Deep Within-Class Covariance Analysis for Robust Audio Representation Learning

Hamid Eghbal-zadeh[1,2] **Matthias Dorfer**[2] **Gerhard Widmer**[1,2]

[1] LIT AI Lab & [2] Institute of Computational Perception
Johannes Kepler University of Linz, Austria
`{hamid.eghbal-zadeh, matthias.dorfer, gerhard.widmer}@jku.at`

Abstract

Deep Neural Networks (DNNs) are known for excellent performance in supervised tasks such as classification. Convolutional Neural Networks (CNNs), in particular, can learn effective features and build high-level representations that can be used for classification, but also for querying and nearest neighbor search. However, CNNs have also been shown to suffer from a performance drop when the distribution of the data changes from training to test data. In this paper we analyze the internal representations of CNNs and observe that the representations of unseen data in each class, spread more (with higher variance) in the embedding space of the CNN compared to representations of the training data. More importantly, this difference is more extreme if the unseen data comes from a shifted distribution. Based on this observation, we objectively evaluate the degree of representation's variance in each class by applying eigenvalue decomposition on the within-class covariance of the internal representations of CNNs and observe the same behaviour. This can be problematic as larger variances might lead to mis-classification if the sample crosses the decision boundary of its class. We apply nearest neighbor classification on the representations and empirically show that the embeddings with the high variance actually have significantly worse KNN classification performances, although this could not be foreseen from their end-to-end classification results. To tackle this problem, we propose *Deep Within-Class Covariance Analysis (DWCCA)*, a deep neural network layer that significantly reduces the within-class covariance of a DNN's representation, improving performance on unseen test data from a shifted distribution. We empirically evaluate DWCCA on two datasets for Acoustic Scene Classification (DCASE2016 and DCASE2017). We demonstrate that not only does DWCCA significantly improve the network's internal representation, it also increases the end-to-end classification accuracy, especially when the test set exhibits a slight distribution shift. By adding DWCCA to a VGG neural network, we achieve around 6 percentage points improvement in the case of a distribution mismatch.

1 Introduction

Convolutional Neural Networks (CNNs) are the state of the art in many supervised learning tasks such as classification, and using the power of convolutional layers, CNNs can learn useful features that are often superior to engineered features, and build internal representations that can achieve high classification performance.

It has been shown that CNNs have a surprising ability to fit data, so much so that they can even perfectly learn from data with random labels [33]. But of course, memorising the training data is

32nd Conference on Neural Information Processing Systems (NeurIPS2018), IRASL workshop, Montréal, Canada.

not sufficient: a model is expected to generalize to unseen data points. Additionally, a robust model has to be able to not only deal with unseen data points that are similar to the training set, but also cope with unseen data points that may come from a slightly different distribution than the training data (*distribution mismatch*). When there is a distribution shift between the training and test sets, robustness of the model's representation becomes more important as it has to classify or embed data points that are quite different from the ones it has observed in the training set.

In this paper, we investigate this by using a well-known DNN architecture (VGG [29]) that is adapted for audio classification [10] and is widely used among researchers. We evaluate VGG on data with as well as without distribution mismatch and observe that while VGG exhibits a reasonable performance on the data without distribution mismatch, its performance significantly drops when tested on data from a shifted distribution.

We start by analyzing the internal representations of the network by using visualisations. As will be seen in the first (a-c) and the 3rd rows (g-i) of Figure 2, the network's internal representations in each class spread more in the embedding space for the unseen data (validation or test) compared to the training data. This is even more extreme when the unseen data comes from a shifted distribution (i).

For an objective evaluation of the amount of the representation's variance in each class, we compute the within-class covariance of the representations of the network for each class, and we apply eigenvalue decomposition to compute the eigenvalues of each class's covariance matrix. We then report the sorted eigenvalues of the within-class covariance of the representations in Figure 3. As the blue curves show, the eigenvalues in unseen data of validation (b and e) and test (c and d) have considerably higher ranges than train data (a and d) for all the datasets we used.

To better understand the effect of such high variance in the quality of generalisation in the representations of our network, we carry out K-nearest neighbor (KNN) experiments on the dataset without, and the dataset with distribution shift. As the results in Figure 4 show, the performance degredation from validation (c) compared to test (d) in case of distribution mismatch is significantly higher compared to the performance drop from validation (a) to test (b) when the test data comes from a similar distribution. This observation is also aligned with what we observed in the visualisations from Figure 2 that showed the data is more spread than validation data, when coming from a shifted distribution.

