Noname manuscript No.
(will be inserted by the editor)

Bregman Pooling: Feature-Space Local Pooling for Image Classification

Alameen Najjar · Takahiro Ogawa · Miki Haseyama

Received: date / Accepted: date

Abstract In this paper, we propose a novel feature-space local pooling method for the commonly adopted architecture of image classification. While existing methods partition the feature space based on visual appearance to obtain pooling bins, learning more accurate space partitioning that takes semantics into account boosts performance even for a smaller number of bins. To this end, we propose partitioning the feature space over clusters of visual prototypes common to semantically similar images (i.e., images belonging to the same category). The clusters are obtained by Bregman co-clustering applied offline on a subset of training data. Therefore, being aware of the semantic context of the input image, our features have higher discriminative power than do those pooled from appearance-based partitioning. Testing on four datasets (*Caltech-101*, *Caltech-256*, *15 Scenes*, and *17 Flowers*) belonging to three different classification tasks showed that the proposed method outperforms methods in previous works on local pooling in the feature space for less feature dimensionality. Moreover, when implemented within a spatial pyramid, our method achieves comparable results on three of the datasets used.

Keywords Image classification · Image representation · Feature pooling · Co-clustering · Bregman divergence

A. Najjar (✉) · T. Ogawa · M. Haseyama
Laboratory of Media Dynamics,
Graduate School of Information Science and Technology,
Hokkaido University, Sapporo, Hokkaido, 060-0814, Japan
e-mail: najjar@lmd.ist.hokudai.ac.jp

T. Ogawa
e-mail: ogawa@lmd.ist.hokudai.ac.jp

M. Haseyama
e-mail: miki@ist.hokudai.ac.jp

1 Introduction

At the heart of modern image recognition lies a local patch-based multi-layer architecture that has significantly evolved during the past decade. This architecture can be summarized as follows. First, handcrafted descriptors (e.g., SIFT [38], HOG [15], SURF [3], etc.) densely sampled from an input image are projected into a codebook space using a common coding method, such as vector quantization (*coding step*). Second, a fixed-length, global image representation is generated via summarizing the encoded descriptors, obtained in the previous step, over the image's area (*pooling step*). In the classification task, this pooled representation is finally fed to a linear (or nonlinear) classifier where both training and class label prediction take place. Extensions to this architecture (e.g., [35, 51, 53]) have dominated standard classification benchmarks (e.g., Pascal VOC [17]) for several years. As mentioned above, this architecture has been refined greatly with improvements aimed at both of its steps. In this paper, we propose a novel extension to this architecture that improves its pooling step.

The idea of pooling originates in the Nobel-winning work of Hubel and Wiesel on the mammalian visual cortex [28] in which they explain a cascaded model of the visual cortex where responses coming from lower simple cells are aggregated before being fed to higher complex cells, rendering them invariant to small spatial transformations. This seminal work has long inspired computer vision researchers to adopt the idea of pooling for the aim of building robust translation-invariant visual recognition systems. Thus, pooling has been a genuine component in visual recognition all the way from the early Neocognitron [22], to the Bag-of-Words (BoW) model [14, 47], up until the recently rediscovered convolutional neural networks [34]. In its most basic adaptation, pooling summarizes the image's features by taking the average (or max) value of their activations [4].

Pooling involves two components: (1) an operator and (2) a neighborhood. While the operator does the summarization function, the neighborhood determines which descriptors are to be pooled together. In conventional pooling (e.g., [14, 47]), the pooling operator is applied to all encoded descriptors of the input image at once, i.e., the pooling neighborhood is defined as the whole area of the image. While the direct advantage of this pooling is added robustness to input translations, its major disadvantage is inevitable information loss. To compensate for part of this loss, an extension to pooling (local pooling) enforces locality via jointly pooling only descriptors that are members of a certain local neighborhood. A local neighborhood could be any subgroup of the image's descriptors that are "close" according to a certain criterion. Based on the space within which local neighborhoods are defined, work on local pooling can be categorized into: (1) image-space and (2) feature-space methods. A local neighborhood in the image space could be a subregion (object) within the image plane. On the other hand, a local neighborhood in the feature space could be a partition (bin) whose members share some aspect in common (e.g., visual similarity). As it might be more straightforward to pool descriptors based on their spatial location within the image, the bulk of the work on local pooling has focused on the image space [30, 35, 45]. However, our method operates in the feature space as we believe in the highly untapped potential this space holds.

