Deep Fisher Score Representation via Sparse Coding

Sixiang Xu, Damien Muselet, Alain Trémeau

Deep Fisher Score Representation via Sparse Coding

Sixiang Xu, Damien Muselet, and Alain Trémeau

Laboratoire Hubert Curien UMR 5516, F-42023, SAINT-ETIENNE, France
{sixiang.xu,damien.muselet,alain.tremeau}@univ-st-etienne.fr

Abstract. Fisher Score has been shown to be accurate global image features for classification. Most of time, it is based on a Gaussian mixture model (GMM). Nevertheless, recent studies show that GMM does not fit well high dimensional data such as the ones extracted by deep convolutional networks. In this paper, we propose to resort to a sparse representation of the centers of the Gaussian functions in order to better cover the high dimensional feature space. This solution has already been used in a framework constituted by independent and off-the-shelf modules and the contribution of this paper is to embed these steps in an end-to-end deep neural network so that all the modules work together for the sole purpose of improving classification performance. Experimental results show that this solution clearly outperforms many alternatives in the context of material, indoor scenes or fine-grained image classification.

Keywords: Fisher Score · Sparse Coding · Orderless Pooling · Classification.

1 Introduction

Deep neural networks have emerged as an essential solution for performing classification tasks. In these networks, convolutional layers extract accurate local features that are pooled to a local feature vector which is sent to fully connected layers for classification. The first networks neglected the pooling step and directly sent the set of local features in the dense layers [20], while the series of ResNet apply a global average pooling to decrease the dimension of the global feature vector and hence reduce the number of parameters of the network [8]. Orderless pooling was widely used before convolutional neural networks (CNN) with the bags of visual words (BOW) [12], VLAD [10] or Fisher Vectors [21] and has shown to provide good results when applied to CNN features [4, 6]. Among them, the Fisher Vectors (FV) were the most promising because they generalize the VLAD and BOW. The main idea of FV is to model the distribution of the training data with a Gaussian mixture and to characterize each data point with the derivatives over the model parameters. This coding approach is referred as Gaussian Mixture Model based Fisher Vector Coding(GMMFVC). Nevertheless, a Gaussian Mixture Model (GMM) seems not to be well adapted to the deep local features since they are lying in a very high dimensional space and require

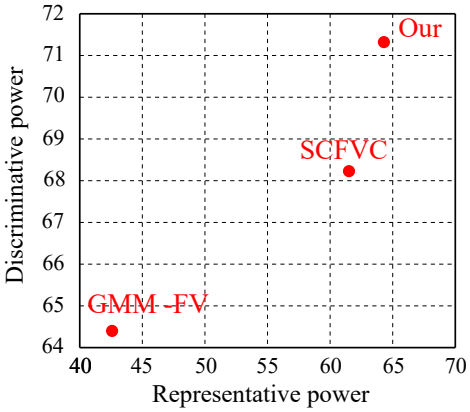


Fig. 1. Discriminative power and representative power of the Gaussian Mixture Model (GMM), the Off-the-shelf Sparse Coding solution (SCFVC) proposed in [15] and our solution. See text for details.

to many Gaussians to be accurately modeled [15]. Liu et al. proposed a smart solution to overcome this problem which consists in sampling the center of each Gaussian from a subspace and therefore benefiting from an infinite number of Gaussians to fit the data distribution [15]. The authors show that this problem can be solved by a classical sparse coding method. Unfortunately, their approach can not take advantage of the main interest of the CNN, i.e. training end-to-end the feature extraction, the pooling and the classification layers. In this paper, we propose a solution to embed all these modules in a deep CNN that can be trained end-to-end. This way, we take advantage of the sparse coding solution of [15] to improve the representative power of our model (compared to the classical GMM) thanks to an infinite number of Gaussians and we also improve the discriminative power (over [15] and GMM) of the different elements (subspace bases and sparse codes) thanks to the end-to-end training.

