EasyChair Preprint

№ 1274

DCASE 2019 Challenge: Audio Tagging with Noisy Labels and the Effect of Model Architecture

Maxim Shugaev, Lehan Yang, Theo Viel and Khoi Nguyen

July 10, 2019

DCASE 2019 CHALLENGE: AUDIO TAGGING WITH NOISY LABELS AND THE EFFECT OF MODEL ARCHITECTURE

Technical Report

Maxim V. Shugaev[1], *Lehan Yang*[2], *Theo Viel*[3], *Khoi T. Nguyen*[4]

[1] University of Virginia, 395 McCormik Road, Charlottesville, VA 22904-4745, USA, mvs9t@virginia.edu
[2] University of Sydney, 456P+HW Camperdown, New South Wales, Australia, xsourse.cc@gmail.com
[3] Ecole des Ponts ParisTech, 6-8 Avenue Blaise Pascal, 77420 Champs-sur-Marne, France theo.viel.enpc@gmail.com
[4] VNG Corporation182 Le Dai Hanh, Ho Chi Minh City, Vietnam, tuankhoi94@gmail.com

ABSTRACT

In this study we investigate the use of noisy-labeled data for pretraining multi-label audio tagging models. The system is implemented based on Neural Network architecture using convolutional and dense layer applied to log-scale mel-frequency spectrograms of the input data. Pretraining on the full noisy dataset is compared with pretraining on a part of the noisy dataset, selected automatically based on a model trained on a curated data, and with use of a curated only data for building an audio tagging system. In addition, the effect of the model architecture on the performance of the system is assessed. In particular, the baseline model consisted of four convolutional blocks followed by a fully connected head is com-pared with models build based on densely connected blocks. The effect of the number of blocks and pooling, for generation of features for fully connected part, is verified. Use of an optimized architecture along with 64 frequency channels in preprocessed mel-spectrograms has allowed us to build a fast baseline model with 5-fold cross-validation label-weighted label-ranking average precision (*lwlrap*) score reaching 0.87 and taking only 60-90 seconds (5-folds) for generating a prediction on the public portion of test dataset. The final system is composed of 15 models trained with slightly modified parameters and selected to minimize correlations between model predictions in the ensemble.

Index Terms— Audio classification, deep learning, convolutional neural network, noisy data

1. INTRODUCTION

The Detection and Classification of Acoustic Scenes and Events (DCASE) is a series of challenges aimed at developing sound classification and detection systems. In particular, DCASE 2019 [1] includes acoustic scene classification (1), audio tagging with noisy labels and minimal supervision (2), sound event localization and detection (3), sound event detection in domestic environments (4), and urban sound tagging (5). This report summarizes results of our study and points out the key features of our system used for Task 2. The objective of Task 2 is building a model based on small amount of reliable, manually-labeled data, and a larger quantity of noisy web audio data in a multi-label audio tag-ging task with a large vocabulary setting. The dataset provided for Task 2, FSDKaggle2019[2], includes curated data composed of 4970 audio files of the total duration 10.5 hours (manually-labeled and verified data from FSD[3]) and noisy data composed of 19,815 clips of the total duration 80 hours (taken from YFCC dataset[4]). The total number of considered classes is 80. A method based on conversion of audio files into log-scale mel-frequency spectrograms followed by use of a convolutional neural network (CNN) has reached the best performance for a variety of tasks in DCASE 2018 challenge[5]. Therefore, we apply this method in our study and focus on two aspects: effective use of data with substantial level of noise for building sound classification system, and optimization of CNN architecture to boost the system performance. The report is organized in the following: the data preprocessing is discussed in Section 2 and is followed by description of the training setup in Section 3.1, use of noisy data in Section 3.2, data augmentation in Section 3.3, CNN model architectures 3.4, and ensembling of individual model for the final system in Section 3.5. The main conclusions of the work are summarized in Section 4.

2. DATA PREPROCESSING

The audio files before use as an input are converted into log-scale mel-frequency spectrograms. The silence with the level be-low 60 dB is trimmed in the beginning and the end of each file. Despite 128 frequency channels in mel-spectrograms were considered initially, and, as demonstrated in Section 3.4, are necessary for deep models, such as DenseNet121[6], smaller models, considered in our study, reach the best performance for 64 frequency channels. Moreover, reduction of the number of channels accelerates the training and inference time. Use of 32 frequency channels also has been tested but leads to decrease of the cross validation (CV) score by 0.016 *lwlrap* in comparison with 64 frequency channel setup. The window size of fast Fourier transform has insignificant effect on the model performance and is set to be equal 1920. During training random 4 second intervals in spectrograms are selected, while files having shorter durations are expanded with using constant zero padding in the beginning and the end of the file. Use of 4 second chunks

