ACOUSTIC SCENE CLASSIFICATION WITH FULLY CONVOLUTIONAL NEURAL NETWORKS AND I-VECTORS

Technical Report

Matthias Dorfer *Bernhard Lehner* *Hamid Eghbal-zadeh*
Christoph Heindl *Fabian Paischer* *Gerhard Widmer*

Institute of Computational Perception (CP-JKU),
Johannes Kepler University Linz, Austria
matthias.dorfer@jku.at

ABSTRACT

This technical report describes the CP-JKU team's submissions for Task 1 - Subtask A (Acoustic Scene Classification, ASC) of the DCASE-2018 challenge. Our approach is still related to the methodology that achieved ranks 1 and 2 in the 2016 ASC challenge: a fusion of i-vector modelling using MFCC features derived from left and right audio channels, and deep convolutional neural networks (CNNs) trained on spectrograms. However, for our 2018 submission we have put a stronger focus on tuning and pushing the performance of our CNNs. The result of our experiments is a classification system that achieves classification accuracies of around 80% on the public Kaggle-Leaderboard.

Index Terms— audio scene classification, convolutional neural networks, deep learning, i-vectors, late fusion

1. INTRODUCTION

This report describes our submissions for Task 1 (Subtask A) – Acoustic Scene Classification (ASC) in the DCASE-2018 Challenge[1][1]. The basic approach to building our final classifier is based on the methodology we developed for the DCASE-2016 Challenge, which took first and second rank in 2016 and has been described in detail in [2]. However, compared to the original version of our system we put a stronger focus on tuning our neural networks for this year's submission. In Summary, the method combines deep Fully Convolutional Neural Networks (FCNN) trained on perceptually weighted spectrograms with an i-vector modeling approach, and fuses these via linear logistic regression.

In the end, we submitted the predictions of four classifiers (see Section 6 below): (1) a averaged ensemble of CNNs; (2) a calibrated ensemble of i-vector-based classifiers; (3) a combination of i-vector and CNN classifiers obtained by averaging; and, as the most complex model, (4) an ensemble of i-vector and CNN results calibrated and fused via linear logistic regression. We estimate the performance of our methods on the publicly available Kaggle-Leaderboard[2]. The best classifier, when evaluated in this way, achieves a classification accuracy of 80.00%.

[1] `http://dcase.community/challenge2018/`

[2] `https://www.kaggle.com/c/dcase2018-task1a-leaderboard`

2. RECAPITULATION: THE DCASE-2016/17 APPROACHES

In DCASE 2016, we proposed a hybrid approach using binaural i-vectors and CNNs [2]. We adapted the i-vector features for ASC by 1) tuning MFCC features by selecting the best performing windowing scheme and cepstral coefficients, 2) extracting i-vectors from different audio channels (left, right, average and difference) and 3) combining the i-vector cosine scores of different channels via score averaging. Our CNN was a *VGG-style* ConvNet trained on short segments of spectrograms that made its final decision by combining the predictions of all the segments from a scene recording. In the end, linear logistic regression was used to fuse the averaged i-vector scores with the CNN prediction scores. In classifying the unseen test data, we combined predictions of the models trained on each fold in the provided cross validation split.

3. FULLY CONVOLUTIONAL NEURAL NETWORKS

In this section we describe the neural network architectures as well as the optimization strategies used for training our audio scene classification networks.

3.1. Network Architecture

Our basic network architecture is depicted in Table 1 (in total we use three slightly modified versions of this architecture for our submission). The feature learning part of our model follows, as in our last years submissions, a *VGG style network* [3] and the classification part of the network is designed as a global average pooling layer [4] over 10 feature maps (one for each class) followed by a *softmax* activation. As an activation function within the network we use Rectified Linear Units (ReLUs). What is a bit special about our model in Table 1 is that we replace some of the Dropout-Layers with Gaussian-Noise-Layers. This helped us to improve the performance on the provided validation set by a bit less than one percentage point compared to only using Dropout. This was also reflected when we checked our performance on the public Kaggle-Leaderboard.

3.2. Data Preparation

For data preprocessing we resample the audio signals to 22050 Hz, and compute a two-channel (left and right) Short Time Fourier Transform (STFT) using 2048-sample windows and a hop-size of

Table 1: Model Specifications. BN: Batch Normalization, ReLU: Rectified Linear Unit, CCE: Categorical Cross Entropy. For training a constant batch size of 100 samples is used.

