Real-Time Automatic Drum Transcription Using Dynamic Few-Shot Learning

Philipp Weber*, Christian Uhle*†, Meinard Müller†, Matthias Lang*
*Fraunhofer Institute for Integrated Circuits (IIS)
Erlangen, Germany
†International Audio Laboratories Erlangen, Germany

2024 IEEE 5th International Symposium on the Internet of Sounds (IS2) | 979-8-3503-6652-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/IS262782.2024.10704130

***Abstract*—This paper proposes the application of dynamic few-shot learning for automatic drum transcription (ADT). The contributions of this work are threefold. First, we adapt dynamic few-shot learning to improve the classification of superimposed events. Secondly, we introduce a novel method for generating training data for ADT. Thirdly, we demonstrate how our model can be applied in real-time without strongly deteriorating the classification performance. We evaluate transcription performance in the presence of melodic instruments for 10 drum classes on three publicly available test datasets and achieve state-of-the-art performance. We show that new drum classes can be learned and performance for known classes can be improved by providing some examples of that respective class during test time.**

***Index Terms*—music information retrieval, automatic drum transcription, few-shot learning, real-time processing**

I. Introduction

Humans learn new tasks given a small number of examples, e.g., identifying a person based on only one provided image. Machine learning implements this ability by means of few-shot learning (FSL) which enables models to learn from only a few examples per class, and generalize effectively to new, unseen classes [1]. Embedding learning maps inputs to lower-dimensional latent representations, where similar inputs are mapped close together and dissimilar inputs are mapped farther apart [1]. Prototype learning [2] aims to represent each class by a prototype, serving as a representative point in the feature space, facilitating efficient classification, which is regarded as robust when dealing with out-of-distribution samples [3]. These techniques are complementary in the following ways. Embedding learning represents data in a compact manner which is beneficial for prototype learning and FSL. Prototype learning enables classification by defining class prototypes.

In this paper, we investigate FSL using prototypes and embedding techniques for automatic drum transcription (ADT), a subfield of automatic music transcription in the area of music information retrieval (MIR) [4]. ADT aims at detecting and classifying drum sounds in audio signals [5]. In this work, we focus on the transcription for 10 drum classes in the presence of melodic instruments such as guitar, piano, and singing. We use FSL to extend pre-trained ADT models to novel classes and to fine-tune them with examples for which classification performance is low.

†A joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institute for Integrated Circuits (IIS).

Possible applications are the area of the Internet of Musical Things (IoMusT) [6] which is a subfield of the Internet of Sound [7]. A Musical Thing is "a computing device capable of sensing, acquiring, processing, or actuating, and exchanging data serving a musical purpose" [6, p. 61995]. ADT helps to create such devices assisting in music production, music education or for creating visualizations of music in streaming or live performance applications. To allow synchronization in live performances, a low delay is needed which, according to [6], should not exceed $60\,\mathrm{ms}$. The authors of [7] emphasize the need of real-time capable MIR techniques as most research projects focus on offline algorithms that are not applicable in many IoMusT scenarios. In applications were the drum sounds of a single drum kit need to be transcribed, FSL is useful to adapt to the present drum sounds. Such an application can be an e-learning software program which transcribes the drum sounds a student produces in real-time and identifies errors by comparing the transcription to a given score.

II. Related Works

Combinations of prototypical learning, embedding learning and FSL are applied for sound event detection (SED) in [8], [9], [10] and [11] and for ADT in [12]. In [13] and [14], the authors apply prototypical learning and embedding learning without using FSL for SED.

Distance-based prototypical approaches have been found to perform poorly for superimposed sounds [14], which poses a challenge for ADT since drum sets are played such that one or more instruments are active at the same time. The authors of [14] propose to classify superimposed sound events by training a neural network to approximate the latent representation of the input as superposition of pre-trained prototypes, without using FSL. The downside of this approach is that the superposition network needs to be retrained when classes used for inference are altered.

In [12], a negative class prototype is created at inference time from the input signal by selecting signal portions without drum sounds. Then, for each target class, the algorithm performs a classification between respective target and negative class which allows for a classification of superimposed drum sounds. This is not possible for real-time applications with low delay.

We use *dynamic* FSL proposed for image classification in [15], which combines prototype learning and embedding

learning. Unlike most other FSL methods, it only uses FSL for novel classes by means of a dedicated generator for novel class prototypes. The prototypes of classes present in initial training, referred to as base classes, are learned end-to-end without FSL. This has the advantage that base classes do not need to be learned again during test time as the model does not "forget" anything it learned during initial training. Dynamic FSL is applied to SED in [9] and [10]. The authors of [9] compare embedding networks with a larger number of trainable parameters in comparison to the four-layer embedding network proposed in [15]. While larger models achieve higher base-class performance, the novel-class performance decreases. In [10], the authors propose to use a larger 14-layer convolutional embedding network with a global temporal pooling layer and apply a binary cross-entropy loss instead of the proposed categorical cross-entropy loss to allow the occurrence of multiple classes simultaneously, similar as in this paper.