For illustration, Fig. 1 displays the representative and discriminative powers of the GMMFVC, the Off-the-shelf Sparse Coding solution (SCFVC) proposed in [15] and our solution. These values are evaluated on the MIT indoor dataset [18] with AlexNet [11]. The representative power is evaluated as $100 - d$, where d is the average distance between the data points and their respective nearest Gaussian center. The discriminative power is the classification accuracy of the method. This Figure clearly shows that inserting the sparse coding and Fisher vector extraction in the network allows to improve both criteria.

The classical sparse coding problem presented in [15] is a regression with L_1 norm regularization (called LASSO regression). Using proximal gradient descent, it can be solved with an iterative algorithm with soft-thresholding, called ISTA [5]. Gregor and Lecun have proposed in [7] to approximate this solver with an unfolded module (LISTA) that can be inserted in a deep network. In this paper, we use LISTA to learn a discriminative dictionary and to extract an adapted sparse code for each input data. These dictionary and sparse code allow us to evaluate the corresponding Fisher vector that is the input of the classification layers. By backpropagating the gradient of the classification loss, we are able to make all these modules (local feature extraction, LISTA, Fisher Vector

and classification) collaborate with the sole objective of improving the performance of the classification task. Experimental tests on three different datasets and three different backbone architectures show that our solution outperform many alternatives.

2 Related Works

Orderless pooling was widely used before the emergence of the CNN-based solutions. The most popular approaches were based on bags of visual words (BOW) [12], VLAD [10] or Fisher Vectors [21]. Inspired by these early methods, some works have evaluated the Fisher vectors or VLAD from deep features for texture or image classification [4, 6]. They show improvements over the SIFT-based counterparts but, in their workflow, the dictionary or Gaussian mixture model are learned independently from the deep features and from the classifier, leaving a large margin of progression.

Thus, the next works have focused on embedding orderless pooling in deep networks to allow end-to-end training. Passalis and Tefas have inserted a Bag-of-Features pooling in deep neural networks thanks to radial basis function neurons [17]. The output of the pooling module is a histogram of the visual words (0^{th} order statistic) learned on the training set.

Instead of counting the occurrences of the visual words in one image, VLAD-based approaches aggregate the residuals between the local features and their nearest visual words (1^{st} order statistic). NetVLAD is the first network that solves this task with an end-to-end training [1] and is later improved by Zhang et al. with Deep Ten [26]. It has been show that first order statistics are more accurate to characterize images in classification tasks and the Fisher vectors go further by using first and second order statistics. Deep FisherNet is an embedded implementation of the GMM Fisher vector [22]. [14] introduces NetFV which extends NetVLAD by appending the second order statistics. The main disadvantage of all these approaches is that they rely on a limited number of codewords or Gaussian centers, which prevents accurate modeling of the data distribution in the high-dimensional deep feature spaces [15].

One interesting solution to cope with this problem has been proposed by Li et al [13]. The authors compute Fisher vectors from a mixture of factor analyzers (MFA), instead of the classical GMM. Their solution is embedded in a deep network which is trainable end-to-end. The idea of MFA is to approximate the data manifold by low dimensional linear spaces and, in this sense, is similar to the idea of sparse coding [15]. Nevertheless, even if the MFA module is embedded in a deep network, the authors show that an accurate initialization of the weights of the network is required to obtain good performance. This initialization consists in running an Expectation-Maximization algorithm on the set of local features that have to be saved in memory. Furthermore, it appears that this second order representation requires high computation costs, high number of parameters to learn and occupies a very large memory space (500k dimensions which is more than the image itself) [9].

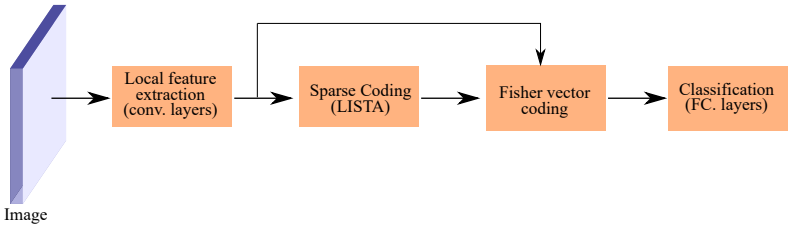


Fig. 2. Workflow of the proposed solution.

