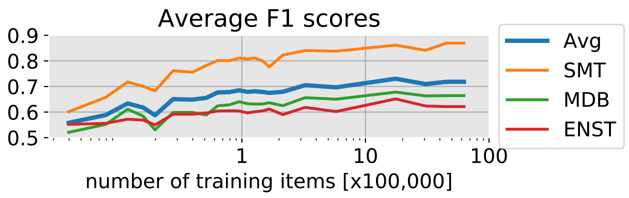

Figure 4: The F1 scores of DrummerNet over training items on each dataset (SMT, MDB, ENST), averaged over KD, SD, and HH. AVG indicates the overall average F1 scores of three datasets.

4.2 Trend of Performance over Training

We did not employ a stopping strategy but trained the network for 6 × 10^6 items (about 13 epochs). As illustrated in Figure 4, the overall performance gradually increases as the training proceeds and approaches converging towards the end of training. This indicates that the proposed loss function is a good proxy of transcription loss. After the initial phase of training, the performance differences among datasets remain consistent, probably due to the different characteristics of drum tracks in each dataset, as will be discussed in Section 4.4.

4.3 Relative Performance against Baselines

In this experiment, we trained our system on the in-house training set without any annotation and evaluated it on a separate test set (also known as 'eval-cross (trained on DTP)', [40]), which is a stronger condition than a usual train/test split scenario in supervised learning ('eval-subset', [40]). This setup allows us to measure the generalization capabilities across the datasets. Specifically, our experiment is equivalent to DTD, 'eval-cross (trained on DTP)' experiment in [40].[7], which is only available on SMT. Therefore, only the performances on SMT are compared in this Section. Overall, the performance of DrummerNet is favorable to that of recent drum transcription systems. With an average F1 score of 0.869 on SMT, the proposed unsupervised DrummerNet outperformed 9 out of 10 systems. The nine systems include ones with deep neural networks and supervised approach (ReLUts, RNN, lstmpB, tanhB, and GRUts [33,34,37,38]), as well as ones with NMF and unsupervised approach (AM1, AM2,

[6] DTP: drum transcription in the presence of percussion

[7] Numbers are omitted in the paper but are available online: https://www.audiolabs-erlangen.de/resources/MIR/2017-DrumTranscription-Survey.

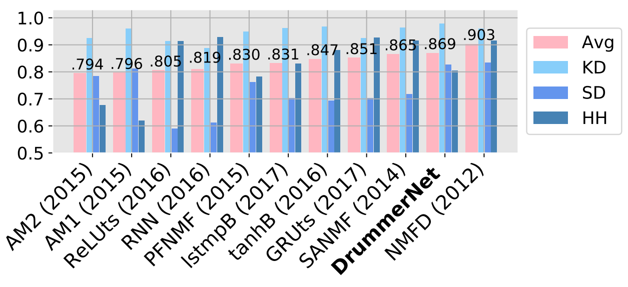

Figure 5: The F1 scores of DrummerNet and other systems on SMT with 'eval-cross' setup, sorted by the ascending order of the overall average. The system names follow [40].

PFNMF, and SANMF [10, 41]). It did not outperform NMFD [20], a system based on the convolutive NMF.

The comparison between DrummerNet and the NMF/unsupervised learning-based systems [10, 41] implies that the proposed deep neural network structure effectively learns relevant representations. Furthermore, DrummerNet allows constant-time inference, unlike NMF and other factorization-based approaches which require iterative optimization in the test time.

What is more interesting is its *generalizability*. All the deep learning based systems [8] present deteriorating performance in the transfer learning scenario (eval-cross) compared to the dataset split scenario (eval-subset). [9] However, less data-driven approaches [10] present similar or even increased performances in eval-cross. This implies that the distributions within datasets are fairly different and biased to certain types of drum tracks and therefore, a transcription system trained with those datasets will be also biased accordingly. This limitation may be attributed to the small sizes of those datasets. Theoretically, supervised deep learning systems may generalize better if trained on a very large dataset, which lacks practicality due to the high annotation cost. In contrast, it is relatively easy to *unbias* DrummerNet. One only needs to control the distribution of drum tracks by their style/genre/sounds without annotating every note.