Within the adopted pipeline (reviewed in Sect. 2), the most notable work on local pooling in the feature space seems to be [8], in which, in the same spirit as that of [29, 55], the image representation is generated via (1) clustering the extracted descriptors over a handful of codewords of a universal codebook learned via k-means clustering [39] and (2) applying the pooling operator within each obtained cluster individually. The final image representation is the (normalized) concatenation of the pooled features. Partitioning of the input data by minimal Euclidean distance (i.e., clustering) assures that only visually similar descriptors are pooled together. In other words, the notion of closeness in the feature space is defined in terms of the visual appearance of descriptors. This method is simple and can be regarded as a straightforward extension to the popular spatial pyramid (SP) model [35] within the feature space.

In this work, we mainly try to determine whether partitioning the feature space using a k-means codebook, i.e., based on visual appearance only as in [8, 29, 55], is optimal for local pooling in the image classification task. While k-means clustering preserves, to some extent, the visual similarity between descriptors, it totally discards any class-related information (i.e., high-level semantics) of the input image. For example, two visually similar descriptors belonging to two semantically different objects (subregions) within the image will be assigned to the same pooling bin. In this case, jointly pooling the two descriptors totally discards the image's semantics.

Motivated by the above observation, we aim at generating pooling bins that are aware of the semantics of the input image. To this end, we propose partitioning the feature space over clusters of visual prototypes common to semantically similar images. The clusters in turn are generated via simultaneously clustering (co-clustering) images and their visual prototypes (codewords). The co-clustering is applied offline on a subset of training data and conducted using Bregman co-clustering [2]. Therefore, contrary to features pooled from appearance-based partitioning [8, 29, 55], our features are aware of the semantic context of the input image within the dataset, which consequently boosts classification performance. Similar to [8], spatial information can be easily encapsulated via implementing our local pooling within an SP or any other similar method.

The remainder of this paper is organized as follows. The classification pipeline within which our method is implemented is reviewed in Sect. 2. A survey of previous works on local pooling in the feature space is given in Sect. 3. Details of the proposed method are given in Sect. 4. In Sect. 5, the proposed method is empirically validated via extensively testing it on four datasets belonging to three different classification tasks. Finally, conclusions are given in Sect. 6.

2 Classification pipeline

In this section, the image classification pipeline within which our method is implemented is reviewed and notations used throughout the paper are defined. We are interested in the coding-pooling pipeline of image classification [6]. This pipeline is summarized in four successive steps: (1) feature extraction, (2) coding, (3) pooling, and finally (4) classification. Individual steps are explained below.

2.1 Feature extraction

Given an input image $I \in \mathbf{I}$ (the image dataset), a set of low-level features (e.g., SIFT) sampled at N different locations is extracted, such that $\mathbf{X} = \{x_i\}_{i=1}^{N}$, where $x_i \in \mathbb{R}^d$ is the d-dimensional low-level feature extracted at location i. Several methods have been proposed in the literature to obtain salient regions within the image from which features are extracted (See [40] for a detailed comparison). However, in the classification task, it has been shown in [35] that better performance is obtained when features are densely sampled from a regular grid covering the image plane.

2.2 Coding

The first step is to train a codebook $\mathbf{B} = [b_1, \cdots, b_K] \in \mathbb{R}^{d \times K}$, where $\{b_i\}_{i=1}^{K}$ is the set of the d-dimensional codewords obtained via unsupervised learning, such as k-means clustering. Note that individual codewords belong to the same space to which the extracted features, of the previous stage, belong. Then, given a coding function ψ, the extracted features ($\mathbf{X}$) of the input image are individually projected into the space of the learned codebook. More formally, each descriptor $x_i \in \mathbb{R}^d$ is mapped to a new representation $v_i \in \mathbb{R}^K$, using a coding function $\psi : \mathbb{R}^d \to \mathbb{R}^K$, such that:

$$v_i = \psi(x_i), \quad \forall\, i \in \{1, \cdots, N\}. \tag{1}$$

The coding function can be thought of as an activation function for the codebook, activating each of the codewords according to the input descriptor [1]. Depending on the coding function used, activations are either continuous or binary-valued. A multitude of coding functions (algorithms) have been proposed in the literature. In the following, we explain three of the most popular ones: Vector Quantization (VQ), Sparse Coding (SC) [53], and Locality-constrained Linear Coding (LLC) [51]. See [10] for a comprehensive survey on coding functions.

