INCREASING DRUM TRANSCRIPTION VOCABULARY USING DATA SYNTHESIS

Mark Cartwright

Music and Audio Research Lab
New York University
New York, New York
mark.cartwright@nyu.edu

Juan Pablo Bello

Music and Audio Research Lab
New York University
New York, New York
jpbello@nyu.edu

ABSTRACT

Current datasets for automatic drum transcription (ADT) are small and limited due to the tedious task of annotating onset events. While some of these datasets contain large vocabularies of percussive instrument classes (e.g. ~20 classes), many of these classes occur very infrequently in the data. This paucity of data makes it difficult to train models that support such large vocabularies. Therefore, data-driven drum transcription models often focus on a small number of percussive instrument classes (e.g. 3 classes). In this paper, we propose to support large-vocabulary drum transcription by generating a large synthetic dataset (210,000 eight second examples) of audio examples for which we have ground-truth transcriptions. Using this synthetic dataset along with existing drum transcription datasets, we train convolutional-recurrent neural networks (CRNNs) in a multi-task framework to support large-vocabulary ADT. We find that training on both the synthetic and real music drum transcription datasets together improves performance on not only large-vocabulary ADT, but also beat / downbeat detection small-vocabulary ADT.

1. INTRODUCTION

Automatic Drum Transcription (ADT) is the task of creating a symbolic score of the percussion instrument events within an audio recording of a musical piece. It is a subtask within Automatic Music Transcription (AMT), which aims to create a symbolic score of all the events within a musical piece. With accurate AMT, tens of millions of musical recordings could be indexed, compared, recommended, mined, and studied at scale using familiar musical concepts like pitch, harmony, rhythm, meter, and tempo. Accurate ADT could also aid in the development of generative rhythm models, descriptive models of rhythmic style, and intelligent digital audio effects that are informed by transcription and style.

ADT researchers typically simplify this problem by focusing solely on detecting the onset time and instrument class of all the notes sounded by percussion instruments in the signal. And very often, they simplify it even further by limiting the problem to only the notes sounded by the bass drum (BD), snare drum (SD), and hi-hat (HH) [1, 2, 3, 4, 5, 6]—a limit that greatly decreasing the utility of these systems. This is due to the tedious and time consuming nature of annotating recordings, which results in ADT datasets that are small and limited, often consisting of just a couple of hours of audio [1, 7, 8, 9]. In addition, while some datasets have annotations of a large number of percussion instrument classes [8, 7], others are limited to the BD, SD, and HH classes [9, 1], and datasets that do have larger vocabularies have minimal examples of these extended classes. The recent successful ADT algorithms have used deep learning architectures incorporating forms of recurrent neural networks [5, 4, 1, 2]. While powerful, these models require training on many audio examples in order to generalize well, making it even more difficult to expand the vocabulary of ADT.

Our goal is to learn ADT models that support a large vocabulary of percussion sounds. To address this, we generated a synthetic dataset that is 126 times larger than four of the most popular drum transcription datasets combined. We constructed this dataset using a large collection of MIDI drum loops, a large collection of drum hit recordings, and non-rhythmic harmonic backgrounds. We then trained and evaluated a multi-task convolutional-recurrent neural network (CRNN) drum transcription model using both the synthetic data and existing real music datasets. However, there is a risk in training with synthetic data—it may not be reflective of the same distribution as "real music", and therefore it may not generalize to it. Despite this, our intuition behind synthesizing and training with such a dataset is that it would expose the model to a wide variety of plausible percussion timbres and rhythmic variations allowing it to generalize to more unseen data.

In this work, we investigate the utility of synthetic data for ADT. Furthermore, to make full use of the real music datasets on the ADT task, our model follows a multi-task learning paradigm. However, both data synthesis and multi-task learning are methods to mitigate the problem of data paucity. Therefore, we also investigate the utility of multi-task learning in ADT when used in combination with synthetic training data.

2. RELATED WORK

As in many domains, there has been a recent shift to solving automatic drum transcription (ADT) with deep learning [1, 2, 5, 4]. Earlier approaches to ADT often fell into one of three categories defined by Gillet and Richard: *segment and classify*, *match and adapt*, *separate and detect* [6]. These approaches often combined multiple machine learning techniques such as support vector machines (SVM), hidden markov models (HMM), and non-negative matrix factorization (NMF) [10, 11]. While these models could perform well on solo drums [11], they often failed in the presence of polyphonic music [12]. Recent approaches that use deep learning incorporate variants of recurrent neural networks (RNNs) and are more robust than their predecessors in the presence of polyphonic music [1, 2, 4, 5]. However, deep architectures require many audio examples to generalize, and therefore due to the limited amount of annotated data, these recent approaches have limited their vocabulary of drum voices to the commonly occurring *bass drum* (BD), *snare drum* (SD), and *hi-hat* (HH). Recently, Wu and Lerch proposed to address data paucity in ADT with a student-teacher learning paradigm that utilizes unlabeled audio data. However, they still limited their work to BD, SD, and HH.

The field of computer vision also has many tasks which do not have enough annotated data to adequately train deep learning models. Some researchers have addressed this problem by generating synthetic datasets with 3D-modeling [13, 14]. Ros et al [13] addressed the problem of urban scene segmentation by generating images from synthetic worlds using a game development platform. They trained their models on a combination of real and synthetic data. Peng et al [14] addressed the problem of object detection using crowdsourced 3D CAD models to generate images. By using synthetic data, they were able to explicitly teach the model to learn invariances to specific transformations.