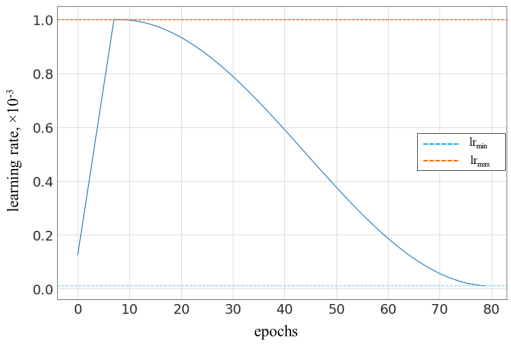

Figure 1: Learning rate scheduling publicity.

instead of 1 or 2 seconds improves the model performance and also allows to avoid significant performance drop at the inference time, when only 8 intervals are selected to generate the prediction for each file. The input image size for the model is 64×256×1. We have tested normalization of data both based on image (1) and global train set statistics (2). The mode performance reaches the similar level; though, the best CV score is reached for global normalization. In the final system we include models trained with both strategies to crease a diversity.

3. TRAINING AND INFERENCE

3.1. Training setup

The training procedure consists of two stages. At the first stage one cycle cosine annealing with warm up applied, as illustrated in Figure 1. The maximum learning rate is 0.001, and the number of epochs is ranged between 50 and 80 for different setups used to create a variety in the final ensemble. Binary cross entropy loss with logits is applied. Depending on the strategy of noisy data use, the training set is composed of the curated data, noisy data, or their mixture, as discussed in Section 3.2. At the second stage we fine-tune the model based on the curated data only. Reduce learning rate when a metric has stopped improving schedule with patience 3 and minimum learning rate 10^{-7} is sequentially applied three time with saving and loading the model with the best CV score. 5-fold cross validation scheme is considered, and the total CV score is computed as an average over the folds. The total training time for one fold is 1-2 hours on P100 GPU, so the training of entire model takes 6-9 hours. Label-weighted label-ranking average precision, *lwlrap*, metric is used for model evaluation.

3.2. Use of noisy data

One of the keys to perform well in the challenge was to find a way to efficiently use the noisy data. In fact, it is much more difficult to obtain 5000 curated labels than 20,000 noisy ones, therefore taking advantage of the "approximately" labelled data can be extremely valuable for building a system. For the challenge, we have used 2 strategies: (1) pretraining on full noisy data and (2) pretraining on a mixture of the curated data with most confidently labeled noisy data.

The first method enables a better weight initialization. Models have already learnt to recognize spectrograms and, therefore, struggle less on curated data. The training policy described previously is then applied. It clearly appeared that noisy labels are not enough for models to perform well. Though, this strategy has the advantage of requiring little work and human decision making.

The second idea is to select some of the noisy labels to include in the training. To achieve this, we create the best possible model using curated data only, and get predictions on noisy data. Then, the n samples that were the most correctly predicted are kept. As some classes were easier to detect than others, we limit the number of noisy samples to 50 per class. We also only kept samples that had one label. The metric used to evaluate our predictions is the logarithmic loss. Those selected samples are used alongside with curated samples during the first stage of the training setup, note that only curated samples are kept for validation. A best choice of n is the key to improve the results. For $n = 5000$, this method slightly outperforms the first strategy based on pretraining with noisy data only. n was chosen by listening the audio samples and by maximizing the local cross-validation score. Finally, we considered two step pretraining with use of the noisy data followed by use of curated data (or a mixture of curated and noisy data), as described below.

Results are shown in the table below. "n = 0", "n = 5000", "n = 15000", and "n = max" refer to pretraining on curated data only, curated data + 5000 files from the noisy data set with the most confident label, curated data + 15000 files from the noisy data set with the most confident label, and only data from the noisy data set, respectively. For "pretraining, n = 0" (and "pretraining, n = 5000") the pretraining procedure consists of two steps: pretraining of the model only on data from the noisy data set followed by pretraining on a curated data (and curated data + 5000 files from the noisy data set with the most confident label, respectively). In all considered cases, the models after pretraining are fine tuned on curated data only, as described in the previous subsection. In the analysis of the model architecture we used pretraining on the mixture of curated data and 5000 files from the noisy data set.

Table 1: Performance of model M3 trained with different strategies of noisy data use evaluated using 5-folds CV. The error represents the standard deviation of the mean.

