A COMBINATION OF VARIOUS NEURAL NETWORKS FOR SOUND EVENT LOCALIZATION AND DETECTION

Technical Report

Daniel Rho[1*], *Seungjin Lee*[1*], *Jinhyeok Park*[1], *Taesoo Kim*[1], *Jiho Chang*[2], *and Jong Hwan Ko*[1],

[1] Sungkyunkwan University, Republic of Korea
{daniel03c195, lsg0385, tskim9439}@gmail.com, pjhsk1996@g.skku.edu, jhko@skku.edu
[2] Korea Research Institute of Standards and Science, Republic of Korea
jiho.chang@kriss.re.kr

ABSTRACT

This technical report describes our approach to the DCASE 2021 task 3: Sound Event Localization and Detection (SELD). We propose a network architecture, a combination of various network layers, which can yield the optimal performance for the SELD task. Furthermore, we propose which augmentation techniques to use to boost the performance of our proposed model with a limited train dataset. In order to further improve the performance, several techniques were applied at training and post-processing stages, such as adaptive gradient clipping, ensemble techniques, and class-wise dynamic thresholds. Evaluation results on the development dataset showed that the proposed approach outperformed the existing baseline model of the task.

Index Terms— Sound event localization and detection, neural architecture search, augmentations

1. INTRODUCTION

The task is to identify the sound source and estimate its direction. To tackle the issue, we followed the basic framework of the baseline [1, 2]. Segmented features are given as inputs and SED branch detects sounds for each class and frame. DOA branch predicts directions in cartesian format. Our approach was to use the same framework as the baseline[1] and only make progress on the model structure and training, post-processing techniques. To outperform the existing baseline model, we ran neural architecture search on this task and determined the model structures to further use to train on this task. Since available training samples were quite limited, many augmentation techniques should be adopted to defer overfitting. In the present study, several ensemble techniques and post-processing methods, such as dynamic thresholds were adopted to improve the performance.

2. PROPOSED METHOD

2.1. Input features

We followed the same feature extracting methods as the baseline [1]. In terms of recording formats, first-order ambisonic (FOA) was used for this work for two reasons. First, the baseline[1, 2] showed better performance on FOA format than on tetrahedral microphone array (MIC) format. Second, FOA has more ways to augment an audio spatially than MIC. Without disentangling sound sources, 16 transformations are applicable to FOA format, whereas only eight transformations to MIC format. This will be explained more precisely in 2.3.2.

First-order ambisonic (FOA) formatted audio files have seven channels: four log-mel spectrograms and three intensity vectors. We extracted features under the same setting as [3]; 64 mel-bands, a 40 ms window and 20 ms hop length at 24 kHz. To match the number of filters of intensity vectors to that of mel-spectrograms, mel-scale of 64 bands was applied to intensity vectors as in [3, 1]. Regarding normalization, we followed the same step as [3, 1].

2.2. Network architecture

Following the baseline of the task [1], the input shape of a spectrogram $D_{time}, D_{freq}, D_{chan}$ were set to 300, 64, and 7 respectively. As in [3], models were designed to predict the SED and DOA on different branches and the idea to use a single branch [1] was not taken.

Finding a suitable network structure for a particular task is not an easy task, especially when there are not many researches on optimal neural architectures on the task. Inspired by [2], we chose to search for optimal models in interpretable ways. After a few rounds of architecture search, we fixed the model structure which performed well on this task.

The final outcome consists of three main stages and SED, DOA branches. Before the first stage, a single convolutional layer and max pooling layer were used to match the number of label frames. More specifically, in order to use the same inputs and labels as the baseline [1], input shape of [300, 64, 7] were mapped to [60, D_{freq}, 32].

The first stage consists of convolutional layers and residual connections. Following equations show the exact structure of the stage.

$$\begin{aligned}&stage_1(x) = \beta(\alpha(x)),\\ &\alpha(x) = cat(act(bn(conv_0(x)) + bn(conv_1(x))), conv_2(x)),\\ &\beta(x) = cat(act(bn(conv_3(x)) + x), x),\end{aligned} \tag{1}$$

where bn denotes batch normalization[4] and act denotes activation function. For an activation function, ReLU [5] function was used. cat means concatenation along channel axis. $conv_1$ and $conv_2$ are strided convolutional layers with the kernel size of one

*Authors contributted equally to this work.

