LOW-COMPLEXITY ACOUSTIC SCENE CLASSIFICATION WITH SMALL CONVOLUTIONAL NEURAL NETWORKS AND CURRICULUM LEARNING

Technical Report

Beniamin Kalinowski, Paweł Kowaleczko, Zuzanna Kwiatkowska, Michał Kośmider, Michał Łopuszyński

Samsung R&D Institute
Audio Intelligence
Warsaw, Poland
{b.kalinowski, p.kowaleczko, z.kwiatkowsk, m.kosmider, m.lopuszynsk}@samsung.com

ABSTRACT

The report presents the results of submission to Task 1B of Detection and Classification of Acoustic Scenes and Events Challenge (DCASE) 2020. Main issue in this task was size limitation of 500 KB for each of submitted models. Such limitations are important when model ought to be implemented on device with low memory size. For this task four different models based on convolutional neural networks were developed, varying from data preprocessing methods, data architectures etc. Crucial techniques used in complexity management were curriculum learning and the use of depth-wise and separable convolutions, along with ensembling models trained on 3 and 10 classes for performance preservation. Best models improved baseline by 10% increase in accuracy and by 60% decrease in log-loss.

***Index Terms*—** acoustic scene classification, convolutional neural networks, low complexity, deep learning

1. INTRODUCTION

Task 1B of Detection and Classification of Acoustic Scenes and Events Challenge (DCASE) 2020 was a task of classifying acoustic scenes, but with a constraint on a model size [1]. There are multiple ways of how one could approach the problem. The first scenario would be to create a model bigger than the given constraints and try to reduce its size using various methods, like pruning [2], quantization [3, 4, 5] or teacher-student models technique [6]. Another scenario, that was chosen by us, was to create a model much smaller than the given constraints, and try to expand it by adding layers or mixing models in ensemble with respect to the assumed maximal size.

2. DATA PROCESSING

Task 1B development set [7] consisted of 40 hours of binaural, 48 kHz 24-bit format audio samples recorded with a single device in 10 cities. The original dataset consisted of 10 classes, but they were aggregated into 3 relatively coarse ones - indoor, outdoor and transportation. Such division was particularly challenging for instance taking into account that classes such as *metro* and *metro station* were put into different categories.

In terms of data processing, we used log-transformed mel spectrograms. In terms of normalization, we had two approaches. In the first one, for each bin of all audio samples, we calculated its mean and standard deviation and we normalized the log-mel spectrograms bins by subtracting its mean and diving by its standard deviation (we denote it as *bz* from bin z-score). In the second one, we calculated the minimal power value in mel spectogram and treating it as a background level we calculated peak (*pk*) normalization to adjust the recordings based on the highest frequency power level present in the sample.

For multi-input models, for a given audio sample we calculated normalized log-transformed mel spectrogram as described, but we also calculated the mean over bins of this particular example before normalization to keep the original spectrogram values for better samples differentiation. In the report, we will use the term *mels* to describe the inputs to the single-input models, and *mels and means* to describe the inputs to the multi-input models. Exact parameters for each model are presented in Table 1.

Model	Window	Hop	Mels	Norm	Means
SRPOL_task1b_1	4096	1024	256	bz	Yes
SRPOL_task1b_2	4096	1024	256	bz	Yes
SRPOL_task1b_3	8192	8192	256	-	No
SRPOL_task1b_4	4096	2048	32	pk	No
	16384	15001	235	pk	No

Table 1: Data processing parameters.

3. ARCHITECTURES AND TRAINING

The following section provides information about architectures and training procedures used in the models development.

3.1. *SRPOL_task1b_1* and *SRPOL_task1b_2*

Both of the models were the weighed soft-voting ensembles of a small convolutional neural network trained on 3 classes with mels and means, denoted as *C3Multi* [8, 9], and a separable convolutional neural network trained on 10 subclasses with mels, denoted as *C10Separable*. The ratio in soft voting was 70% of weight for *C3Multi* and 30% for *C10Separable*. The architecture of *C3Multi* and *C10Separable* are presented in Tables 2 and 3, respectively.

