SOUND EVENT LOCALIZATION AND DETECTION USING CONVOLUTIONAL RECURRENT NEURAL NETWORK

Technical Report

Wen Jie Jee[1]*, *Rohith Mars*[2], *Pranay Pratik*[2], *Srikanth Nagisetty*[2], *Chong Soon Lim*[2],

[1] Nanyang Technological University, School of Physical and Mathematical Science, Singapore,
wjee001@e.ntu.edu.sg
[2] Panasonic R&D Center Singapore, Core Technology Group, Singapore,
{rohith.mars, pranay.pratik, srikanth.nagisetty, chongsoon.lim}@sg.panasonic.com

ABSTRACT

This report details the methods used in the development set of DCASE 2019 Task 3, and the results of the investigations. Data augmentation mixup was used in an attempt to train the model for greater generalization. The kernel size of the pooling layers were also modified to a more intuitive size. In addition, different kernel sizes of the convolutional layers were also investigated and results are reported. Our best model achieved an F-score of 91.9% and a DOA error of 4.588 °on the development set, which showed an improvement of 10% and about 25°, respectively compared to the baseline system.

Index Terms— sound localization, event detection, convolutional recurrent neural network, mixup, regression

1. INTRODUCTION

Sound event localization and detection (SELD) is a challenging and well-researched topic in the area of acoustic signal processing. Signal processing algorithms have been traditionally employed to address this challenging task. However, the performance achieved by such methods are still limited under practical conditions of room reverberation and low signal-to-noise ratio (SNR). To tackle these limitations, novel approaches using deep learning have been proposed recently. The technique used in [1] is one such deep learning-based approach and has been set as the baseline model in the DCASE2019 Task 3 challenge - Sound Event Localization & Detection.

An improvement in the overall performance of [1] was demonstrated in [2]. Both [1, 2] involved the use of convolutional neural network (CNN) to learn the audio spectral information followed by the use of recurrent neural network (RNN) to learn the temporal information. A two-stage training approach was introduced in [2], where the training of the network is split into two branches, i.e., the sound event detection (SED) branch and the direction-of-arrival (DOA) branch. For both branches, the system is trained using a convolutional recurrent neural network (CRNN). However, for training the DOA branch, the trained CNN weights obtained from the SED branch was transferred to the CNN layers in the DOA branch for further training.

In [1], short-time Fourier transform (STFT) representation of the microphone signals were used as input features to the CRNN. However, the introduction of log-mel spectrogram and generalized cross-correlation phase transforms (GCC-PHATs) as audio features in [2], showed an improvement across all evaluation metrics compared to [1]. In this technical report, we show the results we obtained from exploring various techniques aimed to improve the performance in [2] such as,

1. Use of Bark spectrogram as input audio feature
2. Use of data augmentation technique
3. CNN layer changes
4. Pooling layer changes .

The rest of the report is organized as follows. In Section 2, we describe our CRNN architecture and proposed methods and model-tuning parameters in detail. In Section 3, we compare and evaluate our results with [2]. Finally, we put forth possible areas for exploration in Section 4 and conclusions are provided in Section 5.

2. METHODOLOGY

For fair comparison and evaluation, the source code we developed is an extension of the code provided by [2] on GitHub [3]. The time domain microphone signals were first down-sampled to 32 kHz. To obtain the 2D log-mel spectrogram of the time domain signals, we used a STFT parameterized by FFT length of 1024 samples, hop length of 10 ms and Hann window. To convert the STFT representation to mel scale, we used 96 mel bins.

2.1. CRNN Architecture

Our main network architecture , named **TS-C2RNN**, shown in Figure 1 is a modification of the CRNN architecture proposed in [2]. Our modified architecture consists of 4 CNN blocks and 2 RNN layers stacked together. It should be noted that **TS-C2RNN** also employed the same two-stage training strategy in [2]. The output of the Interpolate layer contains 11 class scores, azimuth and elevation values corresponding to each T time frames, where T varies from clip to clip ($T = 6000$ for a 60 s clip using the above STFT parameters). The Interpolate layer ensures that the final number of the time frames is approximately equal to the original number of time frames of the input clip. This is necessary due to the presence of pooling layers after each CNN block.

Techniques mentioned in Section 1 was applied on the **TS-C2RNN** to investigate the effect they have on the final results. The following subsections further describe such techniques.

*The work was done when Jee was an intern at Panasonic R&D Center Singapore.