In machine listening, several researchers have used synthetic data to train models, including for percussion informatics tasks. Van Steelant et al [15] synthesized data using MIDI files and drum sample libraries for the percussive sound classification task. Helen and Virtanen [10] similarly synthesized data for the drum source separation task. Thompson et al [16] synthesized drum patterns for tackling the ADT task using a bar-level classification approach with mel-frequeny cepstral coefficients (MFCCs) and an SVM. While their method used a vocabulary of 6 percussion voices, it had difficulty detecting these voices in the presence of equally mixed polyphonic accompaniment (f-measure 0.48).

Recently, models incorporating deep multi-task learning [17] have achieved state-of-the-art performance on multiple music information retrieval tasks [1, 18]. McFee and Bello [18] used a structured representation of chord qualities along with a multi-task model for large-vocabulary chord recognition. Vogl et al [1] jointly trained a multi-task CRNN for ADT and beat tracking. While they obtained state-of-the-art results, they found minimal improvement jointly training the beat detection task, but they did see improvement when incorporating oracle beat features. Since they had a minimal amount of data with annotated beats, increasing the amount of training data may be helpful in this scenario. In this work, we build upon the results of Vogl et al and use a similar CRNN model for multi-task learning, but we train our model with an abundance of synthetic data.

3. METHODS

3.1. Task Definition

In this work, we define "small-vocabulary" drum transcription as the task of transcribing the onsets of 3 percussion voices: *bass drum* (BD), *snare drum* (SD), and *hi-hat* (HH). We define "large-vocabulary" drum transcription as the task of transcribing the onsets of the following 14 percussion voices which are commonly found in drum sample libraries: *bass drum*, *snare*, *snare (rim)*, *low tom*, *mid tom*, *high tom*, *open hi-hat*, *closed hi-hat*, *ride cymbal*, *crash cymbal*, *conga*, *hand clap*, *clave*, and *bell*.

3.2. Drum Transcription Datasets

To combat the problem of data paucity in large-vocabulary ADT, we use a combination of existing real music datasets (3.69 hours of data) along with a new synthetic dataset of 210k eight-second audio examples (467 hours). The tasks for which these datasets have annotations varies. Some have 3-voice drum annotations and others have annotations for more voices that must be mapped to our 14-voice task. Some of the datasets also have beat annotations. See Table 1 for details about the datasets.

Table 1: *Summary information on the datasets used in this work.*

	RBMA [1]	IDMT SMT [8]	ENST [9]	MDB [7]	SDDS (3.2.1)
Hours	1.67	0.51	1.28	0.23	467
Solo Drums	No	No	Yes	No	No
Onsets	24k	9k	25k	8k	14853k
14-voice	No	No	Yes	Yes	Yes
3-voice	Yes	Yes	Yes	Yes	Yes
Beat	Yes	No	No	No	Yes

3.2.1. Synthetic Drum Dataset (SDDS)

To generate synthetic data, we rendered 60k audio examples from a collection of MIDI drum loops using randomly selected drum samples from a sample library. These examples were then augmented by adding harmonic background noise, stochastic noise, and small pitch shifts, bringing the total number of audio examples to 210k.

The examples were constructed in the following manner. From a release of eight sample libraries[1], we collected all of the one-shot drum samples that were labeled as *bass drum*, *snare*, *snare (rim)*, *tom*, *open hi-hat*, *closed hi-hat*, *ride cymbal*, *crash cymbal*, *conga*, *hand clap*, *clave*, and *bell*. These samples were a mixture of both electronic- and acoustic-sounding drums. We did not have consistent specific tom labels, so we split the tom recordings based on the sum of their median pitch [19] rank and median spectral centroid rank into *low toms* (0–25 percentile), *mid toms* (35–65 percentile), and *high toms* (75–100 percentile). The largest, most diverse of the sample libraries[2] was set aside as the test/validate set, and the remaining libraries were used for the train set. The exception to this statement is the ride cymbals—the test set did not have any recordings labeled as ride cymbals. To compensate, we divided the ride cymbals between the two sets. All of the drum hit recordings were resampled to 44.1 kHz and peak RMS normalized. The pitch, spectral centroid, and RMS energy were computed on frames of size 1024 with 50% overlap. This overall process resulted in 3758 recordings in the train set and 2053 recordings in the test/validate set.

Next, 60k (50k train / 5k validate / 5k test) MIDI drum loops were sampled from the freely available Drum Percussion Midi Archive (800k)[3]—an archive of 800k MIDI drum loops scraped from public web sites, much like how the content of the Lakh MIDI dataset was acquired [20]. We sampled these loops by randomly selecting one measure from a random MIDI file in the collection with the constraints that the measure was less than 3 seconds long and contained at least 3 percussion instruments. The measure was then looped and processed to be 8 seconds in duration with the first downbeat occurring at a random offset from the beginning of the file. Each MIDI loop was then rendered to audio by creating a separate "track" for each percussion voice, randomly selecting a drum sample for each percussion voice, and placing them in the track "monophonically" (i.e. drum hits for the same voice never overlapped) at each note onset time. The training set loops were rendered with the training set of drum samples, and both the validate and test loops were rendered with the testing/validation set of