In *SRPOL_task1b_1*, both *C3Multi* and *C10Separable* were trained for 100 epochs in mini-batches of size 32 using *Adam* optimizer [10, 11]. They were both trained only on *train* fold of *devel-*

Operation	Outputs	Kernel	Stride	Params
Input x: mels	-	-	-	-
Input y: means	-	-	-	-
x=(ConvBlock)(x)*	16	3	1	160
x=(ConvBlock)(x)*	32	3	2	4640
x=(ConvBlock)(x)*	32	3	1	9248
x=(ConvBlock)(x)*	64	3	2	18496
x=(ConvBlock)(x)*	64	3	1	36928
x=AveragePooling(x)	64	-	-	-
Concatenate(x, y)	64	-	-	-
Dense+Softmax	3	-	-	963
Total				70435

* *Conv2D+ReLU+BN*

Table 2: Architecture of *C3Multi* model.

Operation	Outputs	Kernel	Stride	Params
*SepConvBlock**	16	3	1	41
*SepConvBlock**	32	3	2	688
*SepConvBlock**	32	3	1	1344
*SepConvBlock**	64	3	2	2400
*SepConvBlock**	64	3	1	4736
*SepConvBlock**	128	3	2	8896
*SepConvBlock**	128	3	1	17664
AveragePooling	64	-	-	-
Dense+Softmax	10	-	-	1290
Total				37059

* *SeparableConv2D+ReLU+BN*

Table 3: Architecture of *C10Separable* model.

opment dataset, and their bests were chosen according to categorical cross-entropy loss on *evaluate* fold in each epoch. In both trainings, the learning rate for each epoch was calculated as

$$\mathrm{lr}(\mathrm{epoch}) = 0.001 \cdot 0.9^{\lfloor \mathrm{epoch}/2 \rfloor}. \quad (1)$$

Moreover, in *C10Separable* a mixup augmentation technique was used with $\alpha = 0.2$ [12]. Input to both models was normalized with *bz*. Data was shuffled before training and after each epoch.

In *SRPOL_task1b_2*, both *C3Multi* and *C10Separable* were trained on the whole *development* dataset and the last model after all 100 epochs was chosen. They were also trained using *Adam* with same mini-batch, learning rate schedule and loss. No augmentation technique was used. Input to both models was normalized with *bz*. Data was shuffled before training and after each epoch.

3.2. *SRPOL_task1b_3*

The next solution was a single model, with its architecture presented in 4.

It was trained on 3 classes using *Adam* optimizer on 200 epochs with mini-batch of size 167 . The model was chosen based on best performance in terms of accuracy on *evaluate* fold. It was trained using categorical cross-entropy and learning rate was scheduled to follow

$$\mathrm{lr}(\mathrm{epoch}) = 0.01 \cdot 0.9^{\lfloor \mathrm{epoch}/2 \rfloor}. \quad (2)$$

Operation	Outputs	Kernel	Params
*VGGBlock**	16	7x1, 1x7	2000
*VGGBlock**	48	9x1, 1x9	27936
*VGGBlock***	64	11x1, 1x11	79232
*ConvClassifier****	3	3	1731
Total			110899

* *Conv2D+Conv2D+ReLU+BN+MaxPool2D(4)+Dropout(0.3)*
** *Conv2D+Conv2D+ReLU+BN+Dropout(0.3)*
*** *Conv2D+GlobalAvgPool2D+Softmax*

Table 4: Architecture of *SRPOL_task1b_3* model.

In loss, classes were weighed by 1.4 in indoor, 0.2 for outdoor and 0.15 for transportation. The data were not shuffled.

3.3. *SRPOL_task1b_4*

The last model was a single model with architecture presented in 5. The input features are two log mel spectrograms using sizes presented in 1. The second log mel spectrogram transposed is the same size as the first one, thus they are concatenated and stored in two separate channels. Following [13] and [14] the training procedure was using Curriculum Learning divided into 3 splits starting from the easy samples and finishing at processing all training samples.

It was trained on 3 classes using Adam optimizer on 150 epochs with mini-batch of size 128. The model was chosen based on best performance in terms of accuracy on evaluate fold. The learning rate has been selected and is changing according to the cyclical learning rate [15]. The data has been shuffled using Tensorflow shuffle buffer of size 5% of the training data.

Additionally, apart from SpecAugment [16] other augmentation methods like time and frequency warping, time length control have been used[17].

Operation	Outputs	Kernel	Params
*VGGBlock**	64	3x3	1216
*VGGBlock**	96	3x3	55392
*VGGBlock**	64	3x3	55360
*ConvClassifier***	3	3x3	195
Total			112163

* *Conv2D+SeLU+BN+MaxPool2D(4)+AlphaDropout*
** *Conv2D+SpatialDropout+GlobalAvgPool2D+Softmax*

Table 5: Architecture of *SRPOL_task1b_4* model.