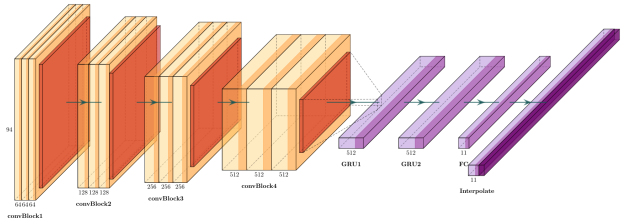

Figure 1: Base architecture **TS-C2RNN**. Each CNN block contains three Conv2D layers followed by 2x2 average pooling. Each CNN layer is followed by batch normalization and ReLU activation. convBlock1 receives the input features. Architecture is drawn with [4].

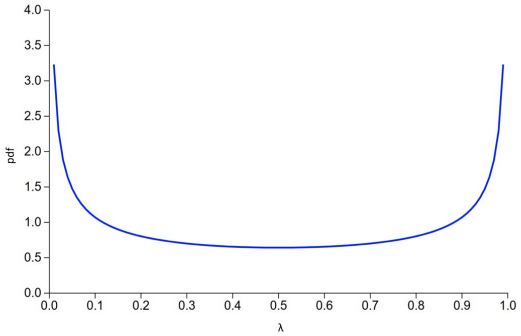

Figure 2: Beta distribution with $\alpha, \beta = 0.2$

2.2. Bark spectrogram

Previous works used mel spectrograms as input features. However, Bark spectrograms are also a type of audio representation based on psychoacoustic models. As such, we are interested to explore if there is any positive effect of using Bark spectrograms over mel spectrograms. FFT length, number of bins, sampling rate and GCC-PHATs as input features remained unchanged.

There are three different equations to convert frequency f to Bark based on different experiments. We used the latest conversion equation [5] which is given by,

$$Bark = 6sinh^{-1}\frac{f}{600}\,. \tag{1}$$

2.3. *mixup* as data augmentation

Ideally, the probability distribution P is known so that the expected risk R can be minimized over P as such,

$$R(f) = \int \ell(f(x), y)dP(x, y)\,,$$

where $f(x)$ is a function that describes the relationship between input vector x and target vector y, ℓ is the loss function that penalizes the difference between the output of $f(x)$ and target y.

While P is unknown is most practical cases, it can be approximated. There are two such approximations raised in [6] namely *empirical risk minimization* [7] and *vicinal risk minimization* [8].

While the vicinal virtual input-target pairs are generated by addition of Gaussian noise in [8], Zhang et al. [6] proposed the generation of virtual input and target pairs as such,

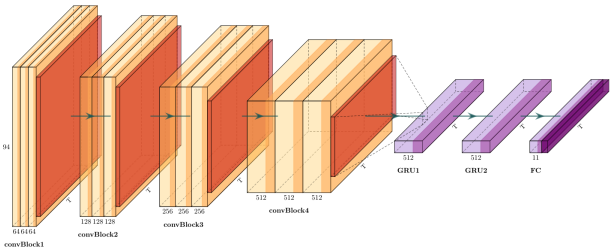

Figure 3: The modified architecture **TS-C2RNN-v2** with the new kernel sizes and without interpolation layer.

$$\begin{aligned} X &= \lambda \times x_1 + (1 - \lambda) \times x_2, \\ Y &= \lambda \times y_1 + (1 - \lambda) \times y_2, \end{aligned} \tag{2}$$

where λ is a weight drawn from the beta distribution with parameters $\alpha, \beta = 0.2$ and x_1, x_2, y_1 and y_2 are two pairs of input-target pairs drawn randomly from the dataset. The parameters α and β are chosen such that the probability density is denser in the domain $0 < \lambda < 0.1$ and $0.9 < \lambda < 1.0$ which can be seen in Figure 2. The average of the loss function can then be minimized over this probability distribution approximation.

2.4. CNN & pooling kernel size changes

Kernel sizes for the pooling layers were changed from 2×2 to 2×1 thereby pooling the feature maps in the frequency dimension only. Performing 2×2 pooling across time and frequency domain will reduce the information for the next CNN layer. To minimize the loss of information in the temporal dimension, which we deemed to be important, we did not perform pooling across the time dimension. As a consequence, when this method was adopted, the Interpolation layer in **TS-C2RNN** was removed. The modified architecture termed **TS-C2RNN-v2** with the new kernel sizes and without interpolation layer is shown in Figure 3. We also explored another modification where the CNN kernel size were changed from 3×3 to 4×2.

