MULTITASK LEARNING AND SEMI-SUPERVISED LEARNING WITH NOISY DATA FOR AUDIO TAGGING

Technical Report

Osamu Akiyama

Osaka University
Faculty of Medicine
oakiyama1986@gmail.com

Junya Sato

Osaka University
Faculty of Medicine
junya.sto@gmail.com

ABSTRACT

This paper describes our submission to the DCASE 2019 challenge Task 2 "Audio tagging with noisy labels and minimal supervision" [1]. This task is a multi-label audio classification with 80 classes. The training data is composed of a small amount of reliably labeled data (curated data) and a larger amount of data with unreliable labels (noisy data). Additionally, there is a difference between data distribution between curated data and noisy data. To tackle this difficulty, we propose three strategies. The first is multitask learning using noisy data. The second is semi-supervised learning (SSL) using input data with a different distribution from labeled input data. The third is an ensemble method that averages models learned with different time windows. By using these methods, we achieved a score of 0.750 with label-weighted label-ranking average precision (lwlrap), which is in the top 1% on the public leaderboard (LB).

Index Terms— Audio-Tagging, Noisy Labels, Multitask Learning, Semi-supervised Learning, Model Ensemble

1. MULTITASK LEARNING

In this task, the curated data and noisy data are labeled in a different manner, therefore treating them as the same one makes the model performance worse. To tackle this problem, we used a multitask learning approach [2]. The aim of multitasking learning is to get synergy between 2 tasks without reducing the performance of each task. Multitask learning learns features shared between two tasks and can be expected to achieve higher performance than learning independently. In our proposal, an encoder architecture learns the features shared between curated and noisy data, and the two separated FC layers learn the difference between the two data (Fig. 1). In this way, we can get the advantages of feature learning from noisy data and avoid the disadvantages of noisy label perturbation. The loss weight ratio of curated and noisy is set as 1:1. By this method, cross validation (CV) lwlrap improved from 0.829 to 0.849 and score on the public LB increased + 0.021.

2. SEMI-SUPERVISED LEARNING

Because treating the noisy labels as same as the curated labels makes the model performance worse, it may be promising to do semi-supervised learning (SSL) using the noisy data without the noisy labels. However, this task is different from the data that SSL is generally applied in two point. The first, there is a difference of data distribution between labeled data and unlabeled data. The second, this is a multi-label classification task. This makes it difficult to apply SSL. In particular, the method that generates labels online like Mean teacher [3] or MixMatch [4] tends to collapse. We tried pseudo label [5], Mean Teacher and MixMatch and all of them are not successful.

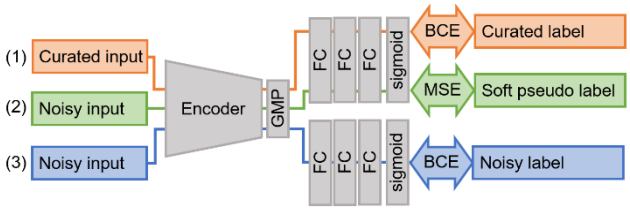

Figure 1: Over all architecture of our proposed model. (1) Basic classification (2) Soft pseudo labeling (3) Multitask learning with noisy labels

Therefore, we propose an SSL method that is robust to data distribution difference and can handle multi-label data (Fig. 1). For each noisy data sample, we guess the label using the trained model. The guessed label is sharpened by sharpening function proposed by MixMatch. We call this soft pseudo label. The basic pseudo label is a one-hot label with only one positive label so that can not apply to multi-label data. In contrast, the soft pseudo label is slightly sharpened label distribution and suits for multi-label data. Learning with soft pseudo labels is performed in parallel with multitask learning. As the temperature of sharpening function, we tried a value of 1, 1.5 or 2 and 2 was the best. Predictions of the trained model are obtained using snapshot ensemble [6] with all the folds and cycle snapshots of 5-fold CV. We used mean squared error (MSE) as a loss function. we set loss weight of semi-supervised learning as 20.

By soft pseudo labeling, The CV lwlrap improved from 0.849 to 0.870 (Table 1, 5-fold soft pseudo label). On the other hand, on the public LB, improvement in score was slight (+0.001). We use predictions of all fold models to generate soft pseudo label so that high CV may be because of indirect label leak. However, even if we use labels generated by only the same fold model which has no label leak, CV was improved as compared to one without SSL (Table 1, 1-fold soft pseudo label). The model trained with the soft pseudo label is useful as a component of model averaging (+0.003 on the public LB).

3. PREPROCESSING

We used both waveform and log mel spectrogram as input data. These two data types are expected to compensate for each other.

3.1. Waveform

We tried a sampling rate of 44.1 kHz (original data) and 22.05 kHz and 44.1 kHz was better. Each input data was regularized into a range of from -1 to +1 by dividing by 32,768, the full range of 16-bit audio.

