FISHER VECTOR BASED CNN ARCHITECTURE FOR IMAGE CLASSIFICATION

Yan Song[1], *Peiseng Wang*[1], *Xinhai Hong*[1], *Ian McLoughlin*[2],

[1] Department of EEIS, University of Sci. and Tech. of China, Hefei, Anhui, China
[2] School of Computing, University of Kent, Medway, UK

ABSTRACT

In this paper, we tackle the representation learning problem for small scale fine-grained object recognition and scene classification tasks. Conventional bag of features(BoF) methods exploit hand-crafted frontend local features, and learn the representations via various machine learning techniques. Convolutional neural networks(CNN) directly learn the representation from raw images and benefit from joint optimization of network parameters in an end-to-end manner. However, the performance of existing representation learning methods is still unsatisfactory for the small-scale recognition tasks. To address this issue, we present a FV coding based CNN(FV-CNN) architecture. FV-CNN has three main advantages in that firstly it is able to exploit activations from the intermediate convolutional layer and a probabilistic discriminative model to derive the FV coding. Secondly, it takes advantage of the end-to-end back-propagation of the gradients to jointly optimize the whole learning process. Finally, it can learn a compact representation. When evaluated on benchmark datasets of fine grain object recognition (Caltech-CUB200), and scene classification (MIT67), accuracies of 88.0% and 82.2% are achieved.

***Index Terms*—** Image Classification, Visual Representation, Convolutional Neural Network, End-to-End Training

1. INTRODUCTION

Effective visual representation plays an important role in modern image classification applications. Over the past few decades, much effort has been devoted to learning representations that are robust to large intra-class variation caused by issues such as image scaling, translation and occlusion.

A typical representation learning pipeline generally consists of frontend feature extraction and backend modeling steps. In conventional image classification methods, the bag of features(BoF) representation has been widely adopted due to its simplicity and efficiency. Local features like SIFT [1] and HoG [2] are first extracted from densely sampled image patches. These are then encoded and aggregated into the BoF representation via various machine learning methods,

Thanks to the support of National Natural Science Foundation of China grant no. 61172158.

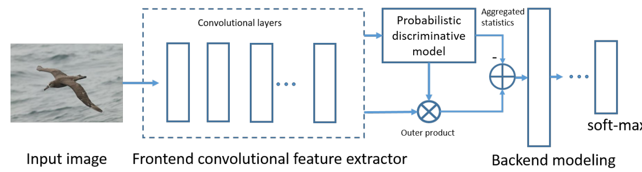

Fig. 1. The FV Coding based CNN architecture

such as sparse coding [3], locality-constrained linear coding (LLC) [4], second-order pooling (O2P) [5], vector of locally aggregated descriptor (VLAD) [6], and Fisher vector (FV) [7]. Among these, O2P, FV and VLAD can achieve outstanding recognition performance with a linear SVM.

Recently, deep convolutional neural networks (CNN) have achieved state-of-the-art performance in large-scale visual recognition tasks [8, 9, 10]. Conceptually, the first few convolutional layers can be considered as the feature extractor, while the later fully connected (FC) layers perform backend modeling. Compared to BoF, CNN models can discriminatively learn the visual representation from raw images in an end-to-end manner. This enables joint optimization of the whole representation learning pipeline, leading to significant improvements in recognition performance. However, for small scale tasks such as fine grained recognition and scene classification, CNN performance is limited due to the relative lack of training data. For such cases, some recent papers propose combining the feature extraction architecture of a CNN with the encoding and aggregating steps, to learn CNN based BoF representations. For example, in [11, 12, 13, 14], the activations from a single FC layer of the pre-trained CNN are extracted as image or patch descriptors to learn the BoF representations. In our previous work [15], the pre-trained CNN is reinterpreted as a probabilistic discriminative model, and the activations from intermediate FC and the output softmax layer are exploited to derive the CNN based FV coding. These methods can capture the interactions between descriptive and discriminative CNN activations in a translationally invariant manner for effective representation.

However, performance is still unsatisfactory, especially when objects are small and appear in clutter. Furthermore, the effect of end-to-end training of the CNN architecture has not been fully explored. In [16], bilinear CNN models were proposed to replace the FC layer, achieving significant improvement for fine-grained recognition. These bilinear CNN models can be considered as the end-to-end training of the O2P based CNN architectures. However, the final representation has very high dimensionality, which makes it impractical for subsequent image classification. In [17], a compact bilinear pooling is proposed, which keeps the same discriminative power with only a few thousand dimensions.

