RECEPTIVE FIELD REGULARIZED CNNS WITH TRADITIONAL AUDIO AUGMENTATIONS

Technical Report

Tobias Morocutti

Johannes Kepler University Linz
Linz, Austria
tobias.morocutti@gmx.at

Diaaeldin Shalaby

Johannes Kepler University Linz
Linz, Austria
diaa.shalaby14@gmail.com

ABSTRACT

This technical report describes our system for Task 1 (Low-Complexity Acoustic Scene Classification) of the DCASE2022 Challenge. Due to the limited allowed complexity of the model to submit, we use a teacher-student approach. The teacher is a Receptive Field (RF) regularized CNN model and the student is a simpler 5-layer CNN with batch normalization, dropout and maxpool layers. In addition, some data augmentation techniques, such as adding gaussian noise, shifting, pitch shifting and time stretching are adopted for expanding the diversity of the dataset. Our system achieves an accuracy of 53.4% and a multiclass cross-entropy (log loss) of 1.279 on the development dataset. The student model has 21,930 parameters and a Multiply accumulate count of 9.775 million.

Index Terms— Acoustic scene classification, convolutional neural networks, low-complexity, teacher-student architecture

1. INTRODUCTION

Detection and Classification of Acoustic Scenes and Events (DCASE) [1] is an annual challenge, attracting attention to the field of machine learning and encouraging research. Task 1 of the challenge was Low Complexity Acoustic Scene classification (ASC) which can be described as a multi-class supervised classification problem. The goal of this task is to recognize a set of given environment classes (e.g., airport, urban park or metro station) from sound recordings. The sample recordings were all 1 second in length and they were collected from 10 different audio scenes from 12 European cities, using four different real and 11 simulated devices [2]. Variants of convolutional neural networks (CNNs) were the top ranking architectures in the previous year's DCASE challenge. Receptive Field (RF) regularized CNNs turned out to be particularly successful (cf. [3] [4] [5]). Moreover, over-parameterized Neural Networks showed better generalization [5] [6] behavior. Submissions for this year's challenge have two computational complexity constraints: The number of parameters, which must not exceed 128K and the maximum number of multiply-accumulate instructions (MACs) per inference, which must not exceed 30 MMAC (Million MACs). The complexity constraints for the challenge make it a perfect candidate for a Teacher-Student CNN approach. This uses two types of classifiers: the first being a large over-parameterized model (teacher) and the second being a much smaller model (student), that adheres to the complexity constraints [7].

2. DATA PROCESSING

2.1. Audio processing

After briefly experimenting with MFCCs, we soon realized that many of last years participants of the challenge used mel-spectrograms for the network input since they contain more information, and can therefore increase the performance of simple CNNs greatly. The input is down-sampled to 22.05 kHz. We used a Short Time Fourier Transform (STFT) with a window size of 2048 and 25% overlap to extract the input features. We then weigh the frequencies in the resulting spectrograms perceptually using the A-weighting curve and apply a Mel-scaled filter bank in a similar fashion to Dorfer et al. [8].

While investigating the hyperparameters for creating the mel-spectrograms we noticed that using 256 Mel frequency bins to create the networks' inputs as opposed to 128 Mel frequency bins enables the networks to detect more useful information.

2.2. Data augmentation

To have more data to train the networks with, and to make our networks generalize better, we augment our whole training dataset once. We experimented with different augmentation techniques and found it most useful to randomly choose from four different augmentations. We add Gaussian noise to the training samples, to help the network learn to be able to classify samples which have some noise themselves. Time stretching, pitch shifting and time shifting helps the network to be more robust in three different ways. The augmentations are applied sequentially in the order stated above with a probability of 0.25 per method.

