THUEE SYSTEM FOR DCASE 2019 CHALLENGE TASK 2

Technical Report

Kexin He *, *Yuhan Shen* *, *Wei-Qiang Zhang* †

Department of Electronic Engineering, Tsinghua University, Bejing 100084, China
hekexinchn@163.com, yhshen@hotmail.com, wqzhang@tsinghua.edu.cn

ABSTRACT

In this report, we described our submission for the task 2 of Detection and Classification of Acoustic Scenes and Events (DCASE) 2019 Challenge: *Audio tagging with noisy labels and minimal supervision*. Our methods are mainly based on two types of deep learning models: Convolutional Recurrent Neural Network (CRNN) and DenseNet. In order to prevent overfitting, we adopted data augmentation using mixup strategy and SpecAugment. Besides, we designed a staged loss function to train our models using both curated and noisy data. We also used various acoustic features, including log-mel energies and perceptual Constant-Q transform (p-CQT), and tried an ensemble of multiple subsystems to enhance the generalization capability of our system. Our final system achieved a lwlrap score of 0.742 on the public leaderboard in *Kaggle*.

***Index Terms*—** Audio tagging, noisy label, model ensemble, DCASE

1. INTRODUCTION

The Detection and Classification of Acoustic Scenes and Events (D-CASE) Challenge is gaining increasing interests among researchers with academic and industrial backgrounds. DCASE 2019 is the fifth edition of this challenge and has been held to support the development of computational scene and event analysis methods. This report describes the methods we used to participate in the task 2 of DCASE 2019 Challenge.

The second task of this year's challenge is *Audio tagging with noisy labels and minimal supervision* [1]. It provides public dataset [2] with baseline and aims to develop competitive audio classification systems using a small set of manually-labeled data and a larger set of noisy-labeled data.

State-of-the-art methods are based on deep neural networks, including Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and Convolutional Recurrent Neural Network (CRNN). We followed this trend and used two types of neural network architectures: a CRNN and a variant of CNN (DenseNet).

Data augmentation has been widely utilized in recent DCASE Challenges. Mixup [3] strategy has been adopted by top teams [4, 5] in DCASE 2018 Challenge. Besides, a new augmentation method named SpecAugment [6] has been proposed recently. In our work, we used the combination of both methods.

Besides, how to use noisy data is the key to tackling this task. We designed a staged loss function to select the most convincing samples from noisy data and train our model using both verified data and convincing unverified data. This training strategy will be illustrated in the following sections.

In addition, we also did some explorations about post-processing and found an effective way of score normalization.

The rest of this report is organized as follows: in Section 2 we describe our methods in detail; we present our experiments and results in Section 3; finally we conclude our work in Section 4.

2. METHODS

2.1. Feature extraction

We used two types of acoustic features in our work, including log-mel energies and perceptual Constant-Q transform (p-CQT). And we also used different parameters, such as frame length, hop length, frequency range, mel bins. As shown in Table 1, we used three feature configurations in total. All features are extracted using librosa [7].

Table 1: Configurations of acoustic features

	Type A	Type B	Type C
Feature	log-mel	log-mel	CQT
Window length	1764	2048	—
Hop length	882	511	512
Low Frequency	0 Hz	0 Hz	55 Hz
High Frequency	22050 Hz	16000 Hz	—
Feature dim	80	128	128
bins per octave	—	—	16

2.2. Data preprocessing

Raw feature data needs further preprocessing before being input to neural networks. The data preprocessing procedures used in our work include sound activity detection (SAD) and data padding.

SAD has shown powerful performance in previous work. We mainly used two kinds of SAD methods: 1) We ignore the silent frames at the beginning and end of each audio. 2) We ignore the silent frames through the whole audio.

In order to deal with the variable lengths of feature presentations, we set a maximum padding length. All shorter feature presentations will be repeated to the padding length. When it is longer, it will be downsampled to align with the padding length.

---

*The first two authors contributed equally.

