HYPERNETWORKS FOR SOUND EVENT DETECTION: A PROOF-OF-CONCEPT

Shubhr Singh[1], *Huy Phan*[1,2], *Emmanouil Benetos*[1,2]

[1]Centre for Digital Music, Queen Mary University of London, UK [2]The Alan Turing Institute, UK

ABSTRACT

Polyphonic sound event detection (SED) involves the prediction of sound events present in an audio recording along with their onset and offset times. Recently, Deep Neural Networks, specifically convolutional recurrent neural networks (CRNN) have achieved impressive results for this task. The convolution part of the architecture is used to extract translational invariant features from the input and the recurrent part learns the underlying temporal relationship between audio frames. Recent studies showed that the weight sharing paradigm of recurrent networks might be a hindering factor in certain kinds of time series data, specifically where there is a temporal conditional shift, i.e. the conditional distribution of a label changes across the temporal scale. This warrants a relevant question - is there a similar phenomenon in polyphonic sound events due to dynamic polyphony level across the temporal axis? In this work, we explore this question and inquire if relaxed weight sharing improves performance of a CRNN for polyphonic SED. We propose to use hypernetworks to relax weight sharing in the recurrent part and show that the CRNN's performance is improved by $\approx 3\%$ across two datasets, thus paving the way for further exploration of the existence of temporal conditional shift for polyphonic SED.

Index Terms— hypernetworks, sound event detection, weight sharing, recurrent networks

1. INTRODUCTION

Environmental sounds carry a rich and complex mixture of information, which is organised and categorized by the human auditory system into a distinct set of concepts known as *sound events* (e.g. dog bark, door slam, car passing by). The task of sound event detection (SED) deals with *identification* and *temporal localisation* of these sound events in a given audio recording and can be broadly divided into two categories - *monophonic* and *polyphonic* SED [1, 2, 3]. The former deals with the presence of a single sound event at a particular instance of time, regardless of the total number of sound events present in an audio recording; on the other hand, the latter deals with more realistic scenarios where multiple sounds may co-occur at any given point of time.

Multiple factors make polyphonic SED a challenging task. First, each sound event has varied acoustic characteristics and an overlap of multiple sound events results in a spectral profile which is distinct from the individual instances. Second, the polyphony at a given time point is unknown and potentially large [4], which magnifies the challenge of designing a robust SED system. Third, another significant challenge that separates polyphonic sound events from music or speech is the lack of definitive sequential structure as compared to the temporal evolution structure of notes or phonemes. Hence, traditional sequential machine learning methodologies such as hidden Markov models are not so effective for polyphonic SED as compared to music or speech processing [1].

The advent of recent approaches based on neural networks have been quite successful for polyphonic SED, especially convolutional recurrent neural networks (CRNNs) [5, 6] and transformer based architectures [7, 8, 9]. In a recent work [10], it was proposed that the hard weight sharing of RNNs makes it difficult for the network to effectively model certain time series data, specifically clinical data, where the relationship between the input feature x and the label y varies across the temporal scale. More formally, the task of an RNN is to estimate the probabilities $p\left(y_t\left(k\right)|x_{1:t}, \theta\right)$ for event classes $k = 1, 2, ..., K$ in frame t, where θ denotes the classifier's parameters and x_t denotes the input feature at time t [11]. In case of temporal conditional shift, the formulation of the task changes to estimation of $p\left(y_t\left(k\right)|x_{1:t}, t, \theta_t\right)$ where θ changes over time. The work in [10] explored this phenomenon for clinical data and showed that relaxation of weight sharing led to improvement in performance of standard recurrent cells.