In this paper, we propose a FV based CNN architecture (FV-CNN) for both fine grained recognition and scene classification tasks, as shown in Fig.1. Like in our previous work [15], the CNN activations from different layers are exploited to cover the frontend feature extraction and backend modeling steps in the traditional representation learning pipeline. The major difference lies in that the proposed FV-CNN architecture can be jointly optimized via end-to-end back-propagation of the gradients. Our proposed architecture follows the paradigm of bilinear CNN models [16] that are related to the two stream hypothesis of visual processing in the human brain [18]. However, a FV component with a probabilistic discriminative model connected by simple outer product and sum operators is presented, which adopts cross-layer pooling like [19], and learns a compact representation. Extensive experiments have been conducted on CUB-200-2011 [20] and MIT67 [21] datasets. Results show that the proposed CNN architecture can achieve better performance on both fine-grained recognition and scene classification tasks, compared with [16, 17]. Accuracies of 88.0% and 82.2% are achieved on the two datasets respectively.

2. REVISITING CNN BASED FV CODING

In the CNN based FV coding methods of [7, 15], the CNN activations $\mathbf{x} \in R^D$ from a single intermediated convolutional layer of a pre-trained CNN model, are extracted as region descriptors [1]. A K-component diagonal GMM model $\boldsymbol{\lambda} = \{\lambda_k = (\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)\}_{k=1}^K$ can be trained off-line from the selected training set via the EM algorithm [23], where $\pi_k \in [0, 1]$, $\boldsymbol{\mu}_k \in R^D$ and $\boldsymbol{\sigma}_k \in R^D$ are the mixture weight, mean vector and diagonal covariance vectors of the k-th component respectively. For an image with T region descriptors $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, the FV $\Psi(\mathbf{X})$ can be represented as

$$\Psi(\mathbf{X}) = \frac{1}{T}\sum_{t=1}^{T}\Psi(\mathbf{x}_t) \tag{1}$$

where $\Psi(\mathbf{x}_t) = (\psi_1(\mathbf{x}_t), \psi_2(\mathbf{x}_t), \ldots, \psi_K(\mathbf{x}_t)) \in R^{K\times D}$, and each component $\psi_k(\mathbf{x}_t)$ is a gradient vector of $p(\mathbf{x}_t|\lambda_k)$ with respect to the mean vector $\boldsymbol{\mu_k}$.

$$\psi_k(\mathbf{x}) = \frac{1}{\sqrt{\pi_k}}\gamma_k(\mathbf{x})\left(\frac{\mathbf{x} - \boldsymbol{\mu}_k}{\boldsymbol{\sigma}_k}\right) \tag{2}$$

[1] A FC layer can also be considered as the convolutional one with kernels that cover their entire region [22].

where $\gamma_k(\mathbf{x})$ is the posterior probability or responsibility of feature $\mathbf{x}$ on Gaussian component λ_k

$$\gamma_k(\mathbf{x}) = \frac{\pi_k N(\mathbf{x}; \lambda_k)}{\sum_{j=1}^{K}\pi_j N(\mathbf{x}; \lambda_j)} \tag{3}$$

It is worth noting that for high-dimensional features, using $\Psi(\mathbf{x}_t)$ is sufficient for good performance [13]. Despite the remarkable improvement over convenient local feature (*e.g.* SIFT) based BoF representations, it is still difficult to model the complex distribution of high-dimensional CNN activations using the GMM obtained via unsupervised learning.

In [15], a pre-trained CNN was re-interpreted as a discriminative probabilistic model, and the output of the softmax layer used as posterior probability for deriving FV coding. In the following section, we extend this to present a FV based CNN architecture for jointly optimizing the feature extraction and backend modeling steps in an end-to-end manner.

3. FV BASED CNN ARCHITECTURE

As in the general representation learning pipeline, the proposed FV based CNN method is divided, as shown in Fig.1, into frontend feature extraction and backend modeling:

Feature Extraction

The feature extraction part consists of the convolutional layers transferred from a pre-trained CNN. The output of the final convolutional layer is a $H \times W \times D$ feature map, which can be viewed as a set of D dimensional local features extracted from $H \times W$ spatial locations.

