AUDIO TAGGING SYSTEM FOR DCASE 2018: FOCUSING ON LABEL NOISE, DATA AUGMENTATION AND ITS EFFICIENT LEARNING

Technical Report

Il-Young Jeong and Hyungui Lim

Cochlear.ai, Seoul, Korea
{iyjeong, hglim}@cochlear.ai

ABSTRACT

In this technical report, we expound on the techniques and models applied to our submission for DCASE 2018: *General-purpose audio tagging of Freesound content with AudioSet labels*. We aim to focus primarily on how to train a deep-learning model efficiently against strong augmentation and label noise. First, we conducted a single-block DenseNet architecture and multi-head softmax classifier for efficient learning with mixup augmentation. For the label noise, we tried the batch-wise loss masking, which eliminates the loss of outliers in a mini-batch. We also tried an ensemble of various models, trained by using different sampling rate or audio representation.

Index Terms— Audio event detection, Convolutional Neural Networks

1. PROPOSED FRAMEWORK

Herein, we present a framework of our submission for DCASE2018. For the detailed information about the dataset and the challenge, please refer [1].

1.1. Preprocessing and batch generation

No preprocessing step was applied in the presented frameworks except data resampling. We tried some other techniques including silence removal, pre-emphasis filtering but we could not find meaningful improvements. In case of data resampling, we tried 16kHz, 32kHz and 44.1kHz (original data). Low sampling rate may lose useful information in high frequency, but it will be easier to analyze longer temporal range since data size goes smaller.

We conduct the batch generation framework as follows. At first, we set each batch to have the same number of classes. In this work, one batch has one data for each class, thus the batch size was 41. We expected that this helps the optimization procedure to be stable and fast.

The length of audio recordings in the dataset is varied from few hundreds milliseconds to about 30 seconds. For efficient mini-batch learning, we fixed the length of data in each batch to be 64,000. It is equivalent to 4s in case of 16kHz data, and shorter for the data with higher sampling rate. If the original recording is longer than this, segment of 64,000 samples was extracted from the random offset. If it has shorter length, on the other hand, we applied zero-padding to the beginning and end of data.

1.1.1. Mixup augmentation

Mixup is an augmentation method which linearly mix two training data [2]. Let x_i and y_i are i-th raw input data in training dataset and its corresponding binary label, respectively, then mixup generates an augmented data $\hat{x}$ which are the mixture of two original data as follows:

$$\hat{x} = \lambda x_j + (1 - \lambda) x_k, \tag{1}$$

where $\lambda \in (0, 1)$. Similarly, the label of generated data is set to be $\hat{y} = \lambda y_j + (1 - \lambda) y_k$. Despite its simplicity, mixup shows meaningful improvements in image classification tasks.

We believe that mixup technique is also, or more, appropriate for audio analysis, since an audio signal captured in real-world can be considered as a linear mixture of various 'source' signals. In this perspective, classify $\hat{x}$ to $\hat{y}$ could be considered as a task which detects multiple sound events occurring simultaneously.

In this work, we set λ to be random variable of Beta(0.4, 0.4). In addition, we set $\lambda > 0.5$, so the data of target class, which is evenly distributed in a batch, is always predominant in the generated data. The class of another data for mixup is randomly selected. Finally, we also apply scale augmentation, which randomly scales the data. This process can be represented as a following equation.

$$\hat{x} = w\lambda x_j / \max(|x_j|) + w(1 - \lambda) x_k / \max(|x_k|), \tag{2}$$

where w is random variable with uniform distribution for the scale augmentation.

1.2. Model architecture

While mixup technique meaningfully prevents overfitting and increases validation/test accuracy, we also found that it makes the minimization of training loss difficult. Therefore, our model and learning strategy is focusing on the training efficiency against strong mixup augmentation.

The full architecture for our model is presented in Fig. 1, and Fig. 2 shows the details of each module in the model.

We tried 2 different models, which are 'logmel-based' and 'waveform-based'. The figures in the paper represent the logmel-based model and the minor changes to the waveform-based model are described individually in each subsection.

1.2.1. Low-level module

A logarithm of mel-scale spectrogram (logmel) is widely used preprocessing step in audio signal analysis. In this work, we applied

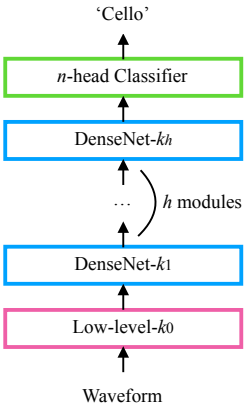


Figure 1: Overall architecture for the model in our submission. The details of each module is represented in Fig. 2

logmel transform as a low-level module in our model by using kapre [3].