[1] http://musician.givegetwin.com/drum-heaven/

[2] *Wave Alchemy - Drum Tools 01 Deluxe*

[3] https://goo.gl/GhV7pc

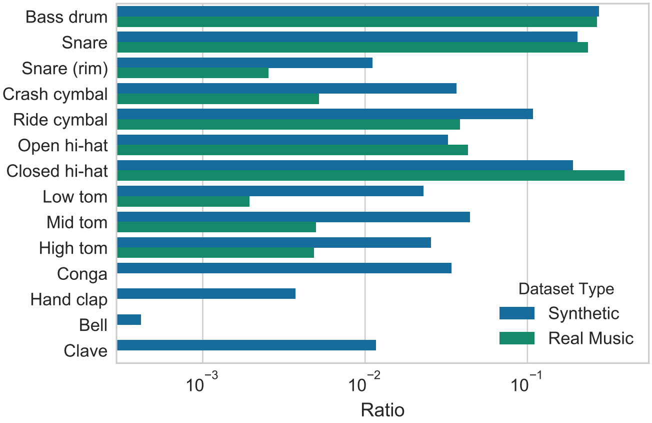

Figure 1: *Distribution of drum onsets in both the Real music (i.e., RBMA13, ENST-Drums, IDMT-SMT, MDB-Drums) and Synth (i.e., SDDS) dataset groups. Note the log-scale of the x-axis.*

drum samples. When rendering the loops, we scaled the amplitude of each drum sample by MIDI velocity as recommended in [21]:

$$r = 10^{r_{\text{dB}}/20} \tag{1}$$

$$b = \frac{v_{\max}}{(v_{\max}-1)\sqrt{r}} - \frac{1}{v_{\max}-1} \tag{2}$$

$$m = \frac{1-b}{v_{\max}} \tag{3}$$

$$a = \begin{cases} (mv+b)^2 & \text{if } v > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where v is the note's MIDI velocity in the range $[0, v_{\max}]$ with $v_{\max} = 127$, r_{dB} is the output amplitude range in dB (set to 60 dB), and a is the resulting amplitude. All percussion voice tracks were mixed together with equal weights.

To generate non-rhythmic background accompaniment without discernible onsets, we selected 20 recordings (10 train, 10 test) containing harmonic instruments from MedleyDB [22]. Each recording was "smeared" in time by processing it with the Pysox [23] reverberator both forward and backward with randomly selected reverberation settings in a range to produce long-tail reverberation. This was repeated 12 times for each recording, each time pitched up an additional semitone, to produce 120 training backgrounds and 120 test/validate backgrounds. All files were trimmed to 30 seconds.

Using the MUDA data augmentation library [24], we then augmented the rendered training set of drum loops by a factor of four by varying both the background and the pitch, each with two variants. The goal of augmentation was to help the model learn invariances to different backgrounds and small changes in pitch. For each file, we selected two random 8 s background segments with random mixing coefficients in the range $[0.01, 0.5]$, and we selected two random semitone pitch shifts sampled from $N(0, 0.05)$. Lastly, to add robustness to noise, we also added white noise with a random mixing coefficient in the range $[0.01, 0.1]$. We repeated this process for the training and validation sets, but we only used one variant of each augmentation step, which resulted in only one processed recording per drum loop. The unprocessed drum loops were discarded. The augmentation process increased the size of the training set from 50k to 200k, while both the testing and validation sets remained at 5k. We will refer to this dataset as the Synthetic Drum Dataset (SDDS).

3.2.2. Real Music Datasets

We also trained and evaluated on four standard drum transcription datasets: RBMA13 [1], IDMT-SMT [8], ENST-Drums [9], and MDB-Drums [7].

RBMA13 [1] is a dataset of 27 fully-produced music tracks in the genres of electronic dance music (EDM), singer-songwriter, and fusion-jazz. It contains both annotated drum onsets and beat / downbeats. While there are several classes of percussion sounds in the recordings, only the BD, SD, and HH are annotated. We used the dataset's 3 predefined cross-validation splits.

IDMT-SMT [8] is a dataset of solo drum recordings consisting of BD, SD, and HH sounds from acoustic drum kits (10 different kits), drum synthesizers, and drum sample libraries. The dataset contains both recordings of isolated, single drum sounds and also recordings of rhythmic patterns using multiple drums. In this work, we only used the recordings of the rhythmic patterns. We randomly split the dataset into 3 cross-validation splits.

ENST-Drums [9] is a dataset of recordings from 3 drummers with different drum kits. It contains drum onset annotations for 20 different classes of percussion sounds. Of these 20 classes, we mapped 11 down to 10 of the classes in our vocabulary. The 9 remaining classes that were out of our vocabulary (e.g., *brush sweep*, *Chinese ride cymbal*) were ignored. While the dataset also contains many solo drum recordings, we only used the subset of recordings with accompaniment. The accompaniment and drums were summed together to create polyphonic mixtures. We used the splits of the drummers for our 3 cross-validation splits.

MDB-Drums [7] is a set of 23 fully-produced music tracks from the MedleyDB dataset [22]. It contains 6 classes of percussion sounds (*bass drum*, *snare drum*, *hi-hat*, *tom*, *cymbal*, *other*) with 21 subclasses (e.g., *snare drum: drag*, *open hi-hat*). Of these 21 subclasses, we mapped 17 down to 9 of the classes in our vocabulary and ignored the remaining 4 classes. We used the dataset's three predefined cross-validation splits.

3.3. Multi-Task Model

In this work, we used a convolutional-recurrent neural network (CRNN) model, very similar to the current state-of-the-art ADT model published in [1]. The model is constructed of 4 blocks of components as described in Table 2 and visualized in Figure 2.

We want to fully utilize the ADT and beat annotations in the real music datasets, but some datasets have 3-voice annotations and others have a larger number of voices that we reduced to our 14-voice vocabulary. Rather than trying to map the 3-voice annotations *up* to 14-voices and dealing with class assignment ambiguity, we instead treat it as two separate tasks—3-voice transcription and 14-voice transcription—and map the 14-voice annotations *down* to 3-voices (i.e., grouping the specific snare and hi-hat annotations, keeping bass drum as is, and ignoring the rest). To support all of these tasks, we designed our model as a multi-task with different outputs and losses for three tasks: 14-voice drum transcription, 3-voice drum transcription, and beat / downbeat detection.