To tackle this problem, we propose *Deep Within-Class Covariance Analysis (DWCCA)*, a deep neural network layer that reformulates the conventional Within-Class Covariance Normalization (WCCN) [13] as a DNN-compatible version. DWCCA is trained end-to-end using back-propagation, can be placed in any arbitrary position in a DNN, and is capable of significantly reducing the within-class covariance of the internal representation in a DNN.

We empirically show that DWCCA significantly reduces the within-class covariance of the DNN's representations, in both cases. Further, we evaluate the generalization quality of the DNN's representations after applying DWCCA by performing nearest neighbor classification on its representations. Our results show that DWCCA significantly improves the nearest neighbor classification results in both cases, hence improving the generalization quality of the representations. And finally we report the end-to-end classification results of the trained models on an acoustic scene classification task, using data from the annual IEEE Challenges on Detection and Classification of Acoustic Scenes and Events (DCASE). It turns out that the classification results for the dataset with distribution shift are significantly improved by integrating the DWCCA layer, while the performance on the dataset without distribution mismatch stayed the same.

2 Related Work

The characteristics of the representations learned in CNNs can be influenced by many factors such as network architecture and the training data. The authors in [20] investigated how architecture topologies affect the robustness and generalization of a representation and showed which representations from different architectures better transfer to other datasets and tasks.

While the topology of the architecture influences the generality of its representations, several authors proposed methods that can improve the internal representation of a DNN. [9] proposed a loss which learns linearly separable latent representations on top of a DNN, by maximising their smallest Linear

Discriminant Analysis (LDA) [11] eigenvalues. And in [1], the authors propose creating a maximally correlated representation from different modalities.

It has been shown that stand-alone CNNs are not very successful at generalizing to unseen data points from shifted distributions in tasks such as Acoustic Scene Classification (ASC), where such distributional shifts are common. ASC is defined as classifying environments from the sounds they produce [2], and often these environments (e.g, home, park, city center) may sound very different in different cities or during different times of the year. Although in [24, 31, 10] CNNs have shown promising results in ASC when the unseen test data has a similar distribution to the training data, in [22, 34] similar CNNs that previously performed successfully in a matched distribution case, suffered significantly from the lack of generalization to a shifted distribution in the test set.

To cope with this drawback of CNNs, [28] investigated CNN manifold data augmentation using Generative Adversarial Networks [12], while in [16, 32, 28] the authors used an ensemble of CNNs as feature extractors and processed the deep features via Support Vector Machines (SVMs), followed by a late fusion of various models. They showed that although CNNs do not generalize well in the distribution shift case, they can learn useful features that can be incorporated for building new classifiers with better generalization properties.

In this paper, we try to better understand these problems. We focus our efforts on investigating the reasons for the performance drop in CNNs when the data distribution is shifted. To this end, we analyze the internal representations in CNNs and propose DWCCA to tackle this issue.

3 Deep Within-Class Covariance Analysis

We start by introducing a common notation which will be used throughout the paper. We then first describe Within-Class Covariance Normalization (WCCN) –a classic machine learning method–, and then show how to cast it into a deep learning compatible version.

3.1 Conventional Within-Class Covariance Normalization

Let $\mathbf{W}$ denote a set of N d-dimensional observations (feature vectors) belonging to C different classes $c \in \{1, ..., C\}$. The observations are in the present case either hand crafted features (e.g. i-vectors [7]) or any intermediate hidden representation of a deep neural network.

WCCN is a linear projection that provides an effective compensation for the within-class variability and has proven to be effectively used in combination with different scoring functions [6, 23]. The WCCN projection scales the feature space in the opposite direction of its within-class covariance matrix, which has the advantage that finding decision boundaries on the WCCN-projected data becomes easier [14]. The within-class covariance $\mathbf{S}_w$ is estimated as:

$$\mathbf{S}_w = \frac{1}{C}\sum_{c=1}^{C}\frac{1}{N_c}\sum_{i=1}^{N_c}(\mathbf{W}^c{}_i - \overline{\mathbf{W}^c})(\mathbf{W}^c{}_i - \overline{\mathbf{W}^c})^T \tag{1}$$

where $\overline{\mathbf{W}^c} = \frac{1}{N_c}\sum_{i=1}^{N_c}\mathbf{W}^c{}_i$ is the mean feature vector of class c and $\mathbf{W}^c{}_i$ are the samples belonging to this class. N_c is the number of observations of class c in the training set. We use the inverse of matrix $\mathbf{S}_w$ to normalize the direction of the projected feature vectors. The WCCN projection matrix $\mathbf{B}$ can be estimated using the Cholesky decomposition as $\mathbf{B} = cholesky(\mathbf{S}_w^{-1})$.