4. RESULTS

In Tables 6 and 7 we present calculated models' sizes and results obtained on *development* dataset, respectively. For *SRPOL_task1b_2*, which was trained on the whole dataset, we report results before retraining on the whole dataset.

5. CONCLUSIONS

In this report we present 4 solutions to Task 1B of DCASE 2020. In general, we use various convolutional neural networks models with

Model	Params	Type	Size
SRPOL_task1b_1	107494	float32	421 KB
SRPOL_task1b_2	107494	float32	421 KB
SRPOL_task1b_3	110899	float32	434 KB
SRPOL_task1b_4	112611	float32	441 KB

Table 6: Models' sizes summary.

Model	I	O	T	Overall
DCASE baseline	82.0%	88.5%	91.5%	87.3%
	0.680	0.365	0.282	0.437
SRPOL_task1b_1	92.6%	94.7%	**96.5%**	94.6%
	0.236	**0.171**	**0.118**	0.175
SRPOL_task1b_2	92.6%	94.6%	96.2%	94.47%
	0.218	**0.171**	0.121	**0.170**
SRPOL_task1b_3	93.1%	**95.0%**	90.3%	92.8%
	0.226	0.173	0.371	0.256
SRPOL_task1b_4	**98.1%**	**95.0%**	93.1%	**95.4%**
	-	-	-	0.217

Table 7: Models results summary. For each model, first row shows accuracy and second shows log loss. Columns represents the first letter of each class.

log-transformed mel spectrograms. In both *SRPOL_task1b_1* and *SRPOL_task1b_2*, we mix 2 models trained on 3 and 10 classes with single- and multi-input, differing with augmentation techniques and training datasets. In *SRPOL_task1b_3*, we use a VGG-style convolutional neural network with depth-wise convolutions. In *SRPOL_task1b_4*, we used further augmentation results with VGG-style convolutional neural network. With our models, we improve baseline by 10% increase in class-wise accuracy and by 60% decrease in class-wise log-loss. To improve the performance of the system, it is necessary to adjust the network structure, convolution type and parameter settings of the CNN. We introduced several architectures of stacked convolution based networks. This suggests that these architectures can be an effective tool that offers good generalization properties for various audio processing tasks. The number of misclassified samples still suggests that there is still a place for an improvement and there is a strong demand for such solutions.

6. REFERENCES

[1] T. Heittola, A. Mesaros, and T. Virtanen, "Acoustic scene classification in dcase 2020 challenge: generalization across devices and low complexity solutions," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, 2020, submitted. [Online]. Available: https://arxiv.org/abs/2005.14623

[2] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," 2018.

[3] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[4] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," in *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 07 2017, pp. 5406–5414.

[5] K. Ullrich, E. Meeds, and M. Welling, "Soft weight-sharing for neural network compression," 02 2017.

[6] J. Wong and M. Gales, "Sequence student-teacher training of deep neural networks," 09 2016, pp. 2761–2765.

[7] 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

[8] M. Kośmider, "Calibrating neural networks for secondary recording devices," Samsung R&D Institute Poland, Tech. Rep., 2019.

[9] T. Nguyen, F. Pernkopf, and M. Kośmider, "Acoustic scene classification for mismatched recording devices using heated-up softmax and spectrum correction," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5 2020, pp. 126–130. [Online]. Available: https://doi.org/10.1109/icassp40776.2020.9053582

[10] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning - Volume 37*, 2015, p. 448–456.

[11] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 2014.

[12] H. Zhang, M. Cissé, Y. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *ArXiv*, vol. abs/1710.09412, 2018.

[13] J. Frankle, D. J. Schwab, and A. S. Morcos, "The early phase of neural network training," 2020.

[14] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 41–48. [Online]. Available: https://doi.org/10.1145/1553374.1553380

[15] L. N. Smith, "No more pesky learning rate guessing games," *CoRR*, vol. abs/1506.01186, 2015. [Online]. Available: http://arxiv.org/abs/1506.01186

[16] 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," *Interspeech 2019*, Sep 2019. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-2680

[17] Y. Hwang, H. Cho, H. Yang, D.-O. Won, I. Oh, and S.-W. Lee, "Mel-spectrogram augmentation for sequence to sequence voice conversion," 2020.