The contributions of this work are further adaptations of the dynamic FSL approach to enhance classification of superimposed drum sounds. We propose to modify the classifier and learn a negative class prototype together with learning base class prototypes. Thereby, unlike in [12], no negative class needs to be sampled at inference time. In contrast to [14], our approach is not limited to matching training and inference classes. As another contribution, we propose a novel method for generating training data including recordings of instruments played by musicians instead of solely using audio files rendered from MIDI files without the downside of time extensive manual labeling effort. Finally, we apply the trained model in real-time still achieving high classification performance.

III. Method

Fig. 1 shows an overview of our proposed method. Input to the model are elements of a dataset $D = \{(x_1, y_1), ..., (x_N, y_N)\}$ with samples x_i and class labels $y_i \in \{1, ..., K\}$ with N being the number of samples and K the number of classes. We use an embedding network $F(\cdot|\theta)$ with learnable parameters θ to compute a latent representation $z_i = F(x_i|\theta) \in \mathbb{R}^d$. D_{base} serves for learning base class prototypes W_{base} from latent representation z_{base}. Similarly, we use D_{novel} to learn novel class prototypes W_{novel} from latent representation z_{novel} and D_{test} to compute latent representation z_{test} for testing the trained model. The classifier $C(z|W_{\text{base}} \cup W_{\text{novel}})$ computes classification scores for base classes and novel classes. The classification scores are derived from the similarities between latent representation z and prototypes of base classes $W_{\text{base}} = \{w_k \in \mathbb{R}^d\}_{k=1}^{K_{\text{base}}}$ and novel classes $W_{\text{novel}} = \{w'_m \in \mathbb{R}^d\}_{m=1}^{K_{\text{novel}}}$. W_{base} are learned in the first training stage without using FSL. The few-shot prototype generator $G(Z', W_{\text{base}}|\theta)$ computes W_{novel} by using W_{base} and latent representations Z' of few novel class examples. $G(\cdot, \cdot|\theta)$ is trained in the second training stage as described in Section III-D.

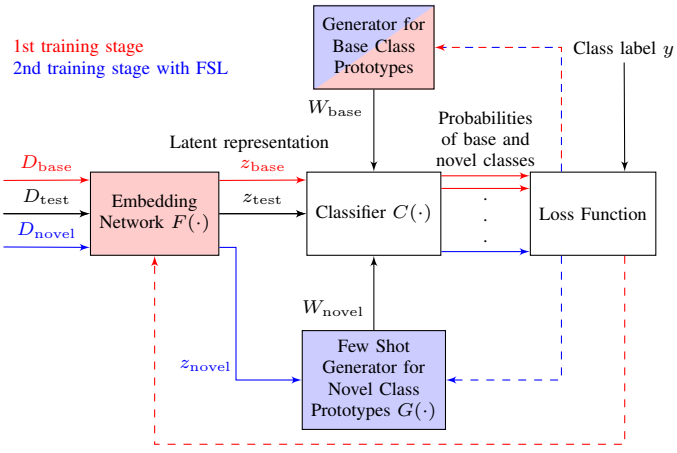

Fig. 1. System overview with embedding network $F(\cdot)$, classifier $C(\cdot)$, loss function, generator for novel class prototypes $G(\cdot)$, base class prototypes W_{base} and novel classes prototypes W_{novel}. Dashed lines are losses.

A. Pre-Processing

Input to the embedding network are mel spectra with 96 frequency bands and an upper cutoff frequency of 16 kHz computed using a 1024-point short-time Fourier transform (STFT) with hop length of 512 samples from single channel audio signals sampled at 48 kHz. The embedding network computes latent representations for blocks of input data, where the number of frames per block is a hyperparameter. The hop length is half a frame. In Section IV-A2, we compare two setting with block length of eight and 21 frames.

B. Embedding Network

The embedding network uses five convolutional layers followed by two dense layers. Each convolutional layer applies 2D convolution with a 3x3 kernel, instance normalization, exponential linear unit (ELU) activation and 2x1 max pooling. Following [15], we excluded the ELU activation in the final convolutional layer. This adjustment permits negative values in the embedding space and has demonstrated enhanced classification performance for novel classes. The outputs are summed up over the time dimension and flattened. The dense layers reduce the dimensionality of the embedding to 384.

C. Classifier

The classifier computes the cosine similarity $s(z, w_k)$ between latent representation z and class prototypes w_k of the k-th class according to

$$s(z, w_k) = \tau \cdot \cos(z, w_k) = \tau \cdot \frac{z \cdot w_k}{\|z\| \|w_k\|} \tag{1}$$

with τ being a trainable scalar value.

We propose to extend the classifier from [15] by learning a single negative class prototype w_{neg} which represents the absence of drum sounds. This enables an independent estimation of the probability of each class and allows high probabilities of several classes simultaneously.

