SOUND SOURCE DETECTION, LOCALIZATION AND CLASSIFICATION USING CONSECUTIVE ENSEMBLE OF CRNN MODELS

Technical Report

Sławomir Kapka

Samsung R&D Institute Poland
Data Intelligence
Warsaw, 00-844, Poland
s.kapka@samsung.com

Mateusz Lewandowski

Samsung R&D Institute Poland
Data Intelligence
Warsaw, 00-844, Poland
m.lewandows4@samsung.com

ABSTRACT

In this technical report, we describe our method for DCASE2019 task 3: Sound Event Localization and Detection. We use four CRNN SELDnet-like single output models which run in a consecutive manner to recover all possible information of occurring events. We decompose the SELD task into estimating number of active sources, estimating direction of arrival of a single source, estimating direction of arrival of the second source where the direction of the first one is known and a multi-label classification task. We use custom consecutive ensemble to predict events' onset, offset, direction of arrival and class. The proposed approach is evaluated on the development set of TAU Spatial Sound Events 2019 - Ambisonic.

***Index Terms*—** DCASE 2019, Sound Event Localization and Detection, CRNN, ambisonics

1. INTRODUCTION

Sound Event Localization and Detection (SELD) is a complex task which naturally appears when one wants to develop a system that posses spatial awareness of surrounding world using audio signals. SELDnet introduced in [1] is a good quality single system baseline designed for this task. In our work we follow the philosophy that if a complex task can be split into simpler ones, one should do so. Thus we decompose SELD task into following subtasks:

- estimating number of active sources (noas),
- estimating direction of arrival of a sound event when there is one active sound source (doa1),
- estimating direction of arrival of a sound event when there are two active sound sources and we posses the knowledge of direction of arrival of a one of this sound events (doa2),
- multi-label classification of sound events (class).

For each of this subtasks, we develop SELDnet-like convolutional recurrent neural network (CRNN) with a single output. We discuss more details on that in section 3. Given such models, we develop a custom consecutive ensemble of this models. This allows us to predict events' onset, offset, direction of arrival and class, what we discuss in details in section 4.

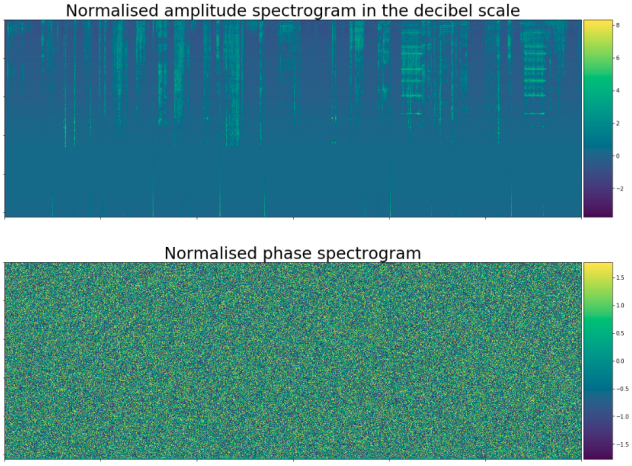

Figure 1: Spectrograms that are feed to networks.

2. FEATURES

DCASE 2019 task 3 [2] provides two formats of TAU Spatial Sound Events 2019 dataset: first order ambisonic (foa) and 4 channels from a microphone array (mic) [3]. In our method we only use ambisonic format.

Each recording is approximately 1 minute long with sampling rate of 48k. We use short time Fourier transform (STFT) with Hann window. We use window of length 0.4s and hop of length 0.2s in STFT to transform a raw audio associated to each foa channel into the complex spectrogram of size 3000x1024. If audio is longer then 1 minute, then we truncate spectrograms. If audio is shorter then 1 minute, then we pad it with zeros.

From each complex spectrogram we extract its module and phase point-wise, that is amplitude and phase spectrograms respectively. We transform amplitude spectrograms to the decibel scale. Finally, we standardize all spectrograms channel-wise and frequency-wise. We end up with spectrograms which looks like in Figure 1.

In summary, from each recording we acquire 4 standardized amplitude spectrograms in decibel scale and 4 standardized phase spectrograms corresponding to 4 foa channels.

	noas	doa1	doa2	class
Input	256x1024x4	128x1024x8	128x1024x8	128x1024x4
Rec	GRU	LSTM	GRU	GRU
Dense	16	128	128	16
Output	1, linear	3, linear	3, linear	11, sigmoid

Table 1: Hyperparameters of specific subtasks.