Input $3 \times 256 \times 128$
5×5 Conv(pad-2, stride-2)-42-BN-ReLU
3×3 Conv(pad-1, stride-1)-42-BN-ReLU
2×2 Max-Pooling + GaussianNoise(1.00)
3×3 Conv(pad-1, stride-1)-84-BN-ReLU
3×3 Conv(pad-1, stride-1)-84-BN-ReLU
2×2 Max-Pooling + GaussianNoise(0.75)
3×3 Conv(pad-1, stride-1)-168-BN-ReLU
Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-168-BN-ReLU
Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-168-BN-ReLU
Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-168-BN-ReLU
2×2 Max-Pooling + GaussianNoise(0.75)
3×3 Conv(pad-0, stride-1)-336-BN-ELU
Drop-Out(0.5)
1×1 Conv(pad-0, stride-1)-336-BN-ELU
Drop-Out(0.5)
1×1 Conv(pad-0, stride-1)-10-BN
GaussianNoise(0.3)
Global-Average-Pooling
10-way Soft-Max

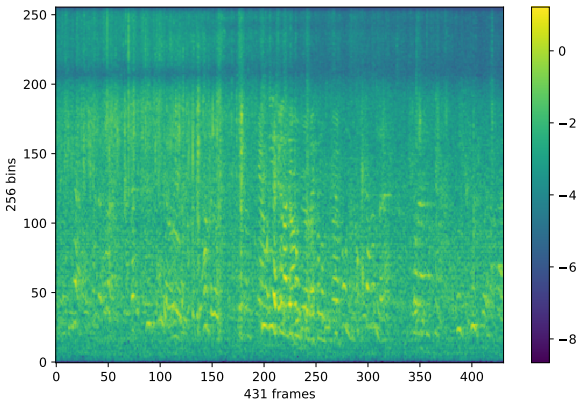

Figure 1: Left channel of perceptually weighted mel-spectrogram used for our classification networks.

512 samples. Given this raw spectrogram, we apply a perceptual weighting to the individual frequency bands of the power spectrogram [5][3]. As a last step, we apply a mel-scaled filterbank yielding 431-frame spectrograms with 256 frequency bins per data point. Figure 1 shows one channel of such a spectrogram. Before presenting this data to our networks we compute the difference between the left and the right channel and concatenate it as a third input feature map with the original two channel spectrogram. All three channels are mean centered and normalized to have standard deviation one along the individual frequency bins.

[3] `librosa.core.perceptual_weighting`

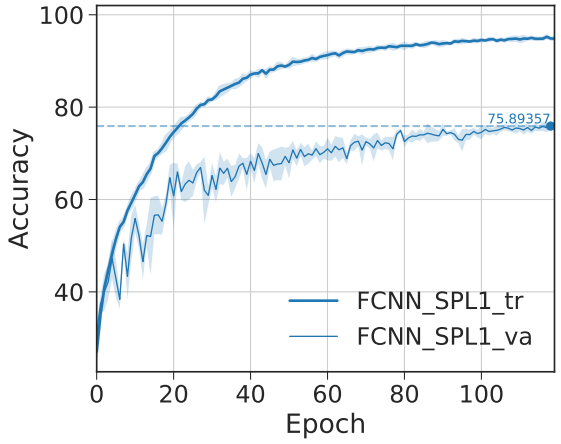

Figure 2: Training and validation accuracy on the provided evaluation split setup.

3.3. Training Procedure

At training time we show the network only randomly selected 128 frame excerpts of the full spectrograms, while presenting the whole spectrogram during testing. This is technically possible as our network is fully convolutional and can therefore processes audio excerpts of varying length. Intuitively presenting shorter excerpts for training should weaken the effect over-fitting to the individual training examples as we end up with a much larger amount of shorter sub-excerpts. To further prevent over-fitting we also apply mixup-data augmentation [6] with an $alpha$ of 0.2.

As optimizer we use the ADAM update rule [7] with an initial learning rate of 0.001 and a mini-batch size of 100 samples. Each model is trained for 120 epochs where we linearly decay the learning rate to zero starting from epoch 25. Figure 2 shows the evolution of training and validation accuracy on the predefined evaluation split. We re-train the model four times using different random seeds and show mean and standard deviation to get a reliable estimate of the model's behavior. Note that the validation accuracy does not decrease towards the end of the full training time. Having a fixed learning rate schedule and this model behavior on the validation set enables us to make use of the full training data for our final models. Therefore we randomly re-split the entire development data set into four distinct folds and retrain our networks. Figure 3 shows the performance of the same model as in Figure 2 trained on the alternative split of the data (mean and standard deviation of the four folds). We observe that the model achieves a much high classification accuracy also on the validation set. This becomes obvious when we keep in mind that the recording locations are now represented in both the training and the validation set. However, as the model's performance does not degrade on the validation set while training we expected that it should still perform well on unseen locations. A submission to the public Kaggle-Leaderboard confirmed this with an average accuracy close to 80%.