3.2 Deep Within-Class Covariance Analysis (DWCCA)

Based on the definitions above we propose Deep Within-Class Covariance Analysis (DWCCA), a DNN-compatible formulation of WCCN. The parameters of our networks are optimized with Stochastic Gradient Descent (SGD), and trained with mini-batches. The deterministic version of WCCN described above is usually estimated on the entire training set, which by the definition of SGD is not available in the present case. In the following we propose a DWCCA Layer which helps to circumvent these problems. Figure 1 shows a schematic sketch of how the DWCCA Layer can be incorporated in a neural network, and its gradient flow.

Instead of computing the within-class covariance matrix $\mathbf{S}_w$ on the entire training set $\mathbf{W}$ we provide an estimate $\hat{\mathbf{S}}_b$ on the observations $\mathbf{W}_b$ of the respective mini-batches. Given this estimate we compute the corresponding mini-batch projection matrix $\hat{\mathbf{B}}_b$ and use it to maintain a moving average projection matrix $\bar{\mathbf{B}}$ as $\bar{\mathbf{B}} = (1 - \alpha)\bar{\mathbf{B}} + \alpha\hat{\mathbf{B}}_b$.

This is done similarly when computing the mean and standard deviation for normalising the activations in batch normalization [17]. The hyper parameter α controls the influence of the data in the current batch b on the final DWCCA projection matrix. The output of this processing step is the DWCCA projected data $\mathbf{W}_b^{'} = \mathbf{W}_b\bar{\mathbf{B}}$ of the respective mini-batch.

The DWCCA Layer can be seen as a special dense layer with a predefined weight matrix, the projection matrix $\bar{\mathbf{B}}$, with the difference that the parameters are computed via the activations of the previous layer, and not learned via SGD. The proposed covariance normalization is applied directly during optimization. For training our network with back-propagation, we need to establish gradient flow. To this end, we implement the DWCCA Layer using the automatic differentiation framework Theano [4] which already provides the derivatives of matrix inverses and the Cholesky decomposition. We refer the interested reader to [30] for details on this derivative.

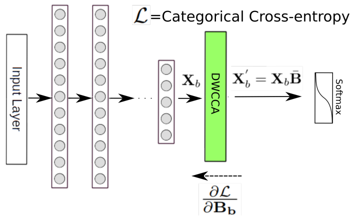

$\mathcal{L}$=Categorical Cross-entropy
Input Layer
$\mathbf{X}_b$
DWCCA
$\mathbf{X}_b^{'} = \mathbf{X}_b\bar{\mathbf{B}}$
Softmax
$\frac{\partial \mathcal{L}}{\partial \mathbf{B_b}}$

Figure 1: Diagram of the proposed layer in a DNN architecture.

4 Empirical Results

In this section, we detail our experimental setup and present our empirical results.

4.1 Dataset

To evaluate the performance of CNNs in situations with and without distribution mismatch, we use the TUT Acoustic Scenes 2016 (DCASE2016) [27] and TUT Acoustic Scenes 2017 (DCASE2017) [26] datasets.

The DCASE2016 dataset consists of 15 different acoustic scenes: lakeside beach, bus, cafe/restaurant, car, city center, forest path, grocery store, home, library, metro station, office, urban park, residential area, train, and tram. All audio material was collected as 3-5 minutes length recordings and then was cut into segments of 30 seconds length. The DCASE2017 dataset consists of the same 15 classes and uses the same recordings in its development set (training and validation) as used for DCASE2016 development set. But these recordings were split into only 10s long segments instead of 30s, which makes the learning and classification task harder. A new evaluation set (unseen test) was recorded for DCASE2017. The reason for choosing these two datasets is that DCASE2016 and DCASE2017 are based on the same development (= training and validation) data, but have different evaluations (= independent test) sets. While the test data for DCASE2016 were collected jointly with the development data, new test recordings were collected for DCASE2017, at a later time, in different places. Hence, there is a distribution mismatch between development and test data in the case of DCASE2017, as the many environments sound different from those in the development set (which will also show, indirectly, in our experimental results). This is detailed in the report of DCASE2017 [26].