Based on this result, we aim to explore if such a phenomenon exists in polyphonic sound events as the fluctuation in polyphony level and uncertain recurrence relations across the temporal axis might induce a shift in the conditional distribution. In such a scenario, weight sharing might not be the optimal choice to model sound events, i.e. the standard recurrent units might not be adequately capturing the dynamic relationship between features and labels across the temporal axis. To the authors' knowledge, the existence of such a phenomenon has not been explored before for polyphonic SED. In this

S. Singh is a research student at the UKRI Centre for Doctoral Training in Artificial Intelligence and Music, supported jointly by UK Research and Innovation [grant number EP/S022694/1] and Queen Mary University of London. E. Benetos and H. Phan are supported by a Turing Fellowship under the EPSRC grant EP/N510129/1.

work, we explore this idea and seek to answer the basic question - does the introduction of relaxed weight sharing improve the performance of CRNNs for polyphonic SED? We exploit hypernetworks [12] to achieve relaxation of weight sharing. Via the conducted experiments, we show that the proposed hypernetwork-based CRNN consistently improved the segment based F-1 score of the CRNN baseline by $\approx 3-4\%$ across two synthetically generated datasets: TUT-SED Synthetic 2016 [13] and Urban-SED dataset [14].

2. HYPERNETWORKS FOR SED

2.1. Dynamic hypernetworks

Hypernetworks is an approach where an auxiliary network (called a "hypernetwork") is used to generate the weights of a primary network (called the "main" network) [12]. Two kinds of hypernetworks were introduced in [12]: *static* and *dynamic* hypernetworks. The static hypernetwork generates the weights for each layer of a feedforward CNN while the dynamic hypernetwork modifies the weights of an RNN or a Long Short-Term Memory (LSTM) network [15] across the time axis. In this work, we use the dynamic hypernetwork to modify the weights of a CRNN architecture. For the rest of this paper, we leave out the term dynamic and refer to the auxiliary LSTM as hypernetwork. The hypernetwork has its own hidden units and its input sequence is constructed by concatenating the input vector x_t at time t and hidden state vector h_{t-1} at time $t-1$ of the main LSTM network.

Adopting the formulation used in [12], a standard LSTM [15] cell consists of a memory cell (c) and 4 gates: input (i), output (o), forget (f), and transformation (g). The transformation gate g is applied before updating the memory cell c. Given an input x_t, and the hidden state from previous time step h_{t-1}, the equation of an LSTM cell at time t is given as

$$i_t = W_h^i h_{t-1} + W_x^i x_t + b^i, \quad (1)$$
$$g_t = W_h^g h_{t-1} + W_x^g x_t + b^g, \quad (2)$$
$$f_t = W_h^f h_{t-1} + W_x^f x_t + b_f, \quad (3)$$
$$o_t = W_h^o h_{t-1} + W_x^o x_t + b_o, \quad (4)$$
$$c_t = \sigma(f_t) \odot c_{t1} + \sigma(i_t) \odot \phi(g_t), \quad (5)$$
$$h_t = \sigma(o_t) \odot \phi(c_t), \quad (6)$$

where $W_x^y \in R^{N_h \times N_x}$ and $W_h^y \in R^{N_h \times N_h}$ are the matrices of the input and hidden state, respectively. y refers to one of $\{i, g, f, o\}$. $b^y \in R^{N_h}$ denotes the bias whereas σ and ϕ denote the sigmoid and tanh functions, respectively.

In the case of hypernetwork, the weights of the input, hidden state and bias of the main LSTM network are a function of embeddings z_x, z_h and z_b, respectively. The embeddings are projected to the parameters of the main network using the matrices $W_{xz} \in R^{N_h \times N_x \times N_z}$, $W_{hz} \in R^{N_h \times N_h \times N_z}$ and $W_{bz} \in R^{N_h \times N_z}$ as shown in (7)-(9):

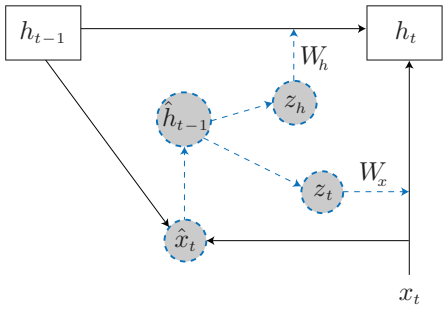

Fig. 1. Block diagram for a dynamic hypernetwork. The dotted circles represent a hypernetwork and the solid blocks represent the main LSTM. Input x_t at time t and hidden state h_{t-1} at time $t-1$ of the main LSTM are concatenated and fed to the hypernetwork, which computes the embeddings z_x and z_h. These embeddings are used to modify the weights W_x and W_h of the main LSTM.