4. THE I-VECTOR METHOD

In this section we describe the i-vector method and the features we use for it.

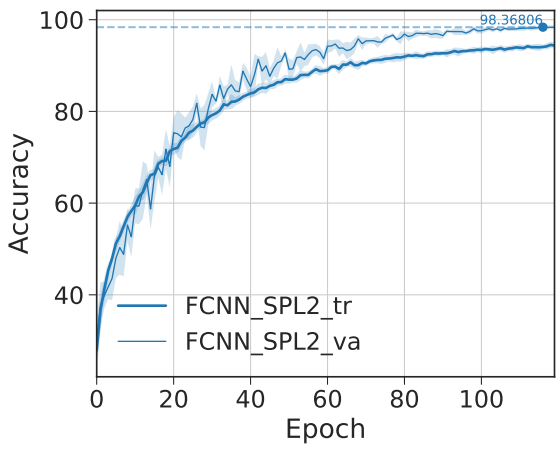

Figure 3: Training and validation accuracy on a random data split.

4.1. Features: MFCCs

We extract the features the same way as in our previous work [8]. We suggest to use 23 MFCCs (without 0^{th} MFCC) extracted by applying a 20 ms observation window without any overlap. 18 MFCC deltas (including the 0^{th} MFCC delta), and 20 MFCC double deltas (including the 0^{th} MFCC double delta) are extracted by applying a 60 ms observation window, placed symmetrically around a 20 ms frame. Regardless of the observation window length, we use 30 triangle shaped mel-scaled filters in the range [0-11 kHz]. Our feature vector has therefore a dimensionality of 61.

Since the audio recordings are rather short, which is problematic in general and a particular problem for i-vector modelling approaches [9], we suggest to augment the data for training and testing. We extract MFCCs from left and right channels, and concatenate the resulting features into a single feature matrix with twice the length in time. Additionally, we extract MFCCs from pitch shifted audio recordings up and down by 100, 200, and 300 cents, respectively.

As a consequence, we end up with a feature matrix that has the same size as the feature matrix extracted from a 140 s audio recording (10 s * 2 (left/right) * 7 (original + shifted); 61 features * 7000 observations) for every audio recording.

4.2. I-vector Backend

In principle, we use the same i-vector extraction pipeline, and the same post-processing of the results as in [2]. A *Universal Background Model (UBM)* is first trained on the MFCCs from the training set. This UBM is then used to learn the i-vector space known as *Total Variability Space (TVS)*. Using UBM and TVS, i-vectors are extracted from the training and test set. The i-vectors are normalized to norm 1 and a *Linear Discriminant Analysis (LDA)* is learned using the i-vectors of the training set. All the i-vectors are then projected into LDA space. As a next step, a *Within-Class Covariance Normalization (WCCN)* [10] is learned from these projected i-vectors. Again, all the i-vectors, which previously were projected using LDA, are further projected via WCCN. From each class, all its projected i-vectors are averaged into one i-vector which is used as the representative of the class for the scoring step. Projected i-vectors from the test set are then scored using a cosine scoring and the class-averaged i-vectors. The class with the maximum score is selected as the label for each test i-vector.

5. SCORE CALIBRATION AND LATE FUSION

As outlined above, we train one i-vector-based model and three different CNNs for each fold. This results in 4 models per fold. We then follow a two-stage fusion and calibration procedure to obtain the test results: first, we calibrate and fuse the models fold-wise. Then, we average the predictions of the fold-wise fused models. Figure 5 shows an overview of our approaches.

We calibrate the prediction scores and fuse the predictions of the individual models using *linear logistic regression*. In particular, for each fusion, we train classifier weights W and class biases b using the validation set[4], and compute the fused prediction $y = \sigma\left(XW + b\right)$, where σ is the softmax function, and X are the class probabilities given by each classifier. As shown in Figure 5, we use this approach for submissions $\text{IVEC}_{\text{calib}}$, and $\text{All}_{\text{calib}}$; for CNN_{avg}, we use plain averaging of the individual models across all folds instead.

6. RESULTS

In the following we list our submitted systems and their respective performances on the public Kaggle-Leaderboard and the final evaluation set.