3.2. Log mel spectrogram

For the log mel spectrogram transformation, we used 128 mel frequency channels. we tried 64 and 256 but model performance decreased. The STFT hop size of 347 was used that makes log mel spectrogram 128 Hz time resolution. Log mel spectrogram was converted from power to dB after all augmentations applied. Thereafter, it was normalized by the mean and standard deviation of each data. Therefore, the mean and standard deviation values change every time and this works as a kind of augmentation. Normalization using the mean and standard deviation of the whole data decreased model performance.

4. AUGMENTATIONS

4.1. Augmentations for log mel spectrogram

4.1.1. MixUp/BC learning

MixUp/BC learning [7, 8] is an augmentation that mixes two pairs of inputs and labels with some ratio. The mixing ratio is selected from the beta distribution. We used alpha of 1.0 for Beta distribution. This makes Beta distribution equal to uniform distribution.

4.1.2. SpecAugment

SpecAugment [9] is an augmentation method for log mel spectrogram consists of three kinds of deformation. The first is time warping that deforms time-series in the time direction. The other two augmentations are time and frequency masking, modifications of Cutout [10], that masks a block of consecutive time steps or mel frequency channels. We applied frequency masking and masking width is chosen from 8 to 32 from a uniform distribution. Time warping and time masking are not effective in this task and we didn't apply them to our models.

4.1.3. Slicing

For training, audio samples which have various time lengths are converted to a certain length by random slicing. The sound samples which have short length than the slicing length are extended to the slicing length by zero paddings. We tried 2, 4 and 8 seconds (256, 521 and 1024 dimensions) as a slicing length and 4 seconds scores the best. Averaging models trained with 4-second slicing and 8-second slicing achieved a better score.

Expecting more strong augmentation effect, after basic slicing, we shorten data samples in a range of 25 - 100% of the basic slicing-length by additional slicing and extend to the basic slicing -length by zero paddings. For data samples with a time length shorter than the basic slicing-length, we shorten data samples in a range of 25 - 100% of original length by additional slicing and extend to the basic slicing-length by zero paddings. We call this additional slicing.

4.1.4. Other augmentations

We used gain augmentation with a factor randomly selected from a range of 0.80 -1.20. We tried scaling augmentation and white noise augmentation but model performance decreased.

4.2. Augmentations for waveform

4.2.1. MixUp/BC learning

We applied MixUp to waveform input. We used alpha of 1.0 for Beta distribution as same as the case of log mel spectrogram.

4.2.2. Slicing

We applied slicing to waveform input. We tried 1.51, 3.02 and 4.54 seconds (66,650, 133,300 and 200,000 dimensions) as a slicing length and 4.54 seconds scores the best. Averaging models trained with 3.02-second slicing and 4.54-second slicing achieved a better score.

4.2.3. Other augmentations

We used scale augmentation with a factor randomly selected from a range of 0.8 - 1.25 and gain augmentation with a factor randomly selected from a range of 0.501 - 2.00.

5. MODEL ARCHITECTURE

5.1. ResNet

We selected ResNet [11] as a classification encoder model for log mel spectrogram. We compared ResNet18, ResNet34 and SE-ResNeXt50 [12] and ResNet34 performed the best. The number of trainable parameters including the multitask module is 44,210,576. We applied a global max pooling (GMP) after convolutional layers to allow variable input length.

5.2. EnvNet

We selected EnvNet-v2 [8] as a classification encoder model for waveform. The number of trainable parameters including the multitask module is 4,128,912. As same as ResNet, we applied a GMP after convolutional layers to allow variable input length.

5.3. Multitask module

For multitask learning, two separate full-connect (FC) layer sequences follow after encoder and GMP. The contents of both sequences are same and consist of FC (1024 cells) – ReLU - DropOut (drop rate = 0.2) - FC (1024 cells) – ReLU - DropOut (drop rate = 0.1) - FC (80 cells) - sigmoid. Sigmoid is replaced by softmax in model # 5 and 6 of EnvNet (Table 1).

6. TRAINING

6.1. ResNet

Adam [13] was used for optimization. Cyclic cosine annealing [14] was used for learning rate schedule. In each cycle, the learning rate is started with 1e-3 and decrease to 1e-6. There are 64 epochs per cycle. We used a batch size of 32 or 64. We used binary crossentropy as a loss function for basic classification and multitask learning with noisy data. We used mean squared error as a loss function for the soft pseudo label.

6.2. EnvNet

Stochastic gradient descent (SGD) was used for optimization. Cyclic cosine annealing was used for learning rate schedule. In each cycle, the learning rate is started with 1e-1 and decrease to 1e-6. There are 80 epochs per cycle. We used binary crossentropy as a loss function for the model using sigmoid and Kullback-Leibler (KL) divergence for the model using softmax. We used a batch size of 16 for the model using sigmoid and 64 for the model using softmax.

7. POSTPROCESSING AND ENSEMBLE

Prediction using the full length of audio input scores better than prediction using test time augmentation (TTA) with sliced audio input. This may be because important components for classification is concentrated on the beginning part of audio samples. Actually, prediction with slices of the beginning part scores better than prediction with slices of the latter part. In order to speed up the calculation, audio samples with similar lengths were grouped together, and the lengths of samples in the same group were adjusted to the same length by zero paddings and converted to mini-batches. The patience for the difference of length within a group (patience rate) was adjusted based on the prediction speed.