To this aim, we compute a similarity matrix of size $K \times 2$ $M = \left[s(z, w_k) \;\; s(z, w_{\text{neg}})\right]_{1 \le k \le K}$ for a K-way classification

TABLE I
USED DATASET

	Usage	Num. Tracks	Dur. [h]	Content	Annotation
SLAKH	Train	1710	118.3	MIDI	Automatic
STAR	Train	1200	80.7	Rec.+MIDI	Automatic
MDB	Test	23	0.37	Recordings	Manual
ENST	Test	64	1.0	Rec.+MIDI	Manual
RBMA	Test	30	1.9	Recordings	Manual

problem with $s(z, w_k)$ being the similarity of latent representation and k-th class prototype and $s(z, w_{\text{neg}})$ the similarity between latent representation and the negative class prototype. Then, we compute the classification score for the k-th class by applying the softmax function to $s(z, w_k)$ and $s(z, w_{neg})$.

D. Few-Shot Prototype Generator

The few-shot prototype generator computes a novel class prototype $w' = G(Z', W_{\text{base}}|\theta)$ given latent representations $Z' = \{z'_i\}_{i=1}^{N'}$ of N' input examples and W_{base} by using two components: The average of the embeddings of the input examples, referred to as w'_{avg}, and an attention-based prototype component w'_{att} [15]. The novel class prototype w' is computed from both components according to

$$w' = \phi_{\text{avg}} \odot w'_{\text{avg}} + \phi_{\text{att}} \odot w'_{\text{att}} \tag{2}$$

where $\odot$ is the element-wise product, and ϕ_{avg} and ϕ_{att} learnable weight vectors.

The attention-based prototype component w'_{att} is obtained according to

$$w'_{\text{att}} = \frac{1}{N'} \sum_{i=1}^{N'} \sum_{b=1}^{K_{\text{base}}} \text{Att}(\phi_q \cdot \overline{z}'_i, k_b) \cdot \overline{w_b} \tag{3}$$

where K_{base} is the number of base class prototypes, $\text{Att}()$ an attention kernel consisting of cosine similarity function and softmax operation, ϕ_q a learnable weight matrix, k_b learnable keys, $\overline{w_b}$ the l_2-normalized base class prototypes and $\overline{z}'_i$ the l_2-normalized latent representations of the input examples. This approach has similarities to the attention mechanism in sequence-to-sequence models, where at each time step an attention score to all other positions of the input sequence is computed [16]. Here, the base class prototypes correspond to the time sequence. For every input example, we compute similarities to all base class prototypes and use this information for creating the novel class prototype. The learnable keys k_b, one per base class prototype, enable the model to learn how relevant each base class prototype is given a certain input.

The attention-based prototype component w'_{att} offers several benefits instead of only using w'_{avg}. In case N' is very low, its average does not yield a robust class representation. Also, unlike w'_{avg}, w'_{att} makes use of information gained during the training of base class prototypes by exploiting similarities between novel class examples and base class prototypes.

E. Datasets

For both training and inference, we organize the drum sounds across all datasets using the following 10 categories: bass drum, snare drum, hi-hat (including closed and pedal hi-hat sounds), open hi-hat, tom, ride cymbals, cymbals, short percussion, tambourine and bell. The category 'short percussion' is used for unvoiced sounds with fast attack, such as sticks, side stick, clave, rim shots, or claps. The category 'cymbals' includes cymbal drum sounds that are not ride cymbal drum sounds.

1) Publicly Available Datasets Used in the Experiments: We use *Slakh2100* (SLAKH) [17] for training, and *MDB Drums* (MDB) [18], *ENST Drums* (ENST) [19] and *RBMA13* (RBMA) [20] for testing. Table I shows the datasets used in the experiments and lists usage, the extent of data, source of annotation and whether they consist of recordings of instruments played by musicians and/or audio files rendered from MIDI files. For STAR, the drums are rendered from MIDI files, while for ENST a part of the backing tracks includes audio composed of rendered MIDI files. We solely use signals with melodic instruments such as guitar, piano, and singing for training and testing. We exclude the tracks from RBMA which do not contain drum sounds.

The drawbacks of SLAKH are the absence of vocals and the fact that all instruments are synthesized from MIDI files, which is not the case for the data to be expected at test time. In machine learning, similar distributions of training and test data are desirable. This motivates the investigation of other ways to generate ADT training data.