In the following, we will use DCASE2016 to represent a data scenario without distribution shift, and DCASE2017 as the distribution shift case. In all experiments, we use the official 4-fold cross validation splits provided with the datasets and report CV results on all folds separately, as well as the performance of our method on the unseen test set.

Table 1: Model Specifications. BN: Batch Normalization, ReLu: Rectified Linear Activation Function, CCE: Categorical Cross Entropy.

Input 1×938 (DCASE2016) or 313 (DCASE2017) $\times 149$
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
DWCCA (if applied)
15-way Soft-Max

4.2 Baseline Systems and Experimental Setup

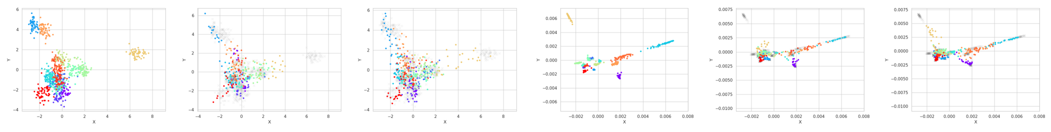
As explained before, we use a VGG-Style CNN proposed in [10] which uses audio spectrograms as inputs. We will refer to this model as *vanilla*.

To evaluate the effect of our proposed method, we apply a DWCCA layer on the output of global average pooling, as can be seen in Table 1. We will refer to this model in our results as *dwcca*. The internal *representations* we will analyze are computed by using the output of global average pooling layer in *vanilla*, and the output of the DWCCA layer in *dwcca*.

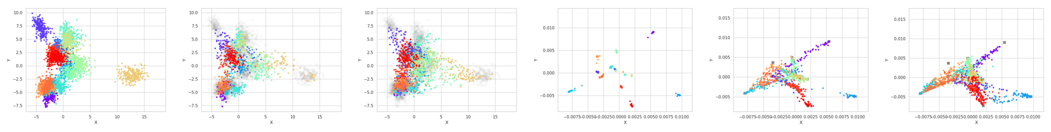
We also provide additional baselines for comparison: We report the performance of the best end-to-end CNN model [24] in the DCASE2016 challenge [25]. This method uses an ensemble of various CNN models. For the DCASE2017 experiments, we also choose the best-performing end-to-end CNN from the DCASE2017 challenge [26] as an additional baseline to vanilla: [34] that uses an ensemble fusion of various ResNets [15] using several channels of audio (left, right and difference).

We report classification results of our end-to-end CNNs on all folds, and on the unseen test set. For the unseen test, we average the probabilities produced by the 4 models trained on the individual training parts in CV folds from the development data. Additionally, we report *calibrated* results where we use a linear logistic regression model in each fold to regress the prediction probabilities of each model to the true labels. This method is better known as *late fusion* and proved successful in [10]. The *probabilistic averaging* for unseen test is done similarly for late fusion, as explained above.

The setup used in *all* of our experiments are as follows: The initial learning rate is set to 0.0001, and the ADAM optimizer is used [19]. We choose a similar α (0.1) to the one used in batchnorm. A model selection with max. patience of 20 is applied and the learning rate is halved in each step of maximum patience. The architecture of our models is specified in Table 1. All models are trained with stratified batch iterators and batch size of 75. In all our experiments, the spectrogram of the whole audio segment (30 sec in DCASE2016 and 10 sec in DCASE2017) was fed to our models. All spectrograms in our experiments are extracted with the *madmom* library [5] using the mono signal, and the *Lasagne* library [8] was used for building neural network architectures. The DWCCA Layer was implemented as a Lasagne-compatible layer in Theano [3].



(a) DCASE2016 training (vanilla) (b) DCASE2016 validation (vanilla) (c) DCASE2016 unseen test (vanilla) (d) DCASE2016 training (dwcca) (e) DCASE2016 validation (dwcca) (f) DCASE2016 unseen test (dwcca)



(g) DCASE2017 training (vanilla) (h) DCASE2017 validation (vanilla) (i) DCASE2017 unseen test (vanilla) (j) DCASE2017 training (dwcca) (k) DCASE2017 validation (dwcca) (l) DCASE2017 unseen test (dwcca)

Figure 2: DNN representations projected to 2D by PCA. The gray embeddings in low opacity (available only in validation and test plots) represent the training data. Note the distribution mismatch for vanilla.