Vector Quantization (VQ) encodes each descriptor by assigning the value 1 to its closest codeword and zeros to the rest. This is done via solving the following constrained least squares fitting problem:

$$\begin{aligned} \arg\min_{\mathbf{V}} \quad & \sum_{i=1}^{N} \|x_i - \mathbf{B}v_i\|^2 \\ \text{subject to} \quad & \|v_i\|_{\ell_0} = 1, \quad \text{and} \quad \|v_i\|_{\ell_1} = 1, \quad v_i \geq 0 \end{aligned} \tag{2}$$

where $\mathbf{V} = [v_1, v_2, \cdots, v_N] \in \mathbb{R}^{K \times N}$ is the matrix of codes obtained for the set $\mathbf{X}$. With a single non-zero element (i.e., $\|v_i\|_{\ell_0} = 1$), these codes are highly sparse. This leads to a high quantization loss, especially when the descriptor being encoded is close to several codewords at the same time.

To alleviate the quantization loss of VQ, Sparse Coding (SC) approximates each descriptor as a sparse linear combination of the codewords. In other words, SC relaxes the cardinality constraint ($\|v_i\|_{\ell_0} = 1$) in Eq. (2). This is achieved via solving the following optimization:

$$\arg\min_{\mathbf{V}} \sum_{i=1}^{N} \|x_i - \mathbf{B}v_i\|^2 + \lambda \|v_i\|_{\ell_1}, \tag{3}$$

where λ is a parameter that controls the sparsity of the obtained code induced by the ℓ_1 norm.

Finally, approximate Locality-constrained Linear Coding (LLC) addresses the non-locality that can occur in SC via encoding each descriptor with its n-nearest codewords. In other words, a new codebook $\mathbf{B}(x_i, n)$ is constructed for each descriptor x_i, such that $\mathbf{B}(x_i, n) = NN_n(x_i, \mathbf{B}) \in \mathbb{R}^{d \times n}$, where n ($n \ll K$) is a constant that defines how localized the coding is. Approximate LLC is formulated as:

$$\begin{aligned} \arg\min_{\mathbf{V}^*} \quad & \sum_{i=1}^{N} \|x_i - \mathbf{B}(x_i, n)v_i^*\|^2 \\ \text{subject to} \quad & 1^T v_i^* = 1, \end{aligned} \tag{4}$$

where $v_i^* \in \mathbb{R}^n$ is the obtained n-dimensional code, later projected into the original space ($\mathbb{R}^K$) of the learned codebook.

2.3 Pooling

At this stage, the matrix $\mathbf{V} \in \mathbb{R}^{K \times N}$ of encoded descriptors is transformed into a fixed-length global image representation $z \in \mathbb{R}^K$. This is achieved via applying the pooling operator $\phi : \mathbb{R}^{1 \times N} \to \mathbb{R}$ to each row of $\mathbf{V}$ separately. The final image representation is the concatenation of the pooled K descriptors, such that:

$$z = [z_1, z_2, \cdots, z_K]^T, \tag{5}$$

where $z_k \in \mathbb{R}$ is given:

$$z_k = \phi\left(\{v_{ki}\}_{i=1}^{N}\right), \quad \forall\, k \in \{1, \cdots, K\}, \tag{6}$$

where v_{ki} is the activation value of the i-th descriptor to the k-th codeword. The most commonly used operators are average (or sum) [35] and max [53] operators in which the output of the pooling step is simply the average and the maximum value of activations, respectively. In other words, ϕ in Eq. (6) is replaced with the suitable operator (avg or max). Recently, several more advanced pooling operators have been proposed in the literature, such as those explained in [33]. The reader is referred to the same paper for a recently published survey on the topic.

2.4 Classification

Both training and class label prediction take place at this stage. The pooled image feature $z \in \mathbb{R}^K$ is (normalized and then) fed to a classifier. A standard classifier choice is Support Vector Machines (SVM) [48].

3 Previous works

As mentioned before, the coding-pooling pipeline has been improved along many directions. However, since it is beyond the scope of this paper to review works in all directions, and since our work improves only pooling within the pipeline, we limit this section to survey methods similar to ours.

As previously explained, based on the space within which local neighborhoods are defined, work on local pooling can be categorized into: (1) image-space and (2) feature-space methods. In the following, we survey previous works on local pooling in the feature-space.

Although the idea of partitioning the feature space to compute correspondences is not a new one [23], generating an image representation by restricting the pooling operator to similar descriptors can be traced back to [29, 55] in Vector of Linearly Aggregated Descriptors (VLAD) and Super Vector (SV) image representations. In these methods, the final image representation is a concatenation of multiple feature vectors pooled from multiple feature bins obtained via clustering the input image's descriptors over the codewords of a universal k-means codebook. This same idea was later explicitly proposed as feature-space local pooling within the BoW model by Boureau et al [8]. The essence of [8, 29, 55] is to partition the feature space using a k-means codebook so that only visually similar descriptors are jointly pooled.