The model receives two feature types as inputs in the **input block**. The first is the log-magnitude, log-frequency short-time Fourier transform (Logf-STFT). To compute this feature, we first

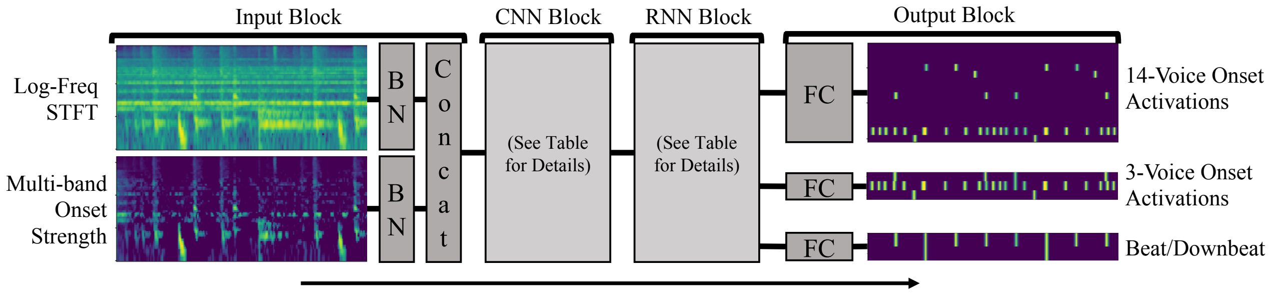

Figure 2: *High-level architecture of multi-task convolutional-recurrent neural network model. See Table 2 for details.*

Table 2: *Detailed architecture of multi-task convolutional-recurrent neural network model. Components that occur in parallel at the same model depth are presented in the same row. The parentheses on the right-hand side of each cell indicate the output size for that component.*

In Block	**Logf-STFT** (799, 64)	**Logf-Onset** (799, 64)	
	BatchNorm (799, 64)	BatchNorm (799, 64)	
	Stack (799, 64, 2)		
CNN Block	32 (3x3) Conv (799, 64, 32)		
	32 (3x3) Conv (799, 64, 32)		
	BatchNorm (799, 64, 32)		
	ReLU (799, 64, 32)		
	30% Dropout (799, 64, 32)		
	64 (3x3) Conv (799, 64, 64)		
	64 (3x3) Conv (799, 64, 64)		
	BatchNorm (799, 64, 64)		
	ReLU (799, 64, 64)		
	30% Dropout (799, 64, 64)		
	64 (1x64) Conv (799, 1, 64		
	BatchNorm (799, 1, 64)		
	ReLU (799, 1, 64)		
RNN Block	(-6:+6) Context Windowing (799, 832)		
	64 BLSTM (799, 128)		
	64 BLSTM (799, 128)		
	64 BLSTM (799, 128)		
Out Block	14 FC (799, 14)	3 FC (799, 3)	2 FC (799,2)
	14-voice	**3-voice**	**Beats**

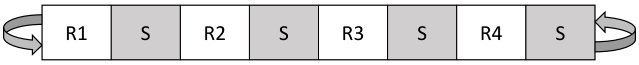

Figure 3: *Round robin sampling for training. S is the synthetic dataset and RN are the real datasets: 1:RBM-13, 2:IMDT-SMT, 3:ENST, 4:MDB*

resample the audio to 22050 Hz and peak normalize it. We then compute the linear-frequency STFT on 1024-sample frames with a ~10 ms (221 sample) hop size. The magnitudes of the linearly-spaced frequency bins are then grouped into log-spaced bins using triangular frequency-domain filters—8 octaves of 8 bins per octave, starting at 40 Hz (i.e, 64 bins). We then log-scale these features. The second input is a multi-band onset signal computed from the Logf-STFT features before we log-scale magnitude. For each frequency bin, we compute the difference function between the current frame and the mean of the previous 22 frames. We then half-wave rectify and log-scale the signal. In the model, these features are batch-normalized [25] and concatenated on top of each other, creating a 2-channel input to the CNN block.

The purpose of the **CNN block** is to model the timbral characteristics of drum onsets. Described in detail in Table 2, this block consists of 3 stacks of convolutions, batch-normalization, rectified linear unit (ReLU) activations, and dropout [26] ($rate = 0.3$) layers. Each of the first two stacks uses 2 layers of 3×3 convolution filters, padded so the output is the same size as the input. The final stack uses one layer of 1×64 convolution filters (without padding) and does not include dropout. This block maintains the temporal resolution of the input.

The purpose of the **RNN block** is to model the temporal dynamics of drum onsets. This block consists of a stack of three bidirectional long short-term memory (BLSTM) [27] components. To more explicitly model the context, the input to the first BLSTM is padded and each temporal frame is concatenated with the 6 previous and 6 subsequent frames.

Each output frame of the RNN block is fed into three task-specific fully-connected (FC) layers in the **output block**: 14-voice transcription, 3-voice transcription, and beat / downbeat detection. In multi-task learning, this architecture is described as “hard parameter sharing” in which tasks share parameters for several layers followed by task-specific layers [17]. The model outputs and training data are encoded as multi-label binary activations with the same temporal resolution as the input (see Figure 2)—i.e, for each temporal frame, a class (i.e, percussion voice or beat / downbeat) bin is 1 if it contains an onset event, and 0 otherwise. The loss for each output is computed using binary cross-entropy.

