

# PLAYER VS TRANSCRIBER: A GAME APPROACH TO DATA MANIPULATION FOR AUTOMATIC DRUM TRANSCRIPTION

Carl Southall, Ryan Stables and Jason Hockman

DMT Lab, Birmingham City University, Birmingham, United Kingdom

{carl.southall, ryan.stables, jason.hockman}@bcu.ac.uk

## ABSTRACT

State-of-the-art automatic drum transcription (ADT) approaches utilise deep learning methods reliant on time-consuming manual annotations and require congruence between training and testing data. When these conditions are not held, they often fail to generalise. We propose a game approach to ADT, termed player vs transcriber (PvT), in which a player model aims to reduce transcription accuracy of a transcriber model by manipulating training data in two ways. First, existing data may be augmented, allowing the transcriber to be trained using recordings with modified timbres. Second, additional individual recordings from sample libraries are included to generate rare combinations. We present three versions of the PvT model: *AugExist*, which augments pre-existing recordings; *AugAddExist*, which adds additional samples of drum hits to the *AugExist* system; and *Generate*, which generates training examples exclusively from individual drum hits from sample libraries. The three versions are evaluated alongside a state-of-the-art deep learning ADT system using two evaluation strategies. The results demonstrate that including the player network improves the ADT performance and suggests that this is due to improved generalisability. The results also indicate that although the *Generate* model achieves relatively low results, it is a viable choice when annotations are not accessible.

## 1. INTRODUCTION

Automatic music transcription (AMT) systems generate a symbolic representation of the instrumentation within an audio recording. There are multiple educational, analytical and creative fields that would benefit from fast and accurately produced music notation. Automatic drum transcription (ADT) systems form a subset of AMT systems which produce notation solely focused on drum instrumentation.

### 1.1 Background


At present, high ADT accuracies have been achieved for audio files containing either just drums or polyphonic mixtures [4, 7, 9–11, 18]. Following the comprehensive

ADT literature review in [22], current state-of-the-art ADT systems utilise either deep learning (DL) or non-negative matrix factorisation (NMF). NMF approaches perform instrument-specific onset detection through an iterative simultaneous update of basis and activation functions via factorisation of an input spectrogram. Recent NMF approaches have introduced specialised update methods (i.e., fixed, adaptive and semi-adaptive) based on the expected end-use application of the algorithm [2] or incorporated additional basis functions to capture harmonic content within polyphonic recordings [23]. Alternatively, DL approaches perform instrument-specific onset detection through supervised frame-based classification. The first DL systems to achieve high ADT performance incorporated recurrent neural networks [14, 19, 20]. More recent approaches now include convolutional neural networks [21] and soft attention mechanisms [15]. As in many fields, augmentation of data (i.e., pitch shifting) during training has aided performance [12, 20].