4.4 Qualitative Analysis

In this section, we will analyze the performance and the behavior of DrummerNet by components, datasets, and metrics, as illustrated in Figure 6. Here, we notice two clear trends. First, across all of the three datasets and the metrics, detecting KD was the easiest, followed by SD and HH (except the precision on SMT). Second, SMT seems to be the easiest, followed by MDB and ENST. *What could be the reasons?*

The first trend is strongly related the proposed loss function. KD has the least within-class variability while being the most distinguishable component (the largest mutual-class variability) due to its solitary frequency range. SD and HH share both the mid and high-frequency ranges

[8] RNN, tanhB, ReLUts, lstmpB, GRUts - RNN-based systems
[9] See Figure 10 (b) of [40]. Note that most of the reported scores in papers also follow eval-subset setup.
[10] SANMF, NMF, PFNMF, AM1, AM2 - NMF-based systems

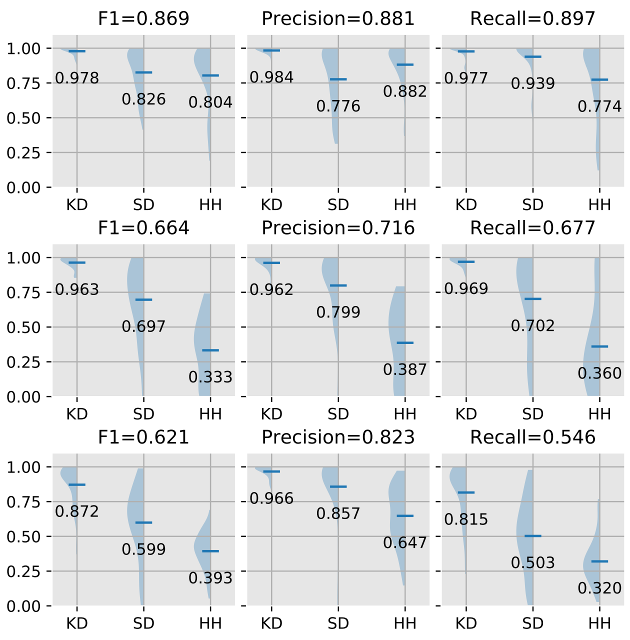

Figure 6: Evaluation of DrummerNet on SMT (top), MDB (middle), and ENST (bottom) datasets.

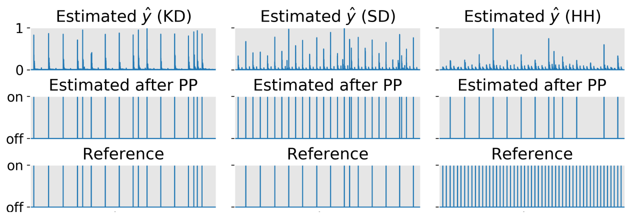

Figure 7: A transcription example of DrummerNet, 'Real Drum 01-12' in SMT - the output of analysis module (top), after peak-picking (middle), and ground truth (bottom); KD, SD, HH (left to right).

and their sounds can vary significantly across drum kits – i.e., larger within-class variability and smaller mutual-class variability. A common pattern, consequently, is the false positive of HH due to SD and vice versa. This is presented in Figure 7, where SD has many false positives due to HH.

The second trend is caused by the mixed use of the probability and the onset velocity in the DrummerNet. Although transcription $\hat{y}$ *is* the estimated amplitude of drum components, the peak-picking method treats $\hat{y}$ as if it was a probability. This discrepancy becomes problematic when the velocities of drum events in a track vary drastically as in the case of MDB and ENST. A failure case is demonstrated in Figure 7, where the HH with strong accents on several occasion caused DrummerNet to miss many of the other HH peaks.

4.5 Ablation Study

We conducted an ablation study where the performance of DrummerNet is compared with that of its variants. Figure 8 shows the reported F1 scores averaged over datasets and components. Please refer to the caption in Figure 8 for the definitions of the system names.