4.3 Experiments Performed

We carried out various experiments to demonstrate the issue we encounter in ASC with CNNs. First, we train our models on DCASE2016 and DCASE2017. For a better understanding of the issues in the representation learned by the CNN, we provide a *visualization* of the VGG representations after being projected into 2D via Principal Component Analysis (PCA) [18]. The result can be found in Figure 2.

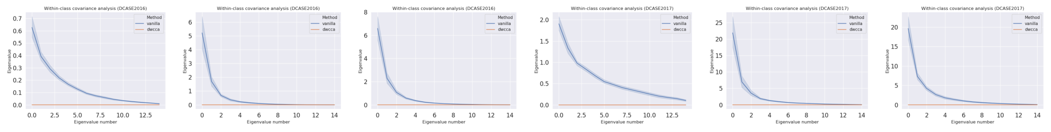
For analysing the *within-class covariance* of the representations, we apply an *eigenvalue decomposition* on the covariance matrix of the representations from each class. This technique is also used in other studies such as [21] to investigate the dynamics of learning in neural networks. These eigenvalues are a good indicator of the covariance of the representations in each class. If high, it means that the representations on that specific class have a high covariance, and if the difference between the highest eigenvalue and the lowest eigenvalue is high it also indicates that the variance of the representation in that class is not spread equally in all dimensions. These results can be found in Figure 3. The shade in this plot represents the variance of eigenvalues over different classes.

To indirectly assess the structure of the internal representation spaces learned, we carry out a *k-nearest-neighbor classification experiment* in these representation spaces; the K-nearest neighbor results on both datasets will be shown in Figure 4.

Finally, we report the *end-to-end classification results* on both datasets, for various methods; these can be found in Table 2. For a deeper understanding of the proposed method, we additionally provided the class-wise f-measure of DWCCA and the baseline VGG (vanilla) in Table 3.

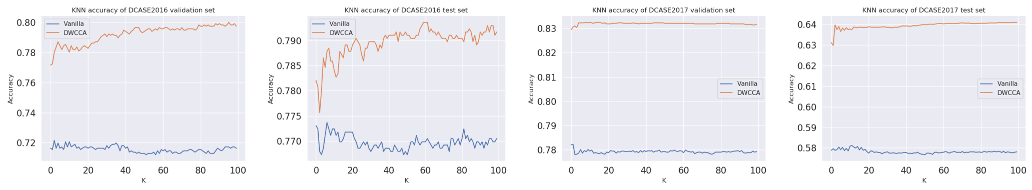
4.4 Results and Discussions

In Figure 2, the network's internal representations in each class are projected into 2D via PCA and each class is represented by a different colour. Looking at first (a-c) and second (d-f) row, it can be seen that for the dataset without mismatched distribution the embeddings of unseen data (validation and test) are spread less after applying DWCCA. Also comparing the unseen embeddings to the training embeddings (with lower opacity and in grey) it can be seen that the unseen embeddings projected closer to the training embeddings after applying DWCCA. Comparing third (g-i) and fourth (j-l) row, it can be seen that for the case of a distribution shift DWCCA also reduces the variance of the embeddings in each class, resulting in them being embedded closer to the training embeddings (grey). This suggests that this property can improve the generalisation of the representations. We will empirically evaluate this hypothesis later in this section by applying KNN classification on the representations.



(a) DCASE2016 training set (b) DCASE2016 validation set (c) DCASE2016 unseen test set (d) DCASE2017 training set (e) DCASE2017 validation set (f) DCASE2017 unseen test set

Figure 3: Eigenvalues of the covariance of the network representation. Shade represents the variance over different classes.



(a) DCASE2016 KNN results on validation set (b) DCASE2016 KNN results on unseen test set (c) DCASE2017 KNN results on validation set (d) DCASE2017 KNN results on unseen test set

Figure 4: KNN results: to predict the label of each unseen data point (validation or test), its representation is assigned to the labels of its K-nearest representation from the DNN's representations of training set. Axis x represents K and y axis shows the classification accuracy for each K.