Another group of second-order pooling works is based on bilinear coding, such as BCNN [14] which is also an end-to-end trainable network and aggregates feature vectors by sum-pooling their outer products. Since this pooled representation always has cumbersome size, SMSO[25] proposes to compress the bilinear pooled features and improves the classification performance.

Our method is inspired by the work of Liu et al. [15], detailed in the next section,. More recently, they have also proposed an improved version of their work in [16], called HSCFV. It uses two dictionaries to code input features and consequently, doubles dimension size of the Fisher vector. Nevertheless, their approach is not embedded in a deep CNN for end-to-end training.

Our method combines all the benefits of these previous solutions: it is embedded in an end-to-end trainable network, it samples an infinite number of Gaussian centers from a learned subspace and it does not require any heavy computation or storage to initialize the weights.

3 Deep sparse coding Fisher vector

Fig. 2 illustrates the complete workflow of our solution whose successive steps are detailed in the next sections.

3.1 From subspace sampling to sparse coding

In order to increase the number of Gaussians that model the distribution of the data, we take advantage of the idea from [15] that sample the Gaussian mean vectors in a subspace spanned by a set of bases. Each mean vector is coded in this "dictionary" B with a code u drawn from a zero-mean Laplacian distribution (to enforce sparsity). Then each local feature vector x extracted from the images and associated with the code u is drawn from a Gaussian distribution $\mathcal{N}(Bu, \Sigma)$ centered on Bu. Fig. 3 illustrates the interest of this approach.

Then, assuming a constant and diagonal covariance matrix as σ and using pointwise maximum to approximate the integral of the distribution, Liu et al. show that the logarithm of the likelihood of x can be estimated as [15]:

$$log(P(x|B)) = \min_u \frac{1}{\sigma^2}||x - Bu||_2^2 + \lambda||u||_1, \tag{1}$$

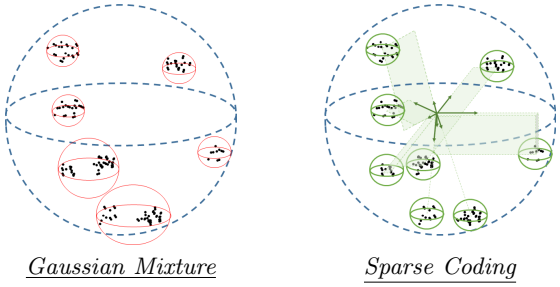


Fig. 3. Some data in a high dimensional space (illustrated by the sphere). Left: With GMM the data distribution is not well fitted because of the limited number of Gaussians. Right: With Sparse Coding, the Gaussian centers are coded sparsely in an adapted basis (green arrows) allowing to create unlimited number of Gaussians and so to fit better the data distribution. The sparsity is illustrated by the low number of basis required to code each center position (lines, planes or parallelograms).

where λ is the scale parameter of the Laplacian distribution of u.

Interestingly, this equation represents the classical problem of sparse coding. Liu et al. proposed to use an off-the-shelf sparse coding solver to learn the dictionary B and infer the code u [15]. Obviously, making use of such independent solver is a good solution to minimize the reconstruction error of x with a sparse code, but it neglects the main goal which is to improve the performance of the classification task.

Hence, we propose in the next section to embed a sparse coding module in a deep neural network that is trained end-to-end. The main advantage of such an approach is that it is learning a dictionary and sparse codes that are accurate to discriminate the different categories in the current dataset.

3.2 Embedding sparse coding with LISTA

Our aim is to find a solution for the following equation:

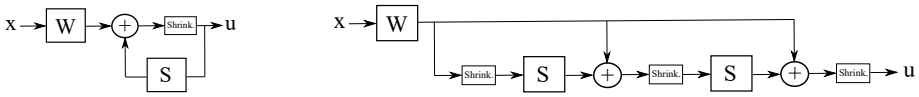
$$\min_u f(u) + \lambda ||u||_1 \tag{2}$$

where $f(u) = ||x - Bu||_2^2$, x is a data point, B the dictionary and u the sparse code of x.