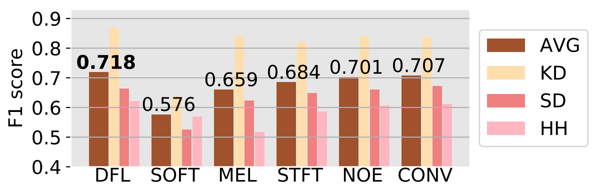

Figure 8: The ablation study results, F1 scores averaged over three datasets per component (KD, SD, HH) and their overall average (AVG). The label indicates as follow: DFL (default DrummerNet as introduced), SOFT (two Softmax layers instead of Sparsemax), MEL (use 128-band melspectrogram instead of CQTs), STFT (use 1024-point STFT instead of CQTs), NOE (not onset enhancement in loss), CONV (3-layer convolutional layers instead of recurrent layers).

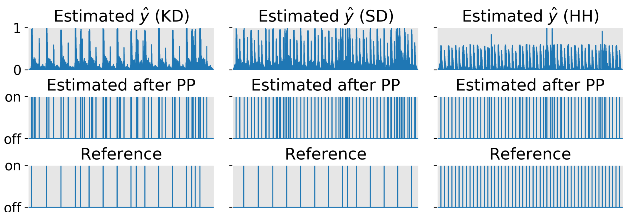

Figure 9: A transcription example of SOFT (DrummerNet with Softmax) , 'Real Drum 01-12' in SMT - the output of analysis module (top), after peak-picking (middle), and ground truth (bottom); KD, SD, HH (left to right).

Sparsemax (DFL vs. SOFT) Among all the variants in this experiment, we observe the most dramatic change in the performance when we replaced Sparsemax with Softmax (SOFT), mostly in a negative way. In SOFT, the two Softmax layers were applied in sequence instead of in-parallel and multiplied, which we tested, but the training was unstable. The transcription $\hat{y}$ of SOFT tends to be much noisier with many false positives, as presented in Figure 9. We conclude that the sparsity induced by Sparsemax is a crucial factor behind the success of the proposed unsupervised transcription.

Figure 9 provides a good example of the performance degradation pattern for each component. As in Figure 8, although the scores of all the three components decrease in SOFT, the degradation is not as critical for HH as in the case of KD/SD. This observation reflects the underlying properties of the different components. KD and SD are sparser than HH, and thus may benefit more from the introduction of Sparsemax.

CQT (DFL vs. MEL vs. STFT) Replacing CQTs with either melspectrograms (MEL) or short-time Fourier transform magnitudes (STFT) results in decreased performance. Unlike CQTs, where different numbers of FFT are used for each octave range, melspectrograms are computed based on single-resolution STFT. This implies that DrummerNet benefits from CQTs which consider multiple temporal and spectral resolutions.

Comparing MEL and STFT, the melfrequency compression helps with the better detection of KD but not SD nor HH. This is explained by the different frequency band weighting of STFT and melspectrogram. Since melfrequency is linear below 1 kHz and logarithmic above 1 kHz [32], melspectrogram allocates relatively more bins below 1 kHz. This means that the loss function in MEL is biased towards the low-frequency range, resulting in training that favors KD over the others.

Onset Enhancement (DFL vs. NOE) The onset enhancement is shown to be boosting the performance, but not significantly (0.017). In the learning curve, we observe that removing the onset enhancement from the loss function results in a large performance degradation during the initial phase of training. This is mainly due to false-positives in the non-transient part.

Recurrent layers (DFL vs. CONV) Overall, replacing three recurrent layers with three convolutional layers does not make significant differences (0.011). This may means i) a long-term relationship may not provide additional information, probably because the transcription largely depends on local information, and ii) the mutual conditioning in the last recurrent layer is not effective in our experiment. In an informal analysis, we observed that with recurrent layers, $\hat{y}$ still has some local temporal correlation, e.g., the activations are smeared over time, probably because that is better to reconstruct the input audio.

5 Conclusion

We introduced DrummerNet, a deep neural network that is trained to transcribe drum tracks without a labeled dataset. In the experiment, DrummerNet achieved strong performance compared to existing systems trained with supervised learning, showing its generalizability towards a real-world drum transcription scenario. Our ablation study showed that Sparsemax and CQT played a crucial role in the successful training of DrummerNet.