3.4. Training

For the experiments in this paper, our model was implemented in Keras [28] and was trained on 8 s examples. We optimized using Adam [29] with gradients clipped at 1.0, a learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $decay = 0$. We used batch sizes of 8 and an "epoch" size of 2000. We reduced the learning rate on plateau (patience=5), and we trained for a maximum of 100 epochs with early stopping set to a patience of 25 epochs.

3.4.1. Task weights

With multiple outputs and loss functions, the optimizer minimizes a weighted combination of the losses:

$$L = \gamma_0 L_0 + \gamma_1 L_1 + \gamma_2 L_2 \tag{5}$$

where γ_i are the loss weights. Tasks with sparse output targets can achieve a low loss by simply predicting constant output. Therefore, if losses are equally weighted, training heuristics such as early-stopping and learning rate reduction will be dominated by tasks with denser outputs. In our training data, the distribution of the 14-voice transcription task is much sparser than the others, resulting in a small loss. To remedy this, we weighted the tasks by the inverse estimated entropy of their event activity distribution:

$$p_i = \frac{n_{events}}{n_{classes} \times n_{timesteps}} \tag{6}$$

$$\gamma_i = (-p_i \log(p_i) - (1 - p_i)\log(1 - p_i))^{-1} \tag{7}$$

We estimated γ_i from the empirical distribution of events in the training set for each task, and weighted the task losses 0.53, 0.16, and 0.31 respectively for 14-voice transcription, 3-voice transcription, and beat / downbeat detection.

In addition, a task loss was masked for training examples without annotations for that particular task. This enables us to train a multi-task model with incomplete data.

3.4.2. Sampling

To have a numerically stable loss when training with incomplete data, the data has to be sampled such that all tasks are represented in each batch. Therefore, a simple random sampling procedure is not adequate. To accommodate this constraint, we utilize a round-robin sampling procedure using Pescador [30] as shown in Figure 3. This sampling procedure cycles through the real music datasets. Each time it samples from a real music dataset, it randomly selects an 8 s time interval from the dataset. Each time it samples from the synthetic music dataset, it randomly selects an 8 s example from the dataset. When training with both real and synthetic data, every other sample is from the synthetic dataset. This helps prevent overfitting due to the large quantity of synthetic data. The round-robin sampling procedure ensure that all active datasets will be present in a single batch, and therefore all tasks will be present as well. However, since the datasets vary in size, examples in smaller datasets will be presented more frequently to the model. This sampling procedure was used in both training and the calculation of the validation loss. However, when testing, we did not use this procedure—outputs were predicted for each example once, using the entire duration of the signal, i.e., the full duration of a music track rather than a 8 s time interval.

3.5. Experiments

To evaluate the effectiveness of our synthetic training data, we trained our model with three variations of training data:

1. **Real**: the real music dataset group (RBMA13, IDMT-SMT, ENST, MDB-Drums)
2. **Synth**: the synthetic dataset (SDDS)
3. **Real + Synth**: and the combination of both the real music dataset group and the synthetic dataset.

To balance the task in each batch, we used the round robin sampling scheme described in Section 3.4.2. When training with the real music datasets, we used the cross-validation splits as noted in Section 3.2.2 for training. To do so, we grouped the corresponding splits of the datasets together, e.g. the first splits from each dataset were grouped together when determining training and validation sets. For validation and testing, we further partitioned the data, using 25% of each split for validation and 75% for testing. In contrast, the synthetic dataset had one large training set and smaller validation and test sets. For consistency with the *Real* and *Real + Synth* variants, training with the *Synth* variant was also repeated three times with different random samples.

Furthermore, to investigate if the addition of large amounts of synthetic data could benefit from a larger model, we also varied the capacity of the model. We trained a "small" model as described in Table 2, and we also trained a "large" model. In the large model, convolution layers that originally had 32 filters were increased to 128 filters, and the number of units in the BLSTM components was increased from 64 to 256.

As noted earlier, we trained our models with a multi-task learning paradigm to make use of the *Real* datasets that only have small-vocabulary annotations. However, both data synthesis and multi-task learning can be viewed as methods to mitigate the problem of data paucity. To investigate how multi-task learning affects the ADT task in combination with data synthesis, we also trained single task models for comparison. These models used the small-capacity configuration and were only trained on *Real + Synth* data. We again evaluated them on *Real* and *Synth* data separately.

For each variation, we trained three models, one for each validation split. For each split, we selected the model with the lowest validation loss for evaluation. To evaluate our model outputs, we estimated the locations of onsets and beats from the output activations using the peak picking method described in [31], in which an output activation sample is selected as a peak if it meets the following criteria:

1. $x(n) = max(x(n - m_{pre}), \ldots, x(n + m_{post}))$
2. $x(n) \geq mean\,(x(n - o_{pre}), \ldots, x(n + o_{post})) + \delta$
3. $n - n_{lastOnset} > w$

where $x(n)$ is an output activation, n is the index of the current sample, $n_{lastOnset}$ is the index of the last identified onset, δ is a threshold parameter, w is the minimum number of samples between onsets, and m_{pre}, m_{post}, o_{pre}, o_{post} are sample offset values that define the window over which the max and $mean$ functions are computed. We tuned peak parameters for each model and task combination using a randomized search with 500 iterations scored on the validation set.

The resulting models were separately evaluated on both the test set of their corresponding split's real music dataset group and of the synthetic dataset. We used a 50 ms evaluation window in all experiments.