One way to solve this equation is to resort to an Iterative Shrinkage/Thresholding Algorithm (ISTA) [5] that iteratively approximates the solution with:

$$u_k = \mathcal{T}_{\lambda t_k}(u_{k-1} - t_k \nabla f(u_{k-1})), \tag{3}$$

where $\mathcal{T}_\alpha$ is a component-wise vector shrinkage function such that $[\mathcal{T}_\alpha(v)]_i = (|v_i| - \alpha)_+ sign(v_i)$, t_k is the step size at iteration k and ∇ is the gradient operator.



ISTA block diagram *Learned ISTA block diagram*

Fig. 4. Block diagrams of ISTA and LISTA. LISTA is an unfolded version of ISTA (2 iterations here).

Evaluating the gradient of $f(u)$ defined above, we get:

$$\begin{aligned} u_k &= \mathcal{T}_{\lambda t_k}(u_{k-1} - 2t_k B^T(Bu_{k-1} - x)), \\ &= \mathcal{T}_{\lambda t_k}((I - 2t_k B^T B)u_{k-1} + 2t_k B^T x), \\ &= \mathcal{T}_{\lambda t_k}(Su_{k-1} + Wx), \end{aligned}$$

where $S = I - 2t_k B^T B$ and $W = 2t_k B^T$.

As mentioned in [7], this equation can be illustrated as a recurrent block diagram as in Fig. 4, left. Fortunately, Gregor and Lecun proposed a fast approximation of ISTA called Learned ISTA (LISTA) [7]. This is an unfolded version of ISTA with a fix number of iterations and that can be plugged into a neural network to provide a sparse code (see Fig.4, right). Embedding this LISTA module in our CNN is a smart solution to learn a dictionary and sparse codes that help to discriminate between the categories of the current task.

3.3 Dictionary based Fisher coding

When a classical GMM is used to model the data distribution, the Fisher code is based on the partial derivatives of the posterior probabilities with respect to the weights, the mean and the standard-deviation parameters of the model [21]. In our case, the model is based on a learned dictionary and we use a particular Fisher coding, as in [15], evaluated as the partial derivative of the log probability of the local features with respect to the dictionary itself:

$$\frac{\partial log(P(x|B))}{\partial B} = \frac{\partial \frac{1}{\sigma^2}||x - Bu^*||_2^2 + \lambda||u^*||_1}{\partial B} = (x - Bu^*){u^*}^T, \tag{4}$$

where $u^* = argmax_u P(x|u, B)P(u)$ (see [15] for details).

This module is very easy to insert in our deep network and provides the pooled features from the input image. These features are then sent to the last fully connected layers for classification. All these modules are constituting our CNN which can be trained end-to-end (see Fig. 2).

4 Experiments

In this section, we are running experimental tests on different datasets and compare our results with those of many alternatives. The datasets and their respective experimental settings are detailed in Sections 4.1 and 4.2. The training

strategy of our network is presented in Section 4.3. Finally, the results and comparisons are commented in Section 4.4.

4.1 Datasets

In order to show the versatility of our solution for image classification tasks, we run experiments on three datasets, which vary between tasks and scales. Note that we always make use of official training-test splits released with the datasets.

MINC-2500 [2] is a large-scale material dataset containing 23 commonly-seen material categories, such as water, wood or paper. There are in total $2,500$ images per category among which $2,350$ are used for training. MIT Indoor 67[18] is a medium but widely-accepted benchmark for indoor scene classification task with 67 indoor categories and 100 images in each category. 80 images per category are used for training. CUB-200-2011[23] consists of $11,788$ images with 200 bird species and is always considered as a fine-grained classification dataset because inter-class difference between bird species is very subtle.

4.2 Experimental settings

Depending on the tested dataset, we use different backbones for fair comparison with other works. Our deep pooling module (DPM) is constituted by a 1×1 convolution layer, a LISTA module with two iterations (see Fig. 4) and the Fisher encoding layer. Then the last layer is a fully connected layer with softmax activation for classification. The loss is the classical cross-entropy.