†This work was supported by the National Natural Science Foundation of China under Grant No. U1836219. The corresponding author is Wei-Qiang Zhang.

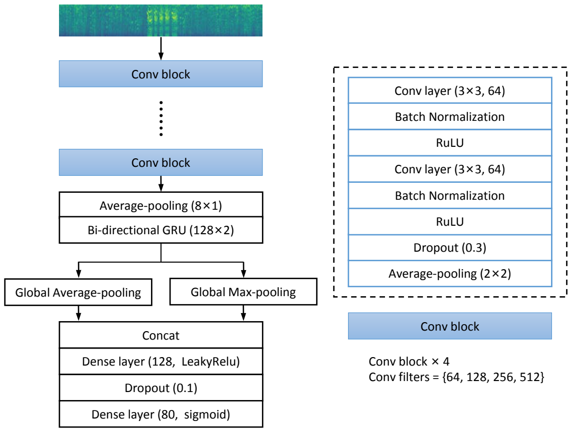

Figure 1: The architecture of CRNN. It consists of 4 convolutional blocks, a bi-GRU and two dense layers. Each convolutional block contains two convolutional layers. After bi-GRU, global average pooling and global max pooling operations are applied to aggregate temporal information, and the results are concatenated together before being input to dense layers.

In our work, the padding length is set 2000. During training, we randomly select continuous 512 frames to feed into the neural network. For test, the whole 2000 frames are used to get predictions.

2.3. Data augmentation

As mentioned above, we combined mixup [3] and SpecAugment [6] for data augmentation.

In mixup, we randomly select a pair of samples from training data. Let x_1, x_2 be the features, and y_1, y_2 be the one-hot labels respectively, the data is mixed as follows:

$$x = \lambda x_1 + (1 - \lambda)x_2 \tag{1}$$

$$y = \lambda y_1 + (1 - \lambda)y_2 \tag{2}$$

where the parameter λ is a random variable with Beta distribution B(0.4, 0.4).

SpecAugment is implemented by time warping, frequency masking and time masking. Detail is available in [6].

2.4. Neural network

2.4.1. CRNN architecture

The architecture of CRNN is illustrated in Figure 1. It begins with four convolutional blocks. Each block contains two convolutional layers, followed by batch normalization, ReLU, dropout and average pooling. Next, an average pooling is adopted on frequency axis to squeeze the frequency dimension to 1. And a bi-directional gated recurrent unit (Bi-GRU) is used to capture temporal context. Then, global max pooling and global average pooling are used on time axis to maintain various information and concatenated together. Finally, two dense layers are applied to output prediction scores for each class.

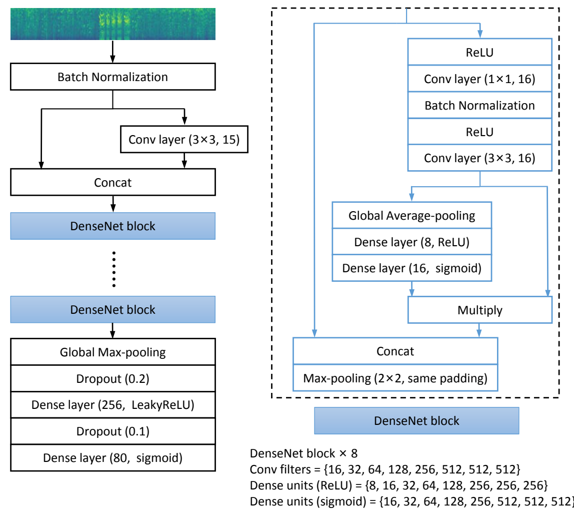

Figure 2: The architecture of DenseNet. Batch normalization is applied to the input acoustic feature, followed by a convolutional layer. The input and output of this convolutional layer are concatenated along channels, followed by 8 DenseNet blocks. Then, global max pooling is applied, and two dense layers are utilized to output final predictions. The configuration of DenseNet block is illustrated in the dotted box.