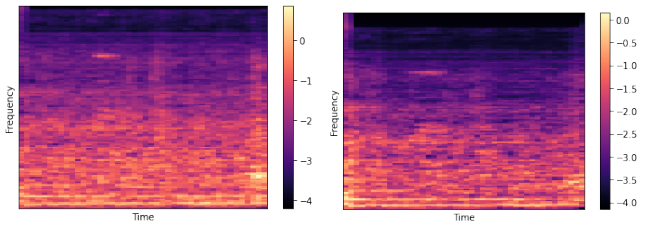

Figure 1: Original vs augmented spectrogram

3. ARCHITECTURES

Both the teacher and the student model are fully convolutional neural networks with an Adaptive Average Pooling at the very end of the network to get an output vector of size ten. In both networks, batch-normalization is applied after each convolutional layer.

3.1. Teacher model

We use a modified version of the damped CNN proposed in [5] as our bigger network. A damped CNN regulates the influence an input has on a convolutional neuron, based on its position in the neuron's receptive field which means that the further the input is from the center the less influence it will have.

The model consists of 5 stages. A stage can be described as multiple layers or blocks. The input and output stage each contain one convolutional layer while each of the three stages in the middle contains four blocks. As illustrated in Figure 2, a block encompasses two sequential pairs of a convolutional layer and a batch-normalization layer.

Teacher network architecture				
Stages	Blocks/Layers	C	K	O
Stage 1	input	1	-	44 x 256
	Conv2dDamped	128	5	21 x 127
	BatchNorm2d	128	-	-
Stage 2	block_{21}	128	3 / 1	21 x 127
	maxpool	128	2 x 2	10 x 63
	block_{22}	128	3 / 3	10 x 63
	block_{23}	128	3 / 3	10 x 63
	block_{24}	128	3 / 3	10 x 63
Stage 3	skip block_{31}	256	3 / 1	10 x 63
	block_{32}	256	1 / 1	10 x 63
	block_{33}	256	1 / 1	10 x 63
	block_{34}	256	1 / 1	10 x 63
Stage 4	skip block_{41}	512	1 / 1	10 x 63
	block_{42}	512	1 / 1	10 x 63
	block_{43}	512	1 / 1	10 x 63
	block_{43}	512	1 / 1	10 x 63
Stage 5	Conv2dDamped	10	1	10 x 63
	BatchNorm2d	10	-	10 x 63
	AdaptiveAvgPool2d	10	-	1 x 1

Table 1: Description of the architecture of the teacher network where a skip block is a block with the convolutional skip connection and dropout layer with dropout probability of $p = 0.1$.
C = number of channels; K = kernel size of either a maxpooling or convolutional layer or the kernel sizes of the first and second convolutional layer of a block; O = output shape

As described in Table 1, the first blocks of the third and fourth stage consist of an additional convolutional layer as skip connection and a Spatial-Dropout layer with dropout probability of $p = 0.1$. Our experiments showed that Spatial-Dropout helps the model to generalize better, and reduces the variance of validation loss.

While the skip connections have a kernel size of 1 the kernel sizes of the convolutional layers in the different blocks differ and are displayed in Table 1. The number of channels doubles from the first to the second and from the second to the third middle stage, starting with 128 channels. Furthermore, we use the ELU [9] activation function to train the teacher network with 3,956,628 parameters and 2,822.779 MMACs in total. The ELU nonlinearity not only outperformed other activation functions like ReLU [10] or Leaky ReLU [11] but additionally ensures a more stable training.

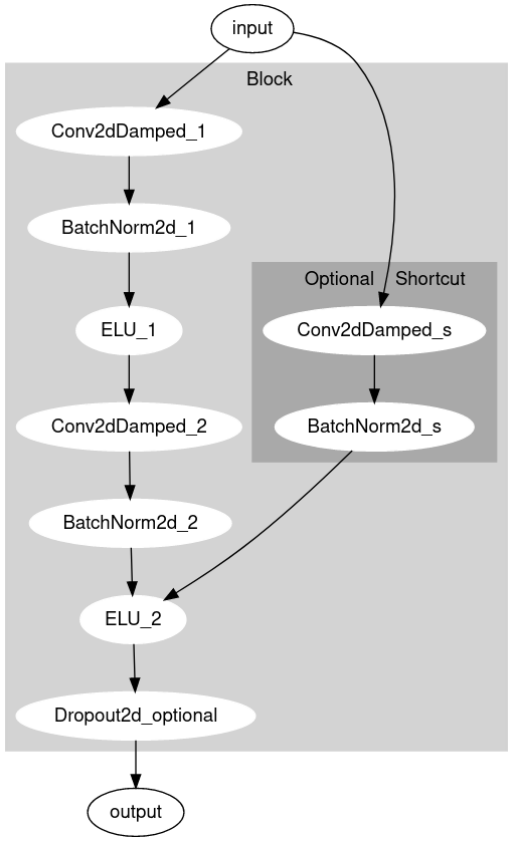

Figure 2: An illustration of the internal structure of a block contained in the teacher network.