The experiment also revealed room for further improvements. Considering the discreteness of the musical notes, a reinforcement learning approach may be more suitable [35], making the prediction more sparse and replacing the peak-picking with trainable action. The onset-enhancement on audio similarity is a function carefully-chosen in order to approximate L_y when x and $\hat{x}$ are given. Unfortunately, the approximation is limited because the exact drum sounds in x are not given, and therefore a perfect reconstruct of (onsets of) the input audio ($L_x(x, \hat{x}) = 0$) does not lead to a perfect transcription ($L_y(y, \hat{y}) = 0$). An alternative way would be measuring a similarity on a (perceptual) representation domain instead of the audio, for example, by learning a loss using forward-backward consistency (also known as a cyclic loss [17]) or known audio features. Lastly, the current synthesizer module is limited to drums as it does not handle the duration of notes. A trainable synthesizer can be used to expand DrummerNet to other instruments [3, 11], eventually leading to an unsupervised universal transcription system combined with instrument recognition.

6 Acknowledgement

We thank Tristan Jehan and Sebastian Ewert for their valuable comments and discussions. We would also like to express our sincere gratitude to Chih-Wei Wu for sharing his insight with us.

7 References

[1] Samer A Abdallah and Mark D Plumbley. Unsupervised analysis of polyphonic music by sparse coding. *IEEE Transactions on neural Networks*, 17(1):179–196, 2006.

[2] Taylor Berg-Kirkpatrick, Jacob Andreas, and Dan Klein. Unsupervised transcription of piano music. In *Advances in neural information processing systems*, pages 1538–1546, 2014.

[3] Merlijn Blaauw and Jordi Bonada. A neural parametric singing synthesizer. *arXiv preprint arXiv:1704.03809*, 2017.

[4] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: a new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, 10 2016.

[5] Sebastian Böck, Florian Krebs, and Markus Schedl. Evaluating the online capabilities of onset detection methods. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 49–54, 2012.

[6] Sebastian Böck and Markus Schedl. Polyphonic piano note transcription with recurrent neural networks. In *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 121–124. IEEE, 2012.

[7] Mark Cartwright and Juan Pablo Bello. Increasing drum transcription vocabulary using data synthesis. *Proc. of the 21st Int. Conference on Digital Audio Effects (DAFx-18). Aveiro, Portugal*, 2018.

[8] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NeurIPS - Workshop on Deep Learning, December 2014*, 2014.

[9] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[10] Christian Dittmar and Daniel Gärtner. Real-time transcription and separation of drum recordings based on nmf decomposition. In *Proceedings of the conference on Digital Audio Effects (DAFx)*, pages 187–194, 2014.

[11] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. Neural audio synthesis of musical notes with wavenet autoencoders. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1068–1077. JMLR. org, 2017.

[12] Derry Fitzgerald. Harmonic percussive separation using median filtering. *Proceedings of the conference on Digital Audio Effects (DAFx), Graz, Austria, 2010*, 2010.

[13] Nicolai Gajhede, Oliver Beck, and Hendrik Purwins. Convolutional neural networks with batch normalization for classifying hi-hat, snare, and bass percussion sound samples. In *Proceedings of the Audio Mostly 2016*, pages 111–115. ACM, 2016.

[14] Olivier Gillet and Gaël Richard. Enst-drums: an extensive audio-visual database for drum signals processing. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 156–159, 2006.

[15] Fabien Gouyon, François Pachet, Olivier 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.

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[17] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Forward-backward error: Automatic detection of tracking failures. In *20th International Conference on Pattern Recognition*, pages 2756–2759. IEEE, 2010.

[18] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 371–386, 2018.

[19] Clément Laroche, Hélène Papadopoulos, Matthieu Kowalski, and Gaël Richard. Drum extraction in single channel audio signals using multi-layer non negative matrix factor deconvolution. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 46–50. IEEE, 2017.