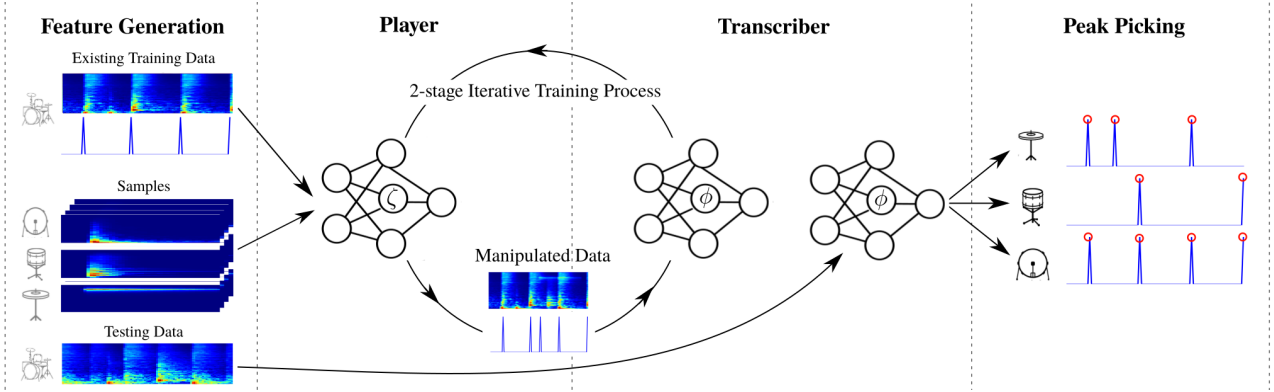
### 1.2 Motivation

Evaluations undertaken in [22] and the MIREX 2017 Drum Transcription Task<sup>1</sup> highlight that state-of-the-art ADT accuracies are achieved by supervised DL approaches. However, success of DL systems is reliant on training data achieved through a time-consuming manual annotation process [24]. Also, if there is a mismatch between training and testing data, these systems will fail to generalise [22]. We propose a game approach to ADT influenced by generative adversarial networks [5], termed *player vs transcriber* (PvT). In an attempt to undermine the accuracy of a transcriber model (i.e., an existing supervised ADT approach), a player model seeks to exploit poorly defined areas of the feature space through a manipulation of training data. This is achieved through learned data manipulation variables in the player network, which are used to define the manipulation coordinates of the transform. Additionally, the player model is able to manipulate the data depending on its content, where existing methods for augmentation typically rely on a set of global variables [8].

The remainder of this paper is structured as follows: Section 2 presents the PvT model and Section 3 provides an overview of the undertaken evaluation. The results and discussion are presented in Section 4 and the conclusions and future work are presented in Section 5.

 © Carl Southall, Ryan Stables and Jason Hockman. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Carl Southall, Ryan Stables and Jason Hockman. “Player Vs Transcriber: A Game Approach To Data Manipulation For Automatic Drum Transcription”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

<sup>1</sup> [http://www.music-ir.org/mirex/wiki/2017:Drum\\_Transcription\\_Results](http://www.music-ir.org/mirex/wiki/2017:Drum_Transcription_Results)



**Figure 1.** Overview of the player vs transcriber (PvT) game approach to automatic drum transcription. The PvT model is achieved through four stages: feature generation, player model, transcriber model and peak-picking.

## 2. METHOD

Figure 1 provides an overview of the proposed PvT system, which is achieved through four stages: Feature generation, data manipulation by a player, activation function creation by a transcriber, and framewise classification by peak picking. At the core of the system is an iterative process of data manipulation (i.e., data augmentation and sample addition) and activation function generation between the player and transcriber. Both models examine the loss functions related to the activation functions created by the transcriber. Here, the two models have diametrically opposed goals; while the player seeks to maximize the loss, the transcriber attempts to minimize the loss. Once the loss function has been optimized during training, testing data may be evaluated by the transcriber and drum events are found through peak picking.

### 2.1 Feature Generation

Input audio (16-bit .wav file sampled at 44.1kHz) is segmented into  $T$  frames using a Hanning window of  $m$  samples ( $m = 2048$ ) with a  $\frac{m}{2}$  hopsize. A logarithmic frequency representation of each of the frames is created using a similar process to [21] using the madmom Python library [1]. The magnitudes of a discrete Fourier transform are converted to a logarithmic scale (20Hz–20kHz) using twelve triangular filters per octave, resulting in a  $84 \times T$  logarithmic spectrogram  $x$  and corresponding target  $y$ .

### 2.2 Player

The aim of the player is to exploit weaknesses within the transcriber through a manipulation of the training data. This is achieved using two processes as demonstrated in Figure 2: data augmentation (Section 2.2.1), which alters the frequential content of existing data; and sample addition (Section 2.2.2), which adds recordings of individual drum hits from drum samples (Section 2.2.3) to the pre-existing training examples. To ensure that the process is *end-to-end* and that the player network can be trained using back propagation, the process must be differentiable. To this end, the entire process is designed around network defined variables  $\theta$  (Sections 2.2.4 and 2.2.5) and avoids operations such as *argmax*.

#### 2.2.1 Data Augmentation

The data augmentation stage is based on existing data augmentation approaches [8], however also aims to portray changes in instrumentation and performance techniques by manipulating the frequency content of pre-existing data. This is achieved using three network-generated variables ( $\theta^p$ ,  $\theta^n$  and  $\theta^g$ ), in which  $\theta^p$  and  $\theta^n$  are used within an pseudo-equaliser function and  $\theta^g$  as an overall gain. Time-step  $t$  of the augmented segment  $x_{aug}$  is calculated using:

$$x_{aug}^t = ReLU(x^t + (s(\theta^p)x^t v) - (s(\theta^n)x^t v))g, \quad (1)$$

where  $v$  and  $s$ , the softmax functions are used to prevent over augmentation by limiting maximum augmentation to either 1 or -1. The rectified linear unit function (ReLU) ensures non-negativity and  $g$  is determined using:

$$g = \theta_g(1 - min_g) + min_g, \quad (2)$$

where  $min_g$  is a hyperparameter that determines the minimum possible gain.