$$W_x^y = \langle W_{xz} z_x \rangle, \quad (7)$$
$$W_h^y = \langle W_{hz} z_h \rangle, \quad (8)$$
$$b^y = \langle W_{bz} z_b \rangle \quad (9)$$

The operation $\langle W, z \rangle$ here denotes a tensor dot product between W and z. As discussed earlier, at each time step, the hypernetwork takes as input $\hat{x}_t$ a concatenated vector of the input x_t and the hidden state h_{t-1} of the main LSTM cell:

$$\hat{x_t} = \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}. \quad (10)$$

Let $\hat{h}_t$ denote the hidden state of the hypernetwork at time t. It is computed as

$$\hat{h}_t = \phi(W_{\hat{h}} \hat{h}_{t-1} + W_{\hat{x}} \hat{x}_t + \hat{b}), \quad (11)$$
$$z_h = W_{\hat{h}h} \hat{h}_{t-1} + b_{\hat{h}h}, \quad (12)$$
$$z_x = W_{\hat{h}x} \hat{h}_{t-1} + b_{\hat{h}x}, \quad (13)$$
$$z_b = W_{\hat{h}b} \hat{h}_{t-1}, \quad (14)$$

where $W_{\hat{x}} \in R^{N_{\hat{h}} \times (N_h + N_z)}$ and $W_{\hat{h}} \in R^{N_{\hat{h}} \times N_{\hat{h}}}$ are the input and the hidden state matrices of the hypernetwork. The projection matrices $W_{\hat{h}h}, W_{\hat{h}x} \in R^{N_z \times N_{\hat{h}}}$ and $W_{\hat{h}b} \in R^{N_z}$ are used for computing z_h, z_x and z_b, respectively. The operation of the hypernetwork is quite similar to a squeeze-and-excitation network [16], where each output feature map from a convolutional layer is reduced to a single value through pooling operation, resulting in a vector of size n, where n is the number of channels. This vector is passed through a small network which produces an embedding that is projected to a vector of size n. This vector is then used for scaling the feature maps.

In practice, equations (7) and (8) are not feasible because memory usage becomes too large for realistic computation.

Hence, the original work proposed an intermediate hidden vector $d(z) \in R^{N_h}$ which is used to linearly scale each row of the main LSTM weight matrices as

$$W_h^y = W_h\left(d\left(z\right)\right) = \begin{pmatrix} d_0(z)W_0 \\ \dots \\ d_{N_h}(z)W_{N_h} \end{pmatrix}. \quad (15)$$

A similar formulation is applied for W_x^y. $d(z)$ is calculated as a linear projection of the hypernetwork embedding z_k, where k refers to any of $\{x, h, b\}$. Using an intermediate vector in this way sacrifices the ability of constructing the entire weight matrix but an adaptive linear scaling of the main LSTM weight matrix at each time step is achieved.

2.2. Model architectures for SED

Starting with a CRNN architecture, we use the presented hypernetwork to modify the weights of a CRNN architecture to introduce weight sharing relaxation to the recurrent part of the network. The CRNN model consists of 4 CNN blocks, followed by either a uni-directional or bi-directional LSTM block and a fully connected layer with sigmoid activation. Each CNN block comprises typical convolution operations with filter size 3×3, rectified linear activation operations, batch normalization and pooling operation with kernel size of 2×2. The number of channels increases across the CNN blocks in the sequence $(64, 128, 256, 512)$ and the input to the LSTM unit is a 512-dimensional tensor. The output is a tensor of size 256 in case of unidirectional LSTM and 512 in case of bidirectional LSTM.