*2) Separate-Tracks-Annotate-Resynthesize Drums (*STAR*):* We propose a new dataset created by **s**eparation of audio **t**racks, **a**nnotation and **r**e-synthesis of drum tracks referred to as STAR. It contains drum sounds synthesized from MIDI files and recordings of melodic instruments played by musicians and singing. To create the STAR dataset the input data must be present as two stems, i.e., the drum stem containing the drum recordings and the non-drum stem containing the recordings of all other instruments and singing. Stems are obtained by processing stereo recordings with the hybrid demucs source separation algorithm which achieves highest performance on drum separation among all compared algorithms in [21]. The signals in MUSDB18 [22] are already provided as instrument stems and no source separation needs to be applied. We create reference annotations by analyzing the drum stems with a 18 class CRNN ADT model [23]. We re-synthesize the drum stems using the reference annotations and 15 professional-

grade virtual drum kits, similar to the ones used in SLAKH. This procedure enables annotations to be obtained without manual labeling effort and guarantees perfect alignment of annotation and audio. The re-synthesized drum stems and the original non-drum stems are loudness normalized to the same target level using BS.1770-4 [24] and mixed. In our opinion, loudness normalization of the stems is an easy and reliable way to obtain mixtures where the loudness levels are on average similar to commercial mixes. We peak normalize the stems after mixing to avoid clipping. The mixes and the re-synthesized drum stems are processed with multiband compression to mimic properties of commercial music recordings.

We exclude 12 items from MUSDB18 which are also present in the test dataset MDB. The remaining 138 items of MUSDB18 are used for validation and testing, while source-separated tracks of 1062 commercial stereo recordings of different genres serve as training data. This results in a dataset with 1200 items with reference annotations and a total length of $80.7\,\mathrm{h}$. We use 1062 tracks for training, 125 for validation and 13 for testing.

Other recent approaches of creating ADT training data are *E-GMD* [25], *ADTOF* [26] [27], and *A2MD* [28]. The authors of *E-GMD* recorded the performances of drummers using an electronic drum kit. This results in MIDI as well as audio files created using different samples from the electronic drum kit. Additionally, velocity information is provided. *E-GMD* supports seven drum classes but does not contain melodic instruments and is targeted towards drum-only transcription. Consequently, we did not use *E-GMD* in this paper. *ADTOF* relies on crowdsourced manual annotations which were post-processed to improve temporal alignment and reduce errors. The authors of *A2MD* use audio-to-MIDI alignment with an additional data inspection step to automatically create annotations. Both datasets do not make use of virtual instruments. However, *ADTOF* supports only five and *A2MD* only three classes. Therefore, we did not include them in the comparisons for this paper.

In contrast to STAR, the approaches used in *ADTOF* and *A2MD* carry the risk of introducing annotation errors. In STAR, the original drum recording is not included in the final mix. Therefore, annotation errors and inaccurate timing introduced by the ADT algorithm used for creating the annotations are also reflected in the synthesized drum stem and do not lead to incorrect annotations. Another advantage of our STAR approach is that it allows for training with a high number of classes. The number of classes is only limited by the supported classes of the ADT algorithm used for creating the reference annotations, in this case to 18. In contrast, increasing the class vocabulary of *ADTOF* or *A2MD* would require additional labeling effort. Furthermore, STAR allows to choose recordings such that they are representative with regards to expected inference data without any restrictions.

F. Training

Training is done in two stages. In both stages the loss is the binary cross-entropy between probabilities obtained by the classifier and reference annotations. We create soft labels by applying label smoothing [29] in the form of adding (to the negative target labels) or subtracting (from the positive target labels) a value drawn from a normal distribution which is scaled by a factor tuned as hyperparameter. Additionally, we use target widening by labeling not only the frame with the drum sound as a positive target with a weight of 1, but also the two adjacent frames of the drum noise with lower weights of 0.6 and 0.3, similar to [30]. We apply data augmentation to the mel spectra in the form of random gain offsets and spectral and temporal masking as described in [31]. Additionally, we balance the number of blocks with and without drum onsets in the training and validation split. To account for imbalanced number of drum sounds of different classes, class weights are computed and applied to the loss such that the loss of a frame containing a rare drum class sound has greater influence than the loss of a frame containing frequent drum class sounds. The class weights are inverse proportional to the class frequency and are clipped to avoid very high weights, which can cause non converging training. The negative class weight is a hyperparameter and tuned such that precision and recall are balanced during training.

1) First Training Stage: The first stage optimizes the parameters of the embedding network and trains base class prototypes and the negative class prototype by minimizing the binary cross-entropy loss. We use the *Adam* [32] optimizer and start at learning rate of 0.0003 which is halved every four epochs. Similar to [10], we use a binary cross-entropy loss instead of the categorical cross-entropy loss proposed in [15].

2) Second Training Stage: The second stage trains the prototype generator as proposed in [15] while still updating the base class prototypes, with the embedding network being frozen. In every batch we randomly pick one base class which is used to train the novel class prototype generator and therefore treated as novel class even though it is already known to the embedding network. We sample five examples of the simulated novel class from the training dataset. The examples are embedded and fed to the prototype generator to generate a novel class prototype. The attention-based prototype component is inferred without the use of the corresponding base class prototype. The classifier then uses the newly-created prototype to classify the current batch.