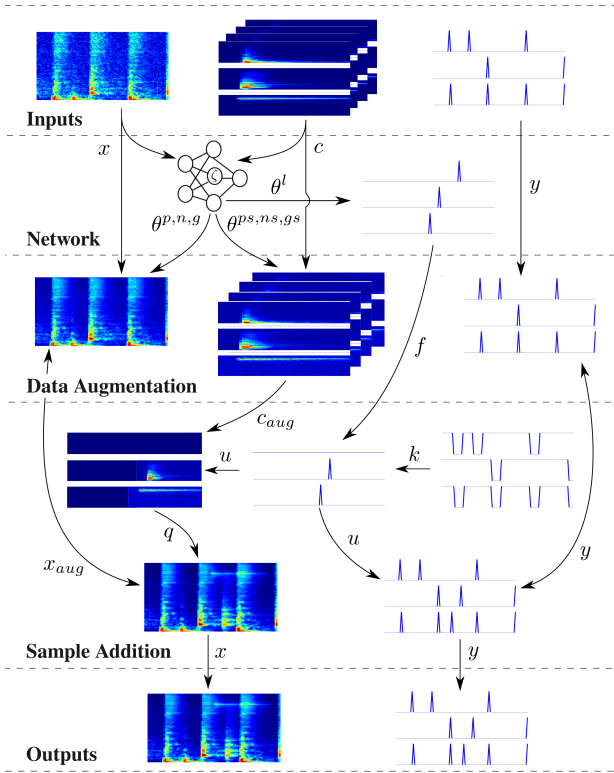
#### 2.2.2 Sample Addition

The sample addition stage aims to reduce transcription accuracy by adding new drum hits to the augmented existing training data using drum samples  $c$  (Section 2.2.3). This is achieved by generating a new spectrogram  $q$  and corresponding target  $u$  and adding them to the existing augmented training spectrogram  $x_{aug}$  and target  $y$ :

$$x = x_{aug} + q_\omega, \quad (3)$$

$$y = y + u_\omega. \quad (4)$$

$\omega$  is the sample number and the total sample number hyperparameter  $\Omega$  determines how many of each sample class are added. Each sample is added in an iterative process with the latest version of  $y$  and  $x_{aug}$  used in the update equations. To create  $q$  and  $u$ , four network determined variables are used:  $\theta^{ps}$ ,  $\theta^{ns}$  and  $\theta^{gs}$  which are variables used to augment each sample using the previously explained augmentation process and  $\theta^l$ , which is used to determine the location of the additional drum samples. For all four network-determined variables, if  $\Omega > 1$  a different variable is used for each sample (i.e.,  $\theta^l = [\theta_1^l, \theta_2^l]$ ). The



**Figure 2.** Overview of player model (Section 2.2) with a data augmentation stage that alters frequency content of pre-existing data and a sample addition stage that adds new drum hits based on generated locations.

generated target  $u$  is created using two variables:  $f$ , an activation function derived from the network determined parameter  $\theta^l$  and  $k$ , a variable that ensures there are no overlaps between the same drum class. The activation function  $f$  for each individual drum sample is derived using:

$$i_\omega = \frac{\theta_\omega^l}{\max(\theta_\omega^l)}, \quad (5)$$

$$f_\omega = \text{ReLU}(i_\omega + \epsilon - \max(i_\omega)) \frac{1}{\epsilon}, \quad (6)$$

where  $\epsilon$  is used to prevent undefined numbers.  $k$  is calculated from the current target  $y$  using a minimum distance between possible locations hyperparameter  $d$ :

$$h_\omega^t = \text{mean}(y^{t-d} : y^{t+d}), \quad (7)$$

$$n_\omega = \frac{\max(h_\omega)}{h_\omega} (h_\omega - \epsilon), \quad (8)$$

$$k_\omega = 1 - \text{ReLU}\left(\frac{n_\omega}{\max(n_\omega)}\right). \quad (9)$$

$u$  is then generated by performing element wise multiplication on the two activation functions  $f$  and  $k$ :

$$u_\omega = k_\omega \odot f_\omega. \quad (10)$$

To calculate the new spectrogram  $q$ , a matrix consisting of all of the possible spectrograms for all  $T$  time steps  $e$  is created using:

$$z_\omega = \text{pad}(c_{aug\omega}, T, T), \quad (11)$$

$$e_\omega^t = z_\omega^{t+b} : z_\omega^{t+b+T}, \quad (12)$$

where  $c_{aug}$  is the augmented current sample spectrogram and  $\text{pad}(c, 1, 1)$  means zero pad  $c$  by 1 in both directions in the time-step dimension. The spectrogram with the sample in the chosen location is then calculated using:

$$q_\omega = \sum_{t=1}^T c_\omega^t u_\omega^t. \quad (13)$$

It is worth noting that the proposed sample addition technique is capable of learning to not add any samples by putting the samples in locations where they overlap other locations and so will be removed.