	CV(*lwlrap*)
$n = 0$	0.858±0.005
$n = 5000$	**0.872**±0.005
$n = 15000$	0.865±0.004
$n = \max$	0.866±0.004
pretraining, $n = 0$	0.870±0.005
pretraining, $n = 5000$	**0.872**±0.005

3.3. Data augmentation

In the setup that has reached the best performance two augmentation techniques, MixUp[7] and spectral augmentation[8], are used. MixUp provides 0.01-0.015 CV boost that can be explained by the fact that sounds, in contrast to images of real objects, are transparent: two simultaneous sounds do not overlap each other but present simulations. Therefore, training the model on a combination of several samples with different sounds improves the model capability of sound recognition under conditions of back-ground noise and presence of several sounds at the same time. The alpha parameter for MixUp is taken to be equal to 0.4. Spectral augmentation with 2 frequency masks with the blanking fraction in the range 0.0-0.15 and 2 temporal masks with the blanking fraction in the range 0.0-0.3 gives additional 0.005 CV improvement. Use of multisample

dropout[9] with 16 samples and 0.1 dropout probability leads to decrease of CV score by approximately 0.002; however, it is applied for training of one model in the final ensemble. Other models are trained without dropout.

3.4. Model architectures

The models tested in our study are summarized in Table 2. The model M0, suggested as the baseline system for DCASE 2019 in Ref.[10], gives 0.855 CV lwlrap for our best setup (Table 3). This model consists of 4 convolutional blocks composed of 2 cascaded 3×3 convolutions, as specified in Table 2. The convolutional part is followed by 2×2 Average Pooling, Global Max Pooling, and two fully connected layers with Parametric Rectified Linear Unit (PReLU) activation function. The initial exploration per-formed for the most common computer vision models has demonstrated that DenseNet models[6] outperform ResNet[11] ResNeXt[12], CBAM-ResNeXt[13], and NasNet[14] architectures for audio classification based on log mel-scale spectrograms. Therefore, when we design models for DCASE 2019 challenge, we tried to increase the number of dense connections. In particular, the base block for our models includes concatenation of the output of the first convolutional layer with the input of the convolutional block, as schematically illustrated in Figure 2. Replacement of the convolutional blocks in M0 model by our base blocks and use concatenate pooling (concatenate output of max and average pooling) boost the overall model performance to 0.868 (M1, Table 3).

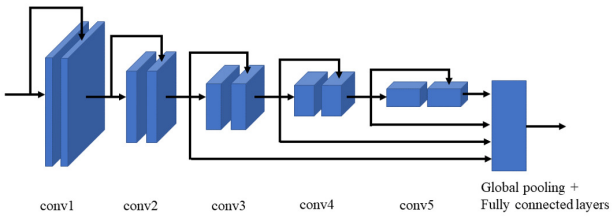

Figure 2: Schematic representation of M3 model depicting concatenation of outputs described in the text and Table 2.

Increase of the number of convolutional blocks from 4 to 5 (M4) does not lead to model improvement and results in decrease of CV to 0.865, Table 3. This decrease suggests that adding skip connections between the fully connected part and intermediate convolutional layers may help with training the model. Adding the skip connections in M1 for the second and third layers, which is referred as pyramid pooling, (M2) does not provide a performance gain in comparison with M1 (the model is shallow enough). However, adding skip connections to M4, as illustrated in Figure 2, (M3) leads to substantial boost of the model CV to 0.872. For comparison, we also tested the performance of DenseNet121[6] model in our pipeline that gave only 0.836 CV. To reach 0.85+ CV we needed to increase the number of frequency channels in mel-spectrograms to 128 followed by rescaling the images to 256×512 resolution. Therefore, we can conclude that deeper models, such as DenseNet121, require more than 64 frequency channels in contrast to models used in our setup.

3.5. Ensembling

Use of 64 frequency channels has allowed us to shorten the inference time. In particular, each model (5-folds) takes only 1-1.5 min for generation of a prediction on the publicly available test set with performing the inference for 8 4-second chunks sampled with regular interval selected based on the length of the file. Therefore, our final system includes 15 5-fold models, which takes approximately 20 and 60 minutes to generate the final prediction on P100 GPU for public and private portion of test set, respectively. The models in the ensemble are trained with slightly different parameters and are selected to minimize correlations between model predictions, given each model has larger CV than 0.864 *lwlrap*. The range of the prediction for each class is rescaled to [0,1] interval for each model, and the system output is calculated as an average of individual outputs of 15 models. On the public portion of the test dataset the system has reached 0.739 *lwlrap*.

4. SUMMARY

This report describes a system build to accomplish Task 2 in DCASE 2019 challenge. The input audio files are converted into log-scale mel-frequency spectrograms and then is used as an input for CNN. Use of 64 frequency channels has allowed us to decrease the model training and inference time keeping the sufficient level of model performance. MixUp and spectral augmentation are used and lead to increase of the model CV by 0.015-0.020 *lwlrap*. Two main topics, use of data with noisy labels and the optimization of the model architecture, are considered. The best model performance is achieved for pretraining the model on a mixture of the curated data and 5000 samples from the noisy dataset, selected automatically based on a model trained on a curated data and having the most confident labels. Increase of the number of noisy samples to 15,000 (out of 19,815) reduces the model performance to the level achieved for pretraining on noisy data only. Use of convolution blocks with dense connections along with concatenate pooling to produce features for fully connected part has boosted the model CV by 0.01. Adding the fifth convolutional block along with skip connections to the head part of the model further increases the model CV by 0.005. The final system is composed of 15 models trained in 5-fold CV setup, and the output is computed as a mean of individual predictions of the models in the ensemble.

