BUILDING LIGHT-WEIGHT CONVOLUTIONAL NEURAL NETWORKS FOR ACOUSTIC SCENE CLASSIFICATION USING AUDIO EMBEDDINGS

Technical Report

Bongjun Kim

3M
Saint Paul, MN, USA
bkim11@mmm.com

ABSTRACT

This technical report describes acoustic scene classification models from our submissions for *DCASE challenge 2021-task1A*. The task is to build a system to perform classification on acoustic scene data. The dataset has 10 acoustic scene labels. Our submissions are Convolutional Neural Network (CNN)-based models which consist of 3 convolutional layers and 1 fully-connected layer. We utilize a small subset of deep audio embedding that has been pre-trained on a large scale of a dataset. We also perform quantization and pruning to reduce the complexity of models to meet the size limit of 128KB for the challenge. We compare the performance of our models with the baseline approach on the provided test dataset. The results show that our models outperform the baseline system.

Index Terms— DCASE, Acoustic scene classification, convolutional neural networks, pruning, quantization

1. INTRODUCTION

This technical report describes our approach to the *DCASE challenge* Task1A (low-complexity acoustic scene classification with multiple devices) [1]. The task of this challenge is to build a system to classify a 10-second recording into each of 10 known acoustic scenes where the audio file was recorded. One of the goals of the challenge is to build lightweight models with a complexity limit of 128KB for non-zero parameters. To build low-complexity, but still high-performing models with a limited training dataset, we utilize a small part of an existing deep audio embedding with data augmentation and model reduction techniques including post-training quantization and pruning.

2. DATASET

The challenge dataset [1] contains audio recordings from 12 European cities in 10 different acoustic scenes using 4 different devices. It also includes synthetic data for 11 mobile devices. The 10 acoustic scenes include airport, indoor shopping mall, metro station, pedestrian street, public square, street with medium level of traffic, traveling by tram, traveling by bus, traveling by an underground metro, and urban park. Each recording in the dataset is 10 seconds and the total amount of audio is 64 hours. The development dataset for the challenge is pre-partitioned into training (13,962 files) and testing (2,970 files).

[1] http://dcase.community/challenge2021/task-acoustic-scene-classification

Table 1: The model architecture of our submissions. Convolution operations for all Conv layers are performed with a stride of 1 and padding of 1. MP indicates 2D-Max Pooling (kernal size: 2×2, stride: 2). **MP* on Layer-3 is Max Pooling operation over time-axis.

Layers	Components	Output shape
Input	Mel-spectrogram	431×256
Layer-1	Conv (3×3, 64) → Relu → MP	215×128, 64
Layer-2	Conv (3×3, 128) → Relu → MP	107×64, 128
Layer-3	Conv (3×3, 64) → Relu → MP → **MP(over time-axis)*	1×32, 64
Layer-4	FC (10) → Softmax	10

3. SYSTEM

3.1. Model architecture

Table 1 shows the architecture of our models. They consist of 3 convolutional layers (Conv) and 1 fully-connected layers (FC). It takes a mel-spectrogram of an audio file as an input representation and its output is class probabilities of 10 acoustic scene classes. In the table, filter sizes and the number of channels of convolutional layers are represented as *Conv (width × height, the number of channels)*.

While using deep audio embeddings such as VGGish [2] or OpenL3 [3] is a good way of building a high-performing model on limited training datasets, it is not easy to use them for the challenge due to its low-complexity requirement (128 KB). However, prior works have shown that using only subsets of convolutional layers of VGGish is still useful in various audio recognition tasks [4, 5, 6, 7]. Therefore, we use pre-trained weights of the first 2 convolutional layers of VGGish to initialize the first 2 layers of our models (Layer-1 and Layer-2 in Table 1). The pre-trained VGGish model is publicly available [2].

Our models take a 10-second of mel-spectrogram and extract features through the 3 convolutional layers, and then perform a max-pooling operation only on time-axis of the output from Layer-3 (before fully-connected layers). We have tried other pooling operations such as mean, Softmax [8], but the max pooling showed the best performance in our experiments.

[2] https://github.com/tensorflow/models/tree/master/research/audioset

Table 2: The number of non-zero parameters of our submissions

Layer	CNN_pr1	CNN_pr2	CNN_pr3	CNN_pr4
Layer-1	385	365	404	385
Layer-2	41,038	39,625	43,312	41,058
Layer-3	55,005	53,473	57,141	55,028
Layer-4	19,970	19,965	19,984	19,968
Total	116,398	113,428	120,841	116,439

3.2. Data augmentation

To overcome the limited training data, we use two well-known data augmentation strategies.

- Mixup [9]: We mix up a pair of randomly chosen training examples (i.e. input spectrogram) with their corresponding labels to construct a new training sample. Mixup is performed for each mini-batch during training. We perform Mixup with alpha of 0.2.
- SpecAugment [10]: Frequency masking and time masking are applied. For each mini-batch, randomly chosen 15% of frequency channels and 40% of time-frames in an input spectrogram are masked.

3.3. Training procedure

To convert raw audio into input representations for our models, we follow the data preparation step in [11]. Each audio file is resampled to 22.05kHz mono and represented by a mel-spectrogram with 256 mel-bins, a window size of 2048 and hop size of 512. Given a 10-second audio file, the size of the input representation is 431×256.

