CLASSIFYING SHORT ACOUSTIC SCENES WITH I-VECTORS AND CNNS: CHALLENGES AND OPTIMISATIONS FOR THE 2017 DCASE ASC TASK

Bernhard Lehner *Hamid Eghbal-zadeh* *Matthias Dorfer*
Filip Korzeniowski *Khaled Koutini* *Gerhard Widmer*

Department of Computational Perception (CP-JKU),
Johannes Kepler University Linz, Austria
bernhard.lehner@jku.at

ABSTRACT

This report describes the CP-JKU team's submissions for Task 1 (Acoustic Scene Classification, ASC) of the DCASE-2017 challenge, and discusses some observations we made about the data and the classification setup. Our approach is based on 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 raw spectrograms. The data provided for the 2017 ASC task presented some new challenges – in particular, audio stimuli of very short duration. These will be discussed in detail, and our measures for addressing them will be described. The result of our experiments is a classification system that achieves classification accuracies of around 90% on the provided development data, as estimated via the prescribed four-fold cross-validation scheme. On the unseen evaluation data, our best performing method achieved 73.8% and 5th place in the team ranking.

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

1. INTRODUCTION

This report describes our submissions for Task 1 – Acoustic Scene Classification (ASC) in the DCASE-2017 Challenge [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 [1]. The method, which will be briefly recapitulated in the next section, combines an i-vector modelling approach with a deep convolutional neural network (CNN) trained on raw spectrograms, and fuses these via linear logistic regression.

However, this year's challenge presented a new difficulty: the provided audio snippets are rather short – 10 seconds rather than the 30 seconds as used in the 2016 challenge. This is bound to be problematic for any learning and classification approach, but particularly so for i-vector-based models, which have been shown, in the speaker recognition literature, to suffer from a 'short utterance problem' [2]. Consequently, we need to increase the amount of information that is available in both the training and testing stage; a corresponding data augmentation strategy will be proposed in Section 3 below.

[1]`www.cs.tut.fi/sgn/arg/dcase2017/challenge/`

In the end, we submitted the predictions of four classifiers (see Section 7 below): (1) a calibrated ensemble of i-vector-based classifiers; (2) a calibrated ensemble of CNNs; (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 openly accessible DCASE-2017 [3] data set, using the prescribed cross-validation scheme. The best classifier, when evaluated in this way, achieves a classification accuracy of 91.29%.

2. RECAPITULATION: THE DCASE-2016 APPROACH

In DCASE 2016, we proposed a hybrid approach using binaural i-vectors and CNNs [1]. 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. OPTIMISING THE DATA

3.1. Features: MFCCs

We extract the features with the Matlab toolbox Voicebox [4]. The reasoning behind the specific parametrisation can be found in our previous work [5]. We suggest to use 23 MFCCs (without 0th MFCC) extracted by applying a 20 ms observation window without any overlap. 18 MFCC deltas (including the 0th MFCC delta), and 20 MFCC double deltas (including the 0th 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.

3.2. Binaural Feature Extraction

Simply averaging both channels into a single monaural signal could reduce the SNR of important cues that are only well captured in one of the two channels. Therefore, we suggest to extract MFCCs from left and right channels, and concatenate the resulting features into a single feature matrix with twice the length in time: Our feature extraction scheme with 20 ms hop size yields 500 observations for the given 10 s audio recordings, and by concatenating the feature vectors from both channels, we end up with 1000 observations. We do not stop here, but further increase the size of our feature matrix by concatenating features from augmented audio recordings.

3.3. Data Augmentation

Since the audio recordings are rather short, which is problematic in general and a particular problem for i-vector modelling approaches [2], we suggest to augment the data for training and testing. Considering various sound sources like engines, voices, bird songs, etc., it is clear that most probably the limited amount of training data will not cover the whole range of variation in terms of 'pitch'. Therefore, we propose to pitch shift the audio recordings up and down by 25, 50, and 100 cents, respectively. Even though a higher degree of alteration might make sense for some audio scenes, we do not apply it in order to avoid the introduction of artifacts due to the pitch shifting algorithm.

As a consequence, for every audio recording 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) instead of just the provided 10 s audio recordings.