Backend Modeling

For backend modeling, we design a new FV-based component with a probabilistic discriminative model connected by simple outer product and sum operator in a direct acyclic graph (DAG), as shown in Fig.1. The component parameters are learnable via end-to-end back-propagation. As mentioned, the k-th FV component $\psi_k(\mathbf{x})$ can be rewritten as

$$\begin{aligned}\psi_k(\mathbf{x}) &= a_k(\mathbf{x})(\mathbf{x} - \boldsymbol{\mu}_k) \\ &= a_k(\mathbf{x})\mathbf{x} - a_k(\mathbf{x})\boldsymbol{\mu}_k\end{aligned} \tag{4}$$

where $a_k(\mathbf{x}) = \gamma_k(\mathbf{x})/\sqrt{\pi_k}\sigma_k$. Given the extracted features of an image $\mathbf{X} \in R^{(H\times W\times D)}$, the resulting FV coding $\Psi(\mathbf{X}) \in R^{K\times D}$ can be represented in matrix form

$$\Psi(\mathbf{X}) = \frac{1}{H\times W}\left(\mathbf{A}(\mathbf{X})^\top\mathbf{X} - \mathbf{A}(\mathbf{X})^\top \odot \boldsymbol{\mu}\right) \tag{5}$$

where $\mathbf{A}(\mathbf{X}) = \left(\mathbf{a}_1(\mathbf{X}), \mathbf{a}_2(\mathbf{X}), \ldots, \mathbf{a}_K(\mathbf{X})\right) \in R^{H\times W\times K}$ is the probabilistic model output at each spatial location. The first term in eqn.(5) can be computed using cross-layer pooling via an outer product operation, while the second term is considered as a regularization term, implmented by a convolutional operation $\odot$. Following [16], $\Psi(\mathbf{X})$ can be derived in an end-to-end manner, given an appropriate CNN strucuture for computing $\mathbf{A}(\mathbf{X})$.

Table 1. Classification results of FC-CNN, CNN-IFV, FB-CNN, CB-CNN-RM/CB-CNN-TS and the proposed FV-CNN on Caltech-CUB200 (or CUB) and MIT67 datasets in terms of MAP (%).The number before and after the slash represents accuracies without and with fine-tuning on target datasets respectively.

Datasets	Basic Net	FC-CNN	CNN-IFV [24]	FB-CNN [16]	CB-CNN-RM [17]	CB-CNN-TS [17]	FV-CNN
CUB	VGG-M	50.1/57.9	47.3/NA	70.6/77.6	63.6/76.1	68.5/76.9	66.6/**81.0**
CUB	VGG-16	57.4/66.1	64.2/NA	80.1/84.00	78.1/83.8	79.5/84.0	76.3/**88.0**
MIT67	VGG-M	61.3/64.4	67.2/NA	70.2/67.1	68.2/67.9	69.3/68.7	66.4/**72.9**
MIT67	VGG-16	64.5/66.7	76.6/NA	77.5/71.1	73.9/73.5	76.1/73.7	75.6/**81.4**

Motivated by [25], we design a probabilistic model to compute $\mathbf{A}(\mathbf{X})$, which consists of two stages: (1) a convolutional layer with K filters $\{\mathbf{w}_j\}_{j=1}^K$ that have spatial support 1×1, producing the output $y_j(\mathbf{x}) = \mathbf{w}_j^\top \mathbf{x}$, and (2) the convolution output is then passed through the soft-max function to obtain $a_k(\mathbf{x})$.

Stage 1. To initialize the parameter $\mathbf{W} = \{\mathbf{w}_j\}_{j=1}^K$ of the newly added convolutional layer, we follow the orthogonal match pursuit (OMP-k) algorithm as detailed in [26]. Given the training set $\mathbf{X}$, OMP-k use an alternating minimization of the following objective function

$$\begin{aligned} & \min_{\mathbf{W},\mathbf{s}} \textstyle\sum_i \|\mathbf{W}\mathbf{s}(\mathbf{x}_i)^\top - \mathbf{x}_i\| && (6) \\ s.t. \quad & \|\mathbf{w}_j\|_2^2 = 1, \forall j \\ and \quad & \|\mathbf{s}(\mathbf{x}_i)\|_0 \leq k, \forall i \end{aligned}$$