G. Real-Time Inference

For real-time inference, we set a requirement of a maximum transcription delay of $60\,\mathrm{ms}$ as suggested in [6] to allow synchronization in live applications. This limits the possible lookahead and consequently the block length as we train the model to detect a drum sound onset when it is located in the center of the current block. Due to target widening, see Section III-F, the classification score of the classifier already rises before the drum sound reaches the center of the current block. The hop length is half a frame, corresponding to $10.7\,\mathrm{ms}$. During the evaluation, we also account for the delay of one frame caused by the STFT. In real-time conditions, we observed the best performance using a block length of eight

frames, corresponding to $85.3\,\mathrm{ms}$. For most of the detected drum sounds this results in a transcription delay of three to five frames, corresponding to 32.1 to $53.5\,\mathrm{ms}$.

In a scenario without real-time restrictions, the model benefits from more temporal context in terms of a higher block length, which is shown in the performance comparison in Section IV-A. We also tried using higher block lengths without increased lookahead by training the model to detect a drum sound onset when it is located earlier than in the center of the current block. This provides more temporal context from past frames, but did not increase performance.

Based on the classification scores computed by the classifier, a drum sound onset is detected if

$$p_{k,n} > \alpha \tag{4}$$

$$p_{k,n} - f(p_{k,n}, \tau_\mathrm{s}) > \alpha_\mathrm{atk} \tag{5}$$

$$n_k - n_{\mathrm{act},k} > \tau_\mathrm{o}. \tag{6}$$

In (4), a threshold α is applied to $p_{k,n}$ which denotes the probability of the k-th class being active in the n-th frame. Equation (5) evaluates if $p_{k,n}$ is in an attack phase by using a recursive averaging $f(\cdot)$ with time constant τ_s. It applies a second threshold α_atk to reduce false positives if $p_{k,n}$ exceeds α but is not rising, meaning that there is no future peak. Unlike the identification of local maxima, this does not cause additional delay. Equation (6) avoids detecting the same drum sound multiple times within τ_o frames. This is done by evaluating the number of frames between current frame n_k and the frame $n_{\mathrm{act},k}$ where the k-th class was last active.

Parameters α, α_atk, τ_s and τ_o are tuned manually on the test split of the training dataset separately for classification with base and novel class prototypes.

IV. Results and Discussion

To assess the classification accuracy, we compare the positions of reference onsets and estimated drum sound onsets using *mir_eval* [33], allowing a tolerance of $\pm\ 30\,\mathrm{ms}$. In real-time scenarios, where predicting future events is not possible, we allow a tolerance of $60\,\mathrm{ms}$ starting from the reference annotation, resulting in an equal total tolerance window length.

In our evaluation, we utilize global and instrument-wise micro F-measures calculated by summing up all true positives, false positives, and false negatives over all tracks instead of computing a mean by averaging F-measures per track. This approach helps avoid bias in cases where some tracks do not contain all drum classes or where the onset count varies significantly among the test tracks.

A. Performance Using Base Class Prototypes

1) `SLAKH` *and* `STAR`*:* We compare two models trained with the proposed `STAR` and `SLAKH` to address the question whether `STAR` leads to a higher performance due to more realistic training data. To avoid a performance bias due to different amount of data, the number of `SLAKH` items is reduced such that the number of tracks in train and validation set matches the size of `STAR`. For both trainings we use the same hyperparameter setting and a block length of eight frames.

TABLE II
F-measure comparison of model trained with `SLAKH` and `STAR` and tested on `MDB`

Class	`SLAKH`	`STAR`
All	0.50	0.62
Kick drum	0.68	0.75
Snare drum	0.59	0.66
Hi-hat	0.66	0.72
Open hi-hat	0.14	0.14
Tom	0.12	0.23
Cymbals	0.05	0.14
Ride cymbals	0.10	0.17
Short Percussion	0.07	0.27
Tambourine	0.00	0.07
Bell	0.00	0.00

TABLE III
F-measure comparison of proposed algorithm with `SOTA` from [23] and ablations

Model	`MDB`	`ENST`	`RBMA`
`SOTA-CRNN`	0.65	**0.70**	0.52
`SOTA-CNN`	0.63	0.61	**0.53**
`Proto-BL8-RT`	0.65	0.56	0.42
`Dense-BL8-RT`	0.64	0.51	0.36
`Proto-BL8-OFF`	0.67	0.66	0.47
`Dense-BL8-OFF`	0.67	0.65	0.43
`Proto-BL21-OFF`	0.69	0.66	0.51
`Dense-BL21-OFF`	**0.70**	0.67	0.51
Ablations:			
`NoICPC-Proto-BL8-RT`	0.58	0.48	0.36
`NoNegC-Proto-BL8-RT`	0.57	0.48	0.35

Table II shows the performance of the two models for 10 transcribed classes on `MDB`. We observe a global F-measure improvement of 0.12 and enhancements in nearly all classes when training with `STAR`.

