AUDIO TAGGING SYSTEM USING DENSELY CONNECTED CONVOLUTIONAL NETWORKS

Il-Young Jeong and Hyungui Lim

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

ABSTRACT

In this paper, we describe the techniques and models applied to our submission for DCASE 2018 task 2: *General-purpose audio tagging of Freesound content with AudioSet labels*. We mainly focus on how to train deep learning models 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 applied the batch-wise loss masking to eliminate 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 tagging, DenseNet, Mixup, Multi-head softmax, Batch-wise loss masking

1. INTRODUCTION

Audio tagging is a research area to find one or more labels from audio signals. It has been studied in various fields including music tagging [1], domestic audio tagging [2], and acoustic scene classification [3]. Similar to audio tagging, sound event detection (SED) is another area of research to provide additional information about temporal boundaries of sound events. Despite some differences such as the optimal size of audio input or integration process of predicted values, their approaches are generally similar [4, 5]. Both of them have recently adopted deep learning-based approaches such as convolutional neural networks (ConvNet) [1, 6] and convolutional recurrent neural networks (CRNN) [7, 8]. In *Detection and classification of acoustic scenes and events* (DCASE) 2017, ConvNet-based [4, 9, 10] and CRNN-based [5, 11, 12, 13] methods outperformed conventional machine learning methods such as hidden Markov model (HMM) [14], non-negative matrix factorization (NMF) [15], and support vector machine (SVM) [16].

As the complexity of the model increases, preventing overfitting caused by lack of audio data has become more critical. To this end, data augmentation methods such as time stretching, pitch shifting, background noise mixing [17] and mixup [18] have been proposed. More recently, large-scale datasets such as Audioset [19] and Freesound dataset (FSD) [20] have been released. While data shortages have shown some improvements, there still exist difficulties associated with ambiguous relationship between an audio and its labels such as imprecisely labeled audio or many possible interpretations of a single sound.

In DCASE 2018, among the various tasks, task 2: *General-purpose audio tagging of Freesound content with AudioSet labels* aims to recognize and tag sound events of diverse nature, including musical instruments, human sounds, domestic sounds, animals, etc. In total of 41 sound event categories are considered, and the audio samples are provided from FSD.

The framework presented in this paper is based on ConvNet, specifically a densely connected ConvNet (DenseNet) [21]. We designed our models to have a single-block architecture, in which, all layers from the very bottom to the top are connected densely. In addition, we applied several techniques including mixup augmentation, multi-head softmax and batch-wise loss masking, expecting robust performance and efficient learning against audio-label ambiguities. Trained models and their ensembles were examined by using a variety of data representations, including low-level transformations and sampling rates.

2. DATASET

Freesound Dataset Kaggle 2018 (FSDKaggle2018) was provided for the challenge [22]. It consists of 11,703 audio recording data, where 9,473 recordings are for training and 1,600 are for evaluation. Each data is labeled into one of 41 audio event categories such as acoustic guitar, bus or laughter. All data is provided in a single-channel format with a sampling rate of 44.1kHz, while the duration is varied form 300 milliseconds to 30 seconds. The number of data per category is not balanced from 94 to 300.

One of the important features of FSDKaggle2018 is that only 3,710 training data labels were verified manually. In case of 5,763 recordings with the non-verified label, some may consist sounds belonging to other categories, or not belonging to any of 41 categories.

3. PROPOSED FRAMEWORK

3.1. Preprocessing and batch generation

Except for data resampling, the proposed framework does not have a preprocessing step. We tried several other techniques, including silence removal and pre-emphasis filtering, but we have not found any meaningful improvements. We applied 16kHz, 32kHz and 44.1kHz (original data) for data resampling. Low sampling rates may lose useful information at high frequencies, but its smaller data size allows to analyze longer time ranges with less computation.

We designed the batch generation framework as follows. First, we set each batch to have the same number of classes. In this work, one batch had one recordings for each class, thus the batch size was 41. We expected that it helps to make the optimization process to be stable and fast.

For efficient mini-batch learning, the length of input data in a mini-batch have to be fixed. We set it to be 64,000 samples, which is the same as 4s for 16kHz data and shorter for data with higher sampling rate. If the original recording is longer than this, a 64,000 sample segment was extracted at random offset. If the length is shorter, zero padding was applied to the beginning and end of the data.

3.1.1. Mixup augmentation

Mixup is an augmentation method which mixes two training data linearly [18]. Let x_i and t_i are i-th raw input data and corresponding binary labels in training dataset, respectively, then mixup generates an augmented data $\hat{x}$ which is a mixture of the two original data as follows:

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

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

We believe that mixup technique is also, or more, suitable for audio analysis, since the captured audio signal in real-world can be considered as a linear mixture of various 'source' signals. In this perspective, classifying $\hat{x}$ to $\hat{t}$ could be thought of as a task which detects multiple simultaneous sound events.