3. ARCHITECTURE

As mentioned in the introduction, each of the subtasks (noas, doa1, doa2 and class) has its own SELDnet-like CRNN. Each of these models is a mere copy of a single SELDnet node with just minor adjustments so that it fits to the specific subtask and also for the sake of regularization.

Each of these models takes as an inputs fixed length subsequence of decibel scale amplitude spectrograms (in case of noas and class subtasks) or both decibel scale amplitude and phase spectrograms (in case of doa1 and doa2 subtasks) from all 4 channels.

In each case input layers are followed by 3 convolutional layer blocks made of convolutional layer, batch norm, relu activation, maxpool and dropout (refer to right node form Figure 2).

The output from the last convolutional block is reshaped so that it forms a multivariate sequence of a fixed length. In the case of doa2, we additionaly concatenate directions of arrivals of associated events with this multivariate sequence. Next there are two recurrent layers (GRU or LSTM) with 128 units each with dropout and recurrent dropout. Next layer is a time distributed dense layer with dropout and with number of units depending on subtask (see Table 1).

Lastly, depending on a subtask, a model has a different output. For noas, the model has just a single time distributed output that corresponds to number of active sources (0, 1 or 2). For doa1 and doa2, the models has 3 time distributed outputs that corresponds to cartesian xyz coordinates as in [1]. Cartesian coordinates are advantageous over spherical coordinates in this task due to their continuity. Lastly, for class, the model has 11 time distributed outputs corresponding to 11 possible classes.

Depending on a subtask, we feed a network with the whole recordings or just thier parts. For noas subtask, we feed all the data. For doa1, we extract only those parts of the recordings where there were just one sound source active. For doa2, we extract only those parts of the recordings where there were exactly two active sound sources. For class, we extract those parts of the recordings where there were at least one active source.

The remaining details about hyperparameters are presented in Table 1. It is important to mention that in all models dropouts and recurrent dropouts are set to 0.2.

As for complexity, the noas, doa1, doa2 and class has respectively 572.129, 753.603, 591.555, 572.299 parameters making total of 2.651.634 parameters.

As for the learning process, for all subtasks we initialised learning process using Adam optimizer with default parameters [4]. Noas and class subtask were learned for 500 epochs with exponential learning rate decay. Every 5 epochs the learning rate were multiplied by 0.95. In doa1 and doa2 subtasks, we run learning process for 1000 epochs without changing learning rate.

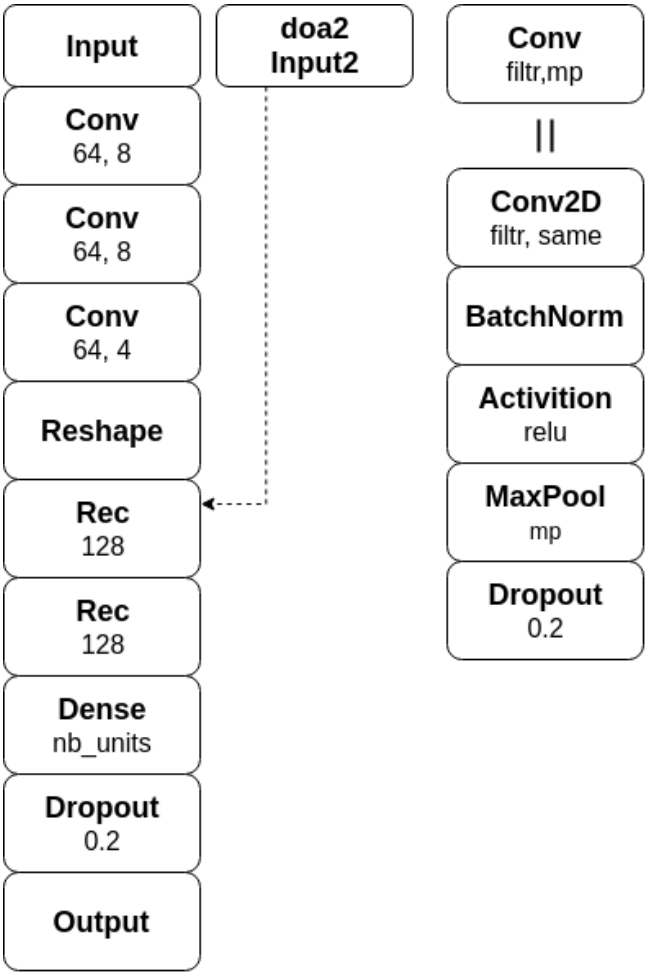

Figure 2: Architectures of networks for subtasks.

