LIGHTWEIGHT CONVOLUTIONAL NEURAL NETWORKS ON BINAURAL WAVEFORMS FOR LOW COMPLEXITY ACOUSTIC SCENE CLASSIFICATION

Technical Report

Nicolas Pajusco, Richard Huang, Nicolas Farrugia

IMT Atlantique, Lab-STICC, Department of Electronics, Brest - France
nicolas.farrugia@imt-atlantique.fr

ABSTRACT

This report describes our submission to DCASE 2020 task 1, subtask B, which is an acoustic scene classification task with the objective of minimizing parameter count. While the vast majority of proposed approaches rely on fixed feature extraction based on time-frequency representations such as spectrograms, we propose to fully exploit the information in binaural waveforms directly. To do so, we train one dimensional Convolutional Neural Networks (1D-CNN) on raw, subsampled binaural audio waveforms, thus exploiting phase information within and across the two input channels. In addition, our approach relies heavily on data augmentation in the temporal domain. Finally, we apply iterative structured parameter pruning to remove the least important convolutional kernels, and perform weight quantization in floating point half precision. We apply this approach to train two network architectures: a 1D-CNN based on VGG-like blocks, as well as a ResNet architecture with 1D convolutions. Our results show that we can train, prune and quantify a small VGG model to make it 20 times smaller than the 500 KB limit (model A) with an accuracy at baseline level (87.6 %), as well as a larger model achieving 91 % of accuracy while being 8 times smaller than the challenge limit. ResNets could be successfully trained, pruned and quantify in order to be below the 500 KB limit, achieving up to 91.2 % accuracy.

Index Terms— raw audio, Convolutional neural networks, auditory scene classification, residual networks, data augmentation, pruning

1. INTRODUCTION

Modern approaches for Deep Learning in computer vision or natural language understanding have been very successful in automatically learning flexible feature extractors, using convolutional neural networks [1], residual networks [2], or attention-based / transformer models [3]. Such feature extractors are trained using large amounts of data, limiting the need of hand-crafted features or representations. In contrast, most deep learning approaches for audio applications still rely on expertly defined, fixed feature extractors based on time-frequency representations, such as spectrograms or mel-spectrograms [4].

In this work, we were interested in testing the hypothesis that using a fixed feature extractor is detrimental for computational complexity, for two reasons. First, considering a spectrogram (or equivalent) as an image-like input may tend to overparametrize the downstream network, as the effort in training for classification becomes a two-dimensional problem. Second, a spectrogram only considers the power in frequency bands, ignoring the phase. In particular, when considering two channels as input, the phase difference between the channels could be informative. As a consequence, our goal is to train networks using end-to-end learning, from feature extraction to classification, using CNN.

Learning from raw waveform is costly, due to the size of input vectors. While approaches such as recurrent networks [5] or dilated convolutions [6] have previously been considered, such approaches need a very large number of parameters. We tackle this problem by proposing an approach that relies on the following ingredients. First, after a careful examination of provided signals, we resampled all data to 18kHz. Next, we use both input channels in a CNN with one-dimensional convolution kernels, coupled with stride and max-pooling to reduce the size of internal feature maps. Third, we use various strategies for data augmentation, in order to challenge the network the learn relevant audio features with degraded or masked versions of the waveforms. This data augmentation strategy is heavily inspired by recent progress in deep learning in computer vision. Finally, we apply parameter pruning, fine tuning and quantization to the best models obtained.

The rest of this report is organized as follows. We begin by detailing our strategies for data augmentation during training in section 2. Next, we describe the two network architectures in section 3. In section 4, we detail how we achieve to compress our models using structured parameter pruning and quantization. We explain our experimental and training setup in section 5, and finally we present and discuss our results in section 6.