where $\|\mathbf{s}(\mathbf{x}_i)\|_0$ is the number of non-zero elements in code vector $\mathbf{s}(\mathbf{x}_i)$. When $k = 1$, OMP-1 is a form of "gain-shape vector quantization", similar to the K-means algorithm when $\mathbf{x}$ and $\mathbf{w}$ have unit length. Specifically, it chooses $k = \arg\max_j |\mathbf{w}_j^\top \mathbf{x}|$, and sets $\mathbf{s}_k(\mathbf{x}_i) = \mathbf{w}_k^\top \mathbf{x}_i$ and all other elements of $\mathbf{s}_k(\mathbf{x}_i)$ to zero. Parameters initialized using OMP-1 can be easily implemented using a basic convolutional layer.

Stage 2. A soft-max layer is used to compute the posterior probability. To simplify evaluating the back-propagation of the gradients, we propose computing the soft-max output as

$$a_k(\mathbf{x}) = \frac{\exp(\alpha_k \mathbf{w}_k^\top \mathbf{x})}{\sum_{j=1}^K \exp(\alpha_j \mathbf{w}_j^\top \mathbf{x})} \tag{7}$$

The parameter α_j is a positive constant that is determined using eqn.(4), to control the decay of corresponding posterior probability with respect to the calculated cosine distance.

4. EXPERIMENT AND ANALYSIS

4.1. Methods

The proposed FV-CNN is a DAG architecture implemented using the MatConvNet toolbox [27]. We consider two basic CNNs pre-trained on ImageNet to design our FV-based CNN architecture, *i.e.* VGG-M [28] and VGG-16 [9]. VGG-16 is more effective with deeper convolutional layers for frontend feature extraction. In both cases, we truncate the CNN structure to the last convolutional layer as the frontend feature extractor, *i.e.* layer 13 (conv$_5$ for VGG-M), and layer 29 (conv$_{5_3}$ for VGG-16). We first resize the image to 512×512 pixels for feature extraction. The extracted features are L2-normalized before backend modeling.

As described in Section 3, the probabilistic model is a basic convolutional layer with filters initialized using OMP-1, followed by a soft-max layer as in eqn. (7). In experiments, the number of probabilistic model components is set to $K = 16$. With 512 channels in the final convolutional layer for both VGG-M and VGG-D, the resulting feature dimension for classification is $16 \times 512 = 8192$. We also evaluate the performance of different K, and similar results have been obtained as shown in Table 2.

The resulting FV is further processed by signed-square root ($\mathbf{y} \leftarrow \text{sign}(\mathbf{x})\sqrt{|\mathbf{x}|}$) ans L2-normalization ($\mathbf{z} \leftarrow \mathbf{y}/\|\mathbf{y}\|_2$). This post-processing has been shown to further improve the effectiveness of Fisher vectors in [7].

With the resulting FV, a basic FC and soft-max layer are used for image classification. Instead of random FC initialization, the linear SVMs are trained and the model parameters used to set the filters of the FC layer. We found this may accelerate the resulting CNN training process. Besides our proposed FV-CNN, we also implement the following methods for comparison with almost the same experimental setting.

FB-CNN: the full bilinear pooling method in [16]. The symmetric network structures corresponding to FB-CNN[M,M] and FB-CNN[D,D] configurations are used for evaluation. Both VGG-M and VGG-16 have 512 channels in the final convolutional layer, and the resulting feature dimension is $512 \times 512 \approx 250K$.

CB-CNN: the compact bilinear pooling method in [17]. With same experimental settings, we evaluate the Random Maclaurin (RM) [29] and Tensor Sketch (TS) [30] bilinear pooling CNNs. Following the configurations in [17], the resulting feature dimension is 8192.

FC-CNN: the classical fine-tuning configuration. Given pre-trained CNNs (*i.e.* VGG-M and VGG-16), the last classification layer is replaced with a randomly initialized k-way layer before fine-tuning. This requires a fixed input image size. For VGG-M and VGG-16 networks, the input size is set to

224×224 and the feature dimension of the FC layer is 4096. **CNN-IFV**: Similar to FB-CNN, FV coding has been used as encoding and aggregating steps instead of FC layers. Following [24], the activations of the last convolutional layer (excluding ReLU) are used as input for FV coding. In CNN-IFV, 64 GMM components are used for FV coding, giving a feature dimension of $64 \times 512 \approx 30K$. CNN-IFV can be considered the initialization step of the proposed FV-CNN architecture.