### 2.2.3 Drum Samples

For drum samples (i.e., isolated drum events) to be utilised within the player model they must be segmented and undergo the same processing as the input features  $x$ . Segmentation of drum events from within larger audio files was achieved automatically through an automatic drum transcription method [13], and subsequently verified manually. Each sample is then cut to a pre-determined sample length with  $b$  frames before the onset. In this work a sample length of 50 is used with  $b = 10$ . The segmented samples are then converted to logarithmic spectrograms  $c$  through the process presented in Section 2.1. More information regarding the extraction of samples is given in Section 3.3.

### 2.2.4 Variations

From the augmentation and sample addition processes we create three different versions of the player model. The first version, termed `AugExist` augments existing training data using only the augmentation stage. The second version, termed `AugAddExist` uses both stages to augment existing training data and add drum samples to the augmented data. The final version, termed `Generate` generates entirely new training data by initializing  $x$  and  $y$  with zeros (i.e., no existing training data).

### 2.2.5 Player Network

The player neural network generates  $\theta$  using a convolutional neural network consisting of two  $3 \times 3$  kernel convolutional layers with max pooling, dropout [17] and batch normalisation [6], followed by a sigmoid fully connected output layer. The first convolutional layer is comprised of 5 channels and the second layer contains 10 channels. The size of the max pooling layer is altered depending on the player parameters (i.e.,  $\Omega$ ,  $v$ ,  $\min_g$ ) so that the total number of trainable parameters of each model is comparable. The input features are the existing training data  $x$  and the different drum instrument samples  $c$  concatenated along the time dimensions. Throughout the remainder of this paper, all trainable player parameters are denoted as  $\zeta$ .

## 2.3 Transcriber

The transcriber model follows the same system outline proposed in [14]. Input features are fed into a pre-trained neural network, which aims to output an activation function  $\tilde{y}$  with spikes in frames where onsets are located. In

this paper, we present a novel system that combines the strength of two recently proposed models. The soft attention mechanism presented in [15] is combined with the convolutional recurrent neural network proposed in [21] to create a convolutional recurrent neural network with a soft attention mechanism output layer. It contains two convolutional layers consisting of 3x3 filters, 3x3 max pooling, dropouts [17] and batch normalization [6], with the first layer consisting of 32 channels and the second is comprised of 64 channels. This is followed by two 20 neuron bidirectional recurrent neural network layers containing long short-term memory cells with peephole connections and a three-neuron sigmoid soft attention mechanism output layer. The attention number is set to 3 [15] and all other unstated variables are the same as the original implementations [15,21]. Throughout the remainder of this paper, all trainable transcriber parameters are denoted as  $\phi$ .

## 2.4 Peak Picking

Once the optimisation process is complete, the peak-picking stage classifies frames of  $\tilde{y}$  as either containing or not containing an onset. We use the mean threshold (MT) peak-picking technique from [15]. First a threshold  $\tau^t$  is determined as:

$$\tau^t = \text{mean}(\tilde{y}^{t-\gamma} : \tilde{y}^{t+\gamma}) * \lambda \quad (14)$$

$$\tau^t = \begin{cases} tmax, & \tau > tmax \\ tmin, & \tau < tmin, \end{cases} \quad (15)$$

where  $\lambda$  is a constant,  $tmax$  and  $tmin$  are the possible maximum and minimum values and  $\gamma$  sets the number of frames used to calculate the mean. The current frame of  $\tilde{y}$  is accepted as an onset if it is the maximum of a surrounding number of frames and above the threshold  $\tau$ :

$$O^t = \begin{cases} 1, & \tilde{y}^t == \max(\tilde{y}^{t-\delta} : \tilde{y}^{t+\delta}) \quad \& \quad \tilde{y}^t > \tau^t \\ 0, & \text{otherwise,} \end{cases} \quad (16)$$

where  $O(t)$  represents an onset at time step  $t$  and  $\delta$  is the number of frames on either side of the current frame  $t$  used to calculate the maximum.