2) Comparison with State of the Art: We compare our algorithms with the offline CNN and CRNN models for eight-classes from [23] which are published as an ensemble of networks on an accompanying website in five different versions. To the best of our knowledge, they still represent state of the art (SOTA) in the eight-class scenario. Two models are trained on synthetic datasets consisting of rendered MIDI files. The remaining three models are trained on `MDB`, `ENST` and `RBMA` which are used as test datasets in this paper. The used *madmom* framework averages the activation functions of all available models. We exclude the one model trained on the test dataset for which performance is being evaluated, and, apply the remaining models with the default peak-picking parameters. For the comparison we reduce our 10 classes to eight to use the same drum classes as those in [23].

The CRNN model from [23] uses a bi-directional RNN architecture, trained on sequences of $4\,\mathrm{s}$ length and is therefore not real-time capable. The CNN model was trained with a temporal context of $250\,\mathrm{ms}$ and would cause higher delays than the targeted $60\,\mathrm{ms}$ in this paper if applied in real time. Table III shows the results of SOTA and our proposed algorithm in several variants which are encoded in the model's name on

the three used test datasets MDB, ENST and RBMA. Furthermore, Table III shows two ablations discussed below. First, we investigate the question of how the use of prototypes affects the performance by training a model which does not use the prototype-based classifier introduced in Section III-C. Instead, we add a dense layer with eight outputs to the embedding network $F(\cdot|\theta)$ followed by a sigmoid function and interpret the output of the dense layer as class probabilities. This variant is referred to as Dense in Table III in contrast to Proto which uses prototypes.

We also assess the question if more temporal context improves the performance by increasing the block length from eight frames (BL8) to 21 frames (BL21) which results in non-real-time capable models marked with OFF. We also evaluate the real-time capable models (RT) under offline conditions. This includes the use of an offline peak picking which identifies local maxima by evaluating probabilities of future frames instead of using Equation (5) and the compensation of the introduced delay.

Table III shows that the performance of our prototype-based real-time algorithm (Proto-BL8-RT) is the same as SOTA-CRNN on MDB with F-measure of 0.65 and lower performance on ENST and RBMA. We did not use any virtual electronic drum kits for creating the STAR training dataset which may lead to the low performance on RBMA which focuses on electronic music. The detection delay averaged over all instruments and tracks on MDB is $42.8\,\mathrm{ms}$ and lies withing the goal of $60\,\mathrm{ms}$, see Section III-G.

Using a dense layer for classification instead of prototypes leads to similar performance in most scenarios. Only the BL8 models are less efficient on ENST and RBMA when using the additional dense layer instead of prototypes.

In a scenario without real-time restrictions, the algorithm without prototypes and block length of 21 frames (Dense-BL21-OFF) outperforms the other variants of our algorithm with F-measure of 0.70 on MDB. This version also achieves better results than SOTA-CNN on MDB and ENST and outperforms SOTA-CRNN on MDB. We would like to stress that our models use a similar embedding network architecture to SOTA-CNN and are, unlike SOTA-CRNN, not able to learn rhythm patterns and repetitive structures outside of the used block length.

The increase in performance when evaluating the BL8 models under offline conditions depends on the test data. We observe a minor performance improvement on MDB and higher performance improvements on ENST and RBMA.

3) Ablation Study: We assess the performance of models trained without our proposed modifications in an ablation study: First, we remove the independent class probability computation (NoICPC) as described in Section III-C and therefore limit the ability to detect superimposed drum sounds. In this case, the softmax function is applied to all classes, including the negative class, meaning that the probabilities of all classes sum up to 1. In Table III this version leads to an F-measure decrease on all datasets.

In a second step, we additionally remove the negative class prototype (NoNegC) and perform the softmax function over all drum classes. This barely affects performance in comparison to the previous step as it can be seen in Table III. However, the removal of the negative class prototype leads to probability curves with higher noise floor as shown in Fig. 2 as the classifier cannot assign probabilities close to 0 to all classes simultaneously if no drum sound occurs.

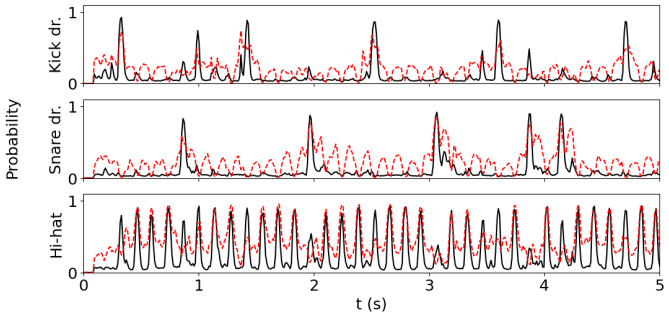

Fig. 2. Class probabilities when transcribing kick drum, snare drum and hi-hat with (solid, black) and without (dotted, red) independent class probability computation and negative class.