Investigations with different sizes of the receptive field showed that the network performs best on the validation set with $\rho = 7$, which corresponds to 44 x 67 pixels. This results in a receptive field big enough to contain a lot of information and small enough to only contain the most relevant information.

3.2. Student model

Since results of the past years showed that using multiple networks for the final prediction can be very helpful, we implemented one student network architecture which has just below one third of the allowed complexity (21,930 parameters / 9.775 MMACs). The student model consists of two middle stages with one block each, the output layer uses 2-dimensional adaptive average pooling in a similar manner as the teacher network.

A block in the student network equals the architecture of a block in the teacher network without the optional Dropout or shortcut-layers. Additionally, due to the requirement to quantize the student models we used conventional convolutional layers instead of the damped version. In total, the student network has 5 convolutional layers and 2 max-pooling layers. To reduce the complexity of the model further, we use only 32 channels throughout the model.

Student network architecture					
Stages	Blocks/Layers	C	K	O	P
Stage 1	input	1	-	44 x 256	-
	Conv2d	32	5 x 6	21 x 127	960
	BatchNorm2d	32	-	-	64
Stage 2	maxpool	32	2 x 2	10 x 63	-
	block$_{21}$:				
	Conv2d	32	3 x 3	10 x 63	9216
	BatchNorm2d	32	-	-	64
	Conv2d	32	1 x 1	10 x 63	1024
	BatchNorm2d	32	-	-	64
	maxpool	32	2 x 5	5 x 12	-
Stage 3	**block**$_{31}$				
	Conv2d	32	3 x 3	5 x 12	9216
	BatchNorm2d	32	-	-	64
	Conv2d	32	1 x 1	5 x 12	1024
	BatchNorm2d	32	-	-	64
Stage 5	Conv2d	10	1 x 1	-	-
	BatchNorm2d	10	-	-	-
	A. AvgPool2d	10	-	-	-

Table 2: Description of the architecture of the student network. C = number of channels, K = kernel sizes, O = output shape, P = number of parameters

3.3. Training

We used the Adam optimizer and a mini-batch size of 22 for the training process of both neural networks. This number enabled an easier tuning of the learning rate scheduler. The teacher and the student network get trained for 25 and 51 or 100 epochs respectively. Training the student model with 51 epochs ensures that we do not overfit to the training data. To improve generalization and robustness we used the Mixup approach [12] with a Beta distribution parameter of 0.3. As stated above, we use an ensemble of three instances of the same student network architecture to produce the final predictions. By averaging the logits of multiple smaller models our system generalizes better than one single large network.

3.3.1. Weight initialization

As weight initialization for the first convolutional layer of a block we take the He-initialization [13] rescaled by a Fixup multiplier [14]. This multiplier equals

$$\frac{1}{\sqrt{i}} \tag{1}$$

where i stands for the block index. For the second convolutional layer of a block we set the weights to zero and for the optional convolutional skip connection we use the original He-initialization.

3.3.2. Distilling the knowledge in a Neural Network

Inspired by the approach in [15], we train our smaller neural network by using soft targets as labels. The soft targets are calculated by using the logits of the bigger model and an additional temperature parameter $T = 5$. More precisely, instead of the standard softmax function

$$q_i = \frac{\exp z_i}{\sum_j \exp z_j} \tag{2}$$

we use a modified version where we divide each input to the softmax function by the temperature such that we get

$$q_i = \frac{\exp \frac{z_i}{T}}{\sum_j \exp \frac{z_j}{T}} \tag{3}$$

where q_i is the probability for class i, z_i is the logit and T represents the temperature.