## 2.5 Training

The player and transcriber are iteratively trained for one epoch each in a two-stage process, in which the player is first trained by updating  $\zeta$  and then the transcriber is trained by updating  $\phi$ . Cross entropy is used to minimise the loss function in both instances with  $1 - y$  used as the player target and  $y$  used as the transcriber target. Both systems are trained using mini-batch gradient descent with the Adam optimiser and an initial learning rate of 0.003. These settings were determined using previous ADT studies [15] and also suited the player network well with no instability observed. Each mini-batch consists of 10 randomly chosen, 100 frame length segments. The data is divided into training, validation and test sets, with the training data used to optimize the systems and the validation used to prevent over fitting and to optimize the player and peak-picking parameters. Training is stopped if there has been no decrease in the transcriber validation loss after 10 epochs.

## 3. EVALUATION

To identify whether the PvT approach improves ADT performance, we compare it with the current state-of-the-art supervised ADT approach in six evaluation conditions, consisting of the three contexts and two evaluation strategies used in [22]. To determine whether similarity between drum samples and existing training data affects performance, two different sample libraries are utilised.

### 3.1 Contexts

For the first context, termed *drum transcription of drum-only recordings* (DTD) we utilise the IDMT-SMT-Drums dataset [2]. For the second context, termed *drum transcription in the presence of percussion* (DTP) we utilise the drum-only tracks within the ENST-Drums *minus one* subset [3] and MDB Drums [16]. The third context, termed *drum transcription in the presence of melodic instruments* (DTM), utilises the full polyphonic audio from the ENST-Drums *minus one* subset, MDB-Drums and RBMA-2013 [21].

### 3.2 Evaluation Strategies

The first strategy, termed *random*, utilises all of the context data and divides the tracks in to 70%, 15% and 15% training, validation and test subsets with three-fold cross validation. The second strategy, termed *subset*, utilises all data for the DTD context and only ENST-Drums for DTP and DTM. This strategy aims to test the generalisability of the systems by utilising the existing subsets within the datasets so that the training and testing data are unrelated.

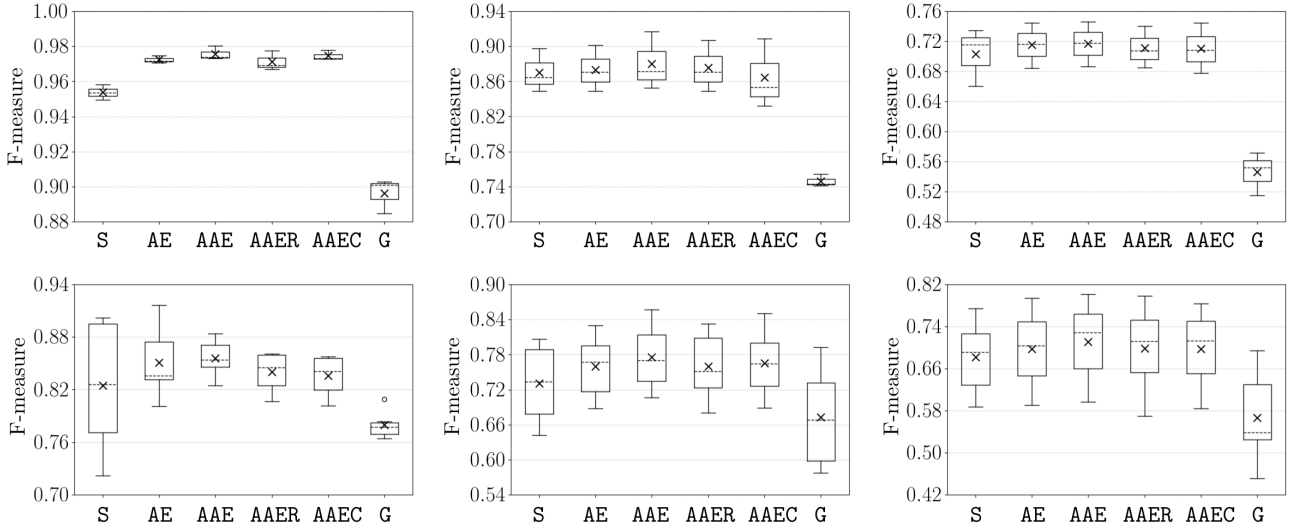
### 3.3 Sample Usage

For drum samples, two sample libraries are used to test for the effect of similarity to existing training data. In addition to these, percussive mixtures are also generated using the content of the unobserved drum samples.

