CP-JKU SUBMISSIONS FOR DCASE-2016: A HYBRID APPROACH USING BINAURAL I-VECTORS AND DEEP CONVOLUTIONAL NEURAL NETWORKS

Hamid Eghbal-Zadeh *Bernhard Lehner* *Matthias Dorfer* *Gerhard Widmer*

Department of Computational Perception, Johannes Kepler University of Linz, Austria
hamid.eghbal-zadeh@jku.at

ABSTRACT

This report describes the 4 submissions for Task 1 (Audio scene classification) of the DCASE-2016 challenge of the CP-JKU team. We propose 4 different approaches for Audio Scene Classification (ASC). First, we propose a novel i-vector extraction scheme for ASC using both left and right audio channels. Second, we propose a Deep Convolutional Neural Network (DCNN) architecture trained on spectrograms of audio excerpts in end-to-end fashion. Third, we use a calibration transformation to improve the performance of our binaural i-vector system. Finally, we propose a late-fusion of our binaural i-vector and the DCNN. We report the performance of our proposed methods on the provided cross-validation setup for the DCASE-2016 challenge. Using the late-fusion approach, we improve the performance of the baseline by 17 percentage point in accuracy.

Our submissions achieved ranks **first** and **second** among 49 submissions in the audio scene classification task of DCASE-2016 challenge.

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

1. INTRODUCTION

In this report, we describe four methods we propose for Task 1 (ASC) in the DCASE-2016 challenge [1]. We provide the performances of our methods on the openly accessible DCASE-2016 dataset. In our challenge submissions, we follow 4 different approaches for audio scene classification. First, we propose a binaural i-vector features extraction scheme using tuned MFCC features for ASD. Second, we examine a score calibration technique with our binaural i-vector system. Third, we use a Deep Convolutional Neural Network (DCNN) trained on spectrograms of audio excerpts in an end-to-end fashion. Finally, we propose a hybrid system which benefits from a late-fusion of the binaural i-vector and the DCNN systems.

The reminder of this report is organized as follows. In the Section 2, the i-vector representation is described. Our novel binaural i-vector extraction scheme is explained in Section 3. The DCNN approach is detailed in Section 4. The late-fusion of binaural i-vectors and DCNN is described in Section 5. In Section 6, the results of ASC on the provided dataset and cross-validation splits are presented. Finally, Section 7 concludes this report.

This work was supported by the Austrian Science Fund (FWF) under grant no. Z159 (Wittgenstein Award) and 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.

[1] `www.cs.tut.fi/sgn/arg/dcase2016/`

2. I-VECTOR FEATURES

2.1. Theory

I-vector [1] representation have been introduced in the field of speaker verification. After its revolutionary success in the field, it was further used with great promise in other areas of audio processing such as language recognition [2], music artist and genre classification [3] and audio scene classification [4].

I-vector features are the product of a Factor Analysis (FA) procedure applied on the statistical representation of an audio excerpt. They provide a fixed-length information-rich low-dimensional representation for short audio segments. To prepare a statistical representation of an audio segment, first a Universal Background Model (UBM) is trained on the acoustic features of sufficient amount of audio files to capture similarities in the acoustic feature space. Further, this UBM is adapted to the acoustic features of each audio segment and the parameters of the adapted model are used as the statistical representation of the audio segment. Finally, the FA procedure is applied on the statistical representation of each audio segment and the factors that have the least changes from one audio segment to the another are estimated. These estimated factors, known as i-vectors are then used instead of the audio excerpts for classification purposes.

The UBM is usually a Gaussian Mixture Model (GMM) trained on frame-level features such as Mel-Frequency Cepstral Coefficients (MFCCs). The statistical representation of an audio segment is then the mean vector of this GMM, which is adapted to the MFCC features of the audio segment.

To apply the FA, the Gaussian mixture model (GMM) mean supervector $\mathbf{M}$ adapted to an audio from audio scene α can be decomposed as follows:

$$\mathbf{M} = m + \mathbf{T}.y \qquad (1)$$

where m is the GMM mean supervector and $\mathbf{T}.y$ is an offset. The low-dimensional subspace vector y is a latent variable with the normal prior and the i-vector w is a MAP estimate of y. The matrix $\mathbf{T}$ is learned by using statistical representations of audio excerpts in the development set via an EM algorithm. More information about the training procedure of $\mathbf{T}$ and i-vector extraction can be found in [1, 5].