2. DATA AUGMENTATION

Our strategy relies heavily on data augmentation (DA), with the underlying hypothesis that combining various forms of DA can yield better flexibility with a smaller set of parameters, as well as better generalization. We use four forms of DA: temporal masking, filtering, noise addition, and CutMix [7]. The various hyperparameters of DA were chosen in preliminary analysis on subsets of the development set. All DA are applied to 99% of the training set randomly at each epoch, and DA is not applied for the validation and test set.

2.1. Temporal masking

Random crop using a rectangular window is an extremely common DA strategy for training 2D CNN in computer vision applications. We adapted this strategy to the temporal domain, by considering a temporal mask that is positioned randomly in the signal. We implement temporal masking by multiplying the signal by a rectangular window. The position of this window is randomly chosen within the total length of the signal, with a total of 1000 possible positions.

The window length is randomly chosen according to a Gaussian distribution, with an average of four seconds and a standard deviation of one second. Importantly, the resulting signal after temporal masking is still 10 seconds long, which enables us to train and validate the network with the full signal length.

2.2. Filtering

We perform DA using filtering in order to augment the variety of the frequency content in the training set, thus challenging the network training to extract the most relevant frequency features when degrading the frequency content of the dataset. As a consequence, we apply eight finite impulse response filter (FIR): three low pass filters, three high pass and two band pass filters. These filters are applied after the temporal masking (if present) on the 10 second long signal. The cut-off frequencies of the low pass and high pass filter are respectively 300, 1000 and 2000 Hz. Two band pass filters are used : one with a bandwidth of 1200 to 3400 Hz, and one with a bandwidth of 340 to 3400 Hz.

2.3. Additive noise

The third DA strategy is to add white Gaussian noise into the signal. The signal to noise ratio is randomly chosen between 6 and 32 dB, by steps of 1 dB. Noise is added in the signal after temporal masking and filtering.

2.4. CutMix

CutMix [7] has been previously introduced as a very efficient DA strategy for training CNNs for computer vision task. The general idea of CutMix is to produce a new sample by concatenating two segments belonging to two different categories, and set the target by weighting according to lengths of each segment. The operation is defined in [7] as :

$$\begin{aligned} \tilde{x} &= \mathbf{M} \odot x_A + (\mathbf{1} - \mathbf{M}) \odot x_B \\ \tilde{y} &= \lambda y_A + (1 - \lambda) y_B \end{aligned} \tag{1}$$

where x_A and x_B are two examples from the original training set, $\mathbf{M} \in \{0, 1\}$ denotes a binary mask, $\odot$ is the element-wise multiplication, λ is sampled from a Beta distribution Beta(α, α) with $\alpha = 1$, $\tilde{x}$ is the input new signal and $\tilde{y}$ is the new target. Note that λ directly corresponds to the proportion of ones in $\mathbf{M}$, as λ controls how much example x_A is "mixed" in $\tilde{x}$.

3. NETWORK ARCHITECTURE

In this section, we present the network architectures that we submit to the challenge. These architectures are based on popular CNN, namely VGG [1], and ResNet [2]. We adapt these architectures for raw binaural waveforms.

3.1. VGG

The first architecture is a one dimensional small standard CNN (models A and B in table 1 and figure 1) composed of successive blocks, each including a sequence of a convolution, Batch norm, Restricted Linear Unit activation (ReLu), and Max Pooling. These blocks are similar to the ones found in networks such as VGG [1]. The particularities of this architecture are (1) the two input channels to deal with binaural sound, (2) one-dimensional operators such as 1D convolutions, 1D max pooling and 1D average pool. We propose two networks, detailed in figure 1.

3.2. ResNet

The second architecture we use is ResNet (models C and D in table 1). ResNet [2] is a very efficient architecture that enables to train very deep neural networks, which have established the state of art result in many computer vision tasks. The proposed 1d-ResNet is based on a basic block described in figure 2, including a shortcut that uses a 1x1 convolution, and convolutions with 3x1 kernels. The architecture that we present here is composed of a first convolutional layer (2 input channels Conv1d, 32 output channels, kernel of size 64, stride of 4) followed by three modules. Each module is made of respectively 3, 4 and 3 basic blocks for model C, and 3,3,3 basic blocks for model D. Convolutions within each module are composed of respectively 32, 64 and 128 feature maps. Note that no max pooling is used in ResNet, and strides of 4 are used in the first convolution of each group of basic block. Average pooling is performed before the final fully connected layer.