6.1. Submissions

We provide four different submissions based on the methods described in the previous sections:

1. CNN_{avg}: Averaging of all CNN models
2. $\text{IVEC}_{\text{calib}}$: Late calibrated fusion of all Binaural I-vectors
3. $\text{All}_{\text{calib-avg}}$: Late calibrated fusion of averaged CNN models and averaged Binaural I-vectors
4. $\text{All}_{\text{calib-sep}}$: Late calibrated fusion of all CNN models and all Binaural I-vectors

As a final prediction for the unseen test set we submit for each system the averaged predictions over all four folds.

6.2. Performance on the Public Kaggle-Leaderboard

Table 2 summarizes the results of our submissions on the public Kaggle-Leaderboard. Our neural network ensemble achieves 80% on the public leaderboard. In contrast, the calibrated fusion of the i-vector system achieves 65.8%. What is surprising is that when combining our best FCNN system with the i-vector system[5] using late fusion we are still able to improve by 0.5% resulting in our best submission at 80.5% .

6.3. Performance on the evaluation set

Will be completed when the official results are released ...

[4] Ideally, we would use a dedicated calibration set to reduce the risk of over-fitting.

[5] Note that our i-vector system is identical to our last year's submission and was not tuned to this years challenge and data.

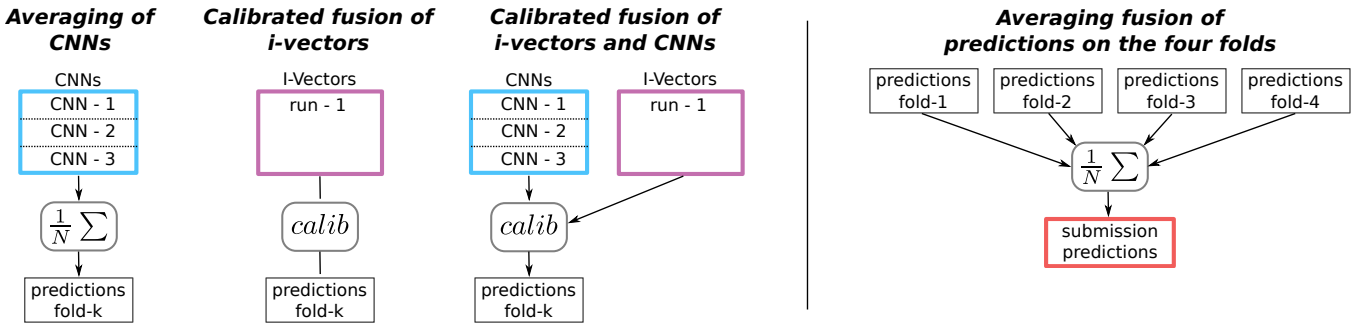

Figure 4: Overview of fusion schemes used for our submissions.

Table 2: Audio scene classification accuracy on the public Kaggle-Leaderboard.

	CNN_{avg}	$IVEC_{calib}$	$All_{calib\text{-}avg}$	$All_{calib\text{-}sep}$
(%)	80.00	65.83	80.50	-

7. CONCLUSION

This short report describes our DCASE-2018 ASC approach for Subtask A of Task 1. It is a combination of three different FCNN trained on perceptually weighted spectrograms and an i-vector system based on MFCCs. An ensemble of three different FCNNs was able to achieve a competitive Kaggle- Leaderboard performance of 80%. When fusing this neural network system with our i-vectors we were able to achieve our best performance of 80.5%.

8. ACKNOWLEDGMENTS

This work was partly supported by the Austrian Ministry for Transport, Innovation and Technology, the Ministry of Science, Research and Economy, and the Province of Upper Austria in the frame of the COMET center SCCH.

9. REFERENCES

[1] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," 2018, submitted to DCASE2018 Workshop. [Online]. Available: https://arxiv.org/abs/1807.09840

[2] H. Eghbal-Zadeh, B. Lehner, M. Dorfer, and G. Widmer, "Cp-jku submissions for dcase-2016: A hybrid approach using binaural i-vectors and deep convolutional neural networks," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

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

[4] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[5] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.

[6] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[7] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[8] H. Eghbal-zadeh, B. Lehner, M. Dorfer, and G. Widmer, "A hybrid approach with multi-channel i-vectors and convolutional neural networks for acoustic scene classification," *arXiv preprint arXiv:1706.06525*, 2017.

[9] A. Kanagasundaram, R. Vogt, D. B. Dean, S. Sridharan, and M. W. Mason, "I-vector based speaker recognition on short utterances," in *Proc. of the 12th Annual Conference of the International Speech Communication Association*. International Speech Communication Association (ISCA), 2011.

[10] A. O. Hatch, S. S. Kajarekar, and A. Stolcke, "Within-class covariance normalization for svm-based speaker recognition." in *INTERSPEECH*, 2006.