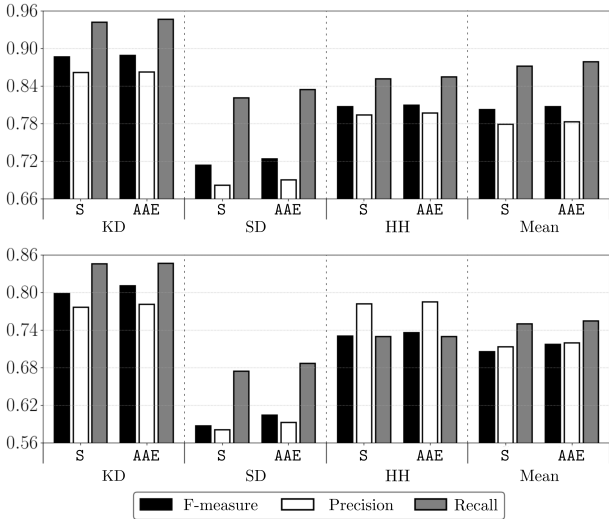
**Training:** The first library, termed *training*, only utilises samples extracted from the training data. For the IDMT-SMT-Drums dataset, a single sample for each observed drum instrument is extracted from each of the 104 tracks. For ENST-Drums the samples are extracted from the included isolated drum files, resulting in a total of 276 samples (21 KD, 146 SD and 109 HH). No samples are extracted from the other datasets; in the DTP and DTM cases the training sample library only consists of ENST-Drum samples.

**Collection:** The second sample library, termed *collection*, consists of drum samples collected from online resources. The collection samples are included as they represent a dataset with a wider diversity and are not included in the existing training data. In total, there are 445 samples (101 KD, 151 SD and 193 HH).

**Polyphonic Instances:** For the DTP versions of the Generate system, the player network output is combined with artificial percussive mixture segments created from other percussive samples (i.e., toms and cymbals) extracted



**Figure 3.** Results for random (top) and subset (bottom) strategies for DTD (left), DTP (middle) and DTM (right) contexts. Crosses denote mean instrument and mean fold F-measure, dashed lines present mean instrument and median fold F-measure and box plots present F-measure range across folds.



**Figure 4.** Individual and mean instrument F-measure, precision and recall scores for existing state-of-the-art system (S) and highest performing AugAddExist PvT model (AAE) using random (top) and subset (bottom) strategies.

from the ENST-Drums isolated files. For the DTM versions of the Generate system the player network output is combined with the artificial percussive mixtures as well as the accompaniment files from the ENST-Drums and MDB-Drums datasets.

### 3.4 Evaluation Methodology

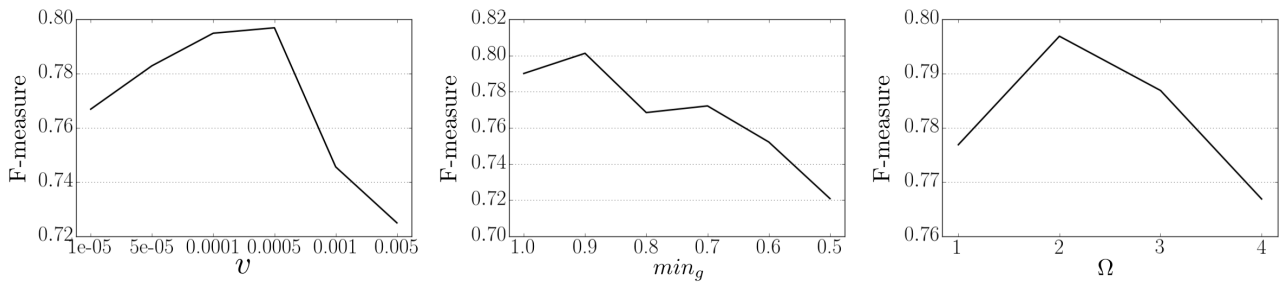
In all evaluations, the three proposed PvT models—AugExist (AE), AugAddExist (AAE) and Generate (G)—are compared with the current state-of-the-art supervised ADT approach (S), which consists solely of the transcriber model. Additionally, two versions of the AAE system are evaluated: for the first (AAER), the  $\theta$  parameters are set to the highest performing random values using a grid search

with the aim of portraying existing data augmentation techniques [8], and the second (AAEC) uses data from the collection samples alone (Section 3.3). The standard precision, recall and F-measure are used as evaluation metrics, with onset candidates being accepted if they fall within 30ms of the ground truth annotations. For all PvT systems  $d$  is set to 3 (approx. 30ms)  $b$  is set to 10 and hyperparameters  $v$ ,  $min_g$  and  $\Omega$  are optimized using grid search. To prevent either networks from overpowering the other, the player max pooling sizes are set so that the number of parameters  $\zeta$  matches that of the transcriber network  $\phi$  (approx. 100,000).