4. MODEL COMPRESSION

4.1. Structured Pruning

Pruning of network parameters is a common technique used to decrease the number of non-zero parameters. We perform structured pruning, i.e. the parameters of whole convolution kernels are set to zero. The importance of convolution kernels is estimated using the L1 norm of its parameters, and the least important kernels are set to zero. Previous studies have shown that structured pruning can lead to high compression rates while keeping good performance on standard computer vision tasks [9, 10]. After the initial training of the model (see section 5.2), we perform pruning using an iterative approach based on fine-tuning (similar to [9]), as follows :

1. Ranking of convolution kernels' importance using the L1 norm.
2. Pruning of the least important ones by setting the corresponding parameters to zero (we used both 10 % and 20 % in our experiments).
3. Fine tuning of the pruned model on the training set dataset, with DA identical to initial training. We fix a relatively low learning rate (1e-5) and train the model using early stopping on validation set (see section 5.2).
4. Repeat from 1 until a stopping criterion is reached.

We use different stopping criterion for pruning : prune while the total number of nonzero parameters is above the challenge allowed maximum (model D), minimize parameter count while keeping an accuracy above the DCASE baseline of 87.3% (model A and C), or minimize parameter count while keeping an accuracy close to the full model (model B).

4.2. Quantization

After having performed training, pruning and fine-tuning iterations, our final step is the quantization of all model parameters. We quantize all inputs and parameters to floating point half precision, which uses 16 bits for each data. This level of precision enables to keep very similar test accuracy when compared with the full precision model, even slightly increasing the accuracy in some cases.

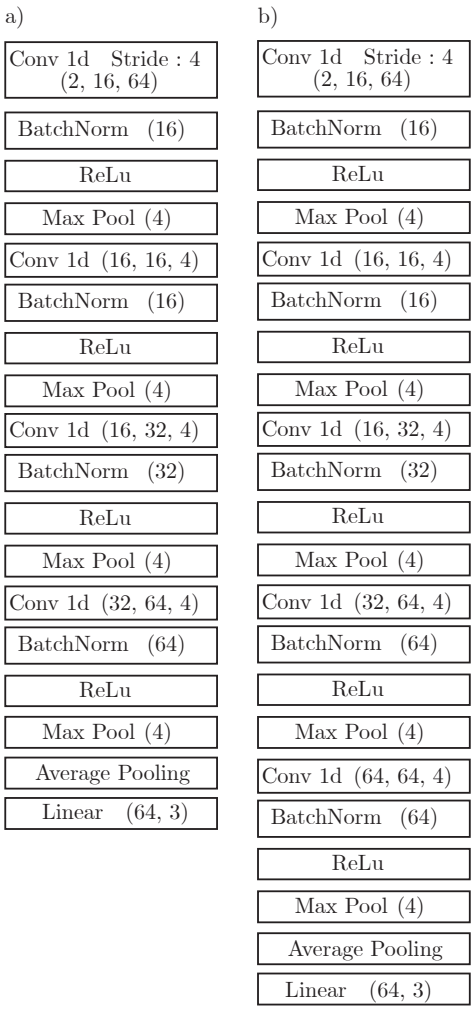

Figure 1: Two simple 1-dimensional CNN on raw waveform. Panel a corresponds to Model A and Panel b corresponds to model B. For Conv1d modules, the numbers correspond respectively to input feature maps, output feature maps, and kernel size. For the Linear module, numbers correspond to number of inputs, number of outputs.