Therefore, within the adopted pipeline, Boureau et al's work [8] seems to be the most prominent on local pooling in the feature space. We believe that this is in part due to its simplicity and the way it is presented as a straightforward extension to the popular SP model within the feature space. However, one of its major drawbacks is the fact that the notion of closeness in the feature space is defined solely in terms of the visual similarity of descriptors, i.e., the pooled feature is indifferent to the high-level semantics of the input image, which, we believe, limits the discriminative power of the pooled feature. Several attempts have been made to generate pooling bins that are aware of the image's semantics. In the following, we introduce those which we are aware of.

Rematas et al [44] proposed pooling features from bins obtained via both k-means and hierarchical clustering. Therefore, the obtained space partitioning is aware of the different categories of the dataset at hand. Using a multiple-kernel framework, Rematas et al [44] report impressive results on several classification benchmarks. This method seems to be the closest to ours. However, it is intended for naive Bayes nearest neighbor classification [5], i.e., it does not fit within the adopted pipeline of this paper.

In a recent work, Fanello et al [19] proposed partitioning the feature space by assigning the image's descriptors with weights learned (in a supervised way) via prediction of the class label of the subregion within which individual descriptors lie. Based on these weights, the pooling operator is applied accordingly. However, since subregions are extracted using a single-layer spatial pyramid (i.e., subregions are predefined square patches), we argue that the learned weights fall short of capturing the image semantics.

To the best of our knowledge, the above discussed methods are the only attempts made to generate pooling bins that are aware of the image's semantics. In the following, we explain how the proposed method addresses this problem. We propose to jointly pool descriptors that are both visually similar and semantically coherent at the same time. To this end, we propose partitioning the feature space over clusters of visual prototypes common to semantically-similar images. The clusters are learned by applying Bregman co-clustering on a subset of training data and serve as semantics-preserving codewords over which the feature space is partitioned. Therefore, contrary to features pooled from an appearance-based partitioning [8, 29, 55], our features are aware of the semantic context of the input image within the dataset, which consequently boosts classification performance. Our method is presented in detail in Section 4.

It should be noted that co-clustering visual words and images has been previously proposed in [25, 37]. However, to the best of our knowledge, we are the first to propose generating pooling bins using co-clustering. In [25, 37], co-clustering is used for the sole purpose of generating a codebook. The image representation are then generated following the traditional BoW procedure. In other words, features are globally pooled not locally as in our method.

4 Our method

In this section, we explain the proposed method. We start by detailing how the feature space is partitioned in Sect. 4.1. Then we explain how the final image representation is generated in Sect. 4.2. Finally, we compare our method to related works in Sect. 4.3.

4.1 Feature-space partitioning

To obtain pooling bins, we need to partition the feature space. This section details this procedure.

4.1.1 Introduction

Given an image's extracted low-level features $\mathbf{X}$, our goal is to find P different neighborhoods $\{x_i\}_{i=1}^{N_p}, \forall\ p \in \{1, \cdots, P\}$, within $\mathbf{X}$, so that members of each neighborhood are semantically coherent. In this work, semantics are defined as the high-level visual traits common to images conveying the same concept, i.e., belonging to the same category, and by

"high-level" we mean characteristics that go beyond the exact appearance of individual images and ascribe to their semantic context within the dataset. Therefore, favoring simplicity, we propose to model semantics as *clusters of visual prototypes* (codewords) common to images belonging to the same category.

To this end, we make use of an established data mining tool called co-clustering [26]. A co-clustering algorithm simultaneously clusters rows and columns of an input data matrix and produces two correlated sets of clusters representing the two dimensions of the input (rows and columns) as an output. Thus, as shown in [25, 37], semantics of a given dataset can be captured, in the form of clusters of visual prototypes, by co-clustering a subset of the dataset's training images represented as a matrix of BoWs. Following the definition (of semantics) above, semantics are captured via finding the set of row (visual words) clusters that frequently appear in certain clusters of the columns (images). This is achieved via finding the set of row and column clusters that minimizes the loss in mutual information between the visual words and the training images. See sect. 4.1.3 for details.

To conduct the co-clustering, we use Banerjee et al [2] in which optimal co-clustering is guided by a search for the nearest matrix approximation that has the minimum Bregman information. Before explaining the co-clustering procedure (Sect. 4.1.3), we introduce two preliminary concepts, Bregman divergence and Bregman information, in Sect. 4.1.2.