4. CONSECUTIVE ENSEMBLE

In this section, we describe the idea of consecutive ensemble which in our opinion is the heart of our approach. This custom binding of our four models allows us to predict events' onset, offset, direction of arrival and class.

We assume that recordings has at most 2 active sound sources at once and sound events occur on a 10 degrees resolution grid. For the sake of readability, we will describe our ensembling algorithm in the case of 1 minute long audios. In our setting such audios after feature extraction has exactly 3000 vectors corresponding to time dimension. Henceforth we will call this vectors as frames. The algorithm itself goes as follows:

1. We feed the features to noas network to predict number of active sources in each frame (see figure 3).

2. We "smooth" noas prediction so that each recording start and end with no sound sources and the difference of noas between each frames is no grater then 1 (like in figure 4).

3. Given noas predictions we deduce number of events, its onsets and the list of possible offsets for each event. If noas in a two consecutive frames increases then we predict that a new event happened at this frame. If in a two consecutive frames noas decreases

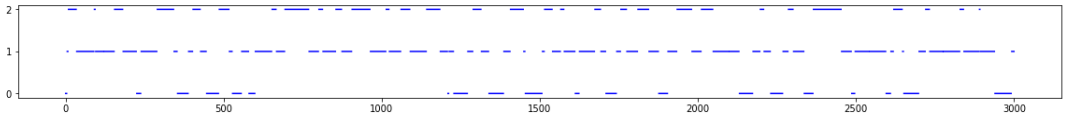

Figure 3: Predicting number of active sources.

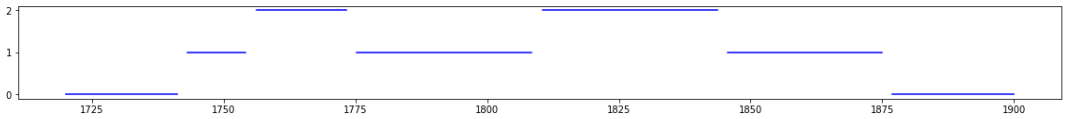

Figure 4: Smoothed out predicted noas.

then we append this frame to all events since last time noas were 0 as a possible offset. For example, consider the case like in the block from figure 5. In this block, 3 events happened E_1, E_2, E_3 with 3 corresponding onsets On_1, On_2, On_3. Events E_1 and E_2 may end at Off_1, Off_2 or Off_3 and event E_3 may end at Off_2 or Off_3.

4. In order to determine which offset corresponds to which event we use doa1 network. We extract chunks (intervals of equal noas) of audio where predicted noas equals 1 and we feed it to doa1 network. For each chunk where noas where 1 we predict the average azimuth, elevation and we round it to the closest multiple of 10 (see figure 5). If two consecutive chunks have the same azimuth and elevation then we conclude that the first event covered two chunks and the second event started and ended between those chunks. If two consecutive chunks have different azimuth or elevation then we conclude that the first event ended when the second chunk started and the second event continued in the second chunk (see figure 6).

5. To determine remaining information about angles we yet need to predict the direction of arrival of events that starts and ends while other event is happening (we call it associated event). We feed chunks where noas is 2 to doa2 network with second input beeing direction of arrival of associated event in cartesian xyz coordinates. Similarly as in step 4, we average the predicted results from chunks and round it to the nearest multiple of 10.

6. Lastly, we predict classes. If an event has chunks where it is happening in isolation (noas = 1), then all such chunks are feed to class network and the most frequent class is taken as a predicted class. If an event has no such chunks, i.e. it is only happening with an associated event, then such chunk (noas = 2) is feed to the network and two most probable classes are extracted. We choose the first one which does not equal to the class of the associated event.

5. RESULTS

We evaluate our approach on the development set of TAU Spatial Sound Events 2019 - Ambisonic using suggested cross-validation. We trained our models on 2 splits out of 4 for every fold even though validation spits does not influence the training process. We show in table 2 the averaged metrics from all folds for our setting and metrics for the baseline. In order to demonstrate the variance among folds, we present in table 3 detailed results on test splits from each fold.

The development set provides distinction for files where there is up to 1 active sound source at once (ov1) and where there are up to 2 (ov2). In table 4 we compare metrics for ov1 and ov2 subsets.

We conclude that our system outperforms the baseline and hence the approach of decomposing SELD task into simpler subtasks is promising. In particular, direction of arrival error looks very appealing due to the fact that models doa1 and doa2 learned estimating doa regardless of event. However, we are aware that some tricks suggested in our solutions fail when one wants to consider a more general setup. For example when there are more than 2 active sources at once or when the grid resolution is more refined.

