ACOUSTIC SCENE CLASSIFICATION USING EFFICIENTNET

Technical Report

Jakub Swiecicki

jakub.swiecicki@gmail.com

ABSTRACT

This technical report describes our solution to task 1b of the DCASE 2020 acoustic scene classification challenge. Our primary focus was to develop a single efficient model. We decided to concentrate on a single model in order to reflect the typical business situation. In our solution we chose to use log-mel spectrograms with deltas and delta-deltas features as a sound sample representation. We augmented the data with multiple techniques - mixup, specaugment, and spectrogram resizing. Our final model used EfficientNet [1] architecture.

Index Terms— Acoustic Scene Classification, DCASE, CNN

1. INTRODUCTION

Task 1b of the DCASE 2020 [2] challenge is focused on classifying acoustic scenes in a situation of multiple recording devices. The goal of the task is to develop a predictive model that generalizes to unknown recording devices. The development dataset consists of samples from 3 real and 6 simulated devices, however the training part of the development dataset uses data from 3 real and only 3 simulated devices. Finally, in case of the evaluation dataset additional one real and 5 simulated devices were used.

During the competition we concentrated on a single model as from our perspective it reflects typical business problems and requirements, where building a big ensemble of models is often impossible or impractical. Moreover, we used no audio embeddings as our intention was to keep our solution as simple as possible.

2. METHODS

As discussed in section 1 we prepared only one model for the competition. Nevertheless, our submission consists of the following versions of this model:

- **m0** - model built only on training part of development dataset,
- **m1** - model built on whole development dataset,
- **m2** - model built on whole development dataset with less regularization than model m1,
- **m3** - ensemble formed by averaging predictions from models m1 and m2.

The first version was submitted in order to verify the degradation of results compared to development set model performance. The remaining versions can be viewed as the actual challenge models.

The models were trained using PyTorch [3].

2.1. Acoustic features

In the submitted systems we used log-mel spectrogram (referred further as spectrogram) as a feature representation. To calculate the spectrogram we used 2560 FFT points, 44.1 KHz sampling rate, hop-length of 700, and 128 frequency bins. The resulting log-mel spectrogram was of size (128, 640). Then, from each spectrogram we subtracted its mean (independently for each frequency bin). It should be noted, that we did not divided spectrogram by its standard deviation.

Each log-mel spectrogram was later augmented (see section 2.2). Finally, we calculated deltas and delta-deltas features on the augmented log-mel spectrogram. One exception to this process was mixup augmentation which was performed after the deltas features were calculated.

To calculate log-mel spectrogram and deltas features we used LibROSA python package [4].

2.2. Augmentation

Below we show augmentation techniques we used in final submission. They are listed in an actual order they were applied.

1. Random crop in the time domain (to 480 data points in time domain).
2. Random resized crop - scale between 0.7 and 1, target aspect ratio between 3.1 and 4.5.
3. SpecAugment (without time warping) [5] - number of masks between 1 and 2, frequency mask rate between 0 and 0.25, time mask rate between 0 and 0.3.
4. Mixup augmentation [6, 7].

The augmentation steps were performed randomly, i.e. each augmentation step was performed with a specified probability: random resized crop - 0.5, spec augment - 0.7, and mixup - 0.7. A lot of augmentation steps were inspired by Ruslan Baikulov's solution to Freesound Audio Tagging 2019 competition [7].

2.3. Model architecture

Our models are based on EfficientNet architecture [1] and its PyTorch implementation in EfficientNet-PyTorch [8]. More specifically we used b3 version of EfficientNet. The decided to change dropout rate to 0.375 in case of models m0 and m1 to achieve more regularization.

2.4. Training

We used Adam optimization algorithm [9] with batch size of 32 and the cross-entropy loss function. In order to regularize our network we used weight decay of 0.0005 in case of models m0 and m1, and 0.0001 in case of model m2. All models were trained for 200 epochs and the learning rate was set according to one cycle policy [10] with the maximum learning rate of 0.001.