4.1.2 Bregman divergences and Bregman information

First introduced in [9], Bregman divergences define a large class of widely used loss functions, such as the squared Euclidean distance, KL divergence, etc. Given a convex function f, the Bregman divergence between two data points $a_1, a_2 \in \mathbb{R}$ is defined as:

$$d_f(a_1,a_2) = f(a_1) - f(a_2) - < \bigtriangledown f(a_2),\ a_1 - a_2 >, \tag{7}$$

where $< a_1, a_2 >$ is the inner product between a_1 and a_2, and $\bigtriangledown$ is the gradient operator. The convexity of f guarantees that $d_f(a_1,a_2)$ is non-negative for all $a_1, a_2 \in \mathbb{R}$. By choosing a suitable convex function (f), the Bregman divergence can generalize several existing distance measures. For instance, using the convex function $f(a) = a \log a$ defined over $a \in \mathbb{R}$, the KL divergence between two points $a_1, a_2 \in \mathbb{R}$ (i.e., $D_{KL}(a_1 \parallel a_2)$) can be expressed as a Bregman divergence as:

$$d_f(a_1,a_2) = a_1 \log(a_1/a_2) - (a_1 - a_2). \tag{8}$$

Based on Bregman divergences, we explain another concept called Bregman information [2]. Given a Bregman divergence (d_f) and a random variable (A), the uncertainty of A can be captured in terms of a useful concept called Bregman information (I_f), defined as the expected (E) Bregman divergence to the expectation, such that:

$$I_f(A) = E\big[d_f(A,\ E(A))\big]. \tag{9}$$

In the following, we explain Bregman co-clustering in which optimal co-clustering is guided by a search for the nearest (in Bregman divergence) approximation matrix that has the minimum Bergman information.

4.1.3 Co-clustering images and visual prototypes

Consider a subset of j training images $C = \{c_v\}_{v=1}^{j}$, spanning L different categories, represented as BoWs generated by using a codebook of m visual prototypes $R = \{r_u\}_{u=1}^{m}$. These images can be regarded as a data matrix $A \in \mathbb{R}^{m \times j}$ of two underlying discrete random variables R and C representing rows (visual prototypes) and columns (images), respectively. The aim here is to simultaneously cluster columns (C) into L categories $\hat{C} = \{\hat{c}_h\}_{h=1}^{L}$ and rows (R) into P clusters $\hat{R} = \{\hat{r}_g\}_{g=1}^{P}$. The obtained co-clustering can be thought of as a pair of mapping functions $\hat{R} = \rho(R)$ and $\hat{C} = \gamma(C)$ operating on the rows and columns, respectively.

According to Bregman co-clustering [2], the optimal solution is the pair (ρ, γ) that constructs the nearest approximation matrix that has the minimum Bregman information, i.e., satisfying:

$$\arg\min_{(\rho,\gamma)} E\big[d_f(A,\ \hat{A})\big], \tag{10}$$

where $\hat{A}$ is the approximation matrix with the minimum Bregman information among the set of approximations that satisfy Eq. (10).

Bregman co-clustering generalizes several well-known co-clustering algorithms, such as [13, 16]. Moreover, the choice of the Bregman divergence is decided by the nature of the input data. Banerjee et al [2] have done an elaborate empirical study on the choice of Bregman divergence and concluded that the KL divergence better runs the co-clustering when the input data is a normalized occurrence matrix of words and documents (visual words and images in our case). However, to accurately investigate which Bregman divergence better suits our data, we (preliminarily) experimented[1] with the KL divergence and the Squared Euclidean Distance (SED). We found that the KL divergence performs slightly better than the SED on our data. This also agrees with [2]. Therefore, we adopt the KL divergence as an objective function in the co-clustering. Thus, as explained

[1] Calculating the Normalized Mutual Information (NMI) between image labels and the obtained clusters. Higher NMI means better co-clustering.

in Sect. 4.1.2, by using a suitable convex function, KL divergence can be expressed as a Bregman divergence as in Eq. (8). This in turn means that Bregman co-clustering reduces to the information-theoretic co-clustering of [16] in which the optimal co-clustering is the one that minimizes the following:

$$\begin{aligned} \Delta MI &= MI(R;C) - MI(\hat{R};\hat{C}) \\ &= D_{KL}\Big(p(R,C) \parallel q(R,C)\Big), \end{aligned} \tag{11}$$

where $MI(R;C)$ is the mutual information between two discrete random variables R and C and is given as:

$$MI(R;C) = \sum_{r\in R, c\in C} p(r,c)\log\left(\frac{p(r,c)}{p(r)p(c)}\right), \tag{12}$$

