SEPARABLE CONVOLUTIONS AND TEST-TIME AUGMENTATIONS FOR LOW-COMPLEXITY AND CALIBRATED ACOUSTIC SCENE CLASSIFICATION

Technical Report

Gilles Puy *Himalaya Jain* *Andrei Bursuc*

valeo.ai, Paris, France

ABSTRACT

This report details the architecture we used to address Task 1a of the of DCASE2021 challenge. Our architecture is based on 4 layer convolutional neural network taking as input a log-mel spectrogram. The complexity of this network is controlled by using separable convolutions in the channel, time and frequency dimensions. We train different models to investigate the benefit of mixup, focal loss and test time augmentations in improving the performance of the system.

The code and trained models are available at `https://github.com/valeoai/SP4ASC`.

1. INTRODUCTION

In this work, we address the Task 1a of DCASE2021 [1]. The task aims at acoustic scene classification (ASC) with a low-complexity model. More specifically, the task is to classify acoustic scenes recorded from different devices into ten predefined classes. The classification model is constrained to fit in 128 KB, which would be 65 K parameters using float16 data type.

The dataset provided for this task is TAU Urban Acoustic Scenes 2020 Mobile [2]. It contains audio recordings from 12 European cities recorded with 4 different devices. In addition, the dataset has simulated data for 11 mobile devices. The development set of the dataset contains recordings from 10 cities, 9 devices (3 real and 6 simulated) while the evaluation set has data from 12 cities and 11 devices. There are 2 cities and 6 devices (1 real and 5 simulated) that are only present in the evaluation set.

2. NETWORK ARCHITECTURE

Our architecture is based on CNN6 described in [3]. This network consists of 4 convolutional layers using 5×5 filters, as in AlexNet architecture [4], followed by a global pooling layer and a final MLP.

The number of parameters in CNN6 is above the required limit of 65 K parameters. We detail in this section our choices to reduce the number of parameters below this limit. Our main ingredient is the change of each of the original 5×5 convolutional layers by separable convolutions along the channel, time and frequency axes. Note that one can find similar ideas in, e.g., MobileNets [5], [6] for videos or [7] for audio. Finally, we also adapt the number of channels in each layer to meet the required constraint in number of parameters.

2.1. Convolutions

Let $\mathsf{X} \in \mathbb{R}^{n \times c_{\text{in}}}$ denote a feature map of spatial dimension n with c_{in} channels. In the original convnet CNN6, the feature maps from one layer to the next satisfy

$$\mathsf{Y} = f_{5\times5}(\mathsf{X}), \tag{1}$$

where $f_{5\times5} : \mathbb{R}^{n\times c_{\text{in}}} \rightarrow \mathbb{R}^{n\times c_{\text{out}}}$ denotes a regular convolution layer with a kernel of size 5×5 (no bias) giving a feature map Y of spatial dimension n with c_{out} channels. This layer contains $25^2 c_{\text{in}} c_{\text{out}}$ parameters.

To reduce the number of parameters, we replace (1) by

$$\mathsf{Y} = h_{3\times1}\left(g_{1\times1}(\mathsf{X})\right) + h_{1\times3}\left(g_{1\times1}(\mathsf{X})\right), \tag{2}$$

where $g_{1\times1} : \mathbb{R}^{n\times c_{\text{in}}} \rightarrow \mathbb{R}^{n\times c_{\text{out}}}$ denotes a regular convolution layer with a kernel of size 1×1, and $h_{3\times1}, h_{1\times3} : \mathbb{R}^{n\times c_{\text{out}}} \rightarrow \mathbb{R}^{n\times c_{\text{out}}}$ denote two different *channel-wise* convolutions with kernels of size 3×1 and 1×3, respectively, dilated by 2 to keep a receptive field similar to the one of $g_{5\times5}$. This reduces the number

Layer	Feat. size	Param. (test)
BN-mel	$1 \times 431 \times 256$	512
Conv-BN-ReLU-Pool	$64 \times 215 \times 128$	512
Conv-BN-ReLU-Pool	$128 \times 107 \times 64$	9088
Conv-BN-ReLU-Pool	$128 \times 53 \times 32$	17280
Conv-BN-ReLU-Pool	$128 \times 26 \times 16$	17280
Global Pooling	128	0
MLP	10	17802
	Total	62474

Table 1: **Network architecture**. The size of feature maps at each layer appears in the middle column under the form $C \times T \times F$ where C, T and F are the number of channels, time dimension and frequency dimension, respectively. The number of parameters in each layer at *test* time is also reported in the last column.