4. RESULTS

Table 3 presents the results of the experiments described in Section 3.5. Each value in the table is a mean metric (f-measure, precision, and recall) averaged over the test sets for the three cross-validation splits. For the small capacity variants evaluated on the

Table 3: *Mean model performance (over CV splits) for all three tasks evaluated on both real music and synthetic datasets while varying the distribution of training data, model capacity, and tasks.* ***F****: f-measure,* ***P****: precision,* ***R****: recall. Bold items indicate best performance per column for each group of training variants. *Asterisk indicates best task performance on Real data across all variants.*

Learning Paradigm	Capacity	Eval. Data	Train. Data	14-voice Trans.			3-voice Trans.			Beat Detect.		
				F	P	R	F	P	R	F	P	R
Multi-task	Small	Real	Real	0.58	0.55	0.61	0.69	0.64	0.74	0.62	0.68	0.58
			Synth	0.43	0.45	0.42	0.61	0.57	0.67	0.61	0.59	0.64
			Real + Synth	**0.68**	**0.67**	***0.70***	***0.77***	***0.77***	**0.77**	**0.74**	**0.74**	**0.74**
		Synth	Real	0.47	0.51	0.43	0.61	0.60	0.62	0.58	0.56	0.60
			Synth	**0.74**	**0.76**	0.72	**0.85**	**0.84**	0.86	**0.70**	0.68	**0.75**
			Real + Synth	0.70	0.64	**0.77**	0.84	0.81	**0.87**	0.69	**0.72**	0.68
	Large	Real	Real	0.63	0.63	0.63	0.72	0.69	0.76	0.57	0.59	0.55
			Synth	0.44	0.42	0.45	0.62	0.58	0.68	0.63	0.58	0.69
			Real + Synth	**0.70**	**0.73**	**0.68**	**0.76**	**0.74**	**0.78**	***0.75***	***0.75***	***0.75***
		Synth	Real	0.48	0.51	0.46	0.63	0.64	0.63	0.60	0.55	0.65
			Synth	**0.77**	**0.78**	**0.77**	**0.87**	**0.86**	**0.87**	**0.75**	0.68	**0.84**
			Real + Synth	0.72	0.71	0.74	0.83	0.81	**0.87**	0.69	**0.72**	0.69
Single-task	Small	Real	Real + Synth	*0.72*	*0.74*	0.69	-	-	-	-	-	-
		Synth	Real + Synth	0.69	0.68	0.71	-	-	-	-	-	-
		Real	Real + Synth	-	-	-	*0.77*	0.74	*0.81*	-	-	-
		Synth	Real + Synth	-	-	-	0.82	0.78	0.87	-	-	-
		Real	Real + Synth	-	-	-	-	-	-	0.64	0.59	0.69
		Synth	Real + Synth	-	-	-	-	-	-	0.65	0.61	0.71

Real test set, the model trained on only *Synth* data performs worse than the model trained on only *Real* data—e.g., f-measure 0.43 vs 0.58 for 14-voice transcription. This is true across all three tasks, though the effect on beat detection is minimal. However, when models are trained on both the *Real* and *Synth* data together, we see a large improvement in performance on all three tasks—e.g., f-measure 0.68 for 14-voice transcription. This trend is present in all three tasks. This implies that both *Real* and *Synth* are useful and complementary when training ADT models. We speculate that the *Synth* data teaches the model a wider variety of percussion timbres, whereas the *Real* data teaches the model to ignore instruments not relevant to the ADT task. This conjecture seems to hold with respect to the models' performance when evaluated on the *Synth* data—the models trained on only the *Synth* perform better or equal to the models trained on both *Synth* and *Real* data.

To further understand the models' performance on the 14-voice transcription task, we also evaluated class-specific performance (see Figure 4). We find that when trained on only *Real* data, the model only predicts 3 classes with f-measure performance above 0.5—*bass drum* (0.68), *snare drum* (0.61), and *closed hi-hat* (0.62). Except for the *open hi-hat* (f-measure 0.11), the model fails to predict all other classes. When *Synth* data is added to the training set, the f-measure performance improves for the *bass drum* (0.73), *snare drum* (0.74), *closed hi-hat* (0.74), and *open hi-hat* (0.55). The model now also has non-zero f-measure for several classes on which the *Real*-trained model completely failed to predict—e.g., *crash cymbals* (0.08), *ride cymbals* (0.45), *low toms* (0.09), *mid toms* (0.05), *high and toms* (0.04). Unfortunately, the performance on all of these classes is quite low with the exception of the ride cymbals. Interestingly, the precision on the tom classes is much higher than the recall—while this may have several causes, one possible cause could be improperly tuned peak-picking parameters, indicating that we should investigate class-specific parameters. There are also several classes on which the model still completely fails—*snares (rim)*, *congas*, *hand claps*, *bells*, *claves*. However, if we revisit the class distribution of drum onsets for the *Real* data in Figure 1, we see that class performance is closely correlated to the data's class distribution. While this could be caused by high-variance performance estimates due to the rarity of these onset events in the evaluation data, if this were the case, one might expect the model to have a high performance for at least one such classes. For comparison, if we look at the model's performance when evaluated on *Synth* data, we see a significant increase in performance for all of these classes except for the *hand claps* and *bells*, which have the lowest representation in the *Synth* data. Therefore while the *Synth* training data may expose the model to a wider variety of timbres, we still need to expose the model to classes of interest in a musical context with other instruments—with its current architecture, the model does not seem to generalize this ability across classes.

From Table 3, we also see that the large capacity models have roughly the same performance as the small models (sometimes slightly worse, sometimes slightly better) when evaluated on *Real* data. Whereas, the large capacity models trained and evaluated on *Synth* data see a small but consistent boost in performance from the higher capacity. Therefore, while increasing model capacity may help when training and evaluating on an abundance of synthetic data from the same distribution, it does not seem to improve model performance when evaluated on real music data.

