CALIBRATING NEURAL NETWORKS FOR SECONDARY RECORDING DEVICES

Technical Report

Michał Kośmider

Samsung R&D Institute Poland
Artificial Intelligence
Warsaw, Poland,
m.kosmider@samsung.com

ABSTRACT

This report describes the solution to Task 1B of the DCASE 2019 challenge proposed by Samsung R&D Institute Poland. Primary focus of the system for task 1B was a novel technique designed to address issues with learning from microphones with different frequency responses in settings with limited examples for the targeted secondary devices. This technique is independent from the architecture of the predictive model and requires just a few examples to become effective.

Index Terms— spectrum correction, multi-device, calibration, frequency response, convolutional neural network

1. INTRODUCTION

Task 1B of the DCASE 2019 challenge [1] extends task 1A with mismatched recording devices. This task has its own dedicated dataset called “TAU Urban Acoustic Scenes 2019 Mobile” [2]. The goal is to create a model capable of predicting acoustic scenes using audio recordings from low quality devices. Additionally the dataset contains a fair amount of examples from a high quality device (referred to as A), but only a limited number from the targeted low quality devices (referred to as B and C).

2. SPECTRUM CORRECTION

The crucial part of this submission was a new technique designed to account for different frequency responses of the devices in the dataset. The procedure is very straight forward. The version used in this submission requires aligned (or at least significantly overlapping) recordings from the same moment in time. Correction is applied to the Short Time Fourier Transform of the audio recording which means that audio can be inverted back to waveform after the correction has been applied or directly transformed into a spectrogram.

Spectrum correction is implemented in two steps. First, the correction coefficients are computed from the spectrum of n aligned pairs of recordings. Then all the recordings are transformed using the computed coefficients. As each coefficient is essentially an average, n can be quite small (even 30 appeared to be sufficient).

Computing the coefficients requires choosing a reference device, for example device A. Correction coefficients for device X are computed by dividing the frequency spectrum of a recording from the reference device by the spectrum of an aligned recording from the device X. Coefficients are averaged over n such pairs. This results in one coefficient per frequency bin.

Correction is applied by multiplying Short Time Fourier Transform of the signal with the correction coefficients on the frequency axis for each point in time. If correctly implemented, spectra after correction should look alike for all the devices.

Note that the reference device can be virtual. For example, in this submission the average spectrum of devices B and C was used as the reference. This was motivated by the fact that in the evaluation dataset source devices were not indicated and a separate neural network was used to reconstruct them with 99.9% accuracy. Using the virtual reference device minimised the error from the unlikely misclassification of the source device for the given recording.

If warranted by the results of this competition this technique and its variants will be described in detail and analysed further in a dedicated article.

3. ARCHITECTURES

Two architectures were used for this submission. Both were simple fully convolutional neural networks. One of them was just a slightly larger version of the other. Dropout was not applied. Convolutional layers used ReLU activation and BatchNormalization [3].

Table 1: Architecture of the smaller network (*basic1*).

layer	outputs	kernel	stride
Conv2D+ReLU+BN	16	3	1
Conv2D+ReLU+BN	32	3	2
Conv2D+ReLU+BN	32	3	1
Conv2D+ReLU+BN	64	3	2
Conv2D+ReLU+BN	64	3	1
AveragePooling	64	-	-
Dense	10	-	-

Table 2: Architecture of the larger network (*basic2*).

layer	outputs	kernel	stride
Conv2D+ReLU+BN	16	3	1
Conv2D+ReLU+BN	32	3	2
Conv2D+ReLU+BN	32	3	1
Conv2D+ReLU+BN	64	3	2
Conv2D+ReLU+BN	64	3	1
Conv2D+ReLU+BN	128	3	2
Conv2D+ReLU+BN	128	3	1
AveragePooling	128	-	-
Dense	10	-	-

4. TRAINING

Models were trained using Adadelta [4] until convergence. Learning rate was reduced by half if accuracy did not improve by at least 10^{-3} for 16 epochs starting at 0.5. Loss function was the focal loss [5] [6] with power of 1 and no class weights. Additionally L2 regularization was added with weight of 10^{-5}. Batch size was set to 64.

The input to the neural network was a single log-scaled mel-spectrogram. Every audio clip was first transformed using STFT to frequency domain with 2048 frequency bins and 512 hop-length and then reduced to 256 mel-bins.

In training set 30% of examples were created using mixup augmentation [7] with $\alpha = 0.2$. Following SpecAugment [8], blocks of time were randomly zeroed out with maximum width of 80 and random frequency bands were zeroed out with maximum height of 27. Maximum width of time warping was 40, but remained unused in all but four out of all the ensembled networks.

Frequency bins of the resulting spectrograms were standardized using mean and standard deviation computed on the appropriate training set (separately for each of the bins).