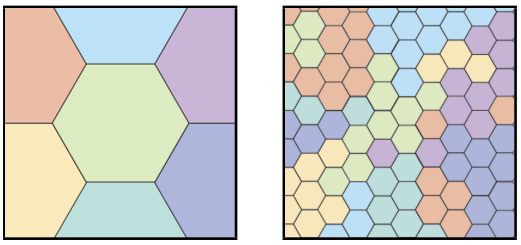
and $q(R,C)$ is a distribution of the form:

$$q(R,C) = p(\hat{R},\hat{C})\; p(R|\hat{C})\; p(C|\hat{C}). \tag{13}$$

Therefore, optimal co-clustering can be obtained by searching for the nearest approximation matrix that has a distribution of the form shown in Eq. (12). To this end, Dhillon et al [16] proposed a neat algorithm that is computationally efficient even for sparse data (our case). As an input, the algorithm takes the joint probability distribution function $p(R,C)$, the number of categories (L), and the number of row clusters (P). Note that since co-clustering is an unsupervised learning, individual images' labels are not considered as an input. As an output, the algorithm produces the pair (ρ,γ). The algorithm starts (at $t=0$) with a random pair (ρ^t,γ^t) which is updated at each iteration (t) via: (1) clustering the rows (R) while keeping the columns (C) fixed and (2) clustering the columns while keeping the rows fixed. The algorithm stops when Eq. (11) is less than a preset threshold.

4.2 Image representation

In this section, we explain how the final image representation is generated. Given an input image $I \in \mathbf{I}$, its set of extracted low-level features ($\mathbf{X}$) are first clustered over the (row) clusters ($\hat{R} = \{\hat{r}_g\}_{g=1}^{P}$) learned via co-clustering training images and their visual prototypes (Sect. 4.1.3) into P different neighborhoods. Then, by using a k-means codebook, each neighborhood is individually pooled into a K-dimensional feature vector (The feature vector is a traditional BoW and the use of a k-means codebook is an essential step in the BoW procedure [14, 47]) ($z_p \in \mathbb{R}^K$), such that:



(a) (b)

Fig. 1 Cartoon representation of (a) an appearance-based partitioning compared to (b) ours. Different colors represent different pooling bins. Number of pooling bins is the same in both spaces. Contrary to (a), our bins (b) are disjoint in the feature space. Our partitioning can be seen as obtained via (1) clustering the input over a large k-means codebook and then (2) aggregating semantically coherent bins according to the result of the co-clustering.

$$\begin{aligned} &z_p = [z_{p1}, z_{p2}, \cdots, z_{pK}]^T, \\ &\text{where} \quad z_{pk} = \phi\Big(\{v_{ki}\}_{i=1}^{N_p}\Big). \end{aligned} \tag{14}$$

The final image representation (z) is then the concatenation of the P individually pooled features (z_p):

$$z = [z_1^T, z_2^T, \cdots, z_P^T]^T \in \mathbb{R}^{PK}. \tag{15}$$

This representation along with the image's label are what passed to the classifier later.

Similar to [8], spatial information can be easily encapsulated in the image representation by repeatedly pooling features locally within the individual S spatial cells of a spatial pyramid (See Sect. 5.1.2 for implementation details). In this case, the final image representation is (z) is then the concatenation of the S individually pooled features (across the S spatial cells) (z_s):

$$z = [z_1^T, z_2^T, \cdots, z_S^T]^T \in \mathbb{R}^{PKS}, \tag{16}$$

where z_s is calculated as shown in Eq. (15).

4.3 Semantically enhanced pooling bins

In this section, we discuss the nature of the feature-space partitioning (pooling bins) obtained in our method and how it compares to the appearance-based partitioning of [8, 29, 55]. As previously explained, the feature space in our method is partitioned by clustering the input image's extracted descriptors ($\mathbf{X}$) over clusters of visual prototypes ($\hat{R}$) learned through Bregman co-clustering. However, given the fact that the co-clustering operates on the training BoWs generated using an m-dimensional k-means codebook ($R = \{r_u\}_{u=1}^{m}$), we can say that our partitioning can be regarded as obtained

in two successive steps: (1) clustering over m ($m \gg P$) k-means codewords followed by (2) aggregating the m clusters of the previous step into P bins using a map ($\hat{R} = \rho(R)$) learned via Bregman co-clustering. Given that the learned map captures the semantic context of the dataset at hand [37], our pooling bins can be regarded as being semantically enhanced compared to those learned in [8, 29, 55], in which the image's descriptors are directly clustered over P codewords of a k-means codebook.