3. RESULTS

We used the dataset **TAU Spatial Sound Events 2019 - Microphone** provided by the challenge organizers for all of our experiments. The network is trained using the 4 predefined folds of the challenge and the final results shown in the tables are the overall results from the test data of all 4 folds in the development set.

The names in Table 1 refer to the trained model of a particular architecture. The names stand for the following,

- **Baseline** Model of the benchmark model released by DCASE2019 Task 3 organizers. Results of **Baseline** displayed are as reported by the authors in [1].
- **SELDNet** Model with the same architecture network in **Baseline**, but input features are log-mel spectrograms and GCC-PHATs. Results of **SELDNet** displayed in Table 1 were as reported by authors in [2].
- **Two-Stage (TS)** Model used in [2]. Results reported here were reproduced and is comparable to [2]. Improvements of about

	no *mixup*				with *mixup*			
	ER	**F-score**	**DOA(°)**	**FR**	**ER**	**F-score**	**DOA(°)**	**FR**
Baseline	0.350	0.800	30.8	0.840	-	-	-	-
SELDNet	0.213	0.879	11.3	0.847	-	-	-	-
Two-Stage (TS)	0.166	0.908	9.619	0.863	0.194	0.888	8.901	0.839
TS-2RNN	0.168	0.907	8.942	0.865	0.184	0.896	7.769	0.855
TS-CRNN	0.186	0.897	9.450	0.857	0.200	0.888	7.866	0.841
TS-C2RNN	0.174	0.901	8.881	0.862	0.176	0.903	7.236	0.856

Table 1: Results of all runs using two-stage training strategy. All TS results reported here are calculated after training and testing all folds predefined in the development set.

Investigation	**ER**	**F-score**	**DOA(°)**	**FR**
Bark spectrograms	0.193	0.889	9.021	0.847
Leaky ReLU	0.190	0.895	7.326	0.848
Maxpooling	0.251	0.868	5.516	0.853
Maxpooling + CNN kernel size	0.240	0.870	5.231	0.857
TS-C2RNN-v2	**0.149**	**0.919**	**4.588**	**0.896**

Table 2: Various results from the investigations into the changes proposed. All of these were ran with *mixup* data augmentation through **TS-C2RNN**.

	mixup on DOA branch only			
	ER	**F-score**	**DOA(°)**	**FR**
Two-Stage	0.175	0.903	8.056	0.861
TS-C2RNN	0.171	0.903	7.486	0.861

Table 3: Results from using *mixup* in DOA branch training only.

10% in F-score and 20° in DOA error from **Baseline** were obtained.

- **TS-2RNN** Same architecture as **Two-Stage** except for the single GRU layer being changed to a stacked two-layered GRU.
- **TS-C2RNN** Base architecture as illustrated in Figure 1.
- **TS-CRNN** Same architecture as **TS-C2RNN** except only 1 GRU layer used as the RNN.

The names in Table 2 refer to the various experiments performed on **TS-C2RNN**. They are denoted as follows,

- **Bark spectrograms** Change of input features from log-mel spectrograms and GCC-PHATs to Bark spectrograms and GCC-PHATs.
- **Leaky ReLU** Use of leaky ReLU instead of ReLU as activation function in the CNN layers.
- **Maxpooling** Use of max pooling layers instead of average pooling with kernel size 2×1.
- **Maxpooling + CNN kernel size** Use of max pooling layers instead of average pooling layers with kernel size 2×1. In addition, the kernel size for the CNN layers were changed from 3×3 to 2×4.
- **TS-C2RNN-v2** Changing of average pooling kernel size from 2×2 to 2×1. This is the modified architecture shown in Figure 3.

3.1. Bark spectrogram as input

Bark spectrograms as input features did not give a better result compared to log-mel spectrograms. The bandwidth in the mel scale is smaller than the Bark scale at higher frequencies giving the mel scale better resolution than the Bark scale at those frequencies. From this result, we may conclude that the higher frequencies play a major role in the SELD task and that frequency resolution plays a similar role as well.

3.2. Changing activation function to Leaky ReLU

By changing the activation function from ReLU to Leaky ReLU, it ensures that the function will not be given a zero value when it is negative, which can lead to dead neurons and the model not being able to learn. However, upon investigation, we notice that the choice of ReLU as an activation function seemed to work better for this task.

3.3. With and without *mixup*

From the results of **Two-Stage** in Table 1, we saw that the F-score decreased while the DOA error improved after applying *mixup*. From this, we proposed a new training strategy in an attempt to obtain improved results in both SED and DOA branches. This is done by applying *mixup* data augmentation technique only on the DOA branch during training.