Looking at Figure 3, we can see that in all plots from dataset with, and dataset without distribution shift, DWCCA significantly reduces the within-class variability. This can be observed by looking at the eigenvalues of the covariances of the representations. An interesting observation is the range of eigenvalues in vanilla: In both datasets, eigenvalues have significantly larger range on unseen data (validation and test) compared to the training data. The maximum eigenvalue in DCASE2016 is around 0.7, while the maximum eigenvalue for unseen is around 7, about 10 times more. Also the maximum eigenvalue of the train set of DCASE2017 is around 2, while the max. eigenvalue on unseen data is around 20 (10 times larger).

By looking at the KNN results in Fig. 4 it can be seen that in both cases (mismatch / no mismatch), the KNN classification accuracy increases by adding DWCCA. Also, while the KNN performance is in a reasonable range on the validation set of both datasets, the test accuraty in the mismatch case (DCASE2017) drops significantly compared to the validation set. Additionally it can be seen that applying DWCCA significantly improves the performance on the test set with shifted distribution, adding an improvement of about 6 percentage point, while the improvement on the test set without mismatch is around 2 percentage points. Looking at the results of end-to-end classifications in Table 2, we see that the performance of vanilla on DCASE 2017 consistently and significantly improves when adding DWCCA, on all development folds as well as on the unseen test data. We observe around 6 percentage points improvement by adding DWCCA to VGG.

Looking at the results of the dataset without mismatch, we see that although the results on all folds were improved by adding DWCCA, the results on the unseen test set do not significantly change. This can be explained better by looking at Figure 2: the embeddings of validation (b) and test (c) indicate that the test data is projected closer to the training set than the validation set. This observation suggests that the unseen test in DCASE2016 might be similar (even more similar than the validation data) to the training set. This can also be confirmed by looking at the results of the *best CNN* baseline, as well as vanilla: the performances on the unseen test set are consistently higher than all the validation folds. Hence, DWCCA could not help as there was not a large generalisation gap between training and test.

It is worth mentioning that both vanilla and DWCCA are single models, trained on mono single channel spectrograms and no ensemble or multi-channel features were used in these experiments. In other words, a single VGG model achieves comparable performances to an ensemble of multi-channel Resnets.

Table 2: Audio scene classification accuracy of different methods on DCASE2016 and DCASE2017 datasets. $*$: Single Model and single channel. †: Ensemble of various models. ‡: Multi-channel augmentation.

			fold 1	fold 2	fold 3	fold 4	test
DCASE'16	no calib	vanilla $*$	75.39	66.80	78.52	75	84.36
		dwcca $*$	82.42	75.78	87.89	84.37	83.60
	calib	vanilla $*$	76.55	67.59	78.27	76.21	85.64
		dwcca $*$	86.55	77.24	83.79	86.90	85.90
		Best CNN †[24]	84.40	78.20	82.70	80.80	86.40
DCASE'17	no calib	vanilla $*$	82.03	73.35	75.95	82.55	62.96
		dwcca $*$	84.03	81.08	79.43	87.41	66.23
	calib	vanilla $*$	82.73	73.93	76.50	84.61	62.47
		dwcca $*$	87.18	84.61	81.28	87.26	68.70
		Best CNN †‡ [34]	86.00	87.80	82.60	86.00	70.00

We also provide class-wise f-measures on the unseen test set for both datasets in Table 3. While on the dataset without distribution shift, the average f1 stays the same by adding DWCCA in both calibrated and non calibrated models, we can observe that there is a boost of *13 percentage points* on the "train" class which was the class with the lowest f1 (both calibrated and non calibrated). It seems that DWCCA does not have a significant impact on classes with high f1: "office" and "beach" which stay the highest correctly predicted classes and do not face significant changes by DWCCA.

On the dataset with distribution shift, we can see a significant improvement of 4 and 7 percentage points on average f1 for non-calibrated and calibrated models, respectively. The worst class in DCASE2017 was "beach" with 32%, which was boosted by *24* and *37* percentage points for non-calibrated and calibrated models, respectively. On the other hand, the best performing class, "forest path", drops by only 2 and 3 percentage points for non-calibrated and calibrated models, respectively.

From the experimental results, we may thus conclude that overall, reducing the within-class covariance of representations using DWCCA results in more robust performance and, in case of a large gap between training and test, DWCCA can improve the generalisation. Additionally, the networks tend to reach a more uniform performance across various classes by improving the performance on the worst classes while not significantly degrading the best performing classes.