4. THE I-VECTOR CLASSIFIER

Our starting point is the i-Vector modelling scheme already introduced for the DCASE-2016 challenge [1]. In principle, we use the same i-vector extraction pipeline, and the same post-processing of the results. 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 length 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)* [6] 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. THE DEEP CONVOLUTIONAL NEURAL NETWORK

Our basic network architecture is depicted in Table 1. The structure is very similar to the one used in our last year's submission. In particular, the feature learning part of our model follows a *VGG style network* [7] and the classification part of the network is designed as a global average pooling layer [8] over 15 feature maps (one for each class) followed by a *softmax* activation. As an activation function within the network we use Exponential Linear Units (ELUs) [9]. The main conceptual difference to last year is that we changed from a sliding window averaging approach for predicting the class labels to a model that predicts on the whole audio sequence.

Table 1: Model Specifications. BN: Batch Normalization, ELU: Exponential Linear Unit, CCE: Categorical Cross Entropy. For training a constant batch size of 50 samples is used.

Input $1 \times 500 \times 137$
5×5 Conv(pad-2, stride-2)-32-BN-ELU
3×3 Conv(pad-1, stride-1)-32-BN-ELU
2×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-64-BN-ELU
3×3 Conv(pad-1, stride-1)-64-BN-ELU
2×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-128-BN-ELU
3×3 Conv(pad-1, stride-1)-128-BN-ELU
3×3 Conv(pad-1, stride-1)-128-BN-ELU
3×3 Conv(pad-1, stride-1)-128-BN-ELU
2×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-0, stride-1)-256-BN-ELU
Drop-Out(0.5)
1×1 Conv(pad-0, stride-1)-256-BN-ELU
Drop-Out(0.5)
1×1 Conv(pad-0, stride-1)-15-BN-ELU
Global-Average-Pooling
15-way Soft-Max

We follow two training protocols, which differ mainly in the way we present the input data to the network. For the first, *CNN*$_{full}$, we resample the audio to 22050 Hz, and compute a Short Time Fourier Transform (STFT) using 2048-sample windows at a frame rate of 50 fps. From this, we compute the magnitude spectrogram and apply a logarithmic filterbank with 24 bands per octave on the frequency band between 20 Hz and 8kHz. This yields 500-frame spectrograms with 137 frequency bins per data point.

For the second, *CNN*$_{excerpt}$, we show the network only randomly selected 2-second excerpts of the full spectrograms during training, while presenting the whole spectrogram during testing. Further, we add spectrograms computed using half and double the window size as channels to the input, resulting in a $3 \times 200 \times 206$ dimensional input space ($3 \times 500 \times 206$ when testing). We also apply a different dropout technique in this configuration. Instead of regular dropout, we drop out complete feature maps, however with lower probabilities (0.1 instead of 0.3, and 0.3 instead of 0.5 respectively). While the validation performances achieved by both training protocols are comparable, they seem to complement each other well in the classifier fusion. This indicates that they focus on different aspects of the input.

We run each experiment (train run on one fold) multiple times (> 20), and only keep the top five models with respect to validation set accuracy. When changing parts of our system we then consider the mean and standard deviation over these runs to get a more reliable estimate of the impact of our modifications.

We use the ADAM update rule [10] with an initial learning rate of 0.002 and a mini-batch size of 50 samples. If accuracy has not improved during the last 20 epochs, we halve the learn rate and continue training from the best parameter configuration (in terms of validation accuracy) found so far. This refinement procedure is repeated up to 15 times.

We observed that the individual top-5 models (which show similar overall accuracy) differ when looking at the individual class performances. To take this into account, we average the predicted class probabilities over the top-5 models for each fold to get one estimate per fold.

6. SCORE CALIBRATION AND LATE FUSION

As outlined above, we train three i-vector-based models and five CNN-based models (in each training protocol) for each fold. This results in 13 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 6 shows an overview of our approach.

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[2], and compute the fused prediction $y = \sigma(XW + b)$, where σ is the softmax function, and X are the class probabilities given by each classifier. As shown in Fig. 6, we use this approach for submissions IVEC$_{\text{calib}}$, CNN$_{\text{calib}}$, and All$_{\text{calib}}$; for All$_{\text{avg}}$, we use plain averaging instead, to see whether training the calibration and fusion on the validation set leads to over-fitting.