B. Performance Using Novel Class Prototypes

Here we evaluate the two novel class prototype scenarios: Learning new classes and fine-tuning of existing classes with sound examples for which performance with base class prototypes is low.

1) Learning New Classes: To evaluate classification performance for new classes, we identify all drum classes from the three test datasets not present in the training data which are snare drum brush sounds in MDB and shaker sounds in RBMA. For each of the two novel classes, we sample five examples to generate a novel class prototype by using the few-shot prototype generator $G(\cdot,\cdot|\theta)$, and we use this prototype for classification. This is done separately for all tracks containing the novel classes. We obtain F-measure values of 0.41 for snare drum brush sounds and 0.21 for shaker sounds. This is a performance in a similar range to using base class prototypes for classes other than the three main drum instrument classes, kick drum, snare drum, and hi-hat, see Table II. This shows that novel classes can be successfully learned from few examples.

2) Fine-Tuning of Existing Classes: We asses the fine-tuning scenario on RBMA as the tracks of this dataset partly use acoustic (track ID 1, 5, 9, 15, 17, 19 and 23) and partly electronic drum sets (all other tracks). We did not use any electronic drum set for the creation of the training dataset STAR and therefore we expect a lower performance on electronic drum sounds (EDrum) in comparison to acoustic drum sounds (ADrum) when using base class prototypes (PrBase). The first two columns of Table IV show that this is the case for most classes. For kick drum, we can, e.g., see a F-measure decrease from 0.67 to 0.41.

To investigate if performance for electronic drum sets can be increased by using novel class prototypes (PrNov) instead of base class prototypes, we randomly sample five drum sounds

TABLE IV
F-MEASURE PERFORMANCE WHEN USING NOVEL CLASS PROTOTYPES (PrNov) OR BASE CLASS PROTOTYPES (PrBase) ON KNOWN CLASSES FOR SIGNALS CONTAINING ELECTRIC DRUM SOUNDS (EDrum) AND ACOUSTIC DRUM SOUNDS (ADrum).

	ADrum	EDrum	
Class	PrBase	PrBase	PrNov
Kick drum	0.67	0.41	**0.56**
Snare drum	0.50	0.38	0.35
Ride cymbals	0.00	0.06	**0.19**
Cymbals	0.21	0.02	**0.10**
Tom	0.37	0.09	**0.28**
Hi-hat	0.44	0.50	0.18
Open hi-hat	0.16	0.01	**0.33**
Short perc.	0.57	0.30	0.24
Tambourine	0.00	0.06	**0.28**
Bell	0.05	0.00	**0.10**

of every class of each track to create novel class prototypes and use them for transcription. The last column of Table IV shows that for most classes (highlighted in bold) the performance on electronic drum sets increases compared to the use of base class prototypes. For kick drum, we can, e.g., see an F-measure increase from 0.41 to 0.56. This is useful for application scenarios where manual tuning is applied to improve performance for dedicated tracks.

We observe a different trend for the drum class hi-hat, where novel class prototypes strongly lower the performance and the base class prototype achieves a slightly higher performance on electronic hi-hat sounds than on acoustic hi-hat sounds. This indicates that electronic hi-hat sounds are similar enough to acoustic hi-hat sounds to be well detected with the base class prototypes.

V. CONCLUSION

We showed that dynamic few-shot learning can be successfully applied to ADT in real time. We achieved a performance that is competitive to SOTA offline algorithms and can learn new classes and fine-tune our model at inference time by providing few examples. We also proposed how realistic training data including recordings of instruments played by musicians can be generated and that this improves transcription performance compared to completely relying on MIDI-synthesized audio.

In future works, we plan to create a version of STAR relying on copyright-free music to be able to make the proposed dataset available to the research community. Furthermore, we hope to improve the transcription performance of models trained with STAR on electronic music by additionally using virtual electronic drum kits during the dataset generation.

REFERENCES

[1] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Computing Surveys*, vol. 53, no. 3, pp. 1–34, 2020.

[2] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in Neural Information Processing Systems*, vol. 30, pp. 4077–4087, 2017.

[3] W. Xu, Y. Xian, J. Wang, B. Schiele, and Z. Akata, "Attribute prototype network for zero-shot learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21969–21980, 2020.

[4] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.

[5] C. Wu, C. Dittmar, C. Southall, R. Vogl, G. Widmer, J. Hockman, M. Müller, and A. Lerch, "A review of automatic drum transcription," *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 26, no. 9, pp. 1457–1483, 2018.

[6] L. Turchet, C. Fischione, G. Essl, D. Keller, and M. Barthet, "Internet of musical things: Vision and challenges," *IEEE Access*, vol. 6, pp. 61994–62017, 2018.