5 Conclusion

In this paper, we presented the DWCCA layer, a DNN compatible version of the classic WCCN which is used to normalize the within-class covariance of the DNN's representation and improve the performance of CNNs on data-points with shifted distributions. Using DWCCA, we improved the performance of the VGG network by around 6 percentage point when the test datapoints were from a shifted distribution. We analysed the embedding's generalisation by reporting KNN classification accuracies and showed that DWCCA also improves the generalisation of DNN representations both for with and without distribution mismatch. We also showed that large within-class covariance of representations can be a sign for bad generalisation and showed that DWCCA can significantly reduce WCC and improve generalisation of the representations.

6 Acknowledgment

This work was supported by the Austrian Ministry for Transport, Innovation & Technology, the Ministry of Science, Research & Economy, and the Province of Upper Austria in the frame of the COMET center SCCH. The authors would like to thank Hasan Bahari of KU Leuven for feedback and fruitful discussions. We also gratefully acknowledge the support of the NVIDIA Corporation with the donation of a Titan X GPU used for this research.

Table 3: The class-wise f1-score comparing the effect of adding DWCCA to Vanilla VGG on DCASE-2016 and DCASE-2017 unseen test sets. Worst-performing classes in vanilla (which benefit most from DWCCA) are shown in bold

	DCASE2016				DCASE2017			
	no calib		calib		no calib		calib	
	vanilla	dwcca	vanilla	dwcca	vanilla	dwcca	vanilla	dwcca
beach	0.96	0.96	0.98	0.96	**0.32**	**0.56**	**0.32**	**0.69**
bus	0.88	0.84	0.88	0.9	0.67	0.66	0.68	0.67
cafe/restaurant	0.62	0.54	0.68	0.62	0.69	0.77	0.64	0.76
car	0.94	0.96	0.96	0.96	0.83	0.84	0.84	0.84
city_center	0.94	0.86	0.93	0.81	0.73	0.66	0.72	0.67
forest_path	0.96	0.98	0.98	0.98	0.89	0.87	0.89	0.86
grocery_store	0.81	0.79	0.84	0.84	0.58	0.69	0.59	0.73
home	0.93	0.92	0.91	0.91	0.69	0.7	0.67	0.7
library	0.63	0.56	0.62	0.64	**0.37**	**0.54**	**0.36**	**0.61**
metro_station	0.75	0.79	0.81	0.85	0.47	0.51	0.45	0.57
office	1	1	0.98	1	0.77	0.72	0.76	0.72
park	0.77	0.78	0.77	0.77	0.39	0.36	0.41	0.46
residential_area	0.79	0.78	0.82	0.77	0.49	0.48	0.5	0.53
train	**0.60**	**0.73**	**0.62**	**0.87**	0.78	0.86	0.76	0.82
tram	0.93	0.91	0.93	0.94	0.65	0.63	0.64	0.59
avg	0.83	0.83	0.85	0.85	0.62	0.66	0.61	0.68

References

[1] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

[2] Daniele Barchiesi, Dimitrios Giannoulis, Dan Stowell, and Mark D Plumbley. Acoustic scene classification. *arXiv preprint arXiv:1411.3715*, 2014.

[3] James Bergstra, Frédéric Bastien, Olivier Breuleux, Pascal Lamblin, Razvan Pascanu, Olivier Delalleau, Guillaume Desjardins, David Warde-Farley, Ian Goodfellow, Arnaud Bergeron, et al. Theano: Deep learning on gpus with python. In *NIPS 2011, BigLearning Workshop, Granada, Spain*, volume 3, pages 1–48. Citeseer, 2011.

[4] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, 2010. Oral Presentation.

[5] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. Madmom: A new python audio and music signal processing library. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 1174–1178. ACM, 2016.

[6] Najim Dehak, Reda Dehak, James Glass, Douglas Reynolds, and Patrick Kenny. Cosine Similarity Scoring without Score Normalization Techniques. *Proceedings of Odyssey 2010 - The Speaker and Language Recognition Workshop (Odyssey)*, 2010.

[7] Najim Dehak, Patrick Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-end factor analysis for speaker verification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 2011.

[8] Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, et al. Lasagne: first release. *Zenodo: Geneva, Switzerland*, 3, 2015.