7. RESULTS

7.1. Submissions

We provide 4 different submissions based on the methods described in the previous sections:

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

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

7.2. Performance on the validation set

In Table 2, all accuracies on ASC are provided. We show the performance of the different methods on the four validation folds as well as the average accuracy over all folds.

Additionally, in Table 3, the class-wise accuracies of the different methods are provided. The baseline method provided with the dataset is given as *Base*.

When looking at the individual models, we see that CNNs outperform the baseline as well as the i-vector system. This is contrary to last year's submission, which suggests that CNNs tend to work better on shorter utterances (10 seconds compared to 30 seconds). Our overall best accuracy of 91.29% is achieved when fusing all models using calibration (model All$_{\text{calib}}$). I-vectors show a similar cross-validation performance as in DCASE-2016, but data augmentation as described in Section 3 was required to achieve this. In contrast, CNNs seem to work significantly better than last year without any data augmentation. As a final observation we highlight the performance gain when fusing both CNNs and i-vectors into one system. This is surprising as CNNs outperform i-vectors on almost all of the classes (except for city center and park).

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

Table 2: Audio scene classification accuracy on the provided DCASE-2017 validation set with provided cross-validation splits.

(%)	fold1	fold2	fold3	fold4	avg
IVEC$_{\text{calib}}$	84.27	85.17	80.73	87.61	84.45
All$_{\text{avg}}$	86.58	87.98	88.58	87.68	87.70
CNN$_{\text{calib}}$	88.72	90.03	89.09	88.29	89.03
All$_{\text{calib}}$	91.62	92.24	89.68	91.62	91.29

Table 3: Class-wise accuracies of different methods on the provided development data. The results are averaged over all four folds.

(%)	Base	IVEC$_{\text{calib}}$	All$_{\text{avg}}$	CNN$_{\text{calib}}$	All$_{\text{calib}}$
Beach	75.3	85.9	88.8	90.4	91.0
Bus	71.8	94.6	98.1	98.1	98.4
Cafe/Rest.	57.7	79.5	79.5	87.2	89.7
Car	97.1	96.2	97.4	97.1	97.4
City center	90.7	87.8	86.5	85.3	88.1
Forest path	79.5	93.6	95.5	95.2	96.5
Grocery store	58.7	91.0	96.5	94.6	97.4
Home	68.6	77.7	82.4	85.5	86.8
Library	57.1	75.0	88.8	90.1	92.3
Metro station	91.7	76.3	89.1	86.5	91.0
Office	99.7	92.0	95.5	95.5	96.5
Park	70.2	84.6	65.1	70.2	80.5
Resident. area	64.1	65.7	74.4	73.1	74.4
Train	58.0	79.8	87.5	93.9	93.9
Tram	81.7	87.2	91.4	93.0	95.5
Overall	74.8	84.5	87.7	89.0	91.3

7.3. Performance on the evaluation set

In Table 4, the results of our methods on the unseen evaluation set are listed, along with the baseline provided by the organisers. All methods provide higher accuracies than the baseline, and the method where we fuse our two rather different approaches (All$_{\text{calib}}$) performs best, yielding 73.8% accuracy and 5^{th} place in the team ranking. Contrary to the results on the validation set, our i-vectors perform better than our CNN by almost 4 ppt. However, the linear logistic regression-based fusion of both methods results in a synergy. Simply averaging leads to a worse performance compared to the i-vectors alone.

Considering the results of our CNNs, it becomes clear that the greatest strength of deep learning – the ability to model extremely complex relationships between input and target – holds an omnipresent risk of non-obvious overfitting: it is hard to tell where generalisation ends and overfitting begins when it is unclear in what aspects the validation and evaluation data sets will differ.