2.2. Our I-vector pipeline

The block-diagram of our i-vector pipeline is shown in Figure 1. As can be seen, the i-vector pipeline has 3 steps: 1) development, 2)

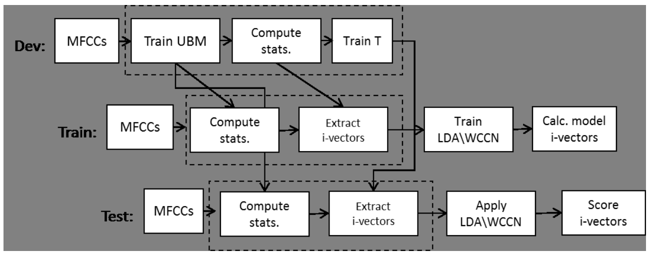

Figure 1: Block-diagram of our i-vector pipeline.

training and 3) testing.

During the *development step*, the UBM is trained on the development set and the statistic representation of the development audio segments are calculated. Then using these statistics, the $\mathbf{T}$ matrix is learned on the statistics of development set's audio segments.

In the *training step*, using the i-vector models UBM and $\mathbf{T}$, i-vectors of the training set are extracted. Using the training set i-vectors, a Linear Discriminant Analysis (LDA) [6] and a Within-Class Covariance Normalization (WCCN) [7] model are trained. LDA and WCCN projections are used for i-vector post-processing. They improve the i-vector representation and reduce the within-class variability in i-vector space. Further, LDA-WCCN projected i-vectors are used to calculate *model i-vectors*. For each class, the average of the projected i-vectors in training set is stored as a *model i-vector*.

In the *testing step*, using the models from the previous step and the MFCCs of the test set we extract test i-vectors. Then, these i-vectors are projected by LDA and WCCN models trained in the training step. Finally, each projected test i-vector is scored against all model i-vectors. The class of the model i-vector with the highest score is chosen as the predicted class.

2.3. Our I-vector setup

We trained our UBMs with 256 Gaussian components on MFCC features extracted from audio excerpts. The UBM, $\mathbf{T}$ matrix, LDA and WCCN projections are trained on the training portion of each cross-validation split. The details of our MFCC features are explained in the following section. We set the dimensionality of the i-vectors to 400. To score the projected i-vectors, we used a cosine scoring as explained in [8].

3. TUNING THE I-VECTOR FEATURES

3.1. MFCCs

In [9] it was shown that it is useful to find a good parametrisation of MFCCs for a given task. Therefore, the first step is to improve the performance of MFCCs which we extract with the Matlab toolbox Voicebox [10].

In order to include all the components that are involved, we do this after the complete i-vector pipeline is implemented. The following results are always averaged from a four-fold CV, unless explicitly mentioned otherwise.

3.1.1. Windowing Scheme

In the first experiment we want to evaluate different observation window lengths. For this, we place the different observation windows symmetrically around the frame that was always fixed on 20 ms. Thus, independent of the actual observation window, we always end up with exactly the same amount of observations. In Table 1 we provide some results of different windowing schemes for MFCCs and their deltas and double deltas. As can be seen, the impact of using different overlaps is quite severe on the results of the MFCCs. It turns out that a 20 ms window without overlap gives best accuracy for MFCCs.

	win=20 ms		win=60 ms		win=100 ms	
	acc	f1	acc	f1	acc	f1
$MFCC$	68.95	61.73	61.84	57.05	60.61	56.53
Δ	61.62	62.53	64.02	65.53	60.68	62.11
$\Delta\Delta$	61.54	57.83	62.05	60.17	59.49	62.59

Table 1: Results of MFCC observation window tuning. Row $MFCC$: 20 MFCCs with 0^{th} coefficient; Δ: 20 MFCC deltas; $\Delta\Delta$: 20 MFCC double deltas. Gray cells indicate the configurations that were combined for further experiments.

$MFCC$		Δ		$\Delta\Delta$	
w/ 0^{th}	w/o 0^{th}	w/ 0^{th}	w/o 0^{th}	w/ 0^{th}	w/o 0^{th}
68.95	71.43	61.62	56.34	61.54	50.77

Table 2: The impact of the 0^{th} MFCC. It can be seen that it is important to include just the 0^{th} MFCC deltas and double deltas, but not the 0^{th} MFCC itself.