The hidden-state size of hypernetworks $\hat{h}$ was varied between $\{64,\ 128\}$ but the size of output embedding z_k was fixed to a value of 8. For simplicity, we address the CRNN with hypernetwork as HCRNN and the original CRNN network as simply CRNN. Based on the hidden-state size of the hypernetwork, we identify the HCRNN with hidden size 64 as HCRNN-64 and similarly the hypernetwork with hidden size 128 is assigned the name HCRNN-128. A visual depiction of the hypernetwork is shown in Fig. 1.

To showcase the advantage of the proposed HCRNN, we use the original CRNN as the baseline in our experiments depicted in Section 3. The architecture of the CNN block for the CRNN baseline is identical to that of the HCRNN. The difference in the two networks was in the LSTM block. For the HCRNN, the main network consisted of a single LSTM layer with an input size of 512 and the hidden-state size of 256. The LSTM block of the CRNN baseline was varied between bidirectional (hidden-state size of 512) and unidirectional mode (hidden-state size of 256).

3. EXPERIMENTS

3.1. Datasets

We employed two datasets for our experiments: TUT-SED Synthetic 2016 [13] and Urban-SED [14]. TUT-SED Synthetic 2016 consists of 100 recordings synthesized using isolated instances of 16 sound event classes, namely alarms&sirens, baby crying, bird singing, bus, cat meowing, crowd applause, crowd cheering, dog barking, footsteps, glass smash, gun shot, horse walk, mixer, motorcycle, rain and thunder. The total duration of the dataset is 566 minutes. 60% of the dataset was used as training set, 20% as validation set and remaining 20% as test set. The dataset has strong annotations, implying that annotation of every audio recording consists of the sound event labels along with their onset and offset timestamps. The maximum polyphony of the dataset is 5. Of note, the dataset is not balanced with respect to the number of instances and the total duration of each event class.

The Urban-SED dataset is also a synthetic dataset consisting of 10,000 soundscapes with sound event annotations generated using Scaper [14], an open-source library for soundscape synthesis and augmentation. All recordings are of equal length (10 seconds) and were sampled at 44.1kHz. Similar to TUT-SED synthetic 2016, this dataset is also strongly labelled with each audio recording consisting of minimum 1 and maximum 9 sound events classes from the list: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren and street music with Brownian noise as the background sound. The Brownian noise is the same for all the audio recordings.

3.2. Metrics

In all experiments, we used the segment-wise F-score [17] for evaluation of the SED models. The segment-wise metrics compare the system output and ground truth in short time segments (1 second in our case) as opposed to frame-wise metrics where the prediction of each frame is compared to the ground truth to calculate the metric score. The class-wise F-score was computed on the segment level.

3.3. Feature extraction

Each audio file was transformed to a 96-band log-mel spectrogram with a window size of 1024 samples and hop length of 256 samples at a sampling rate of 22.5 kHz. The frequency range for the log-mel spectrogram was 50-11,025 Hz. All data samples were z-score normalized across the entire time-frequency representation using the mean and standard deviation of the training set. To deal with variable lengths of the input, the spectrograms were split into fixed-size chunks using a window length of 2 seconds and hop length of 0.5 seconds. We used the training/validation/test splits provided in the meta files of the datasets discussed in Section 3.1.

3.4. Experimental settings

All the network models described in Section 2.2 were trained under the same settings, including a maximum of 250 epochs with a batch size of 64, a learning rate of $4e^{-5}$ and early stopping with a patience of 10 epochs. Adam [18] with default parameters was used as optimizer for all the experiments.

Table 1. Class-wise and overall segment-wise F1 scores (%) obtained on the TUT-SED Synthetic 2016 dataset.