2.5. Validation setup

The development dataset was provided with proposed train/test split. During development phase we followed the official split and the results provided in this report are also based on it. The first submitted model (m0) was developed on the proposed train set, while other models were developed with the whole development dataset, after model performance was calculated with the proposed split.

2.6. Inference

The first step of data augmentation was to randomly crop a spectrogram to 480 data points in time domain. In order to account for that, at inference time we splitted each spectrogram into 3 overlapping spectrograms of size 480, so that the whole audio sample was used for inference.

3. RESULTS

All submitted systems shared the architecture but model m0 was trained on training dataset, while model m2 was less regularized version of model m1.

We achieved the highest macro-mean accuracy of 71.9% and the lowest log-loss of 0.79 with model m3. The performance statistics were calculated on the test set according to official train/test split. We present more detailed results in tables 1, 2, and 3.

Table 1: Accuracy by model and class.

Class	m0-m1	m2	m3
airport	57.6%	60.6%	63.0%
bus	82.2%	80.8%	82.8%
metro	72.1%	75.4%	78.1%
metro_station	75.1%	71.7%	75.1%
park	82.2%	78.5%	82.8%
public_square	58.6%	54.5%	57.9%
shopping_mall	58.9%	55.6%	62.3%
street_pedestrian	42.1%	47.5%	50.2%
street_traffic	84.8%	87.2%	85.9%
tram	79.8%	77.4%	80.8%
average	69.3%	68.9%	71.9%

The results show that the system performance differed between different classes and between devices. We obtained the best results for device a, as it was overrepresented in the development sample. We also observe that the systems performance for synthetic devices s1-s3 was similar to performance for devices s4-s6 even though devices s4-s6 was present only in the test set, while s1-s3 were also included in the training set.

4. DISCUSSION

Our systems show that even without using complex solutions, big ensembles, and pretrained embeddings results, that are significantly better than provided benchmark, are achievable. Nevertheless, the results show that the system is not perfect. First of all, we achieved significantly better performance for the overrepresented devices. Secondly, we observed strong differences in performance across classes and across devices.

During our experiments we observed that the model should probably be trained longer than the chosen number of epochs. However, due to time and resource constraints, we were not able to extend our solution.

Table 2: Log-loss by model and class.

Class	m0-m1	m2	m3
airport	1.018	1.255	0.976
bus	0.526	0.550	0.443
metro	0.716	0.725	0.639
metro_station	0.753	0.907	0.722
park	0.566	0.738	0.530
public_square	1.165	1.448	1.148
shopping_mall	1.073	1.353	1.056
street_pedestrian	1.441	1.649	1.355
street_traffic	0.561	0.473	0.466
tram	0.636	0.636	0.719
average	0.846	0.973	0.790

Table 3: Accuracy by model and device.

Device	m0-m1	m2	m3
a	75.8%	72.4%	76.1%
b	69.4%	71.5%	74.8%
c	70.9%	70.3%	73.3%
s1	68.8%	69.7%	71.5%
s2	66.7%	65.5%	68.5%
s3	68.5%	70.0%	71.8%
s4	69.4%	68.5%	70.9%
s5	66.7%	67.0%	69.4%
s6	67.9%	65.5%	70.6%

5. REFERENCES

[1] M. Tan and Q. V. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” *ICLM*, 2019.

[2] http://dcase.community/challenge2020/.

[3] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” *arXiv e-prints*, p. arXiv:1912.01703, Dec. 2019.

[4] https://librosa.github.io/librosa/index.html.

[5] 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 e-prints*, p. arXiv:1904.08779, Apr. 2019.

[6] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond Empirical Risk Minimization,” *arXiv e-prints*, p. arXiv:1710.09412, Oct. 2017.

[7] https://github.com/lRomul/argus-freesound.

[8] https://pypi.org/project/efficientnet-pytorch.

[9] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *arXiv e-prints*, p. arXiv:1412.6980, Dec. 2014.

[10] L. N. Smith, “A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay,” *arXiv e-prints*, p. arXiv:1803.09820, Mar. 2018.