	Error rate	F-score	DOA error	Frame recall	Seld score
Train	0.03	0.98	2.71	0.98	0.02
Val.	0.15	0.89	4.81	0.95	0.08
Test	0.14	0.90	4.75	0.95	0.08
Base	0.34	0.80	28.5	0.85	0.22

Table 2: Average results from all 4 splits.

	Error rate	F-score	DOA error	Frame recall	Seld score
Split 1	0.13	0.91	6.01	0.95	0.07
Split 2	0.16	0.88	6.01	0.95	0.09
Split 3	0.11	0.93	4.93	0.96	0.06
Split 4	0.17	0.86	5.89	0.96	0.10

Table 3: Results on test splits from each fold.

6. SUBMISSIONS

We created 4 submission for the competition:

- ConseqFOA (`Kapka_SRPOL_task3_2`)
- ConseqFOA1 (`Kapka_SRPOL_task3_3`)
- ConseqFOAb (`Kapka_SRPOL_task3_4`)
- MLDcT32019 (`Lewandowski_SRPOL_task3_1`)

The first three submissions use the approach described in the above sections. The only difference is that ConseqFOA is trained on all four splits from development dataset. ConseqFOA1 is trained on splits 2,3,4. ConseqFOAb is trained on all splits but the classifier in this version was trained using categorical crossentropy instead of binary crossentropy loss. The outcome of this version was very similar to ConseqFOA and hence we give it a try.

	Error rate	F-score	DOA error	Frame recall	Seld score
Ov 1	0.07	0.94	1.28	0.99	0.04
Ov 2	0.18	0.87	7.96	0.93	0.11

Table 4: Results on ov1 and ov2 subsets.

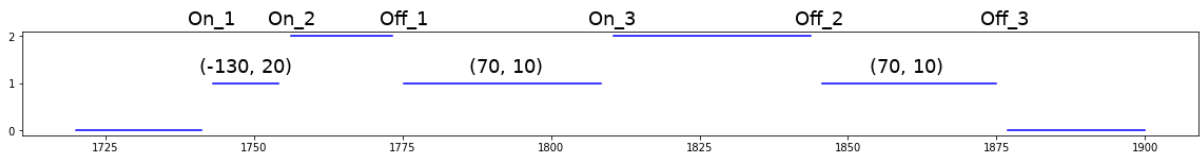

Figure 5: Onsets, offsets and direction of arrivals.

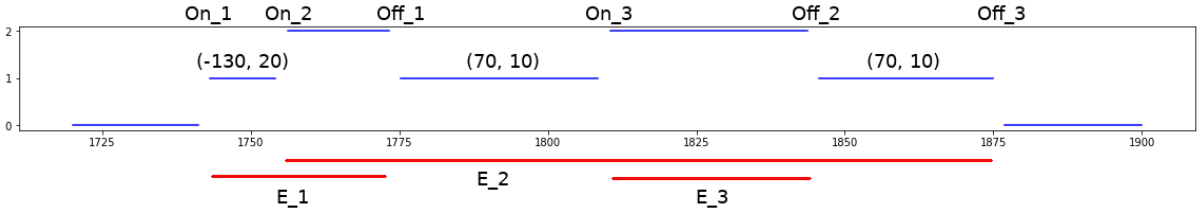

Figure 6: Deduced offsets using doa of single sources.

Submission MLDcT32019 uses different approach. It works in the same way as original SELDnet architecture but with the following differences:

- We implemented Squeeze-and-Excitation block [5] after the last convolutional block. We pass the output from the last convolutional block through two densely connected neural layers with respectively 1 and 4 neurons, we multiply it with the output of the last convolutional block and we pass it further to recurrent layers.
- We set all dropouts to 0.2.
- We used specAugment [6] as an augmentation technique to double the training dataset.
- We replaced recurrent layer GRU units with LSTM units.

7. REFERENCES

[1] S. Adavanne, A. Politis, J. Nikunen, and T. Virtanen, "Sound event localization and detection of overlapping sources using convolutional recurrent neural networks," *IEEE Journal of Selected Topics in Signal Processing*, pp. 1–1, 2018.

[2] http://dcase.community/challenge2019/task-sound-event-localization-and-detection.

[3] S. Adavanne, A. Politis, and T. Virtanen, "A multi-room reverberant dataset for sound event localization and uetection," in *Submitted to Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, 2019. [Online]. Available: https://arxiv.org/abs/1905.08546

[4] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[5] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018, pp. 7132–7141.

[6] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," 2019. [Online]. Available: https://arxiv.org/abs/1904.08779