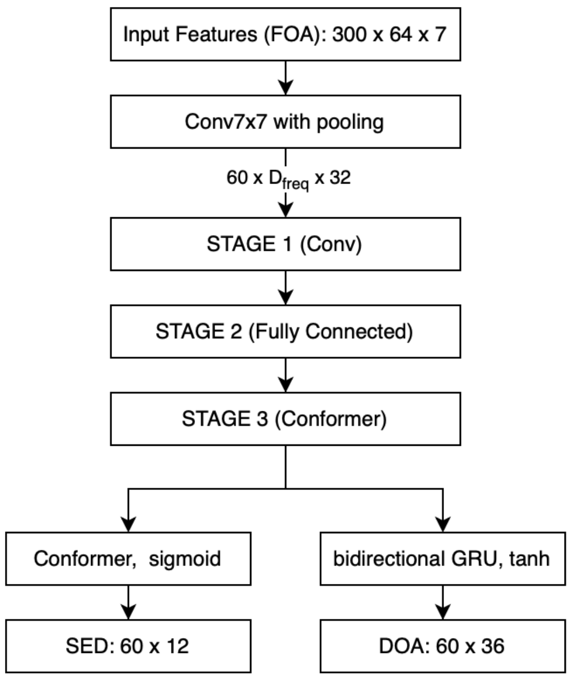

Figure 1: Proposed neural architecture

to match the output feature map size of that of $conv_0$. To prevent shrinkage in time-domain, strides were not set in time axis and only frequency dimension was allowed to set strides to a value, greater than one. $conv_0$ and $conv_3$ share the same number of channels and kernel size. The output of the first stage has the shape of $[D_{time}/5, D'_{freq}, D'_{chan}]$.

The second stage reshapes 2D outputs from the first layer to 1D outputs $[D_{time}/5, D'_{freq} * D'_{chan}]$ and applies a fully connected layer, which is equivalent to applying an one-dimensional convolutional layer with the kernel size of one. ReLU [5] function was used and no batch normalization [4] was used in the second stage.

The third stage is a stack of conformer encoder layers[6]. More precisely, it consists of two conformer encoder layers. Each layer has four attentional heads and the size of heads were set to 24.

For SED and DOA branches, a single conformer encoder layer and a stack of bidirectional gated recurrent units (GRU) [7] were used respectively. DOA branch was designed to output directions in cartesian formats as in [1]. SED branch has four attentional heads like the third stage, but the size has doubled. Two bidirectional GRU layers were used as a DOA branch. As in [3], sigmoid, hyper tangent functions were used respectively for SED, DOA branches.

2.3. Augmentations

Augmentation methods are essential when it comes to the performance, due to limited training samples. Simply scaling models leads to improvement only on train dataset, and not on validation nor test datasets. Furthermore, the challenge does not allow to use an external dataset in any ways. Therefore, we came up with as many augmentation techniques as possible to defer overfitting.

2.3.1. Masking

Among methods proposed in SpecAugment[8], augmentation by masking input features was adopted. SpecAugment[8] introduced two types of masks: time domain mask and frequency domain mask. In this work, only frequency domain mask was adopted, because time domain mask was found to degrade the performance on this task. We speculate that this degradation is due to the nature of the task [1]. Unlike automatic speech recognition task, for which [8] was proposed, detecting start-points and end-points of sound events are critical. Time-masking makes it harder to predict exact timing of sound events and this might have affected models in a negative way.

2.3.2. FOA domain spatial augmentation

To improve the task performance of the system, FOA domain spatial augmentation[9] was used to augment a limited train dataset. Among augmentation techniques, already used in audio related tasks, such as voice activity detection and automatic speech recognition, the spatial augmentation method[9] seems to be one of the simplest ways to spatially augment datasets to the best of our knowledge. As in [9], channels of input features were randomly swapped and theirs signs were randomly reversed to change the directions of sound sources. Since spectrograms and intensity vectors are correlated, they must be transformed equally.

Since disentangling multiple sound sources thoroughly in this task is impossible, applicable transformation on FOA format is limited. A transformation is applicable only if location of each sound source can change without distorting its sound, regardless of its locations. For example, changing the sign in x axis will affect directions of sound sources in the same way, regardless of their directions. For a FOA format, there are 48 applicable transformations. Nevertheless, only 16 transformations were selected[9], due to the fact that the official dataset for the task [1, 3] are known to have a limited range of elevation angle between -45° and 45°. Table 1 shows those 16 transformations.