## 4. RESULTS AND DISCUSSION

### 4.1 Random and Subset

Figure 3 presents the random and subset results for the six implemented systems in the three contexts. The crosses represent the mean instrument F-measures and the box plots present the median and range across the folds. In all cases the trained versions of the AugExist (AE) and AugAddExist (AAE) systems achieve a higher mean F-measure and median F-measure than the existing state-of-the-art supervised method (S), with the box plots showing that this improvement is consistent in the majority of the folds. Results from t-tests across folds highlight that the improvements made by all AugExist and AugAddExist systems in the random DTD evaluation are significant (i.e.,  $\rho < 0.05$ ). Larger improvements are seen in the subset evaluation—designed specifically to test the generalisability of the systems [22]—which suggests that the PvT model does improve generalisability. For all PvT variations, the trained AugAddExist player versions (AAE) achieve higher accuracies than the grid search-set AugAddExist system (AAER). This demonstrates the worth of utilising a player network to learn the weaknesses of the transcriber model and its ability to manipulate each of the segments based on the content. Although the Generate



**Figure 5.** Mean fold, mean instrument and mean training strategy F-measure results of different AugAddExist settings for PvT model hyperparameters  $v$  (left),  $min_g$  (middle) and  $\Omega$  (right).

systems do not achieve a higher F-measure they achieve a high accuracy relative to the amount of human input required within the process. Again, this is more apparent in the subset and easier contexts. The lower improvements achieved by the AugAddExist and Generate systems in the DTP and DTM contexts can be attributed to the limitation of samples from just the ENST dataset. This highlights that the more diverse the sample library, the larger the improvement in performance. However, the results from the system trained using the collection sample library (AAEC) show that too much diversity can cause the system to underfit, resulting in a slightly lower improvement being observed. The random- $\theta$  and collection sample library versions of the AugExist and Generate versions were also implemented but not included within the results as the trends are the same as the AAER and AAEC systems.

## 4.2 Individual Instrument

Figure 4 presents the mean fold, mean context individual and mean drum instrument F-measure, precision and recall scores for the supervised and highest performing AugAddExist systems. In all cases the AugAddExist system achieves higher F-measure precision and recall scores for all of the observed drum instruments with 0.015 increase in mean instrument F-measure in random and 0.035 increase in subset. The largest relative improvement is within the snare drum class, which further suggests the increase is due to greater generalisability, as it has proven to be the most difficult instrument to generalise [22].

## 4.3 Player Settings

Figure 5 presents mean instrument, mean fold and mean training strategy F-measure results for the AugAddExist system with different user defined hyperparameter settings. The left diagram presents results for different values of  $v$ , the middle diagram for different values of  $min_g$  and the right diagram different values of  $\Omega$ .  $v = 0.0005$  achieved the highest accuracies with lower values not allowing enough manipulation and higher values causing class overlap.  $min_g = 0.9$  achieved the highest accuracy overall however, within the DTP and DTM contexts lower values achieved similar results. This is possibly due to the fact that a larger diversity of playing technique is present within those contexts. Adding a maximum of two extra drum hits ( $\Omega = 2$ ) resulted in the highest accuracies with larger values causing too much overlap between instruments.

## 4.4 Understanding What The Player Does

By observing the player model training it is possible to gain an understanding of poorly defined areas of the feature space within ADT datasets. When the player performs data augmentation, a maximum amount of augmentation is selected most of the time (i.e., the output of the ReLU function in eq. 1 is close to either 1 or -1). This suggests that there are substantial gaps in coverage of training datasets in the feature space for the different instrumentation. Within the sample addition stage the player model consistently attempts to overlap drums with both other drum instruments and other instrumentation within the existing training data. This suggests that the datasets only contain limited observations of overlapping instrument combinations.