Category	Avg. dur. per event (s)	Total dur. (s)	Bi-baseline	Uni-baseline	HCRNN-128	HCRNN-64
Horsewalk	6.4	1,614	57.1	50.4	**64.9**	67.4
Rain	8.2	3,975	64.3	73.4	60.4	**74.8**
Bird_singing	6.1	2,298	**71.8**	64.7	71.7	52.9
Gunshot	1.7	534	**78.2**	64.0	71.8	67.8
Alarms&sirens	8.2	4,405	84.4	**87.2**	80.1	94.6
Crowd_cheering	8.1	4,825	63.2	61.9	**67.4**	67.0
Motorcycle	7.0	3,691	**56.5**	38.2	51.8	42.6
Mixer	7.9	4,020	77.3	47.5	**97.7**	91.0
Baby_crying	6.9	2,007	40.6	56.8	**58.8**	27.4
Footsteps	7.1	1,173	56.3	84.5	**85.3**	74.2
Crowd_applause	7.3	3,278	61.9	55.9	**74.4**	68.1
Dog_barking	5.0	716	72.5	23.9	65.6	**91.3**
Bus	7.8	3,464	41.4	49.7	**50.7**	38.1
Cat_meowing	2.1	941	31.5	**39.7**	34.7	23.9
Thunder	5.9	3,007	45.6	**50.1**	44.1	44.3
Glass_smash	1.2	621	72.1	**73.1**	59.1	68.4
Average	-	-	60.9	57.6	**64.9**	62.1
Overall	-	-	58.4	54.0	**62.7**	60.9

Table 2. Class-wise and overall segment-wise F1 scores (%) obtained on the Urban-SED dataset.

Category	Bi-baseline	Uni-baseline	HCRNN-128	HCRNN-64
Air_conditioner	48.9	48.8	**49.3**	44.9
Car_horn	75.2	70.5	**77.2**	64.1
Children_playing	52.5	49.3	**63.8**	61.6
Dog_bark	59.5	54.2	49.2	**63.4**
Drilling	**66.5**	54.3	55.1	61.1
Engine_idling	58.9	56.4	**71.6**	63.9
Gun_shot	64.2	67.1	68.2	**70.6**
Jackhammer	59.5	65.9	**66.9**	58.1
Siren	66.3	64.6	**71.0**	62.7
Street_music	**61.9**	54.1	58.5	62.2
Average	61.5	58.5	**63.1**	61.7
Overall	60.9	57.8	**62.3**	61.2

The trained models were evaluated on the test sets across 10 epochs and the mean of the results across the epochs was reported as the final score.

4. RESULTS AND DISCUSSION

Evaluation results across the test sets of TUT-SED-Synthetic 2016 and URBAN-SED datasets are presented in Tables 1 and 2, respectively. In the tables, Uni-baseline and Bi-baseline represent the unidirectional and bidirectional baseline, respectively. As can be seen, HCRNN-128 consistently outperforms the baselines across the selected datasets. A preliminary investigation of the results from Table 1 shows that HCRNN-128 achieved a significant improvement over the baseline for classes such as Horsewalk, crowd cheering,crowd applause and Mixer. A common thread across these classes is relatively long average duration per event occurrence and also the presence of a large number of samples as compared to other classes. In case of event classes with short average duration per occurrence (e.g., gunshot, glass_smash and cat meowing), the performance of the HCRNN-128 deteriorates as compared to the baselines. Since the dataset is highly imbalanced in terms of duration, a definitive trend of model performance cannot be confirmed across classes, although employing hypernetworks does improve the overall class-wise and overall segment-wise F-scores of the baselines.

In case of a class and duration balanced dataset like Urban-SED, HCRNN-128 improves upon the performance of the baseline networks across the majority of the classes as shown in Table 2. HCRNN-128 shows consistent improvement across all classes except *Dog _bark* over the unidirectional baseline and also across majority of classes in case of the bi-directional baseline, considering the fact that the bidirectional baseline has more trainable parameters as compared to HCRNN-128 and it also takes into account future information as opposed to the HCRNN-128, which is unidirectional.

The improvement in performance of the HCRNN due to the increase in the hidden state size from 64 to 128 is consistent across datasets and in line with the observation from [12], where increasing the hypernetwork hidden state size improved the results on different natural language processing tasks. One potential reason for this could be that since the concatenated input and the hidden vector of the main LSTM network are linearly projected to a lower dimension, an aggressive compression might lead to loss of critical information.

5. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel line of investigation into CRNN models for polyphonic SED. Through our preliminary experiments, we observed that a relaxed weight sharing mechanism led to an overall improvement in the performance of the baseline models, thus indicating that there might be a shift in the functional relationship between features and labels across the temporal axis.

While theoretically temporal conditional shift should be more common for short duration events, factors like polyphony level (which is high in the TUT-SED Synthetic 2016 database) and background noise, which are more prominent in longer duration events could also lead to temporal conditional shift. We would need to conduct more experiments to empirically ascertain the nature of temporal conditional shift in future work. One methodology to explore the presence of conditional shift could be enforcing an LSTM network to have different parameters for each time step. Although hypernetworks have the flexibility of modifying the weights of an LSTM, it is not confirmed that they use a different scaling vector for each time step, hence enforcing a condition to have a different scaling vector for each time step might provide more insights on the impact of hard weight sharing on the performance of a CRNN model.

6. REFERENCES

[1] Annamaria Mesaros, Toni Heittola, Tuomas Virtanen, and Mark D. Plumbley, "Sound event detection: A tutorial," *IEEE Signal Processing Magazine*, vol. 38, no. 5, pp. 67–83, 2021.

[2] An Dang, Toan H. Vu, and Jia-Ching Wang, "A survey of deep learning for polyphonic sound event detection," in *2017 International Conference on Orange Technologies (ICOT)*, 2017, pp. 75–78.

[3] Tuomas Virtanen, Mark Plumbley, and Dan Ellis, *Computational Analysis of Sound Scenes and Events*, 09 2017.

[4] Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen, "Recurrent neural networks for polyphonic sound event detection in real life recordings," in *Proc. ICASSP*, 2016, pp. 6440–6444.

[5] Sharath Adavanne, Pasi Pertilä, and Tuomas Virtanen, "Sound event detection using spatial features and convolutional recurrent neural network," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 771–775, 2017.

[6] Yuanbo Hou, Qiuqiang Kong, Shengchen Li, and Mark Plumbley, "Sound event detection with sequentially labelled data based on connectionist temporal classification and unsupervised clustering," in *Proc. ICASSP*, 2019.

[7] Qiuqiang Kong, Yong Xu, Wenwu Wang, and Mark D. Plumbley, "Sound event detection of weakly labelled data with cnn-transformer and automatic threshold optimization," 2020.

[8] Koichi Miyazaki, Tatsuya Komatsu, Tomoki Hayashi, Shinji Watanabe, Tomoki Toda, and K. Takeda, "Conformer-based sound event detection with semi-supervised learning and data augmentation," 2020.

[9] Zhirong Ye, Xiangdong Wang, Hong Liu, Yueliang Qian, Rui Tao, Long Yan, and Kazushige Ouchi, "Sound event detection transformer: An event-based end-to-end model for sound event detection," 2021.

[10] Jeeheh Oh, Jiaxuan Wang, Shengpu Tang, Michael W. Sjoding, and Jenna Wiens, "Relaxed parameter sharing: Effectively modeling time-varying relationships in clinical time-series," *CoRR*, vol. abs/1906.02898, 2019.

[11] Emre Çakır, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, 02 2017.

[12] David Ha, Andrew Dai, and Quoc V. Le, "Hypernetworks," in *Proc. ICLR*, 2017.

[13] "TUT-SED Synthetic 2016 dataset," `https://webpages.tuni.fi/arg/paper/taslp2017-crnn-sed/tut-sed-synthetic-2016`.

[14] Justin Salamon, Duncan Macconnell, Mark Cartwright, Peter Li, and Juan Bello, "Scaper: A library for soundscape synthesis and augmentation," in *Proc. WASPAA*, 2017, pp. 344–348.

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

[16] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[17] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, 2016.

[18] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd International Conference on Learning Representations (ICLR)*, 2015.