(X, Y, Z)	(-X, Y, Z)	(X,-Y, Z)	(-X,-Y, Z)
(Y, X, Z)	(-Y, X, Z)	(Y,-X, Z)	(-Y,-X, Z)
(X, Y,-Z)	(-X, Y,-Z)	(X,-Y,-Z)	(-X,-Y,-Z)
(Y, X,-Z)	(-Y, X,-Z)	(Y,-X,-Z)	(-Y,-X,-Z)

Table 1: Combinations of dimensions and their signs

2.3.3. Random magnitude

The overall volume of an audio sample can be changed by simply adding a random scalar value to a given log-mel spectrogram. We varied the magnitude of the samples in two ways: one is to add a constant to change the overall volume of a sample and the other is to add different values for each frame. More specifically, two random scalars were selected for the first and the last frames. The values for other frames were calculated by linearly interpolating those two values. This array of values were added to log-mel spectrograms.

The first approach, adding a random constant, leads to better validation and test performance on the development dataset. The second approach further improved the performance than the first approach.

2.4. Ensemble methods

Ensemble methods are frequently used to boost the performance. These methods can be divided into two groups; intra-model level and inter-model level ensemble methods. In this work, both types of methods were used to improve the performance.

Among intra-model ensemble methods, two methods were used for the task. The first method is boosted deep neural network (bDNN) [10], which presented a way to aggregate intra-model predictions. The second method is stochastic weight averaging (SWA)[11], which averages the weights over the course of training steps to form the final weights.

To adopt the first method, as in [10], a model makes predictions on parts of an audio sequence and those predictions are aggregated to form a single prediction for the given audio sequence. More specifically, a model is given a window of 300 frames of inputs and the window slides with the stride of 5 frames. Overlapping windows creates overlapping predictions and these overlapped predictions were averaged to form a single prediction about the input sequence.

In terms of inter-model prediction aggregation, every model we made has a similar performance on the development dataset. Therefore, simple average predictions were used. Each model outputs a prediction per an audio sample and these outputs were averaged to form the final prediction.

2.5. Adaptive gradient clipping

The gradient clipping algorithm[12] is a way to clip gradients, used to update model weights. The algorithm clips gradients, so that the norm of gradients does not exceeds a hyperparameter λ. The equation for the algorithm for the gradient G can be expressed as follows:

$$G \to \begin{cases} \lambda \frac{G}{\|G\|} & if\ \|G\| \quad > \lambda \\ G & Otherwise \end{cases} \tag{2}$$

The clipping algorithm can reduce the instability during training process, especially when a learning rate is high. One drawback of the algorithm is its high sensitivity to λ. To overcome this issue, adaptive gradient clipping method [13] propose the following equation,

$$G_i^l \to \begin{bmatrix} \lambda \frac{\|W_i^l\|_F^*}{\|G_i^l\|_F} G_i^l & if\ \frac{\|G_i^l\|_F}{\|W_i^l\|_F^*} > \lambda \\ G_i^l & otherwise. \end{bmatrix}, \tag{3}$$

where $\|W_i^l\|_F^*$ equals max($\left\|W_i^l\right\|_F$, ϵ) and ϵ was set to 1e-3. As shown in the equation, [13] proposed two modifications. One is layer and node-wise gradient clipping. l specifies a layer and i specifies a node. The other modification is to use relative Frobenius norm of gradients to that of weights. W^l denotes weights of layer l. Throughout this work, λ was set to 0.02.

3. EXPERIMENTS

Two different model structures were used for this task. We trained several models using these two structures and selected a few of them to form ensemble models. The two model structures share the same model configuration and the only difference between them is whether pooling size in frequency axis in the first convolutional layer is one or two. The output channels and kernel size of the first stage, more precisely, those of $conv_0$ and $conv_3$ were set to 96 and 3. The strides of the first stage were set to one for the time axis and three for the frequency axis. The output channels of the second layer were set to 192.

AdaBelief optimizer [14] was used to train models with the learning rate of 0.001. The batch size was fixed to 256. For SED and DOA, binary cross entropy and masked mean squared error were used respectively[1]. As in [3], loss weights for SED, DOA branches were set to 1 and 1,000. Stochastic weight averaging [11] was applied after 80 epochs. Ensembles of three to four models were used for final results.