The effect is much smaller on the results of deltas and double deltas. Nevertheless, we consider it useful to extract deltas and double deltas separately with a 60 ms observation window, and combine them with the 20 ms MFCCs into one single feature vector.

3.1.2. Number of Coefficients

After fixing observation window lengths for MFCCs and deltas and double deltas, we evaluate the amount of coefficients that is actually useful in our specific setting. Often, the 0^{th} coefficient is ignored in order to achieve loudness invariance, which also makes sense for this task. The results in Table 2 support this intuition, where we can see that including the 0^{th} coefficient leads to reduced accuracy for MFCCs.

Nevertheless, it turned out to be quite useful if the delta and double delta of the 0^{th} MFCC is included in the feature vectors. In Table 2 it can be seen that the performance of the MFCC deltas drops from 61.6% to 56.3% accuracy without the 0^{th} MFCC deltas. The performance of the MFCC double deltas drops from 61.5% to 50.8% without the 0^{th} MFCC double deltas.

In a series of further experiments conducted, the amount of coefficients that turned out to be useful was determined, separately for MFCCs, deltas and double deltas.

3.1.3. Final MFCC configuration

According to the results of the previously conducted experiments, 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

	fold 1 acc	fold 2 acc	fold 3 acc	fold 4 acc	avg acc
BASE	79.7	64.3	77.1	75.3	74.1
Monaural MFCC	85.52	65.86	79.19	77.05	76.91
Binaural MFCC	85.86	76.55	77.52	83.22	80.79

Table 3: Comparing the performance of our tuned MFCCs with the provided MFCCs in conjunction with the i-vector procedure. Row BASE: original MFCC provided by the DCASE organisers; Monaural MFCC: tuned MFCCs on averaged single-channel; Binaural MFCC: multi-channel tuned MFCCs.

frame. Regardless of the observation window length, we use 30 triangle shaped mel-scaled filters in the range [0-11 kHz].

3.2. Binaural Feature Extraction

Most often, the binaural audio material is down-mixed into a single monaural representation by simply averaging both channels. This could be problematic in cases where an important cue is only captured well in one of the channels, since averaging would then lower the SNR, and increase the chance that it gets missed by the system. The analysis of both channels separately would alleviate this problem.

Not only do we extract MFCCs from both channels separately, but also from the averaged monaural representation as well as from the difference of both channels. All in all, we extract MFCCs from four different audio sources, resulting in four different feature space representations per audio file. An experiment where we concatenated the MFCCs into a single feature vector did not lead to improved i-vector representations, therefore we opt for a late fusion approach.

3.3. Late Fusion

The aforementioned separately extracted MFCCs yield four different i-vectors which in turn result in four different scores per audio file. In order to fuse those scores, we suggest to compute the mean of them. Additionally, we utilise some sort of a bagging approach for the unseen Test set, where we combine the output of the models trained on the four CV folds. All in all, we combine 16 scores in order to yield the classification result of one audio file.

4. DEEP 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. The specific network architecture used is depicted in Table 4. The feature learning part of our model follows the *VGG style networks* for object recognition and the classification part of the network is designed as a global average pooling layer as in the *Network in Network* architecture. The input size of our network is a one channel spectrogram excerpt with size 149×149. This means we train the model not on whole sequences but only on small ”sliding” windows. The spectrograms for this approach are computed as follows: The audio is sampled at a rate of 22050 samples per second. We compute the Short Time Fourier Transform (STFT) on 2048 sample windows at a frame rate of 31.25 FPS. Finally we post-process the STFT with a logarithmic filterbank with 24 bands, logarithmic magnitudes and an allowed passband of 20Hz

Table 4: Model Specifications. BN: Batch Normalization, ReLu: Rectified Linear Activation Function, CCE: Categorical Cross Entropy. For training a constant batch size of 100 samples is used.