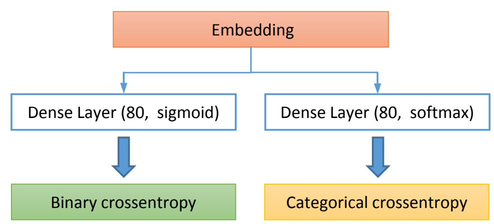

Figure 3: Combination of sigmoid and softmax activation.

2.4.2. DenseNet architecture

Shown in Figure 2 is the architecture of DenseNet. Our module is similar to that in [4]. In this module, the feature maps of previous layers can propagate to later layers, which can effectively alleviate the vanishing-gradient problem and encourage feature reuse. In each DenseNet block, we use Squeeze-and-Excitation Network [8], which can adaptively recalibrate channel-wise feature responses by explicitly modelling interdependencies between channels.

2.4.3. Choice of final activation

Since this is a multi-label and multi-class classification task, sigmoid is naturally the primary choice of the activation in final layer. However, as shown in Table 2, the average number of positive labels in training dataset is very close to 1, and single-label data takes up the majority. So softmax is also a good option.

In order to combine the advantages of both sigmoid and softmax, we design a new structure, in which the output embedding before the final layer will pass through two dense layers. As shown in Figure 3, one dense layer with sigmoid activation will be optimized with binary crossentropy loss, and the other dense layer with softmax activation will be optimized with categorical crossentropy loss. The outputs of both dense layers are ensembled to get final prediction.

Table 2: The number of positive labels in training dataset

#positive labels	train curated	train noisy
1	4269	16566
2	627	2558
3	69	504
4	4	141
5	0	38
6	1	4
7	0	4
average #positive labels	1.157	1.211
percentage of single label	85.9%	83.6%

2.5. Staged loss function for noisy data

In this task, only a small amount of data is manually labeled, and a large quantity of data contains noisy label. Since the noisy data is not verified to have groundtruth label, we try to use only the most convincing noisy data. Inspired by the batch-wise loss masking in [4], we propose a staged loss function to learn from noisy data.

To make it specific, we firstly use the verified data to train our system for several epochs. Then, we use both the verified and unverified data. However, in order to use only the most convincing noisy data, we adopt a loss masking similar to the work in [4]. The difference is that we ignore the noisy samples with the top *k* loss in a batch rather than set a threshold value and ignore samples with higher loss. Finally, after training for more epochs, we abandon the noisy data and finetune our model with only the verified data. Our staged training strategy has made huge improvements according to our experiments.

2.6. Post processing

For inference, we use score normalization strategy for further improvements. Let $x_{i,j}$ be the prediction score for the i-th class in the j-th sample. We normalize the prediction scores for each class. The normalization procedure goes as follows:

$$\overline{x}_i = \frac{\sum_{j=1}^{N} x_{i,j}}{N}, \tag{3}$$

$$\hat{x}_{i,j} = \frac{x_{i,j} - \overline{x}_i}{\sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(x_{i,j} - \overline{x}_i\right)^2 + \varepsilon}} \tag{4}$$

$$\widetilde{x}_{i,j} = \frac{\hat{x}_{i,j} - \min_j \hat{x}_{i,j}}{\max_j \hat{x}_{i,j} - \min_j \hat{x}_{i,j} + \varepsilon} \tag{5}$$

where N is the total number of samples in evaluation dataset, ε is a sufficiently small value to avoid division by zero. For each class in evaluation dataset, we normalize the predictions scores to zero mean and unit variance. Then, we set min and max zoom to keep the scores between 0 and 1. According to experimental results, score normalization can raise the evaluation score by approximately 0.002 on average.

3. EXPERIMENTS

3.1. Experiment setup

Adam [9] is used for gradient based optimization. The learning rate is 0.001 and batch size is 64. We split our training dataset into four folds. Then we train four models using any three folds for training and the other fold for validation.