When testing on MIT-67 and CUB-200 2011 datasets, we follow the settings adopted by the state of the art methods [25, 14]. The input image size is 448x448 and the backbone networks are either the pretrained VGG-D (a.k.a VGG-16) or Alexnet. Our DPM is plugged on the last convolutional layer for VGG-D and on the Fc6 layer for Alexnet. The 1×1 convolutional layer in our DPM does not change the input feature size and the sparse code in LISTA has 100 elements.

For the tests on MINC-2500, the network backbone is the pretrained ResNet-50[8]. The 1×1 convolutional layer in our DPM reduces the input feature size to 128 and the provided sparse code in LISTA has 32 elements. While training, we follow the data augmentation settings of [24]: the input image is resized to 256x256, 8% to 100% of the area of the of image is cropped with a random aspect ratio between $\frac{3}{4}$ and $\frac{4}{3}$ and the crop is resized to 224x224. 50% chance horizontal and vertical flip is applied. At test time, we use central crop of 224x224 as input.

4.3 Training details

For training our network, three consecutive steps are conducted. First, we run a PCA on a small subset of feature vectors (around $10,000$) extracted from the backbone outputs and initialize the 1×1 convolutional layer of our DPM with these PCA parameters. Second, inspired by [3], we apply a warming-up process that consists in training our DPM and FC layer (while the backbone

Table 1. Comparison of the classification accuracy (%) with closed-related alternatives on three datasets and three backbone architectures.

	Approaches	MIT AlexNet	MIT VGG16	CUB AlexNet	CUB VGG16	MINC ResNet50
Off-shelf	Baseline	58.4[19]		53.3[19]	60.4[14]	
	GMMFVC	64.3[15]	72.6[a][16]	61.7[15]	70.1[a][16]	
	SCFVC	68.2[15]	77.6[a][16]	66.4[15]	77.3[a][16]	
	HSCFVC		79.5[a][16]		80.8 [16]	
End-to-end	Baseline		64.51[25]		70.4[14]	79.1[25]
	Deep Ten					80.4[24]
	NetVLAD				81.9[14]	
	NetFV		78.2[14]		79.9[14]	
	FisherNet		76.4[13]			
	MFAFVNet	69.89[b][13]	78.01[b][13]			
	B-CNN		77.6[14]		84.0[14]	79.05[25]
	SMSO		79.45[25]		**85.01**[25]	81.3[25]
	Our	**70.3**	**80.22**	**73.4**	84.28	**81.5**

[a] *These methods were trained with VGG19 (not VGG16) with 2 scales, whereas the other approaches from the column are trained with a single scale.*

[b] *Since MFAFVNet works on patches and not on images, we have selected in [13] the results provided with the nearest patch scale from our settings (*160×160*).*

is frozen) with an objective function which is the sum of the cross-entropy loss and the sparse coding loss (see Eq. (1)). Finally, the whole network is fine-tuned end-to-end under the supervision of the sole cross-entropy loss.

For training, we use stochastic gradient descent as optimization algorithm with a mini-batch size of 64, a weight decay of $5e^{-4}$ and a momentum of 0.9. The learning rate is 0.004 during the warming-up. During the end-to-end finetuning, it starts from 0.004 and is divided by 10 when the training loss meets a plateau.

4.4 Results

The top-1 classification accuracy of our approach and many alternatives are resumed in Table 1. The results of the related works are extracted from different papers that are referenced in this table. Note that our CNN is trained on single-scale images while many state-of-the-art approaches are training on multi-scales, so we have carefully selected the results that allows fair comparisons, even if some results in Table 1 are from multi-scale training.

The methods called 'Off-the-shelf' use independent modules that are not fine-tuned together while the 'Finetuned' group contains approaches that use end-to-end trainable networks. We notice that the results provided by fine-tuned networks overall outperform those of the Off-the-shelf solutions. This shows that it is always better to make the modules work together to optimize the same loss instead of independently optimizing them. Also our approach is built upon

SCFVC which produces more discriminant second-order pooled features than the classical Fisher vector or VLAD. The proposed smart combination of these two advantages make our method outperform most of the alternatives for all the datasets and backbones.