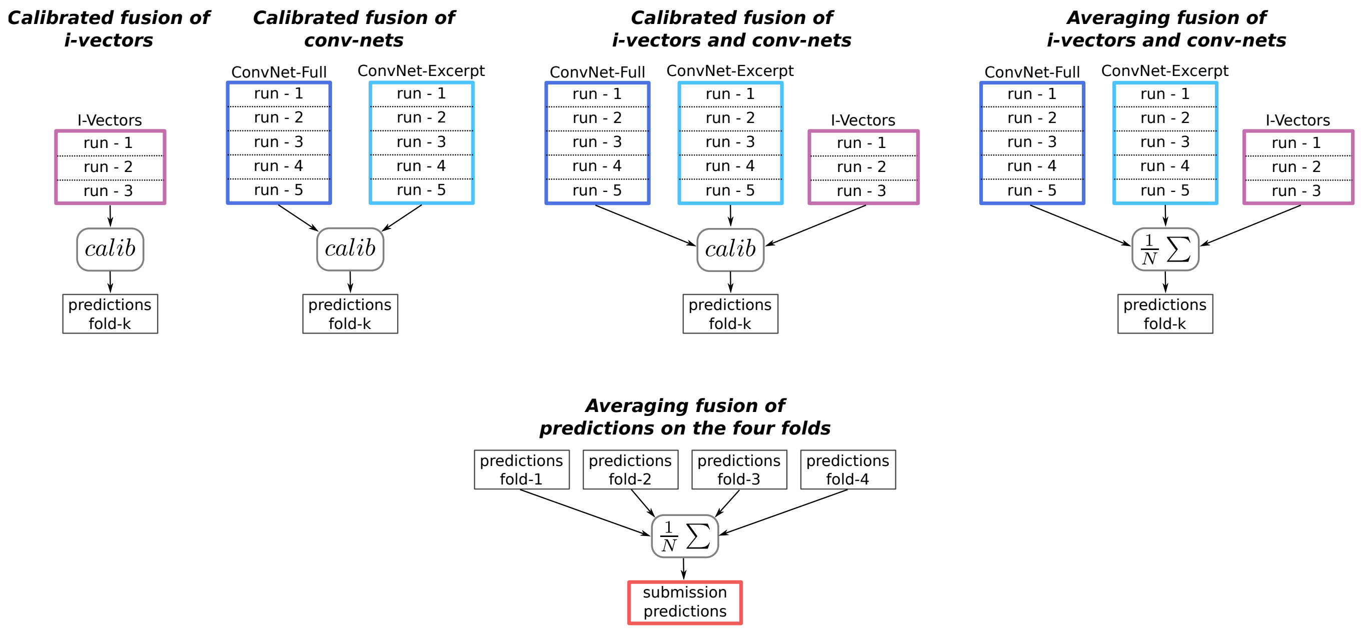

Figure 1: Overview of fusion schemes used for our submissions.

Table 4: Accuracies of the different methods on the evaluation data set, compared to the provided baseline.

(%)	Base	IVEC$_{\text{calib}}$	All$_{\text{avg}}$	CNN$_{\text{calib}}$	All$_{\text{calib}}$
Evaluation	61.0	68.7	66.8	64.8	73.8

8. CONCLUSION

This short report has described how we extended and optimized our DCASE-2016 ASC approach in order to address the challenges of the 2017 DCASE ASC task – in particular, the short duration of provided audio snippets. The results on the provided DCASE-2017 data again show that the fusion approach seems superior to simpler classification models.

Generally, classification accuracies around 90% in a hard 15-class discrimination task such as this, where humans, under similar conditions, achieve recognition rates around 50% (at least according to a test we ran with students at our university, using the DCASE-2016 listening test infrastructure[3]) seem surprisingly high. The lower results on the new evaluation data confirm our initial concerns regarding overfitting.

9. 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. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of a Titan X GPU used for this research.

[3]`http://www.cs.tut.fi/sgn/arg/dcase2016/task-acoustic-scene-classification`. Thanks to the DCASE organizers for their help!

10. REFERENCES

[1] 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.

[2] 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.

[3] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017.

[4] M. Brookes, "Voicebox: Speech Processing Toolbox for Matlab," Website, 1999, available online at http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html; visited on November 1st, 2013.

[5] 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.

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

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

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

[9] D. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” *International Conference on Learning Representations (ICLR) (arXiv:1511.07289)*, 2015.

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