5. RESULTS

Systems described in section 6 used the entire development dataset, either during training or calibration. Because of this, it was impossible to report results on the official train-test split. Instead table 3 shows results for a single neural networks trained in various configurations using the official test-train split. Every configuration was tested on both architectures. Any configuration that used spectrum correction outperformed its counterpart that did not. Figure 1 shows this more clearly.

Table 3: Accuracy on the development dataset for both architectures (for device A and devices B&C separately) with and without augmentation, normalization and spectrum correction.

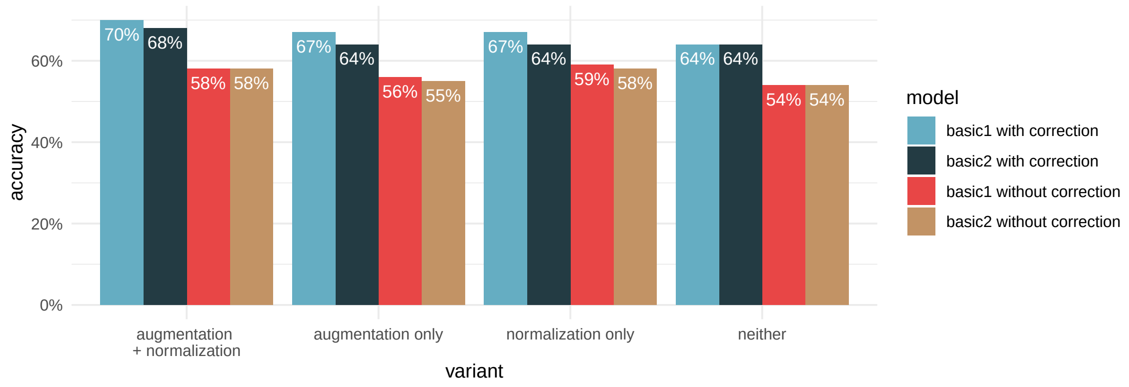
augmentation	Yes	Yes	Yes	Yes	-	-	-	-
normalization	Yes	Yes	-	-	Yes	Yes	-	-
correction	Yes	-	Yes	-	Yes	-	Yes	-
(A) basic1	72%	73%	70%	72%	71%	71%	68%	69%
(B&C) basic1	**70%**	58%	67%	56%	67%	59%	64%	54%
(A) basic2	71%	72%	68%	71%	72%	70%	69%	70%
(B&C) basic2	68%	58%	64%	55%	64%	58%	64%	54%

6. SUBMITTED SYSTEMS

The four submitted systems were all ensembles, each created in a slightly different way. The simplest one was created using soft-voting and models trained on all development dataset examples. Two systems were created using calibrated soft-voting. Calibration for each of the trained models was done on a randomly created validation split (different split per model), using isotonic regression for each class. Fourth submission used soft-voting and combined models from all the other systems. Each system was based on networks with best accuracy of at least **70%**. Most of the selected systems were using *basic1* architecture.

- *Kosmider_SRPOL_task1b_1* Calibrated soft-voting on random split #1 using 36 models which achived **74.8%** on the public and **73.8%** on private leaderboard.

Figure 1: Accuracy for devices B&C on the development dataset for both architectures in various configurations.



- *Kosmider_SRPOL_task1b_2* All models soft-voting using 124 models which achived **74%** on the public and **76%** on the private leaderboard.
- *Kosmider_SRPOL_task1b_3* Calibrated soft-voting on random dataset split #2 using 31 models.
- *Kosmider_SRPOL_task1b_4* Soft-voting using 46 models trained on the entire development dataset.

7. CONCLUSION

Spectrum correction appears to be a very effective method to combat mismatched frequency responses. It enables training of cross device predictive models and requires a modest number of examples to become effective. At the same time, accuracy for devices with dominating number of examples doesn't appear to be diminished. Applying it to a very simple model, resulted in an improvement in accuracy of over 20% relative to the base model. Ensembling and dataset augmentation further improved the results. While a single raw neural network reached only around 54%, the submitted system achieved **76%** on the private leaderboard.

8. REFERENCES

[1] http://dcase.community/challenge2019/.

[2] 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

[3] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[4] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012. [Online]. Available: http://arxiv.org/abs/1212.5701

[5] Y. Liping, C. Xinxing, and T. Lianjie, "Acoustic scene classification using multi-scale features," DCASE2018 Challenge, Tech. Rep., September 2018.

[6] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *CoRR*, vol. abs/1708.02002, 2017. [Online]. Available: http://arxiv.org/abs/1708.02002

[7] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *CoRR*, vol. abs/1710.09412, 2017. [Online]. Available: http://arxiv.org/abs/1710.09412

[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.