[9] Matthias Dorfer, Rainer Kelz, and Gerhard Widmer. Deep Linear Discriminant Analysis. *International Conference on Learning Representations (ICLR)*, 2016.

[10] Hamid Eghbal-zadeh, Bernhard Lehner, Matthias Dorfer, and Gerhard Widmer. Cp-jku submissions for dcase-2016: a hybrid approach using binaural i-vectors and deep convolutional neural networks, 2016.

[11] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 1936.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[13] A Hatch and A Stolcke. Generalized linear kernels for one-versus-all classification: application to speaker recognition. *IEEE International Conference on Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings.*, 2006.

[14] Ao Hatch. Within-class covariance normalization for SVM-based speaker recognition. *Interspeech*, 2006.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Rakib Hyder, Shabnam Ghaffarzadegan, Zhe Feng, and Taufiq Hasan. BUET bosch consortium (B2C) acoustic scene classification systems for DCASE 2017. Technical report, DCASE2017 Challenge, September 2017.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, 2015.

[18] Ian Jolliffe. Principal component analysis. In *International encyclopedia of statistical science*, pages 1094–1096. Springer, 2011.

[19] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? *arXiv preprint arXiv:1805.08974*, 2018.

[21] Yann Le Cun, Ido Kanter, and Sara A Solla. Eigenvalues of covariance matrices: Application to neural-network learning. *Physical Review Letters*, 66(18):2396, 1991.

[22] Bernhard Lehner, Hamid Eghbal-Zadeh, Matthias Dorfer, Filip Korzeniowski, Khaled Koutini, and Gerhard Widmer. Classifying short acoustic scenes with i-vectors and cnns: Challenges and optimisations for the 2017 dcase asc task. Technical report, DCASE2017 Challenge, Tech. Rep, 2017.

[23] Ming Li, Andreas Tsiartas, Maarten Van Segbroeck, and Shrikanth S Narayanan. Speaker verification using simplified and supervised i-vector modeling. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7199–7203. IEEE, 2013.

[24] Erik Marchi, Dario Tonelli, Xinzhou Xu, Fabien Ringeval, Jun Deng, Stefano Squartini, and Björn Schuller. The up system for the 2016 DCASE challenge using deep recurrent neural network and multiscale kernel subspace learning. Technical report, 2016.

[25] Annamaria Mesaros, Toni Heittola, Emmanouil Benetos, Peter Foster, Mathieu Lagrange, Tuomas Virtanen, and Mark D Plumbley. Detection and classification of acoustic scenes and events: Outcome of the dcase 2016 challenge. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 26(2):379–393, 2018.

[26] Annamaria Mesaros, Toni Heittola, Aleksandr Diment, Benjamin Elizalde, Ankit Shah, Emmanuel Vincent, Bhiksha Raj, and Tuomas Virtanen. Dcase 2017 challenge setup: Tasks, datasets and baseline system. In *DCASE 2017-Workshop on Detection and Classification of Acoustic Scenes and Events*, 2017.

[27] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. Tut database for acoustic scene classification and sound event detection. In *Signal Processing Conference (EUSIPCO), 2016 24th European*, pages 1128–1132. IEEE, 2016.

[28] Seongkyu Mun, Sangwook Park, David Han, and Hanseok Ko. Technical report, DCASE2017 Challenge, September 2017.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[30] Stephen P Smith. Differentiation of the cholesky algorithm. *Journal of Computational and Graphical Statistics*, 1995.

[31] Michele Valenti, Aleksandr Diment, Giambattista Parascandolo, Stefano Squartini, and Tuomas Virtanen. Dcase 2016 acoustic scene classification using convolutional neural networks. In *Proc. Workshop Detection Classif. Acoust. Scenes Events*, pages 95–99, 2016.

[32] Zheng Weiping, Yi Jiantao, Xing Xiaotao, Liu Xiangtao, and Peng Shaohu. Acoustic scene classification using deep convolutional neural network and multiple spectrograms fusion. Technical report, DCASE2017 Challenge, 2017.

[33] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[34] Shengkui Zhao, Thi Ngoc Tho Nguyen, Woon-Seng Gan, and Jones Douglas L. ADSC submission for DCASE 2017: Acoustic scene classification using deep residual convolutional neural networks. Technical report, DCASE2017 Challenge, 2017.