The total loss of the student model consists of the soft target loss and the true label loss, with weights of 20 and 1 respectively. Even though we weigh the true label loss relatively small, the performance would suffer if we would not include it in the loss at all.

3.3.3. Learning Rate Scheduler

We monitored the validation loss and adapted the learning rate schedulers such that the loss curve stays as flat as possible.
For training the teacher model, we therefore implemented a linear learning rate scheduler with multiple phases. For the first three epochs we train our network with learning rates of 1e-4, 9e-5 and 5e-5. After we train our network for four epochs with the same learning rate, we decrease it in the next four epochs linearly to 1e-5 before we descend it linearly to 2e-6 for the next eleven epochs. For the final six epochs our learning rate stays the same.

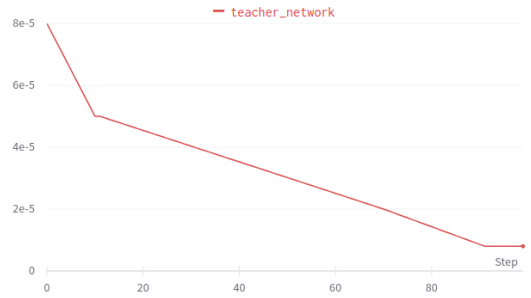

Figure 3: Learning rate schedulers of the teacher network.

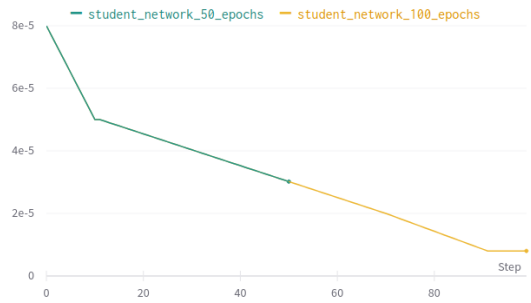

Figure 4: Learning rate schedulers of the student network.

We used simpler settings for the learning rate scheduler of the student models. After the learning rate descends linearly from 8e-5 to 5e-5 in the first 10 epochs, it decreases to 8e-6 for the next 80 epochs before the network is trained with this learning rate for another 10 epochs. In case we train the network only for 51 epochs, we use the exact same learning rate scheduler.

4. RESULTS

Our four submissions consist of one model trained just on the training set for 100 epochs and three models trained on the whole development set for 51 epochs. All of our submitted predictions were created using the average logits of three student networks. For one submission we used a dropout probability of 0.4 instead of 0.3. We got our forth predictions by training the teacher model with a batch normalization layer at the beginning of the network. In Table 3 below is a short overview of all the settings which differ in our submissions.

Settings	model 1	model 2	model 3	model 4
Epochs	51	100	51	51
Dropout p	0.3	0.3	0.3	0.4
BN at the beginning	False	False	True	False
Trained on the whole development set	True	False	True	True

Table 3: Settings of the different submitted student models ranked by lowest validation loss when trained on the training set.

Scenes	model 1		model 2	
	Log loss	Accuracy	Log loss	Accuracy
Airport	1.3	37.9 %	1.205	42.7 %
Bus	0.983	56.3 %	1.116	50.0 %
Metro	1.272	42.9 %	1.21	43.2 %
Metro Station	1.388	41.7 %	1.337	42.5 %
Park	0.575	73.1 %	0.607	74.4 %
Public Square	1.59	26.8 %	1.506	28.8 %
Shopping Mall	0.928	57.8 %	1.05	52.5 %
Street Pedestrian	1.66	24.1 %	1.502	30.7 %
Street Traffic	0.623	72.2 %	0.722	69.1 %
Tram	1.236	48.3 %	1.236	48.7 %

Table 4: The performance on the validation set by class of the two best performing models.

As clearly visible in Table 4 our networks perform much better in some acoustic scenes than in others. Since the results of the baseline network show a very similar unevenness of the performance on the different scenes we can conclude that scenes like public square or street pedestrian are harder to detect for convolutional neural networks than ones like street traffic and park.