We found padding augmentation is effective TTA. This is an augmentation method that applies zero paddings to both sides of audio samples with various length and averages prediction results. We think this method has an effect to emphasize the start and the end part of audio samples.

For model averaging, we prepared models trained with various conditions. Especially, we found that averaging with models trained with different time window is effective (Table 1 #7)

The model weights of each cycle were saved and used for snapshot ensemble. In order to reduce prediction time, the cycles and padding lengths used for the ensemble were chosen based on CV. The predictions selected are 5 fold × (model #1 × cycle = 2, 4, 7 and 8 × padding = 8, 64 + model #2 × cycle = 1, 2, 6 and 7 × padding = 8, 64 + model #3 × cycle = 2, 4 and 6 × padding = 8, 64 + model #4 × cycle = 2 and 3 × padding = 0, 32k + model #5 × cycle 3 and 5 × padding = 8k, 32k + model #6 × cycle = 7 and 9 × padding = 8k, 32k) = 170 predictions (submission 1). The weights of each model for averaging are model # 1: 2: 3: 4: 5: 6 = 3: 4: 3: 1: 1: 1, which is chosen based on CV. In the simplified version submission, the selected predictions are 5 fold × (model #1 × cycle = 2, 4, 7 and 8 × padding = 8 + model #2 × cycle = 1, 2, 4, 6 and 7 × padding = 8 + model #3 × cycle = 2, 4 and 6 × padding = 8 + model #4 × cycle = 2 and 3 × padding = 8k + model #5 × cycle 3 and 5 × padding = 8k + model #6 × cycle = 6, 7 and 9 × padding = 8k) = 95 predictions (submission 2). The weights for averaging are the same as submission 1.

8. COMPARISON

Table 1 shows the results of each learning condition. The score is lwlrap of 5-fold CV. Table 2 shows the results of each model averaging condition.

Table 1: Comparison of each learning condition. CV lwlrap is calculated based on the best epoch of each fold in 5-fold CV except for #8, which is calculated based on the final epoch.

#	condition	CV lwlrap
1	ResNet34, 512 epoch/cycle × 1, slice length = 512, batch size = 64	0.724
2	#1 + MixUp, frequency masking and gain	0.829
3	#2 changed to 64 epoch/cycle × 8	0.829
4	#3 + multitask (model #1)	0.849
5	#4 + 5-fold soft pseudo label, batch size = 32, + additional slicing, 64 epoch/cycle × 7, use #1 weights as pretrained weights (model #2)	0.870
6	#5 changed to 1-fold soft pseudo label	0.858
7	#4 changed to slice length = 1,024 (model #3), 64 epoch/cycle × 6	0.840
8	EnvNetV2, 400 epoch/cycle × 1, slice length = 133,300, batch size = 16, augmentation = MixUp, gain and scaling, multitask, softmax	0.809
9	#8 changed to sigmoid, batch size = 64, 80 epoch/cycle × 3, use #8 weights as pretrained weight (model #4)	0.814
10	#8 changed to 80 epoch/cycle × 5, use #8 weights as pretrained weight (model #5)	0.818
11	#10 changed to 80 epoch/cycle × 10, slice = 200,000 (model #6)	0.820

Table 2: Comparison of model averaging.

#	condition	CV lwlrap
1	model #1 cycle = 1-8, pad = 8, 32	0.868
2	model #2 cycle = 1-7, pad = 8, 32	0.886
3	model #3 cycle = 1-6, pad = 8, 32	0.862
4	model #4 cycle = 1-3, pad = 8k, 32k	0.815
5	model #5 cycle = 1-5, pad = 8k, 32k	0.818
6	model #6 cycle = 5-10, pad = 8k, 32k	0.820
7	#1 + #3	0.876
8	#1 + #2 + #3	0.890
9	#4 + #5 + #6	0.836
10	submission 1	0.896
11	submission 2	0.895

9. REFERENCES

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

[2] S. Ruder, "An Overview of Multi-Task Learning in Deep Neural Networks," *arXiv preprint arXiv:1706.05098*, 2017.

[3] A. Tarvainen, and H. Valpola. "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," *arXiv preprint arXiv:1703.01780*, 2017.

[4] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "MixMatch: A Holistic Approach to Semi-Supervised Learning," *arXiv preprint arXiv:1905.02249*, 2019.

[5] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *ICML Workshop on Challenges in Representation Learning*, 2013.

[6] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot Ensembles: Train 1, get M for free," *arXiv preprint arXiv:1704.00109*, 2017.

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

[8] Y. Tokozume, Y. Ushiku, and T Harada, "Learning from Between-class Examples for Deep Sound Recognition," *arXiv preprint arXiv:1711.10282*, 2017.

[9] 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.

[10] T. DeVries, and G. W. Taylor, "Improved Regularization of Convolutional Neural Networks with Cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[11] 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.

[12] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," *arXiv preprint arXiv:1709.01507*, 2017.

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

[14] I. Loshchilov, F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," *arXiv preprint arXiv:1608.03983*, 2016.