4.2. Datasets and results

Datasets
We evaluate using the Caltech-CUB200 fine grained recognition dataset [20] and MIT67 scene classification [21] dataset. CUB200 contains 11,788 images of 200 bird species which is challenging since inter-category differences are small, and easily overwhelmed by factors such as pose, viewpoint and object location. The MIT dataset comprises 6700 images of 67 indoor scene categories, with 100 images per category. Most scenes are collections of objects organized in a highly variable layout, with some subtle cross-category differences. We use standard training and test splits for both datasets.
Experimental results
The experimental results on both Caltech-CUB200 and MIT67 datasets are shown in Table 1. The results of Caltech-CUB200 indicate that FB-CNN and CB-CNN outperform FC-CNN and CNN-IFV by a large margin. CB-CNN can achieve the similar performance as FB-CNN with compact representation. The dimension of CB-CNN-RM and CB-CNN-TS is 8192, and that of FB-CNN is about 250K. Without fine-tuning, the performance of FV-CNN is slightly worse than FB-CNN and CB-CNN, and better than CNN-IFV. When VGG-M is used as the basic network, classification accuracy of 81.0% can be achieved, which outperforms FB-CNN by 2.4%. This demonstrates the effectiveness of jointly optimizing the whole representation learning pipeline. When VGG-16 is used as the basic network, classification accuracy of 88.0% with fine-tuning can be achieved. This outperforms FB-CNN trained without and with bounding box (84.0% and 85.1% respectively) while the feature dimension of $16 \times 512 = 8192$ is the same as in CB-CNN.

The MIT67 dataset differs from Caltech-CUB200 by having quite large intra-category variation. From Table 1, we can see that CNN-IFV outperforms FC-CNN by 6.0% and 12.1% for VGG-M and VGG-16 basic networks respectively. FB-CNN without fine-tuning on MIT67 still outperforms CNN-IFV but it is surprising to see that after fine-tuning, the performance of FB-CNN and CN-CNN is degraded by about 2%-3%. This is probably due to the small training set size, and large number of convolutional parameters in FB-CNN and CB-CNN [17]. Unlike FB-CNN and CB-CNN, fine-tuning of FV-CNN can still improve performance; from 66.4% to 72.9% for VGG-M, and from 75.6% to 81.4% for VGG-16 respectively. This is comparable to the accuracy of 82.1% in [15], which exploits the activations from FC7 and output soft-max layers to derive FV coding. However, the representation is more compact. This may be attributed to the fact that FV-CNN benefits from a reduced number of convolutional parameters, and the regularization term introduced in eqn. (4).

Table 2. Classification results on Caltech-CUB200 and MIT67 datasets with different K components in probabilistic discriminative model in terms of MAP (%). The basic network is VGG-16.The number before and after the slash represents accuracies without and with fine-tuning on target datasets respectively.

datasets	16	32	64	128
CUB	76.3/88.0	78.0/88.1	78.5/88.1	78.5/**88.6**
MIT67	75.6/81.4	76.4/81.8	77.6/81.6	77.2/**82.2**

FV-CNN architecture with different configurations
We further conduct experiments on both tasks using FV-CNN with different number K of components in the probabilistic model. From the results in Table 2 we can see that the classification accuracies on both datasets are steady. However, when $K = 128$, the feature dimension is about 125K and the accuracy of 88.6% can be achieved on Caltech-CUB200. To the best of our knowledge, this is the highest reported performance for this dataset.

5. CONCLUSION

In this paper, we presented an FV-based CNN architecture, termed FV-CNN, for fine grained object recognition and scene classification. FV-CNN exploits the intermediate convolutional layers for frontend feature extraction, and a probabilistic discriminative model is initialized and fine-tuned in CNN training. This jointly optimizes parameters for the whole representation learning pipeline. Evaluation on two benchmarks, *i.e* fine grain object recognition (Caltech-CUB200), and scene classification (MIT67) have shown the effectiveness of FV-CNN. Accuracies of 88.0% and 82.2% can be achieved for these datasets respectively.

6. REFERENCES

[1] David G. Lowe, “Distinctive image features from scale-invariant keypoints,” vol. 60, no. 2, pp. 91–110, 2004.