5. REFERENCES

[1] "DCASE 2019." [Online]. Available: http://dcase.community/challenge2019

[2] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, and X. Serra, "Audio tagging with noisy labels and minimal supervision," ser. Submitted to DCASE2019 Workshop, 2019. [Online]. Available: https://arxiv.org/abs/1906.02975

[3] 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 *Hu X, Cunningham SJ, Turnbull D, Duan Z, editors. 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.

Table 2: Description of neural network architectures. Convolutional blocks are encapsulated in square brackets, and the first two numbers in each line represent the kernel size and the number of filters. Concatenate referees to concatenation of the input and the result of the first convolutional layer in a block. *Pyramid Pooling* refers to concatenation of the results of the second, third, fourth (and fifth) convolutional blocks. In the last row, numbers represent the number of output features of each linear layer.

Feature size	M0 (baseline)	M1	M2	M3	M4
64×256 × 1	Log mel-spectrograms				
64×256 × 64	$\begin{bmatrix} 3\times3, 64, \text{BN}, \text{ReLU} \\ 3\times3, 64, \text{BN}, \text{ReLU} \end{bmatrix}$		$\begin{bmatrix} 3\times3, 64, \text{BN}, \text{ReLU} \\ \text{Concatenate} \\ 3\times3, 64, \text{BN}, \text{ReLU} \end{bmatrix}$		
	2×2 Avr Pooling				
32×128 × 128	$\begin{bmatrix} 3\times3, 128, \text{BN}, \text{ReLU} \\ 3\times3, 128, \text{BN}, \text{ReLU} \end{bmatrix}$		$\begin{bmatrix} 3\times3, 128, \text{BN}, \text{ReLU} \\ \text{Concatenate} \\ 3\times3, 128, \text{BN}, \text{ReLU} \end{bmatrix}$		
	2×2 Avr Pooling				
16×64 × 256	$\begin{bmatrix} 3\times3, 256, \text{BN}, \text{ReLU} \\ 3\times3, 256, \text{BN}, \text{ReLU} \end{bmatrix}$		$\begin{bmatrix} 3\times3, 256, \text{BN}, \text{ReLU} \\ \text{Concatenate} \\ 3\times3, 256, \text{BN}, \text{ReLU} \end{bmatrix}$		
	2×2 Avr Pooling				
8 ×32 × 512	$\begin{bmatrix} 3\times3, 512, \text{BN}, \text{ReLU} \\ 3\times3, 512, \text{BN}, \text{ReLU} \end{bmatrix}$		$\begin{bmatrix} 3\times3, 512, \text{BN}, \text{ReLU} \\ \text{Concatenate} \\ 3\times3, 512, \text{BN}, \text{ReLU} \end{bmatrix}$		
	2 ×2 Avr Pooling	2 ×2 Avr Pooling	None	2×2 Avr Pooling	
4 ×16 × 1024				$\begin{bmatrix} 3\times3, 1024, \text{BN}, \text{ReLU} \\ \text{Concatenate} \\ 3\times3, 1024, \text{BN}, \text{ReLU} \end{bmatrix}$	
	Adaptive Max Pooling 512 features	Adaptive Concat Pooling 1024 features	Adaptive Concat Pyramid Pooling 1792 features	Adaptive Concat Pyramid Pooling 3840 features	Adaptive Concat Pooling 2048 features
	BN, 128, PReLU, BN, 80		BN, 256, PReLU, BN, 80		

Table 3: The performance of the considered model architectures evaluated as average over 5-fold CV. The error represents the standard deviation of the mean.

Model	M0	M1	M2	M3	M4	DenseNet121
CV(lwlrap)	0.855±0.006	0.868 ±0.005	0.867±0.003	**0.872** ±0.005	0.865 ±0.004	0.836 ±0.005

[4] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li, "Yfcc100m: The new data in multimedia research," *Commun. ACM 59, 6473*, 2016.

[5] "DCASE 2018." [Online]. Available: http://dcase.community/challenge2018

[6] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

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

[8] 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.

[9] H. Inoue, "Multi-sample dropout for accelerated training and better generalization," *arXiv preprint arXiv:1905.09788*, 2019.

[10] Q. Kong, Y. Cao, T. Iqbal, Y. Xu, W. Wang, and M. D. Plumbley, "Cross-task learning for audio tagging, sound event detection and spatial localization: Dcase 2019 baseline systems," *arXiv preprint arXiv:1904.03476*, 2019.

[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] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[13] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.

[14] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.