In this study, we set λ to be random variable of Beta distribution of $\alpha = \beta = 0.4$. 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. Another data class for mixup was randomly selected. Finally, we also applied the scale augmentation, which randomly scales the data. This process can be represented by the following equation.

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

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

3.2. Model architecture

While mixup technique meaningfully prevents overfitting and increases validation/test accuracy, it also makes the minimization of training loss to be difficult. Therefore, our model and learning strategy are focused on efficient training against strong mixup augmentation.

The overall architecture of the presented model is presented in Fig. 1. And Fig. 2 shows the details of each module in the model. It is noted that we tried 2 different models, which are 'logmel-based' and 'waveform-based', while Fig. 2 only represents logmel-based model. The minor changes for waveform-based model are described in each subsection.

3.2.1. Low-level module

The logarithm of mel-scale spectrogram (logmel) has been widely used as a preprocessing step of audio analysis. In this work, we applied the logmel transform as a low-level module in our model and implemented it using kapre. [23].

Detailed low-level module is described in Fig. 2 (a). First, the input waveform of a size (64000, 1), which denotes (sample, feature), is normalized by using Batch normalization (BN) [24], then transformed into a logmel domain with two dimensions, time and frequency. For the logmel transformation, we used 1024 window size with 128 shift and 64 mel-frequency bins. After applying BN, considering the frequency bins as a filter, it is reshaped to a size (time, frequency, 1) and considered as a grayscale image. As described in the next subsection, we aimed to conduct a single-block densely-connected architecture, so the output features of convolution layer is concatenated with its inputs.

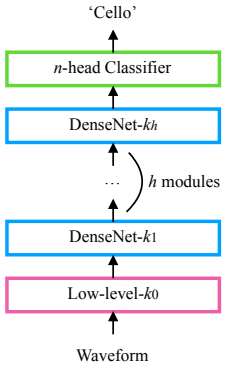


Figure 1: Overall architecture of the presented models. Details of each module are shown in Fig. 2

For the waveform-domain model, the low-level module is simplified and modified as follows:

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

3.2.2. DenseNet Module

We designed our model based on DenseNet [21]. Although the original DenseNet model divided its architecture into several blocks and applied densely-connected layer within each block, our model consists 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 densely-connected blocks, or the number of layer that disconnects the concatenation, slows down the training speed.

Fig. 2 (b) shows the details of DenseNet module. Because the size of the filter continues to increase over concatenation, 1x1 convolution is first applied to reduce it before 3x3 convolution. We also applied Squeeze-and-Excitation Network [25] to support efficient training by adding a few parameters. 2x2 max-pooling is applied to the last layer of each DenseNet module.

For waveform-based model, it was modified as follows:

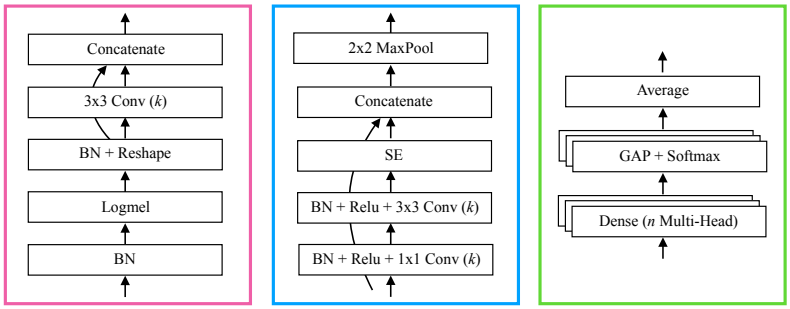
- 3x3 convolution is replaced by 1x3 convolution.
- 2x2 max-pooling is replaced by 1x2 max-pooling.

3.2.3. Classifier module

In general, the goal of classification task is to predict a binary target output vector such as [1, 0, 0]. When mixup is applied, on the other hand, it needs to predict the real values in the range of (0, 1) such as [0.9, 0.1, 0] or [0.7, 0.3, 0]. When a stronger mix-up is applied, the more target values tend to be close to 0.5.

To efficiently train the mixup model, we modified the existing 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 particularly in training with a strong mixup augmentation for the following reasons. Since the target values of augmented data are in the range of (0, 1), the values of each softmax can be varied even if these average is same as its target. Moreover, because softmax output is bounded in the range of (0,



(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 filter size of convolution and the number of softmax layer respectively. BN: batch normalization, Concatenate: feature concatenation, Relu: rectified linear unit, Conv: linear convolution, MaxPool: max-pooling, GAP: global average pooling.