[20] Henry Lindsay-Smith, Skot McDonald, and Mark Sandler. Drumkit transcription via convolutive nmf. In *International Conference on Digital Audio Effects (DAFx), York, UK*, 2012.

[21] Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623, 2016.

[22] Brian McFee, Matt McVicar, Stefan Balke, Vincent Lostanlen, Carl Thomé, Colin Raffel, Dana Lee, Kyungyun Lee, Oriol Nieto, Frank Zalkow, Dan Ellis, Eric Battenberg, Ryuichi Yamamoto, Josh Moore, Ziyao Wei, Rachel Bittner, Keunwoo Choi, nullmightybofo, Pius Friesch, Fabian-Robert Stöter, Thassilo, Matt Vollrath, Siddhartha Kumar Golu, nehz, Simon Waloschek, Seth, Rimvydas Naktinis, Douglas Repetto, Curtis "Fjord" Hawthorne, and CJ Carr. librosa/librosa: 0.6.3, February 2019.

[23] Brian CJ Moore. *An introduction to the psychology of hearing*. Brill, 2012.

[24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Advances in neural information processing systems 2017 Workshop*, 2017.

[25] Jouni Paulus and Anssi Klapuri. Drum sound detection in polyphonic music with hidden markov models. *EURASIP Journal on Audio, Speech, and Music Processing*, 2009:14, 2009.

[26] Jouni Paulus and Tuomas Virtanen. Drum transcription with non-negative spectrogram factorisation. In *Signal Processing Conference, 2005 13th European*, pages 1–4. IEEE, 2005.

[27] Graham E Poliner and Daniel PW Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*, page 048317, 2006.

[28] Colin Raffel, Brian Mcfee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel P. W. Ellis, C Colin Raffel, and Eric J. Humphrey. mir_eval: a transparent implementation of common mir metrics. In *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*, 2014.

[29] Axel Roebel, Jordi Pons, Marco Liuni, and Mathieu Lagrangey. On automatic drum transcription using non-negative matrix deconvolution and itakura saito divergence. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 414–418. IEEE, 2015.

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[31] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):927–939, 2016.

[32] Malcolm Slaney. Auditory toolbox. *Interval Research Corporation, Tech. Rep*, 10(1998), 1998.

[33] Carl Southall, Ryan Stables, and Jason Hockman. Automatic drum transcription using bi-directional recurrent neural networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 591–597, 2016.

[34] Carl Southall, Ryan Stables, and Jason Hockman. 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)*, pages 606–612, 2017.

[35] Carl Southall, Ryan Stables, and Jason 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.

[36] Carl Southall, Chih-Wei Wu, Alexander Lerch, and Jason 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.

[37] Richard Vogl, Matthias Dorfer, and Peter Knees. Recurrent neural networks for drum transcription. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 730–736, 2016.

[38] Richard Vogl, Matthias Dorfer, and Peter Knees. Drum transcription from polyphonic music with recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 201–205. IEEE, 2017.

[39] Richard Vogl, Gerhard Widmer, and Peter Knees. Towards multi-instrument drum transcription. *Proc. of the 21st Int. Conference on Digital Audio Effects (DAFx-18). Aveiro, Portugal*, 2018.

[40] Chih-Wei Wu, Christian Dittmar, Carl Southall, Richard Vogl, Gerhard Widmer, Jason Hockman, Meinard Muller, and Alexander Lerch. A review of automatic drum transcription. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 26(9):1457–1483, 2018.

[41] Chih-Wei Wu and Alexander Lerch. Drum transcription using partially fixed non-negative matrix factorization with template adaptation. *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 257–263, 2015.

[42] Chih-Wei Wu and Alexander Lerch. Automatic drum transcription using the student-teacher learning paradigm with unlabeled music data. In *Proc. Int. Soc. Music Inf. Retrieval Conf.*, pages 613–620, 2017.

[43] Chih-Wei Wu and Alexander Lerch. From labeled to unlabeled data–on the data challenge in automatic drum transcription. *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[44] Kazuyoshi Yoshii and Masataka Goto. Unsupervised music understanding based on nonparametric bayesian models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5353–5356, 2012.