Models were trained under different hyperparameter settings, such as the magnitude of label smoothing [15] and λ in adaptive gradient clipping[12]. Five single models were selected among others, based on their test results. After that we selected three best ensemble models. Tables 2 and 3 shows the test results on the development set of single models and ensemble models respectively.

Models	ER	F	DER(%)	DERF	SELD
model 1	0.4369	0.6688	15.74	0.7214	0.2836
model 2	0.4132	0.6894	15.48	0.7297	0.2700
model 3	0.4317	0.6675	13.65	0.6849	0.2887
model 4	0.4381	0.6733	15.71	0.7257	0.2816
model 5	0.4471	0.6684	14.90	0.7209	0.2851

Table 2: Test results of single models on the development dataset

Models	ER	F	DER(%)	DERF	SELD
2+4+5	0.3895	0.7107	14.34	0.7252	0.2583
1+2+4+5	0.3829	0.7137	14.25	0.7249	0.2559
1+2+3+4+5	0.3843	0.7129	13.80	0.7140	0.2585
1+2+4+5 [1]	0.3807	0.7336	14.71	0.8101	0.2297

[1] means the final results after applying dynamic thresholds.

Table 3: Test results of ensemble models on the development dataset

4. CONCLUSION

We proposed a framework on DCASE2021 task3 which can be divided into three main parts. The first part is the model structure.The proposed combination of various types of layers has shown a great improvement on the development dataset. The second part is, adopted augmentation methods, such as frequency masking, spatial augmentation, and random magnitude which significantly deferred overfitting. Lastly, a number of inter and intra-model ensemble techniques have shown effectiveness in terms of task performance enhancement. Alongside the three main parts, by adopting many other techniques at training and post-processing stages, we were able to boost the performance even further and outperformed the baseline system on the development dataset.

5. ACKNOWLEDGMENT

This work was partly supported by the Institute of Information and Communication Technology Planning Evaluation (IITP) grant on AI Graduate School Program (IITP-2019-0-00421) and AI Industry Technology RD program (IITP-2021-0-00066), funded by the Korea government.

6. REFERENCES

[1] "Sound event localization and detection with directional interference." [Online]. Available: http://dcase.community/challenge2021/task-sound-event-localization-and-detection

[2] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 428–10 436.

[3] A. Politis, S. Adavanne, and T. Virtanen, "A dataset of reverberant spatial sound scenes with moving sources for sound event localization and detection," in *Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2020)*, November 2020. [Online]. Available: https://arxiv.org/abs/2006.01919

[4] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 448–456. [Online]. Available: http://proceedings.mlr.press/v37/ioffe15.html

[5] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines." in *ICML*, J. Fürnkranz and T. Joachims, Eds. Omnipress, 2010, pp. 807–814. [Online]. Available: http://dblp.uni-trier.de/db/conf/icml/icml2010.html#NairH10

[6] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Proc. Interspeech 2020*, 2020, pp. 5036–5040. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-3015

[7] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: https://www.aclweb.org/anthology/D14-1179

[8] 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," in *Proc. Interspeech 2019*, 2019, pp. 2613–2617. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-2680

[9] L. Mazzon, M. Yasuda, Y. Koizumi, and N. Harada, "Sound event localization and detection using foa domain spatial augmentation." [Online]. Available: https://dcase.community/documents/challenge2019/technical_reports/DCASE2019_MazzonYasuda_93.pdf

[10] X.-L. Zhang and D. Wang, "Boosting contextual information for deep neural network based voice activity detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 2, pp. 252–264, 2015.

[11] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," 2018.

[12] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1310–1318. [Online]. Available: http://proceedings.mlr.press/v28/pascanu13.html

[13] A. Brock, S. De, S. L. Smith, and K. Simonyan, "High-performance large-scale image recognition without normalization," 2021.

[14] J. Zhuang, T. Tang, Y. Ding, S. Tatikonda, N. Dvornek, X. Papademetris, and J. Duncan, "Adabelief optimizer: Adapting stepsizes by the belief in observed gradients," *Conference on Neural Information Processing Systems*, 2020.

[15] G. Pereyra, G. Tucker, J. Chorowski, L. Kaiser, and G. E. Hinton, "Regularizing neural networks by penalizing confident output distributions," *CoRR*, vol. abs/1701.06548, 2017. [Online]. Available: http://arxiv.org/abs/1701.06548