Detailed low-level module is described in Fig. 2 (a). First, input waveform is normalized by using Batch normalization (BN) [4], then transformed into logmel domain, which has two dimensions of time and frequency. After applying BN for each frequency, it is reshaped to have the size of (time, frequency, 1) so it can be considered as a grayscale image. As described in the next subsection, we aimed to conduct a single-block densely-connected architecture, so output features of convolution layer is concatenated with its inputs.

In case of waveform-domain model, the preprocessing module is simplified and modified as follows:

- logmel and BN+Reshape layer is removed and input data after BN is directly concatenated to Conv outputs.
- 3x3 Conv layer is replaced to 1x3 Conv.

1.2.2. DenseNet Module

Numerous previous studies have tried to overcome the difficulty in training the deep architecture by applying the short-path from lower layers to higher ones. For example, in highway networks, some layer input is passed to layer output depending on the gate function [5]. In case of the residual networks (Resnet), on the other hand, layer input is directly added to layer output [6]. In densely-connected networks (DenseNet), layer input is also directly transfered to next layer as Resnet, it is concatenated with layer inputs [7].

We designed our model based on DenseNet. Although the original DenseNet model divided its architecture into several blocks and applied densely-connected layer within each block, our model consist of a single block architecture so the very first logmel or waveform can be reached even to the very last layer. In our experiments, increasing the number of blocks in models which means the number of layer that disconnects the concatenation makes the training slower.

Fig. 2 (b) shows the details of DenseNet module. Since the filter size of layer output keeps increasing due to concatenation, it is first reduced by using 1x1 convolution, and 3x3 convolution is performed. We also applied Squeeze-and-Excitation Network [8], which is expected to help efficient training by adding a few more parameters. 2x2 max pooling is applied for the last layer of each DenseNet module.

In case of waveform-based model, it is modified as follows:

- 3x3 convolution is replaced to 1x3 convolution.
- 2x2 max pooling is replaced to 1x2 max pooling.

1.2.3. Classifier module

In general, the goal of classification task is to predict the target label which is binary i.e. [1, 0, 0]. When mixup is applied, on the other hand, it needs to predict the real values in the range of (0, 1) i.e. [0.9, 0.1, 0] or [0.7, 0.3, 0]. The stronger mixup is applied, the more target values tend to be close to 0.5.

To train the mixup model efficiently, we modify the conventional softmax output layer to have a multi-head architecture, where output is obtained by averaging multiple softmax outputs as Fig. 2 (c).

We expect it will be helpful especially for training with strong mixup augmentation because of following reason. Since the output range of softmax layer is bounded to (0, 1), all the softmax have to be 1 to make their average to be 1. When the target value is in the range of (0, 1), each softmax output is allowed to have some error if those average can predict the exact target value. More margin is allowed when the target is closed to 0.5 because the softmax outputs in the range of (0, 1) can have larger variance. In our experiments with various n-multi-head settings, we found larger n helps to accelerates the training procedure, while there was no improvement in terms of maximum validation accuracy.

1.2.4. Overall frameworks

Our entire model is conducted by using abovementioned modules. For the logmel- and waveform-based model, the detailed parameters for modules are as follows.

- Logmel-based: Low-level-15, 8 DenseNet modules of k=(16, 32, 64, 128, 256, 512, 512, 512), 8-head Classifier. About 11M trainable parameters (except fixed parameters in logmel layer)
- Waveform-based: Low-level-1, 15 DenseNet modules of k=(2, 4, 8, 16, 32, 64, 128, 256, 512, $\cdots$, 512), 8-head Classifier. About 16M trainable parameters.

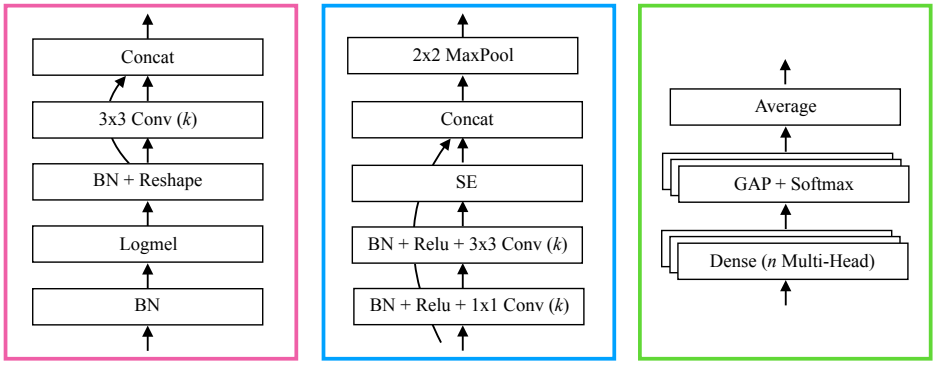
1.3. Optimization

Training procedure was done by using keras [9]. Adam was used for optimization [10]. Although it adaptively controls the learning rate (lr) by itself, we found that manual decaying the learning rate helps the optimization even for Adam. We set lr to be 1e-3 for first 150k mini-batch iteration, 1e-4 for next 100k and 1e-5 for the last 50k. Note that the actual learning rate for each minibatch is based on this lr parameter and adaptation algorithm of Adam.