[2] Navneet Dalal and Bill Triggs, “Histograms of oriented gradients for human detection,” in *Proc. of CVPR*, 2005, pp. 886–893.

[3] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang, “Linear spatial pyramid matching using sparse coding for image classification,” in *Proc. of CVPR*, June 2009, pp. 1794–1801.

[4] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, T. Huang, and Yihong Gong, "Locality-constrained linear coding for image classification," in *Proc. of CVPR*, June 2010, pp. 3360–3367.

[5] João Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu, "Semantic segmentation with second-order pooling," 2012, pp. 430–443.

[6] Herve Jegou, Matthijs Douze, Cordelia Schmid, and Patrick P1717rez, "Aggregating local descriptors into a compact image representation.," in *Proc. of CVPR*, 2010, pp. 3304–3311.

[7] Jorge Sanchez, Florent Perronnin, Thomas Mensink, and Jakob J. Verbeek, "Image classification with the Fisher vector: Theory and practice.," *International Journal of Computer Vision*, vol. 105, no. 3, pp. 222–245, 2013.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. of NIPs*, 2012, pp. 1106–1114.

[9] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *Proc. of ICLR*, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," pp. 770–778, 2015.

[11] Yunchao Gong, Liwei Wang, and Ruiqi Guo, "Multi-scale orderless pooling of deep convolutional activation features," in *Proc. of ECCV*, 2014, pp. 392–407.

[12] Lingqiao Liu, Chunhua Shen, Lei Wang, Anton van den Hengel, and Chao Wang, "Encoding high dimensional local features by sparse coding based Fisher vectors," in *Proc. of NIPs*, 2014, pp. 1143–1151.

[13] Mandar Dixit, Si Chen, Dashan Gao, Nikhil Rasiwasia, and Nuno Vasconcelos, "Scene classification with semantic Fisher vector," in *Proc. of CVPR*, 2015.

[14] Yan Song, Ian Mcloughlin, and Lirong Dai, "Deep bottleneck feature for image classification," in *Proc. of ICMR*, 2015, pp. 491–494.

[15] Yan Song, Xinhai Hong, Ian Vince McLoughlin, and Li-Rong Dai, "Image classification with cnn-based fisher vector coding," in *2016 Visual Communications and Image Processing, VCIP 2016, Chengdu, China, November 27-30, 2016*, 2016, pp. 1–4.

[16] Tsung-Yu Lin, Aruni Roy Chowdhury, and Subhransu Maji, "Bilinear CNN models for fine-grained visual recognition," in *Proc. of ICCV*, 2015, pp. 1449–1457.

[17] Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell, "Compact bilinear pooling," in *Proc. of CVPR*, 2016, pp. 317–326.

[18] M. A. Goodale and A. D. Milner, "Separate visual pathways for perception and action.," *Trends in Neurosciences*, vol. 15, no. 1, pp. 20, 1992.

[19] Lingqiao Liu, Chunhua Shen, and Anton van den Hengel, "The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification," in *Proc. of CVPR*, 2015, pp. 4749–4757.

[20] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," Tech. Rep., 2011.

[21] Ariadna Quattoni and Antonio Torralba, "Recognizing indoor scenes," in *Proc. of CVPR*, 2009, pp. 413–420.

[22] Jonathan Long, Evan Shelhamer, and Trevor Darrel, "Fully convolutional network for semantic segmentation," *Proc. of CVPR*, 2015.

[23] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[24] Mircea Cimpoi, Subhransu Maji, and Andrea Vedaldi, "Deep filter banks for texture recognition and segmentation," in *proc. of CVPR*, 2015, pp. 3828–3836.

[25] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *proc. CVPR*, 2016, pp. 5297–5307.

[26] Adam Coates and Andrew Y. Ng, "The importance of encoding versus training with sparse coding and vector quantization," in *Proc. of ICML*, 2011, pp. 921–928.

[27] A. Vedaldi and K. Lenc, "Matconvnet – convolutional neural networks for MATLAB," vol. abs/1412.4564, 2014.

[28] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," 2014.

[29] Purushottam Kar and Harish Karnick, "Random feature maps for dot product kernels," in *AISTATS*, 2012, pp. 583–591.

[30] Ninh Pham and Rasmus Pagh, "Fast and scalable polynomial kernels via explicit feature maps," in *proc. of SIGKDD*, 2013, pp. 239–247.