As for the staged training, we design a data generator to generate different proportions of training data during different stages. In the first stage, all data comes from curated dataset. In the second stage, the proportion of curated dataset is equal to noisy dataset. In the third stage, only curated dataset is used. In the second stage, the top *k* samples with the highest loss on noisy dataset would be masked. In our experiments, *k* is 10.

For CRNN architecture, the first stage runs for 8k iterations, the second stage runs for 12k iterations, and the third stage runs for 3k iterations. For DenseNet architecture, the first stage runs for 5k iterations, the second stage runs for 8k iterations, and the third stage runs for 2k iterations. The models with the best validation performance on each fold are selected.

3.2. Evaluation metric

The primary competition metric is label-weighted label-ranking average precision (lwlrap). This measures the average precision of retrieving a ranked list of relevant labels for each test clip (i.e., the system ranks all the available labels, then the precisions of the ranked lists down to each true label are averaged). LRAP is calculated as follows, and lwlrap is the macro-average of per-class LRAP. [10]

$$\mathrm{LRAP}(y, \hat{f}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} \frac{1}{\|y_i\|_0} \sum_{j:y_{ij}=1} \frac{|\mathcal{L}_{ij}|}{\mathrm{rank}_{ij}} \tag{6}$$

where $\mathcal{L}_{ij} = \left\{k : y_{ik} = 1, \hat{f}_{ik} \geq \hat{f}_{ij}\right\}$, $\mathrm{rank}_{ij} = \left|\left\{k : \hat{f}_{ik} \geq \hat{f}_{ij}\right\}\right|$, $|\cdot|$ computes the cardinality of the set, and $\|\cdot\|_0$ computes the number of nonzero elements in a vector.

3.3. Model ensemble and submissions

Model ensemble is successful in boosting the system's performance according to previous work. We ensemble our models using geometric average as follows:

$$y_{\text{ensemble}} = \exp \frac{1}{N} \sum_n w_n \log y_n \tag{7}$$

where N is the number of subsystems, y_n is the output score of each subsystem, and w_n is the weight coefficient for each subsystem.

We submitted two prediction results using different weights:

1) Zhang_THU_task2_1.output.csv: achieved our highest lwlrap score of 0.742 on public leaderboard in *Kaggle*.

2) Zhang_THU_task2_2.output.csv: achieved our highest local lwlrap scores in each cross-fold validation, with a lwlrap score of 0.739 on public leaderboard in *Kaggle*.

4. CONCLUSION

In this report, we present our methods and techniques used in the task 2 of DCASE 2019 Challenge. We adopted mixup and SpecAugment for data augmentation. We applied two types of deep learning model including CRNN and DenseNet. Besides, a staged loss function is applied to learn from both curated and noisy data. Our final system achieved a lwlrap score of 0.742 on the public leaderboard in *Kaggle*.

5. REFERENCES

[1] E. Fonseca, M. Plakal, F. Font, D. P. Ellis, and X. Serra, "Audio tagging with noisy labels and minimal supervision," *arXiv preprint arXiv:1906.02975*, 2019.

[2] E. Fonseca, J. Pons Puig, X. Favory, F. Font Corbera, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra, "Freesound datasets: a platform for the creation of open audio datasets," in *Proceedings of the 18th ISMIR Conference; 2017 oct 23-27; Suzhou, China.[Canada]: International Society for Music Information Retrieval; 2017. p. 486-93.* International Society for Music Information Retrieval (ISMIR), 2017.

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

[4] I.-Y. Jeong and H. Lim, "Audio tagging system for DCASE 2018: focusing on label noise, data augmentation and its efficient learning," DCASE2018 Challenge, Tech. Rep., 2018.

[5] T. Iqbal, Q. Kong, M. D. Plumbley, and W. Wang, "Stacked convolutional neural networks for general-purpose audio tagging," *Tech. Rep., DCASE Challenge*, 2018.

[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," *arXiv preprint arXiv:1904.08779*, 2019.

[7] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.

[8] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[10] https://www.kaggle.com/c/freesound-audio-tagging-2019/overview/evaluation.