Lastly, the models trained separately on the 14-voice and 3-voice transcription tasks typically performed very similarly to their multi-task counterparts (f-measure ± 0.02) with the single-task 14-voice model performing a bit better on the *Real* data (f-measure 0.72 vs 0.68). In contrast, the single-task beat detection model

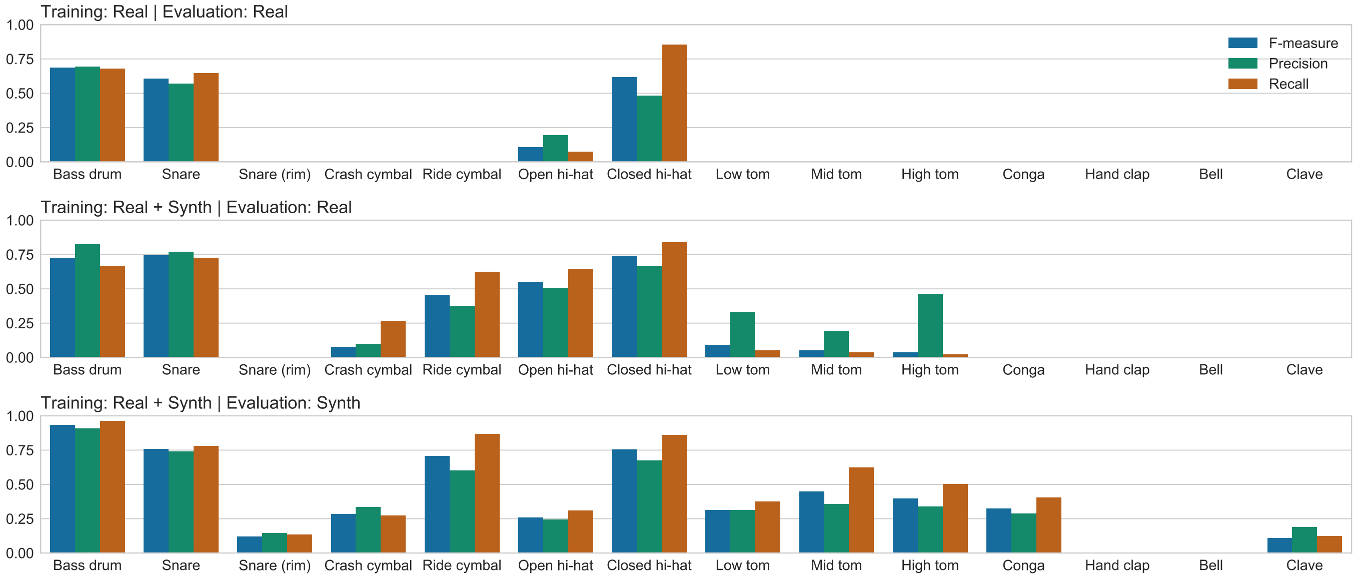

Figure 4: *Mean model performance (over CV splits) for 14-voice transcription performance broken down by class.* ***Top****: Model trained and evaluated on the real music datasets (RBM-13, IDMT-SMT, ENST, MDB).* ***Middle:*** *Model trained on both the real music dataset group and synthetic dataset (SDDS), and evaluated on the real music datasets.* ***Bottom:*** *Trained on both the real music datasets and synthetic dataset, and evaluated on the synthetic dataset.*

performed worse than its multi-task counterpart when evaluated on the *Real* data (f-measure 0.64 vs 0.74). Furthermore, we had the least amount of annotated *Real* data for the beat detection task (only RBMA-13 has beat / downbeat annotations). Thus, while multi-task learning in conjunction with synthetic data does seem to considerably aid on some tasks (e.g., beat / downbeat detection), this is not true for all tasks, and for 14-voice drum transcription it seems to actually hinder performance. These results seem to indicate that the benefit of synthetic data possibly overwhelms the benefit of multi-task learning for the ADT task.

5. CONCLUSION

In this work, we addressed the problem of data paucity for large-vocabulary automatic drum transcription (ADT) by generating a large synthetic dataset. We found that training with synthetic data can improve performance not only on ADT but also on beat detection. Improvements were observed on both 3-voice and 14-voice transcription tasks. On the 14-voice task, training with synthetic data increased performance for five classes on which the model without synthetic data training failed completely. Unfortunately, there is still a lot of room for improvement for 14-voice drum transcription. For synthetic data to help, it needs to be utilized in conjunction with real music data. In fact, it seems that it may need at least some minimum amount of annotated real music data in each class of interest, a problem that is exacerbated by the class imbalance in real music ADT training data. In our experiments we did not investigate what this minimum threshold is. However, these results imply that determining this minimum and focusing efforts to annotate up to this minimum for classes of interest could be a reasonable next step for improving large-vocabulary drum transcription. Another reasonable next step could be to resynthesize new percussion tracks with a variety of timbres for annotated datasets that have separate accompaniment and percussion tracks (e.g,, MDB-Drums). We also investigated the benefits of multi-task learning for ADT in combination with training on synthetic data. In our experiments, we trained both single task and multi-task models, and we found that multi-task learning potentially harmed the ADT task, but greatly benefited our auxiliary beat / downbeat detection task.