[7] L. Turchet, M. Lagrange, C. Rottondi, G. Fazekas, N. Peters, J. Østergaard, F. Font, T. Bäckström, and C. Fischione, "The internet of sounds: Convergent trends, insights, and future directions," *IEEE Internet Things J.*, vol. 10, no. 13, pp. 11264–11292, 2023.

[8] Y. Wang, J. Salamon, N. J. Bryan, and J. Pablo Bello, "Few-shot sound event detection," in *Proc. ICASSP*, pp. 81–85, 2020.

[9] Y. Wang, N. J. Bryan, J. Salamon, M. Cartwright, and J. P. Bello, "Who calls the shots? rethinking few-shot learning for audio," in *Proc. WASPAA*, pp. 36–40, 2021.

[10] Y. Wang, N. J. Bryan, M. Cartwright, J. P. Bello, and J. Salamon, "Few-shot continual learning for audio classification," in *Proc. ICASSP*, pp. 321–325, IEEE, 2021.

[11] D. Yang, H. Wang, Y. Zou, Z. Ye, and W. Wang, "A mutual learning framework for few-shot sound event detection," in *Proc. ICASSP*, pp. 811–815, IEEE, 2022.

[12] Y. Wang, J. Salamon, M. Cartwright, N. Bryan, and J. Bello, "Few-shot drum transcription in polyphonic music," in *Proc. 21st ISMIR*, 10 2020.

[13] P. Zinemanas, M. Rocamora, M. Miron, F. Font, and X. Serra, "An interpretable deep learning model for automatic sound classification," *Electronics*, vol. 10, no. 7, p. 850, 2021.

[14] P. Zinemanas, M. Rocamora, E. Fonseca, F. Font, and X. Serra, "Toward interpretable polyphonic sound event detection with attention maps based on local prototypes," in *Proc. 6th DCASE*, pp. 50–54, Universitat Pompeu Fabra, 2021.

[15] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *Proc. CVPR*, pp. 4367–4375, 2018.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. 30* (I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds.), pp. 5998–6008, 2017.

[17] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, "Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity," in *Proc. WASPAA*, pp. 45–49, IEEE, 2019.

[18] C. Southall, C.-W. Wu, A. Lerch, and J. Hockman, "MDB drums: An annotated subset of MedleyDB for automatic drum transcription," 2017.

[19] O. Gillet and G. Richard, "ENST-drums: an extensive audio-visual database for drum signals processing," in *Proc. 7th ISMIR*, pp. 156–159, 2006.

[20] R. Vogl, M. Dorfer, G. Widmer, and P. Knees, "Drum transcription via joint beat and drum modeling using convolutional recurrent neural networks," in *Proc. 18th ISMIR* (S. J. Cunningham, Z. Duan, X. Hu, and D. Turnbull, eds.), pp. 150–157, 2017.

[21] A. Défossez, "Hybrid spectrogram and waveform source separation," *CoRR*, vol. abs/2111.03600, 2021.

[22] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017.

[23] R. Vogl, G. Widmer, and P. Knees, "Towards multi-instrument drum transcription," in *Proc. 21st DAFx'18*, 2018.

[24] ITU-R Recommendation BS.1770-4, *Algorithms to measure audio programme loudness and true-peak audio level*, 2015.

[25] L. Callender, C. Hawthorne, and J. Engel, "Improving perceptual quality of drum transcription with the expanded groove MIDI dataset," 2020.

[26] M. Zehren, M. Alunno, and P. Bientinesi, "ADTOF: A large dataset of non-synthetic music for automatic drum transcription," in *Proc. 22nd ISMIR*, pp. 818–824, 2021.

[27] M. Zehren, M. Alunno, and P. Bientinesi, "High-quality and reproducible automatic drum transcription from crowdsourced data," *Signals*, vol. 4, no. 4, pp. 768–787, 2023.

[28] I. Wei, C. Wu, and L. Su, "Improving automatic drum transcription using large-scale audio-to-midi aligned data," in *Proc. ICASSP*, pp. 246–250, IEEE, 2021.
[29] C.-B. Zhang, P.-T. Jiang, Q. Hou, Y. Wei, Q. Han, Z. Li, and M.-M. Cheng, "Delving deep into label smoothing," *IEEE Trans. on Image Processing*, vol. 30, pp. 5984–5996, 2021.
[30] S. Böck and M. E. P. Davies, "Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation," in *Proc. 21th ISMIR* (J. Cumming, J. H. Lee, B. McFee, M. Schedl, J. Devaney, C. McKay, E. Zangerle, and T. de Reuse, eds.), pp. 574–582, 2020.
[31] D. S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," in *20th Interspeech, 2019* (G. Kubin and Z. Kacic, eds.), pp. 2613–2617, ISCA, 2019.
[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd ICLR* (Y. Bengio and Y. LeCun, eds.), 2015.
[33] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "MIR_EVAL: A transparent implementation of common MIR metrics.," in *Proc. 15th ISMIR*, vol. 10, p. 2014, 2014.