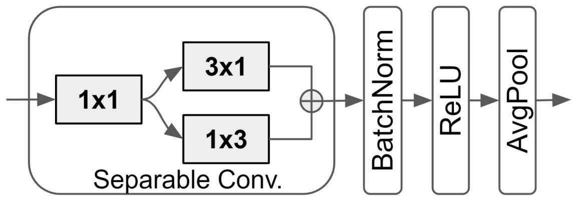

Figure 1: **Conv-BN-ReLU-Pool layer**. The separable convolution block shows the architecture we use in place of 5×5 convolutions in CNN6.

of parameters to $(c_{\rm in} + 6)c_{\rm out}$. Note that none of the convolutional layers contain any bias parameter.

2.2. Batch normalisation

Our network is trained using a batch normalisation layer after each convolutional layer (2), adding $4c_{\rm out}$ parameters per layer. However, at testing time, we reduce the number of parameters by merging each convolution layer with its following batch normalisation layer, hence effectively adding $c_{\rm out}$ parameters per layer, in the form of a bias, after each convolution (2).

For completeness, we detail now how we merge the parameters of the convolutional and batch normalisation layers after training. At the end of the training process, the $\ell^{\rm th}$ channel of Y in (2) after batch normalisation satisfies

$$\tilde{y}^{(\ell)} = \gamma^{(\ell)} \left[\frac{w_{3\times1}^{(\ell)} * \tilde{x}^{(\ell)} + w_{1\times3}^{(\ell)} * \tilde{x}^{(\ell)} - \mu^{(\ell)}}{\sigma^{(\ell)}} \right] + \beta^{(\ell)}, \quad (3)$$

where $*$ denotes the 2D convolution, $\mu^{(\ell)}$, $\sigma^{(\ell)}$, $\gamma^{(\ell)}$ and $\beta^{(\ell)}$ are the running mean, running standard deviation, scale and bias parameters of the batch normalisation layer on the $\ell^{\rm th}$ channel, $\tilde{x}^{(\ell)}$ denotes the $\ell^{\rm th}$ channel of $g_{1\times1}(\mathsf{X})$, and $w_{3\times1}^{(\ell)}$, $w_{1\times3}^{(\ell)}$ the corresponding channel-wise convolutional filters. At test time, we compute

$$\tilde{y}^{(\ell)} = \tilde{w}_{3\times1}^{(\ell)} * \tilde{x}^{(\ell)} + \tilde{w}_{1\times3}^{(\ell)} * \tilde{x}^{(\ell)} + \tilde{\beta}^{(\ell)}, \quad (4)$$

where

$$\tilde{w}_{\cdot\times\cdot}^{(\ell)} = \frac{\gamma^{(\ell)}}{\sigma^{(\ell)}}\ w_{\cdot\times\cdot}^{(\ell)} \quad \text{and} \quad \tilde{\beta}^{(\ell)} = \beta^{(\ell)} - \frac{\gamma^{(\ell)}}{\sigma^{(\ell)}}\ \mu^{(\ell)}. \quad (5)$$

Hence, in addition to the parameters involved in (2), $c_{\rm out}$ additional parameters $\tilde{\beta}^{(\ell)}$ are involved after batch normalisation at test time.

2.3. Complete Architecture

The network architecture is presented in Table 1. It contains 64 330 parameters at train time and 62 474 at test time.

The input is a log-mel spectrogram computed from the complete audio signal of 10 seconds, sampled at 44.1 KHz, using 2048 points for the FFT, a hop size of 1024, and 256 mel-bins. The generated spectrogram has size 431×256.

The first layer, denoted by BN-mel, consists of a batch normalisation of each mel-bin. This layer is transformed into an affine layer at test time by merging its scale and bias parameters with the estimated running statistics (as in (5)). At test time, this layer contains 512 parameters.

The layers denoted by Conv-BN-ReLU-Pool, shown in Figure 1, consist of a convolutional layer satisfying (1), a batch normalisation layer, a ReLU activation, and an average pooling layer applied sequentially. The average pooling layer reduces the resolution by 2 in the time and frequency axes with a kernel of size 2×2.

The global pooling layer consists of (i) a global averaging pooling along the frequency axis followed by (ii) a global averaging *and* a max pooling along the time axis, the results of both pooling being summed together to yield a feature vector of size 128 that enters the final MLP for classification. This MLP contains two layers with a hidden dimension of 128 using a ReLU activation in the hidden layer.

Syst.	MixUp	CE	FL	TTA	Log loss	Accuracy
1		✓		✓	0.90	66.8%
2	✓	✓		✓	0.93	66.2%
3			✓	✓	0.88	68.7%

Table 2: **Performance of different train/test strategies**. Scores obtained on the validation set of [2] using different training strategies – MixUp or not, cross entropy (CE) or focal loss (FL) – and test-time augmentations (TTA) or not.

3. TRAINING PROCEDURE

The model is trained for 200 epochs, with a batch size of 32, a weight decay of 10^{-5}, using AdamW with a starting learning rate of 10^{-3} and a cosine annealing scheduler decreasing the learning rate to 10^{-5}. We use two dropout layers [8]: the first on the input of the final MLP and the second on its hidden layers. These layers drop each neuron with probability 0.2.