While the distribution of the full SDDS dataset is prohibitive due to its size, we have made both a small portion of the data available for download along with the trained multi-task and single-task models with highest performance on the *Real* datasets. Furthermore, we have also released a Python package to generate a similar dataset given collections of drum samples and MIDI files.[4]

In summary, data synthesis is a promising approach to combat the problem of data paucity in ADT and to increase the vocabulary size of ADT systems, but additional work is needed to investigate how to further improve performance on rare percussion classes. In addition, multi-task learning can also be a powerful tool to take advantage of limited training data, but its benefit is not consistent across tasks when combined with synthetic data. In future work, we hope to further investigate when training on auxiliary tasks is beneficial in music information retrieval.

6. REFERENCES

[1] Richard Vogl, Matthias Dorfer, Gerhard Widmer, and Peter Knees, “Drum transcription via joint beat and drum modeling using convolutional recurrent neural networks,” in *Proc. of the Int’l Society for Music Information Retrieval Conf.*, 2017, pp. 150–157.

[4] `https://github.com/mcartwright/dafx2018_adt`

[2] Richard Vogl, Matthias Dorfer, and Peter Knees, "Drum transcription from polyphonic music with recurrent neural networks," in *Proc. of the Int'l Conf. on Acoustics, Speech and Signal Processing*. 2017, pp. 201–205, IEEE.

[3] Chih-Wei Wu and Alexander Lerch, "Automatic drum transcription using the student-teacher learning paradigm with unlabeled music data," in *Proc. of the Int'l Society for Music Information Retrieval Conf.*, 2017, pp. 613–620.

[4] Carl Southall, Ryan Stables, and Jason Hockman, "Automatic drum transcription for polyphonic recordings using soft attention mechanisms and convolutional neural networks," in *Proc. of the Int'l Society for Music Information Retrieval Conf.*, 2017, pp. 606–612.

[5] Carl Southall, Ryan Stables, and Jason Hockman, "Automatic drum transcription using bi-directional recurrent neural networks," in *Proc. of the Int'l Society for Music Information Retrieval Conf.*, 2016, pp. 591–597.

[6] Olivier Gillet and Gaël Richard, "Transcription and separation of drum signals from polyphonic music," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 3, pp. 529–540, 2008.

[7] Carl Southall, Chih-Wei Wu, Alexander Lerch, and Jason Hockman, "Mdb drums–an annotated subset of medleydb for automatic drum transcription," in *Int'l Society for Music Information Retrieval Conf. Late-breaking and Demo Papers*, 2017.

[8] Christian Dittmar and Daniel Gärtner, "Real-time transcription and separation of drum recordings based on nmf decomposition," in *DAFx*, 2014, pp. 187–194.

[9] Olivier Gillet and Gaël Richard, "Enst-drums: an extensive audio-visual database for drum signals processing," in *ISMIR*, 2006, pp. 156–159.

[10] Marko Helen and Tuomas Virtanen, "Separation of drums from polyphonic music using non-negative matrix factorization and support vector machine," in *Proc. of the European Signal Processing Conf.* 2005, pp. 1–4, IEEE.

[11] Olivier Gillet and Gaël Richard, "Automatic transcription of drum loops," in *Proc. of the Int'l Conf. on Acoustics, Speech, and Signal Processing*. 2004, vol. 4, pp. iv–iv, IEEE.

[12] Olivier Gillet and Gaël Richard, "Drum track transcription of polyphonic music using noise subspace projection," 2005, pp. 92–99.

[13] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2016, pp. 3234–3243.

[14] Xingchao Peng, Baochen Sun, Karim Ali, and Kate Saenko, "Learning deep object detectors from 3d models," in *Proc. of the Int'l Conf. on Computer Vision*. 2015, pp. 1278–1286, IEEE.

[15] Dirk Van Steelant, Koen Tanghe, Sven Degroeve, Bernard De Baets, Marc Leman, Jean-Pierre Martens, and T De Mulder, "Classification of percussive sounds using support vector machines," in *Proc. of the annual machine learning conference of Belgium and The Netherlands, Brussels, Belgium*. 2004, pp. 146–153, Citeseer.

[16] Lucas Thompson, Matthias Mauch, and Simon Dixon, "Drum transcription via classification of bar-level rhythmic patterns," in *Proc. of the Int'l Society for Music Information Retrieval Conf.*, 2014, pp. 187–192.

[17] Sebastian Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.

[18] Brian McFee and Juan P Bello, "Structured training for large-vocabulary chord recognition," in *Proc. of the Int'l Society for Music Information Retrieval Conf.*, 2017, pp. 188–194.

[19] Alain De Cheveigné and Hideki Kawahara, "Yin, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

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

[21] Roger B Dannenberg, "The interpretation of midi velocity," in *Proceedings of the International Computer Music Conference*, 2006, pp. 193–196.

[22] Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan P Bello, "Medleydb: A multitrack dataset for annotation-intensive mir research," in *Proc. of the Int'l Society for Music Information Retrieval Conf.*, 2014, vol. 14, pp. 155–160.

[23] Rachel Bittner, Eric Humphrey, and Juan P Bello, "Pysox: Leveraging the audio signal processing power of sox in python," in *Int'l Society for Music Information Retrieval Conf. Late-Breaking and Demo Papers*, 2016.

[24] Brian McFee, Eric J Humphrey, and Juan P Bello, "A software framework for musical data augmentation," in *Proc. of the Int'l Society for Music Information Retrieval Conf.* 2015, pp. 248–254, Citeseer.

[25] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[26] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[27] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] François Chollet, "Keras," `https://github.com/fchollet/keras`, 2015.

[29] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[30] Brian McFee, Eric Humphrey, and Christopher Jacoby, "Pescador," `https://github.com/pescadores/pescador`, 2016.

[31] Sebastian Böck, Florian Krebs, and Markus Schedl, "Evaluating the online capabilities of onset detection methods," in *ISMIR*, 2012, pp. 49–54.