5 Conclusion

Fisher vectors are very accurate features for classification but require many Gaussians when applied on high-dimensional deed features. One way to cope with this problem is to code sparsely the Gaussian centers in an adapted basis in order to increase the number of available Gaussians and better fit the data distribution. In this paper, we have shown that this coding can be embedded in a deep network allowing to adapt the basis and sparse code such that they optimize the classification performance. We have also proposed a training strategy that can easily and quickly initialize the network parameters before finetuning. With the support of the end-to-end learning and a powerful Fisher score representation, our method outperforms many alternatives on three different datasets.

References

1. Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: Netvlad: Cnn architecture for weakly supervised place recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
2. Bell, S., Upchurch, P., Snavely, N., Bala, K.: Material recognition in the wild with the materials in context database. Computer Vision and Pattern Recognition (CVPR) (2015)
3. Branson, S., Van Horn, G., Belongie, S., Perona, P.: Bird species categorization using pose normalized deep convolutional nets. arXiv preprint arXiv:1406.2952 (2014)
4. Cimpoi, M., Maji, S., Vedaldi, A.: Deep filter banks for texture recognition and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3828–3836 (2015)
5. Daubechies, I., Defrise, M., Mol, C.: An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Comm. Pure Appl. Math. **57**, 1413–1457 (2004)
6. Gong, Y., Wang, L., Guo, R., Lazebnik, S.: Multi-scale orderless pooling of deep convolutional activation features. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 392–407 (2014)
7. Gregor, K., LeCun, Y.: Learning fast approximations of sparse coding. In: Proc. International Conference on Machine learning (ICML'10) (2010)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)
9. jacob, P., Picard, D., Histace, A., Klein, E.: Efficient codebook and factorization for second order representation learning. In: Proc. International Conference on Learning Representations (ICLR) (2019)

10. Jégou, H., Perronnin, F., Douze, M., Sánchez, J., Pérez, P., Schmid, C.: Aggregating local image descriptors into compact codes. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **34(9)** (2012)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25**, 1097–1105 (2012)
12. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). vol. 2, pp. 2169–2178 (2006)
13. Li, Y., Dixit, M., Vasconcelos, N.: Deep scene image classification with the mfafvnet. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5746–5754 (2017)
14. Lin, T.Y., RoyChowdhury, A., Maji, S.: Bilinear convolutional neural networks for fine-grained visual recognition. IEEE transactions on pattern analysis and machine intelligence **40**(6), 1309–1322 (2017)
15. Liu, L., Shen, C., Wang, L., Hengel, A.v.d., Wang, C.: Encoding high dimensional local features by sparse coding based fisher vectors. In: Advances in Neural Information Processing Systems(NIPS) (2014)
16. Liu, L., Wang, P., Shen, C., Wang, L., Van Den Hengel, A., Wang, C., Shen, H.T.: Compositional model based fisher vector coding for image classification. IEEE transactions on pattern analysis and machine intelligence **39**(12), 2335–2348 (2017)
17. Passalis, N., Tefas, A.: Learning bag-of-features pooling for deep convolutional neural networks. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 5766–5774 (2017)
18. Quattoni, A., Torralba, A.: Recognizing indoor scenes. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 413–420. IEEE (2009)
19. Sharif Razavian, A., Azizpour, H., Sullivan, J., Carlsson, S.: Cnn features off-the-shelf: An astounding baseline for recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (June 2014)
20. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Proc. International Conference on Learning Representations (ICLR'15) (2015)
21. Sánchez, J., Mensink, T., Verbeek, J.: Image classification with the fisher vector: Theory and practice. International Journal of Computer Vision **105** (12 2013)
22. Tang, P., Wang, X., Shi, B., Bai, X., Liu, W., Tu, Z.: Deep fishernet for image classification. IEEE transactions on neural networks and learning systems **30**(7), 2244–2250 (2019)
23. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011)
24. Xue, J., Zhang, H., Dana, K.: Deep texture manifold for ground terrain recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 558–567 (2018)
25. Yu, K., Salzmann, M.: Statistically-motivated second-order pooling. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 600–616 (2018)
26. Zhang, H., Xue, J., Dana, K.: Deep ten: Texture encoding network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 708–717 (2017)