Layer-1 and Layer-2 are initialized with parameters from convolutional layers of the pre-trained VGGish model. The last convolutional layer (Layer-3) and one fully-connected layer (Layer-4) are randomly initialized. During training, the first two convolutional layers (Layer-1 and 2) are fixed (not updated). The cross-entropy loss and Adam optimizer with learning rates of 0.0002 are used. A model is trained for 500 epochs and the model that shows the lowest validation loss was chosen.

3.4. Model size reduction

In order to meet the model complexity requirement of the challenge, we apply pruning on weights of our models. The L1 unstructured pruning method provided in PyTorch library is used. We first prune weights of a model, and then re-train the model on the training set (Adam optimizer with learning rates of 0.0002). When a model is re-trained as a part of the pruning process, none of the layers is fixed. To test the effect of pruning rates, we apply 4 different pruning rates on the initial model, resulting in 4 different models. Table 2 shows the number of non-zero parameters of our submissions (CNN_pr1 to 4).

To further reduce the number of non-zero parameters of a model, post-training quantization is applied to the pruned models.

Table 3: Log-loss and accuracy(%) for the baseline system and our models. All the proposed models show higher average of class-wise log-loss and accuracy than the baseline system.

Classes	Baseline	CNN_pr1	CNN_pr2	CNN_pr3	CNN_pr4
Airport	1.429 (40.5%)	**1.180** (55.4%)	1.198 (55.4%)	1.212 (55.1%)	1.211 (56.4%)
Bus	1.317 (47.1%)	0.673 (75.8%)	**0.672** (77.4%)	0.674 (76.1%)	0.756 (72.7%)
Metro	1.318 (51.9%)	1.028 (60.9%)	1.032 (60.3%)	1.004 (60.6%)	**0.992** (60.9%)
Metro station	1.999 (28.3%)	1.327 (53.9%)	**1.235** (56.9%)	1.284 (54.5%)	1.287 (53.9%)
Park	1.166 (69.0%)	**0.628** (77.4%)	0.650 (77.1%)	0.669 (74.1%)	0.641 (78.1%)
Public square	2.139 (25.3%)	1.305 (51.9%)	1.346 (49.5%)	1.294 (53.2%)	**1.204** (56.6%)
Shopping mall	1.091 (61.3%)	1.048 (61.3%)	**1.030** (64.3%)	1.084 (61.3%)	1.068 (62.3%)
Street, pedestrian	1.827 (38.7%)	**1.210** (53.5%)	1.220 (51.9%)	1.224 (51.5%)	1.386 (44.1%)
Street, traffic	1.338 (62.0%)	0.729 (78.5%)	**0.705** (78.5%)	0.718 (79.5%)	0.734 (79.5%)
Tram	1.105 (53.0%)	0.968 (64.9%)	0.995 (63.9%)	0.93 (66.6%)	**0.811** (70.3%)
Average	1.473 (47.7%)	1.010 (63.4%)	**1.008** (63.5%)	1.009 (63.3%)	1.009 (63.5%)

We use static-quantization method provided in PyTorch library [3]. It converts a data type of a model from float32 to qint8, so it reduces the size of a model by four times. After quantization, the size of the larget model from our submissions (CNN_pr3) becomes 118 KB (120,841 $\times$ 8bit / 8 bits per byte /1024 = 118) meets the model size requirement ($<=$128KB)

4. EVALUATION

We evaluate our models on the provided validation dataset containing 2,970 recordings. As the performance metrics, macro-average multiclass cross-entropy (i.e. log-loss) and classification accuracy are used. For the challenge, submissions will be ranked by log-loss. We compare our models with the baseline system which consists of three CNN layers and one fully connected layer. More details of performance metrics and baseline system can be found in the challenge website [4].

Table 3 shows class-wise log-loss and accuracy for each model. It shows that our submissions achieved higher average of class-wise log-loss than the baseline system.

[3] https://pytorch.org/docs/stable/quantization.html

[4] http://dcase.community/challenge2021/task-acoustic-scene-classification

5. CONCLUSION

We presented low-complexity acoustic scene classification models for DCASE challenge 2021-task1A. To build high-performing and low-complexity models, we utilized a small subset of deep audio embedding and applied post-training quantization as well as model pruning techniques. We also used several well-known data augmentation methods to boost the accuracy of our models on limited datasets. The experiment results showed that our models outperformed the baseline system provided by the challenge organizers.

6. REFERENCES

[1] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, November 2018, pp. 9–13. [Online]. Available: https://arxiv.org/abs/1807.09840

[2] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, *et al.*, "Cnn architectures for large-scale audio classification," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 131–135.

[3] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, "Look, listen, and learn more: Design choices for deep audio embeddings," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3852–3856.

[4] B. Kim and B. Pardo, "Sound event detection using point-labeled data," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 1–5.

[5] B. Kim and S. Ghaffarzadegan, "Self-supervised attention model for weakly labeled audio event classification," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[6] B. Kim, "Convolutional neural networks with transfer learning for urban sound tagging," DCASE2019 Challenge, Tech. Rep, Tech. Rep., 2019.

[7] B. Kim and B. Pardo, "Improving content-based audio retrieval by vocal imitation feedback," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 4100–4104.

[8] J. Salamon, B. McFee, P. Li, and J. P. Bello, "Dcase 2017 submission: Multiple instance learning for sound event detection," Tech. Rep., 2017.

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

[10] 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.

[11] K. Koutini, H. Eghbal-Zadeh, M. Dorfer, and G. Widmer, "The receptive field as a regularizer in deep convolutional neural networks for acoustic scene classification," in *2019 27th European signal processing conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.