Fig. 1 provides a cartoon representation of an appearance-based partitioning compared to a semantically enhanced one (ours). For simplicity, the feature space is illustrated as a square and individual partitions are illustrated as hexagons. Different colors represent different pooling bins. Notice that the similarity between the two spaces is that both have the same number of pooling bins (number of unique colors), i.e., the pooled image representation has exactly the same dimension in both spaces. However, they differ in that pooling bins in our partitioning are disjoint in the feature space which is not the case in the appearance-based one.

5 Experiments

In this section we empirically validate the proposed method by showing the results of experiments conducted on four popular image datasets belonging to three different classification tasks, namely generic-object classification, scene classification, and fine-grained classification. This section is organized as follows. The protocol we followed in our experiments is detailed in Sect. 5.1. The obtained results are discussed in Sect. 5.2 and Sect. 5.3.

5.1 Experimental protocol

Our experimental protocol is explained in this subsection. The image datasets used are briefly reviewed in Sect. 5.1.1. Then implementation details are given in Sect. 5.1.2.

5.1.1 Image datasets

In our experiments, we used *Caltech-101*, *Caltech-256*, *15 Scenes*, and *17 Flowers* image datasets. Individual datasets are briefly introduced in the following:

Caltech-101 [21]: This is a widely used dataset suitable for the generic-object classification task. It consists of 9144 images exhibiting a variety of objects spanning 102 different categories (e.g., *person*, *cougar*, etc.). The number of images per category ranges from 31 to 800. Images come in an approximate resolution of 200×300 pixels each.

Caltech-256 [24]: This is a challenging generic-object classification dataset that consists of 30607 images organized in 257 categories of the same nature as those of *Caltech-101*. The number of images per category is 80 to 827. Images come in an approximate resolution of 200×300 pixels each.

15 Scenes [20, 35, 42]: This is a common choice for the task of scene classification, and the dataset consists of 4485 images organized in 15 different categories of indoor (e.g., *kitchen*, *bedroom*, etc.) and outdoor (e.g., *forest*, *highway*, etc.) scenes. Each category has 200 to 400 images on average. Images come in an average size of 250×300 pixels each.

17 Flowers [41]: This is a dataset of 1360 high-resolution flower images organized in 17 different categories. Each category has 80 images. Images have large scale, pose and light variations. *17 Flowers* is a challenging fine-grained classification dataset.

5.1.2 Implementation details

Favoring the reproducibility of our results, the implementation details of our experiments are explained in this section.

Pre-processing: Images were first converted to grayscale and then reduced in resolution so that the longest side was less than or equal to 300 pixels.

Feature extraction & description: Using VLFeat toolbox [49], low-level features were densely sampled over a rectangular grid of 16×16 pixel patches with a sampling rate (distance between the centers of two successive patches) of 4 pixels. Unless otherwise noted, a 128-dim SIFT descriptor was then computed for each extracted patch.

Codebooks: Standard k-means clustering was used to generate codebooks. The number of codewords was always set to 4096.

Coding, pooling (operator), and normalization: Unless otherwise noted, the combination of sparse coding and max pooling was used in our experiments. The final image representation is always ℓ_2-normalized.

Co-clustering: We applied Bregman co-clustering on a subset of the training data for $P = \{8, 16, 32, 64\}$. It should be noted that co-clustering is applied offline, i.e., at the time of generating the image representation and not during training or testing.

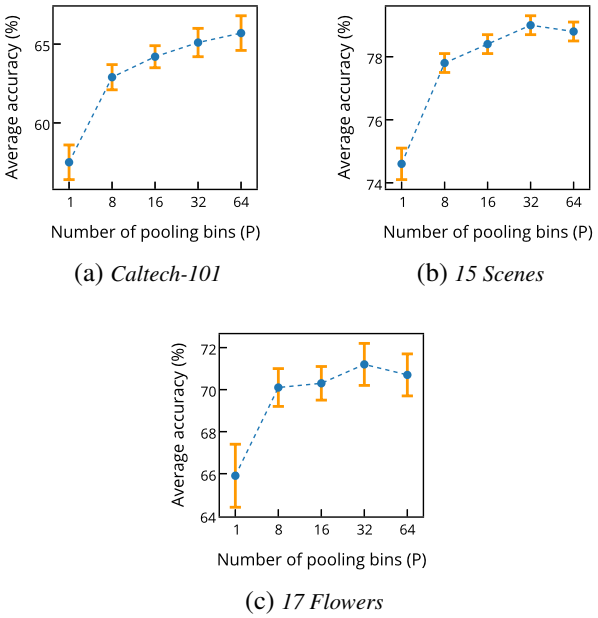
Spatial information: We used a three-layer spatial pyramid of 21 cells (1×1, 2×2, 4×4) whenever spatial information was included. Similar to [8], the image's low-level features within each spatial cell are (1) clustered around the P feature bins. Then, (2) pooled accordingly. The final image representation is the concatenation of the locally pooled features across all cells.