Devices	model 1		model 2	
	Log loss	Accuracy	Log loss	Accuracy
A	0.888	59.9 %	0.877	60.6 %
B	1.207	48.6 %	1.222	47.2 %
C	1.097	52.4 %	1.119	51.5 %
S1	1.188	47.5 %	1.193	47.0 %
S2	1.214	47.9 %	1.223	50.2 %
S3	1.124	52.3 %	1.094	54.1 %
S4	1.313	46.7 %	1.341	45.3 %
S5	1.238	45.6 %	1.251	44.8 %
S6	1.343	42.2 %	1.357	43.1 %

Table 5: The performance on the validation set by device of the two best performing models.

The relatively low log loss on samples recorded on the device A which can be seen in Table 5 can be explained due to the fact that almost three fourths of the data of the training split are recordings from said device.

5. CONCLUSION

In this work, we described our data processing, data augmentation, network architectures, training of those networks and, finally, the results. We tested many different configurations of the mel-spectrograms. We experimented with many different network architectures and hyperparameters, ending up with an oversized teacher network and a relatively small student network architecture small enough to be allowed to use three instances of them.

6. ACKNOWLEDGEMENT

We are both students of the bachelor's degree in Artificial Intelligence at the Johannes Kepler University Linz. This challenge has been a fantastic opportunity for us to get exposed to an interesting research problem. Hence, we would like to thank the Institute of Computational Perception at the JKU and its head Professor Gerhard Widmer for giving us this unique opportunity and letting us use the institute's resources for training. In addition we want to thank Rainer Kelz, Paul Primus and Khaled Koutini for giving us guidance during the challenge.

7. REFERENCES

[1] I. Martín-Morató, F. Paissan, A. Ancilotto, T. Heittola, A. Mesaros, E. Farella, A. Brutti, and T. Virtanen, "Low-complexity acoustic scene classification in dcase 2022 challenge," 2022. [Online]. Available: https://arxiv.org/abs/2206.03835

[2] 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, pp. 56–60. [Online]. Available: https://arxiv.org/abs/2005.14623

[3] K. Koutini, S. Jan, and G. Widmer, "CP-JKU submissions to DCASE21: Cross-device audio scene classification with wide sparse frequency-damped CNNs," DCASE2021 Challenge, Tech. Rep., June 2021.

[4] B. Kim, S. Yang, J. Kim, and S. Chang, "QTI submission to DCASE 2021: Residual normalization for device-imbalanced acoustic scene classification with efficient design," DCASE2021 Challenge, Tech. Rep., June 2021.

[5] K. Koutini, F. Henkel, H. Eghbal-zadeh, and G. Widmer, "CP-JKU submissions to DCASE'20: Low-complexity cross-device acoustic scene classification with RF-regularized CNNs," DCASE2021 Challenge, Tech. Rep., 2020.

[6] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proceedings of the National Academy of Sciences*, vol. 116, p. 201903070, 07 2019.

[7] M. Brahimi, M. Saïd, B. Kamel, and A. Moussaoui, "Deep interpretable architecture for plant diseases classification," 09 2019, pp. 111–116.

[8] M. Dorfer and G. Widmer, "Training general-purpose audio tagging networks with noisy labels and iterative self-verification," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, November 2018, pp. 178–182.

[9] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," 2015. [Online]. Available: https://arxiv.org/abs/1511.07289

[10] A. F. Agarap, "Deep learning using rectified linear units (relu)," 2018. [Online]. Available: https://arxiv.org/abs/1803.08375

[11] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," 2015. [Online]. Available: https://arxiv.org/abs/1505.00853

[12] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," 2017. [Online]. Available: https://arxiv.org/abs/1710.09412

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," 2015. [Online]. Available: https://arxiv.org/abs/1502.01852

[14] H. Zhang, Y. N. Dauphin, and T. Ma, "Fixup initialization: Residual learning without normalization," *CoRR*, vol. abs/1901.09321, 2019. [Online]. Available: http://arxiv.org/abs/1901.09321

[15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015. [Online]. Available: https://arxiv.org/abs/1503.02531