## 5. CONCLUSIONS AND FUTURE WORK

To overcome the requirement of time-consuming manual annotation, we proposed PvT, a game approach to automatic drum transcription. The player model is trained to alter the training data so that the accuracy of the transcriber model is reduced. The three implemented versions of the PvT model—AugExist (AE), AugAddExist (AAE) and Generate (G)—are evaluated alongside the existing supervised state-of-the-art ADT (S) and a grid search-set AugAddExist approach (AAER) using two evaluation strategies and three contexts. The results highlight that the trainable PvT model does improve ADT performance with AugAddExist achieving the highest accuracy in all evaluations. The Generate model also provides a viable option when annotated training data is not accessible. Although two approaches to alter the training data have been implemented, more are possible. Future work could explore trainable methods for moving existing drum events or synthesizing new drum samples within the player network, as there are no structural constraints on what the player can do. For polyphonic cases, the unobserved instrumentation could also be included within the player network so that further combinations can be generated. Another possible direction is to increase the number of observed drum instruments (i.e., including toms and crash cymbals), which is easily done using the Generate model. Open source versions of the PvT model are available online.<sup>2</sup>

<sup>2</sup><https://github.com/CarlSouthall/Player-Vs-Transcriber>

## 6. REFERENCES

- [1] 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 ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, 2016.
- [2] Christian Dittmar and Daniel Gärtner. Real-time transcription and separation of drum recordings based on NMF decomposition. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 187–194, Erlangen, Germany, 2014.
- [3] Olivier Gillet and Gaël Richard. Enst-drums: an extensive audio-visual database for drum signals processing. In *Proceedings of the 7th International Society for Music Information Retrieval Conference (ISMIR)*, 2006.
- [4] Olivier Gillet and Gaël Richard. Transcription and separation of drum signals from polyphonic music. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(3):529–540, 2008.
- [5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.
- [7] Henry Lindsay-Smith, Skot McDonald, and Mark Sandler. Drumkit transcription via convolutive NMF. In *Proceedings of the International Conference on Digital Audio Effects Conference (DAFx)*, York, United Kingdom, 2012.
- [8] Brian McFee, Eric J Humphrey, and Juan Pablo Bello. A software framework for musical data augmentation. In *ISMIR*, pages 248–254, 2015.
- [9] Marius Miron, Matthew E. P. Davies, and Fabien Gouyon. An open-source drum transcription system for pure data and max MSP. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 221–225, Vancouver, Canada, 2013.
- [10] Axel Röbel, Jordi Pons, Marco Liuni, and Mathieu Lagrange. On automatic drum transcription using non-negative matrix deconvolution and Itakura-Saito divergence. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 414–418, Brisbane, Australia, 2015.
- [11] Mathias Rossignol, Mathieu Lagrange, Grégoire Lafay, and Emmanouil Benetos. Alternate level clustering for drum transcription. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pages 2023–2027, Nice, France, August 2015.
- [12] Carl Southall. MIREX 2017 drum transcription submissions. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017.
- [13] Carl Southall, Nick Jillings, Ryan Stables, and Jason Hockman. ADTWeb: An open source browser based automatic drum transcription system. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017.
- [14] Carl Southall, Ryan Stables, and Jason Hockman. Automatic drum transcription using bi-directional recurrent neural networks. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 591–597, New York City, United States, 2016.
- [15] 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 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 606–612, Suzhou, China, 2017.
- [16] Carl Southall, Chih-Wei Wu, Alexander Lerch, and Jason Hockman. MDB drums an annotated subset of medleyDB for automatic drum transcription. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017.
- [17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [18] Lucas Thompson, Simon Dixon, and Matthias Mauch. Drum transcription via classification of bar-level rhythmic patterns. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, pages 187–192, Taipei, Taiwan, 2014.
- [19] Richard Vogl, Matthias Dorfer, and Peter Knees. Recurrent neural networks for drum transcription. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 730–736, New York City, United States, 2016.
- [20] Richard Vogl, Matthias Dorfer, and Peter Knees. Drum transcription from polyphonic music with recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 201–205, New Orleans, United States, 2017.
- [21] Richard Vogl, Matthias Dorfer, Gerhard Widmer, and Peter Knees. Drum transcription via joint beat and

drum modeling using convolutional recurrent neural networks. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 150–157, Suzhou, China, 2017.

- [22] Chih-Wei Wu, Christian Dittmar, Carl Southall, Richard Vogl, Gerhard Widmer, Jason Hockman, Meinard Müller, and Alexander Lerch. A Review of Automatic Drum Transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(9):1457–1483, 2018.
- [23] Chih-Wei Wu and Alexander Lerch. Drum transcription using partially fixed non-negative matrix factorization with template adaptation. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 257–263, Malaga, Spain, 2015.
- [24] Chih-Wei Wu and Alexander Lerch. Automatic drum transcription using the student-teacher learning paradigm with unlabeled music data. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 613–620, Suzhou, China, 2017.