1), more margin is allowed when the target is closed to 0.5. In experiments using various n-multi-head settings, we found that larger n helps to accelerates the training procedure, while its maximum validation accuracy did not show the meaningful difference.

3.2.4. Overall frameworks

Our entire model is conducted by using above mentioned modules. For the logmel- and waveform-based model, the detailed parameters for each module is 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.
- 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.

3.3. Optimization

Our experiment was implemented using Keras [26]. Adam [27] was used for optimization. Although it adaptively controls the learning rate (*lr*) by itself, we found that manual decay of learning rate helps the optimization even for Adam. We set *lr* to be 10^{-3} for first 150k mini-batch iterations, 10^{-4} for next 100k and 10^{-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 minibatch iteration and the best model was saved for the evaluation. The computation time for 1k iteration was about 150 s (logmel-based model) and 200 s (waveform-based model) using NVIDIA Tesla P100 GPU.

3.3.1. Batch-wise loss masking

Another consideration for optimization was label noise. The 3.7k data was verified from the 9.5k data for training and validation and the remainder was not guaranteed the true label. In this case, this data with false labels may not only lead to lower classification performance, but also disturb optimization because the model is trained to handle those outliers. Therefore, we believed that it will be helpful if those noise data can be detected and eliminated.

In this work, we used an iterative detection strategy which is called batch-wise loss masking in this paper. 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

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

where $t_{n,c}$ and $y_{n,c}$ denote the label and classification results for c-th class of n-th data, respectively. On the other hand, if we know which data is labeled correctly and which is not, we can modify the loss function to ignore the noise data as follows:

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

where m_n is 1 if the n-th data is correctly labeled and 0 if not. Since the optimal m is not known in the real-world situations, it is required to be estimated.

In this study, we used two factors to determine the values of m. First, verified data can always be considered as a true label. On the other hand, if some data show particularly high loss in the current model, then it can be considered as an outlier with wrong label. From these factors, we set m for each minibatch iteration as folllows:

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

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 was empirically set to be 0.8 in this work. This modification removes some data with the largest errors in the gradient calculation. In addition, since the outlier data is selected in a batch, it is expected that data with noise will be gradually found. In our experiments, this masking technique improved the cross-validation accuracy about 1 percent point.

3.4. Inference and ensemble

Unlike the training phase, the entire data longer than 64,000 sample is fed directly into the model. The presented model can handle the

Table 1: Comparison of MAP@3 scores for models using different audio representations and sampling rates.

Model	16kHz, 4s	32kHz, 2s	44.1kHz, 1.45s	ensemble
logmel	0.932	0.940	0.942	0.947
waveform	0.924	0.925	0.915	0.933
ensemble	0.944	0.949	0.948	**0.954**

variable length data and the output size is always the same due to the global average pooling layer. However, we applied zero padding for shorter data, because we believe that the length of zero-padded region implies the information about the data length, which can be an important clue for recognition.

For the ensemble of multiple models, we took the geometric mean of the model outputs. The logmel-based model was given a weight of 1.5, considering that it outperformed the waveform-based model in our experiments. The ensemble process can be represented as follows:

$$\tilde{y} = \exp(\sum_{e} (p_e \log y_e)), \tag{8}$$

where y_e and p_e denote the output and the ensemble weight of e-th model, respectively. The final output was obtained after normalizing $\tilde{y}$ to its l_1-norm.

4. RESULTS

4.1. Comparison of the low-level modules, sampling rate and temporal length

The first experiment evaluated classification performance at various low-level modules and sampling rates. It is noted that changing sampling rate directly affects to the temporal length of the analysis window since the number of samples in the batch generation was fixed.

Each model/sampling rate setting is conducted by using ensembles of 5 cross-validation models, and Table 1 shows those MAP@3[1] score. From those results, we found that logmel-based models outperform waveform-based ones, while those ensemble shows meaningful improvements. Although the results were less sensitive to sampling rate, however, it seems that the higher sampling rate leads to better classification performance, particularly in case of logmel-based models. Again, ensembles of models with various sampling rate/temporal length improves MAP@3 score. The ensemble of all different settings achieved 0.954, which is the state-of-the-art in this task[2].

4.2. Effects of multi-head classifier

The main aim of the multi-head classifier module was to accelerate the minimization of training loss. To observe the effect of the number of softmax layers, we compared the history of losses over mini-batch iteration. For this experiments, we used the logmel-based model and 44.1kHz sampling rate. Other settings were the same as previous experiments (mixup, batch-wise loss masking, learning rate, etc.). Fig. 3 shows the convergence of training loss for $n = 1$ and $n = 8$ in the first 100k iteration. Although the number of parameters in 7 added layers is around 0.6M, which is only 6% of $n = 1$ model, but the $n = 8$ model shows meaningful faster convergence.