Input $1 \times 149 \times 149$
5×5 Conv(pad-2, stride-2)-32-BN-ReLu
3×3 Conv(pad-1, stride-1)-32-BN-ReLu
2×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-64-BN-ReLu
3×3 Conv(pad-1, stride-1)-64-BN-ReLu
2×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-128-BN-ReLu
3×3 Conv(pad-1, stride-1)-128-BN-ReLu
3×3 Conv(pad-1, stride-1)-128-BN-ReLu
3×3 Conv(pad-1, stride-1)-128-BN-ReLu
2×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-0, stride-1)-512-BN-ReLu
Drop-Out(0.5)
1×1 Conv(pad-0, stride-1)-512-BN-ReLu
Drop-Out(0.5)
1×1 Conv(pad-0, stride-1)-15-BN-ReLu
Global-Average-Pooling
15-way Soft-Max

to 16kHz. The parameters of our models are optimized with mini-batch stochastic gradient decent and momentum. The mini-batch size is set to 100 samples. We start training with an initial learning rate of 0.02 and half it every 5 epochs. The momentum is fixed at 0.9 throughout the entire training. In addition we apply an $L2$-weight decay penalty of 0.0001 on all trainable parameters of our model.

For classification of unseen samples at test time we proceed as follows. First we run a sliding window over the entire test sequences and collect the individual class probabilities for each of the window. In a second step we average the probabilities of all contributions and assign the class with maximum average probability.

5. SCORE CALIBRATION AND LATE FUSION

5.1. Score Calibration

To calibrate the binaural i-vector cosine scores, a calibration transformation is used. We use *linear logistic regression* to train our transformation models using the scores of the validation set and its labels. To calibrate the test set scores, we apply the models learned via validation set scores to transform the test set scores. The transformed scores are used for the final prediction. More information about our score calibration technique can be found in [11].

5.2. Late fusion

Figure 2 shows a block-diagram of the proposed late-fusion method. After extracting binaural i-vectors, their final score matrix on the test set is calculated. In addition, the DCNN is trained and the soft-max activations on the test set are calculated. Using a linear logistic regression score calibration similar to what we described in the previous section, the scores of binaural i-vectors and soft-max activations of DCNN are combined into a single score matrix. The projection models are learned using the binaural i-vector scores

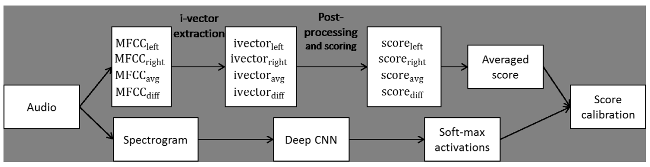

Figure 2: Block-diagram of the late-fusion between binaural i-vectors and DCNN.

Table 5: Audio scene classification accuracy on the provided DCASE-2016 test set with provided cross-validation splits. Methods marked with an asterisk ($*$) used the score calibration projection which was trained on the same set as the test set because of the lack of validation set in the provided cross-validation splits.

(%)	fold1	fold2	fold3	fold4	avg
DCNN	80.69	75.52	77.85	83.90	79.49
BMBI	85.86	76.55	77.52	83.22	80.79
$*$ CBMBI	87.93	80.34	81.54	85.62	83.86
$*$ LFCBI	94.48	85.17	87.25	92.81	89.93

and soft-max activations of the validation set. The binaural i-vector scores and soft-max activations of the test set are then fused together using the models learned on the validation set. This fused score is used for the final prediction.

6. RESULTS

6.1. Submissions

We provided 4 different submissions based on the methods described in the previous sections for the DCASE-2016 challenge. Our submissions are:

1. DCNN: Deep Convolutional Neural Network (explained in Section 4)
2. BMBI: Binaural MFCC Boosted I-vectors (explained in Section 3)
3. CBMBI: Calibrated Binaural MFCC Boosted I-vectors (explained in Section 3)
4. LFCBI: Late Fusion of CNN and Binaural I-vectors (explained in Section 5)

6.2. Performance on the validation set

In Table 5, 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 6, the class-wise accuracy of different methods are provided. The GMM-MFCC baseline method provided with the dataset, can be found as *Base*. Since the available dataset for DCASE-2016 challenge provides cross-validation splits with only training and test portions, we use the test set also for the calibration step (e.g. for computing the calibration projection in the i-vector pipeline). Similarly, we use the validation sets of each fold for model selection in the DCNN approach.