Validation accuracy was evaluated for every 1k iteration and the best model was saved for the submission. The computation time for 1k iteration was about 150s (logmel-based model) and 200s (waveform-based model) using Nvidia Tesla P100 GPU.

1.3.1. Batch-wise loss masking

Another consideration in optimization is label noise. In 9.5k data for training and validation, the labels of only 3.7k data is verified and the rest is not guaranteed to have the true label. In this case,



(a) Low-level-k module (b) DenseNet-k module (c) n-head classifier module

Figure 2: Details of each module in the presented model. k and n denote the fileter size of convolution and the number of softmax layer. BN: batch normalization, Concat: feature concatenation, Relu: rectified linear unit, Conv: linear convolution, MaxPool: max pooling, GAP: global average pooling.

these data with false label may not only leads to lower classification performance, but also slower optimization because the model is trained to handle those outliers. Therefore, we expect that it will be helpful if those noise data can be detected and eliminated.

In this work, we tried the batch-wise loss masking as follows. First, the conventional loss function for a mini-batch is defined as

$$J = \sum_n C_n, \tag{3}$$

where C_n is cross-entropy for a single data in a mini-batch, which is defined as

$$C_n = \sum_c t_{n,c} \log(y_{n,c}). \tag{4}$$

On the other hand, if we know which data is labeled correctly and which is not, then we can modify the loss function to ignore noised data as follows:

$$\hat{J} = \sum_n m_n C_n, \tag{5}$$

where m_n is 1 if n-th data is labeled correctly and 0 if not.

We considered two factors to decide the values of m. First, in case of verified data, it can be considered as true label. On the other hand, if some data show especially high loss in the current model, then it can be considered as outliers with noised-label.

$$m_n = \begin{cases} 1 & \text{if } v_n = 1 \text{ or } C_n < \mu, \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where v_n denotes whether n-th data is manually verified or not. μ is defined as follows in this work:

$$\mu = \alpha \times \max_n C_n, \tag{7}$$

where α is empirically set to be 0.8 in this work. This modification eliminates the several data with largest error to gradient calculation. In addition, data which will be eliminated in cross-entropy is chosen for every batch, and it is expected to allow to find noised data gradually. In our experiments, we found that this masking technique improves the cross-validation accuracy about 1 percent point.

1.4. Model ensemble and submission

For this challenge, We submitted 3 prediction results with different model ensemble, under the team name 'Cochlear.ai (COCAI)'.

- Jeong_COCAI_task2_1.output.csv : Ensemble model is conducted by using all the tried sampling rates (16, 32, and 44.1kHz) and models (logmel- and waveform-based). Because logmel-based model shows relatively better accuracy in the cross-validation experiments, ensemble them by using weighted geometric mean as follows:

$$y_{ensemble} = \exp \frac{1}{N} \sum_n w_n \log y_n, \tag{8}$$

  where N denotes the number of models for ensemble and w_n denotes the relative weight for n-th model. We empirically set w_n to be 0.6 for the outputs from logmel-based models and 0.4 for waveform-based models. Each setting is trained by using 5-fold cross validation (CV), thus total number of models for ensemble was 3 (sampling rate) $\times$ 2 (model) $\times$ 5 (CV). The Mean Average Precision @ 3 (MAP@3) score in public leaderboard was 0.975.
- Jeong_COCAI_task2_2.output.csv: Same as Submission 1, but only 16kHz and 32kHz were used for ensemble. Total number of models was 2 (sampling rate) $\times$ 2 (model) $\times$ 5 (CV). It also scored 0.975.
- Jeong_COCAI_task2_3.output.csv (Candidate for Judges' award): Only 32kHz sampling rate and logmel-based model was used, thus 5 models of CV were used for ensemble. It scored 0.972.

2. REFERENCES

[1] E. Fonseca, M. Plakal, F. Font, D. P. Ellis, X. Favory, J. Pons, and X. Serra, "General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline," *arXiv preprint arXiv:1807.09902*, 2018.

[2] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz,

"mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[3] K. Choi, D. Joo, and J. Kim, "Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models with keras," *arXiv preprint arXiv:1706.05781*, 2017.

[4] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[5] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks." in *CVPR*, vol. 1, no. 2, 2017, p. 3.

[8] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *arXiv preprint arXiv:1709.01507*, vol. 7, 2017.

[9] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

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