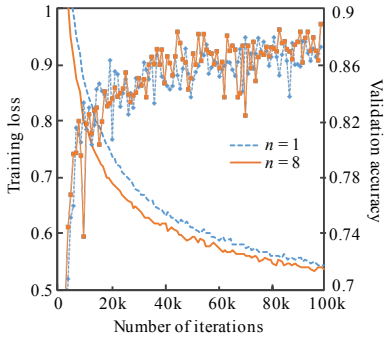


Figure 3: Training loss convergence and validation accuracies of 1-head and 8-head architecture. Loss over 1 and accuracy below 0.7 are clipped for visual convenience.

We leave the following discussions as future works. At first, the effect of multi-head layer on the test data have to be verified from more experiments. In addition, while the single-block DenseNet architecture has many filters in the last hidden layer, we expect that the number of parameters of 0.6M can be reduced by modifying the model architecture.

5. DISCUSSION

The proposed framework has shown meaningful results in the challenge, but there is room for improvement. First, the presented techniques, including single-block DenseNet architecture, Squeeze-and-Excitation Network, multi-head softmax and batch-wise loss masking, require further experimentation in various condition for verify its effectiveness. Here, the various condition may include applying to different models, datasets or tasks.

We also plan to improve the classification performance of waveform-based model. Although the logmel operation is similar to convolution layer except the squared and log operations, the waveform-based model showed relatively a lower performance compared to the logmel-based model. We believe that improving waveform-based model will be also helpful for the ensemble result.

Another important consideration is minimization of model size, which is currently 11M to 19M parameters for a single model. The smaller the size of model is, the easier to be implemented in devices with less power consumption and smaller size. Therefore, finding the minimal model size maintaining the detection performance will improve the usability in real-world applications.

6. CONCLUSION AND FUTURE WORKS

This paper described the audio tagging system submitted in DCASE 2018 task 2. We primarily focused on finding a technique that efficiently learns strongly augmented data. We presented a single-block DenseNet model, multi-head softmax layer, as well as batch-wise loss masking. We also tried to ensemble models of various low-level modules and sampling rate, and it achieved the state-of-the-art results.

[1] https://www.kaggle.com/c/freesound-audio-tagging#evaluation
[2] https://www.kaggle.com/c/freesound-audio-tagging/leaderboard

7. REFERENCES

[1] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," *arXiv preprint arXiv:1606.00298*, 2016.

[2] Y. Xu, Q. Huang, W. Wang, P. J. Jackson, and M. D. Plumbley, "Fully dnn-based multi-label regression for audio tagging," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

[3] S. Mun, S. Park, D. K. Han, and H. Ko, "Generative adversarial network based acoustic scene training set augmentation and selection using svm hyper-plane," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[4] D. Lee, S. Lee, Y. Han, and K. Lee, "Ensemble of convolutional neural networks for weakly-supervised sound event detection using multiple scale input," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[5] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, "Surrey-cvssp system for dcase2017 challenge task4," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[6] H. Zhang, I. McLoughlin, and Y. Song, "Robust sound event recognition using convolutional neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 559–563.

[7] E. Cakir, S. Adavanne, G. Parascandolo, K. Drossos, and T. Virtanen, "Convolutional recurrent neural networks for bird audio detection," in *Signal Processing Conference (EUSIPCO), 2017 25th European*. IEEE, 2017, pp. 1744–1748.

[8] Y. Hou, Q. Kong, and S. Li, "Audio tagging with connectionist temporal classification model using sequential labelled data," *arXiv preprint arXiv:1808.01935*, 2018.

[9] Y. Han, J. Park, and K. Lee, "Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification," *the Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[10] I.-Y. Jeong, S. Lee, Y. Han, and K. Lee, "Audio event detection using multiple-input convolutional neural network," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[11] H. Lim, J. Park, K. Lee, and Y. Han, "Rare sound event detection using 1d convolutional recurrent neural networks," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[12] E. Cakır and T. Virtanen, "Convolutional recurrent neural networks for rare sound event detection," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[13] S. Adavanne and T. Virtanen, "A report on sound event detection with different binaural features," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[14] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *Signal Processing Conference, 2010 18th European*. IEEE, 2010, pp. 1267–1271.

[15] A. Mesaros, T. Heittola, O. Dikmen, and T. Virtanen, "Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class activity annotations," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 151–155.

[16] A. Temko and C. Nadeu, "Classification of acoustic events using svm-based clustering schemes," *Pattern Recognition*, vol. 39, no. 4, pp. 682–694, 2006.

[17] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

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

[19] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 776–780.

[20] 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 International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[21] 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.

[22] 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.

[23] 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.

[24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015, p. 448456.

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

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

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