For the final submission of the DCASE-2016 challenge, our models are tested on an unseen test set. On this unseen test set, our DCNN, BMBI, CBMBI and LFCBI submissions achieved 83.3%, 86.4%, 88.7% and 89.7% accuracy, achieving 14th, 5th, 2nd and 1st place in the challenge among 49 submissions, respectively.

Table 6: The class-wise accuracy comparing the performance of different methods for classification of different audio scenes on DCASE-2016 provided test dataset. The results are averaged for all the folds. Methods marked with an asterisk ($*$) used the score calibration projection which was trained on the same set as the test set because of the lack of validation set in the provided cross-validation splits.

(%)	Base.	DCNN	BMBI	*CBMBI	*LFCBI
Beach	69.3	92.11	78.95	86.84	92.11
Bus	79.6	77.37	79.47	87.11	95.00
Cafe/Rest.	83.2	80.27	62.87	78.72	93.92
Car	87.2	84.61	96.18	96.18	96.18
City center	85.5	83.79	90.19	90.01	88.52
Forest path	81.0	94.05	94.84	96.03	98.81
Grocery store	65.0	93.80	94.86	89.72	95.11
Home	82.1	72.29	59.15	71.01	89.17
Library	50.4	75.14	75.56	78.13	85.93
Metro station	94.7	88.52	83.92	84.10	91.89
Office	98.6	73.18	97.22	90.50	97.22
Park	13.9	58.61	78.33	81.81	86.94
Resident. area	77.7	67.54	63.60	72.06	76.00
Train	33.6	63.45	73.18	72.95	76.74
Tram	85.4	90.66	86.99	84.47	87.88

Results shows that our DCNN and BMBI methods perform similarly on average. However, a closer look at Table 6 reveals that the two methods have different strengths and weaknesses. Looking at the CBMBI results, we observe that the calibration improves the performance of BMBI. Since we observed that DCNN and BMBI methods do not behave similarly on different classes, we expect that a combination of the two improves the fused system's performance. As expected, the LFCBI method outperforms DCNN, BMBI and CBMBI systems as shown in our results. It suggests that binaural i-vector features and the representation learned by DCCN from spectrograms contain complementary information about the audio scenes. As a result, by combining the two we achieve an improvement for the LFCBI system.

7. CONCLUSION

In this report, we proposed 4 different methods for ASC. We used a deep CNN which uses spectrograms of audio excerpts and is trained in end-to-end fashion. We further proposed a novel binaural i-vector extraction scheme using different channels of the audio. Finally, we proposed a late-fusion of the two methods which improved the overall and class-wise performance of ASC.

8. REFERENCES

[1] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *Audio, Speech, and Language Processing, IEEE Transactions on*, 2011.

[2] N. Dehak, P. A. Torres-Carrasquillo, D. A. Reynolds, and R. Dehak, "Language recognition via i-vectors and dimensionality reduction." in *INTERSPEECH*. Citeseer, 2011.

[3] H. Eghbal-zadeh, B. Lehner, M. Schedl, and G. Widmer, "I-vectors for timbre-based music similarity and music artist classification," in *ISMIR*, 2015.

[4] B. Elizalde, H. Lei, G. Friedland, and N. Peters, "An i-vector based approach for audio scene detection," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.

[5] P. Kenny, "Joint factor analysis of speaker and session variability: Theory and algorithms," *CRIM, Montreal,(Report) CRIM-06/08-13*, 2005.

[6] B. Scholkopft and K.-R. Mullert, "Fisher discriminant analysis with kernels," *Neural networks for signal processing IX*, 1999.

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

[8] N. Dehak, R. Dehak, J. R. Glass, D. A. Reynolds, and P. Kenny, "Cosine similarity scoring without score normalization techniques." in *Odyssey*, 2010, p. 15.

[9] B. Lehner, R. Sonnleitner, and G. Widmer, "Towards Lightweight, Real-time-capable Singing Voice Detection," in *Proceedings of the 14th International Conference on Music Information Retrieval (ISMIR 2013)*, 2013.

[10] 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.

[11] N. Brümmer, "Focal multi-class: Toolkit for evaluation, fusion and calibration of multi-class recognition scorestutorial and user manual," *Software available at http://sites. google. com/site/nikobrummer/focalmulticlass*, 2007.