For all submitted systems, we use SpecAugment [9] on the log-mel spectrogram using two masks of size at most 128 on the time axis and two masks of size at most ΔF on the frequency axis. We also investigate the benefit of MixUp [10] with $\alpha = 0.2$ during training to improve the log loss at test time. The systems are trained on the training split provided by [2] and evaluated on the provided validation split.

All but one system are trained by applying a softmax on the final logits and using the cross-entropy loss. The last system is trained with the focal loss.

Focal Loss. The focal loss [11] is a popular objective function for computer vision tasks that suffer from significant class imbalance in terms of number of samples per class and class difficulty, e.g., object detection, semantic segmentation. In a nutshell, the focal loss drives away the network's focus from easy well-classified examples, that eventually dominate training, towards harder incorrectly classified ones, that can be more informative.[1] Mukhoti et al. [12] have recently shown that the focal loss mitigates the overconfidence pathological behavior of deep neural networks [13] leading to well calibrated classifiers. Even though smaller networks, such as the one we propose here, do not necessarily suffer from overconfidence [13], we argue the utility of the focal loss in addressing difficult or ambiguous samples that are typically in-between similar classes and that can be prone to annotation error, e.g., `metro_station` *vs.* `metro`, `street_pedestrian` *vs.* `public_square`. Given such a sample at an intermediate training point, if the network predicts a low probability for the correct class the focal loss will insist on increasing its probability. Conversely, if the network predicts a high probability for the correct class, then it will not insist further in reducing the negative log-likelihood (NLL) and avoiding overconfidence, unlike the cross entropy loss that aims to reduce the NLL equally for all samples, even correct ones with already high probability predictions. We expect the focal loss to align the predicted and target distributions, while increasing the entropy of the predicted distribution [12], leading to better logs loss scores. In our experiments, we consider the fixed focal loss with $\gamma^{FL} = 1$. More elaborate strategies for scheduled or adaptive γ^{FL} during training can be designed according to the dataset and the

[1]We use the following definition of the focal loss: $\mathcal{L}^{FL}(\hat{p}_t)=-(1-\hat{p}_t)^{\gamma^{FL}} \log(\hat{p}_t)$, where $\hat{p}_t$ is the predicted probability of the correct class.

neural network architecture [12], but we did not explore this direction.

4. TEST TIME EVALUATION

The network presented above is trained using parameters in float32 type. At the end of the training procedure, each batch normalisation layer is combined with its previous convolution layer to reduce the number of parameters. The resulting 62 474 parameters are then cast in float16, yielding a model size of 122 KB to be saved on disk.

Before testing, these float16 parameters are loaded on GPU and cast into float32 type. The log-mel spectrograms are computed using float32 precision and all computations are thus done with a float32 precision on the GPU.

System 1 "ce_tta": This system is trained using cross entropy, SpecAugment with $\Delta F = 32$ and dropout. We also use SpecAugment at test time to improve the log loss and the accuracy. We average the softmax predictions from 30 different augmentations. This test time augmentations permit us to improve the log loss from 1.10 to 0.90 and the accuracy from 65.8% to 66.8% on the validation set.

System 2 "ce_mu_tta": This system is trained using cross entropy, SpecAugment with $\Delta F = 16$, and MixUp. We also use SpecAugment at test time, averaging the softmax predictions from 30 different augmentations. This test-time augmentations permit us to improve the log loss from 0.96 to 0.93. The accuracy is unchanged, reaching 66.2% on the validation set.

System 3 "fl_tta": This system is trained with focal loss, SpecAugment with $\Delta F = 32$. We also use SpecAugment at test time, averaging the softmax predictions from 30 different augmentations. This test-time augmentations permit us to improve the log loss from 0.95 to 0.88 and the accuracy from 66.7% to 68.3% on the validation set.

All scores obtained on the validation set of [2] are reported in Table 2. Note that because of test-time augmentations, these scores vary slightly from one evaluation to another.

5. REFERENCES

[1] I. Martín-Morató, T. Heittola, A. Mesaros, and T. Virtanen, "Low-complexity acoustic scene classification for multi-device audio: analysis of DCASE 2021 Challenge systems," *arXiv:2105.13734*, 2021.

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

[3] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, 2012.

[5] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861*, 2017.

[6] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[7] E. Kazakos, A. Nagrani, A. Zisserman, and D. Damen, "Slow-fast auditory streams for audio recognition," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 855–859.

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[9] 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," in *Proc. of Interspeech*, 2019, pp. 2613–2617. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-2680

[10] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=r1Ddp1-Rb

[11] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[12] J. Mukhoti, V. Kulharia, A. Sanyal, S. Golodetz, P. H. Torr, and P. K. Dokania, "Calibrating deep neural networks using focal loss," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[13] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1321–1330.