Comparing between the results of **Two-Stage** and **TS-C2RNN** in Table 1 and 3, an improvement could be seen across all four evaluation metrics for **Two-Stage**. In contrast for **TS-C2RNN**, only three metrics showed an improvement while DOA error increased. Training for DOA branch is built upon the trained weights of the SED branch. For results in the DOA branch to improve, SED results are also essential. Although *mixup* appear to decrease the score for SED predictions, its effect on the SED branch, while affecting

it negatively, appears to have a positive result on the DOA branch after the knowledge is transferred to the DOA branch.

The negative effects of *mixup* on the SED branch appeared to be suppressed by increasing the number of layers. This can be seen from the F-score converging to 0.903 in the two scenarios of *mixup* applications in Table 1 and 3. The DOA error could be improved further by learning from the trained weights of a *mixup*-applied SED branch instead of a non-*mixup*-applied SED branch although that would adversely affect the results of SED predictions. Thus, a balance must be found in the use of *mixup*, depending on the use case and the allowance for error in SED and DOA predictions.

3.4. Changes to the pooling layers

The change of pooling type from average pooling to maxpooling shows a drop in the SED evaluations compared to **TS-C2RNN** but an improvement in the DOA evaluations. However, **TS-C2RNN-v2** which uses average pooling kernel size of 2×1 without interpolation shows the best improvements with the best score across all evaluation metrics. For the SED evaluation metrics, **TS-C2RNN-v2** achieved an error rate of 0.149 and an F-score of 91.9%. For the DOA evaluation metrics, it achieved a DOA error of 4.588 °and frame recall of 0.896. The improvement by not pooling across the temporal dimension is clear, as can be seen in both the max and average pooling cases. By not pooling across the time domain, important information is retained for the next layer.

4. FUTURE WORKS

For future studies, the changing of parameters α and β in *mixup* can be investigated. The parameters were so chosen so as not to create too many vastly different virtual input-target pairs. There might be a beneficial improvement if the λ is less heavily weighted to one side of the input-target pair.

Another possible branch for investigation is the parameters used for extracting features such as sampling rate and number of mel bins used. The audio could be sampled at a frequency of 8 kHz since there is not much audible information located in the high frequency domain. Instead of resampling, the use of a low-pass filter could also be used to remove audio in the higher frequency. Increasing the number of mel bins could also lead to higher SED score.

A preprocessing technique that had not be tried and could yield a slight improve in the results is the use of the pre-emphasis filter. Other scales beside the Bark and mel scale could also be tried such as the equivalent rectangular bandwidth (ERB) and gammatone scale.

5. CONCLUSION

In DCASE2019 task 3 challenge, we conducted various experiments such as the use of a data augmentation technique *mixup*, the use of a different input feature (Bark spectrogram), kernel size changes to the CNN layers as well as pooling layers and the employment of different types of pooling layer. From the evaluation metrics, we observed that the best result is achieved by **TS-C2RNN-v2** which uses a 2×1 pooling kernel size. Our proposed **TS-C2RNN-v2** achieved an improvement of about 10% and 25° in the F-score and DOA error, respectively over the baseline system.

6. 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*, vol. 13, no. 1, pp. 34–48, March 2019.

[2] Y. Cao, Q. Kong, T. Iqbal, F. An, W. Wang, and M. D. Plumbley, "Polyphonic sound event detection and localization using a two-stage strategy," *arXiv preprint arXiv:1905.00268*, 2019. [Online]. Available: https://arxiv.org/pdf/1905.00268.pdf

[3] "Polyphonic-sound-event-detection-and-localization-using-a-two-stage-strategy," https://github.com/yinkalario/Two-Stage-Polyphonic-Sound-Event-Detection-and-Localization.

[4] "Latex code for making neural networks diagrams," https://github.com/HarisIqbal88/PlotNeuralNet.

[5] S. Wang, A. Sekey, and A. Gersho, "An objective measure for predicting subjective quality of speech coders," *IEEE Journal on Selected Areas in Communications*, vol. 10, pp. 819–829, 1992.

[6] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[7] V. N. Vapnik, "Statistical learning theory," vol. 1, 1998.

[8] O. Chapelle, J. Weston, L. Bottou, and V. Vapnik, "Vicinal risk minimization," in *Advances in neural information processing systems*, 2001, pp. 416–422.