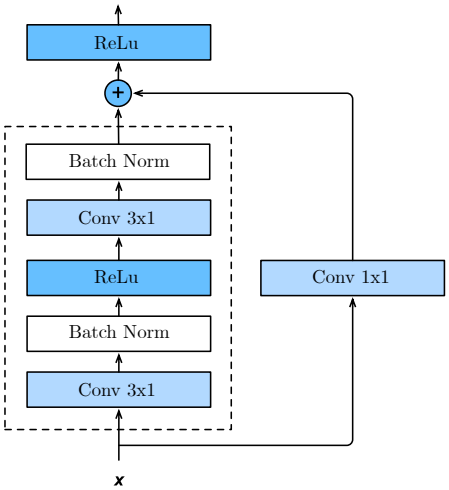

Figure 2: ResNet Block (figure adapted from [8]).

5. EXPERIMENTS

5.1. Datasets

The dataset for this task is based on TAU Urban Acoustic Scenes 2020 3Class [11]. All samples were recorded from the same device in different sites (shopping mall, metro, bus, ...). There are three possible acoustic scene categories : transportation, indoor and outdoor. Each audio sample are 10 second long, binaural, sampled at 48 kHz in 24 bits precision.

An extensive inspection of frequency content and frequency coherence of the development set has indicated that most of the signal energy is below 9000 Hz. Therefore, we resample all audio to 18 kHz in 16 bits precision, using the Fourier method ("resample" function in SciPy [12]). We generate a validation set using 20% of the available training data, and use the remaining 80% as training set.

5.2. Training protocol

All models are trained using an Adam optimizer with a starting learning rate of 0.001 and a batch size of 64. We use a scheduler to divide the learning rate by 2 when the loss on the validation set does not improve during five epochs. The model with the best accuracy on the validation set is kept and tested on the test set. DA is used on the training data only. Model C and D are trained using only CutMix, while Model A and B are trained using Temporal masking, filtering and additive noise. For all models, the training protocol is perfomed in the following sequence:

- Training until early stopping as indicated by validation set performance,
- Iterative structured pruning on parameters and fine tuning, as described in section 4,
- Quantization to floating point half precision, and final evaluation on the test set.

Model name	Accuracy	Loss	Total params	Non-zeros params	Size (KB)
Model A	87.6	0.360	13632	12160	23.8
Model B	90.9	0.288	30080	29888	58.4
Model C	87.6	0.379	398400	130730	255.3
Model D	91.2	0.269	373696	238896	466.6

Table 1: Performance and model complexity of the submitted models. Parameters are encoded in 16 bit floating point half precision. Note that according to subtask B rules, batch norm layers are not included in the calculation of model parameters.

6. RESULTS AND DISCUSSION

Table 1 presents the results obtained by the four models in the test data of the development dataset, as well as the parameter count. As specified in the task rules, batch norm layers are not included in parameter count. As our approach does not use any predefined feature extraction (e.g. spectrogram), we included the whole model in the calculation of model parameters, including the first convolutional layer. Layer-wise parameter count, including pruning rates, are detailed on our github repository, as well as model definitions in pytorch (`https://github.com/brain-bzh/dcase-2020-task1-subtaskB`).

Our results show the feasibility of using raw binaural waveforms to train a model 20 times smaller than the 500 KB limit (model A), as well as a model achieving 90.9 % of accuracy, while being 8 times smaller than the challenge limit (model B). We also provide results for larger ResNet models approaching the 500 KB (model C and D). Note that model D does not perform significantly better than model B in terms of macro-average accuracy, but has a lower loss, which potentially indicates a better generalization power.

7. REFERENCES

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[4] 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 *2017 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE, 2017, pp. 131–135.

[5] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, "Samplernn: An unconditional end-to-end neural audio generation model," *arXiv preprint arXiv:1612.07837*, 2016.

[6] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[7] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6023–6032.

[8] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into Deep Learning*, 2020, https://d2l.ai.

[9] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[10] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5058–5066.

[11] 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

[12] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, *et al.*, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.