Image representation: We conducted two sets of experiments (In Sect. 5.2 and Sect. 5.3 respectively). The image is represented with a different representation in each. In Sect. 5.2, the image is represented with a concatenation

of P BoWs. Thus, the final image representation is of PK dimensions which is along with image label are passed to the classifier. On the other hand, in Sect. 5.3 the image is represented with a concatenation of S ($S = 21$) locally pooled features each of which is of a dimension equals to PK. Thus, the final image representation is of PKS dimensions which is along with the image label are passed to the classifier.

Classification: We adopted the one-versus-all methodology by training one SVM classifier per class using the library reported in [18]. The cost parameter was determined by cross-validation within the training data of the target dataset. Following the common practice of training/testing, we used 30 training images per class for *Caltech-101*, 60 for *Caltech-256*, 100 for *15 Scenes*, and 40 for *17 Flowers*. The rest were used for testing.

Evaluation: Average classification accuracy and standard deviation, over e runs, are reported as classification results. The number of runs (e) is set to 10 for all datasets except for *17 Flowers*, where training/testing data splits are provided by the authors.



(a) *Caltech-101* (b) *15 Scenes*

(c) *17 Flowers*

Fig. 2 Classification accuracy (%) comparison between the baseline ($P = 1$) and our method ($P > 1$). Local pooling in the feature space always improves classification performance over the baseline for all datasets. Over-binning ($P > 32$) degrades performance on (b) and (c).

5.2 Pooling locally in the feature space

In this section, we empirically analyze the performance of the proposed method within the feature space only. In other words, spatial information is not included at all here (i.e., our method is not implemented within an SP). Thus, *results reported in this subsection are by no means intended to be compared with the published state-of-the-art methods*. For such a comparison, please refer to Sect. 5.3, which is dedicated to this purpose. This style of reporting experimental results has been previously adopted by others including Fanello et al [19] and Khan et al [32]. We start by assessing the performance improvement our method brings to the baseline, in Sect. 5.2.1. Then we compare our method to a closely related work on local pooling in the feature space, in Sect. 5.2.2.

5.2.1 Contribution to the baseline

The purpose of this study was to empirically assess the performance improvement our method brings to the baseline, i.e., how locally pooling image features from a space partitioning obtained by Bregman co-clustering boosts the classification performance of the baseline. As a classification baseline, we adopted the Bag-of-Features (BoW) model, implemented as detailed in Sect. 5.1.2. We chose to analyze the contribution of our method in generic-object, scene, and fine-grained classification scenarios. Thus, experiments were conducted on *Caltech-101*, *15 Scenes*, and *17 Flowers* image datasets.

Fig. 2 compares classification performance of the baseline ($P = 1$) to that of our method implemented for an increasing number of pooling bins $P \in \{8, 16, 32, 64\}$. Note that $P = 1$ means that no local pooling is conducted. In other words, the image is represented with a traditional BoW. This BoW along with the image label are what passed to the classifier. From the results, it is clear that local pooling in the feature space always improves classification performance over the baseline for all datasets. This was observed in a previous work [19]. Moreover, doubling the number of pooling bins always boosts performance on the first dataset. However, for both the second and third datasets, performance degrades when 64 pooling bins are used. In summary, performance boost ranges between 5.4% and 8.2% for *Caltech-101*, 3.2% and 4.4% for *15 Scenes*, and 4.2% and 5.3% for *17 Flowers*. Remember that this boost in performance is achieved while no SP is used, i.e., the only spatial cell is the image plane itself. However, when we use an SP (of 21 spatial cells), performance boost becomes much smaller. In other words, the boost our method brings to the baseline saturates with the increase in the number of cells. See Sect. 5.3.1 for more details.

To confirm that our implementation of the baseline achieves results comparable to the recently published results, we implemented the baseline within a spatial pyramid following the details of Sect. 5.1.2. We obtained 76.8 ± 0.8 and 82.7 ± 0.3 on *Caltech-101* and *15 Scenes*, respectively. These results are very close to (slightly better than) those in [50] in which

similar experimental settings were followed. As for *17 Flowers*, we are aware that the baseline performance is way behind what has been reported recently in [12, 33], in which low-level features are both RGB colors and dense SIFTs extracted at multiple scales. The purpose of using this dataset here is just to assess our method in the feature space on a fine-grained image classification dataset implemented within a simple but widely used baseline.

5.2.2 Comparison to a closely related work

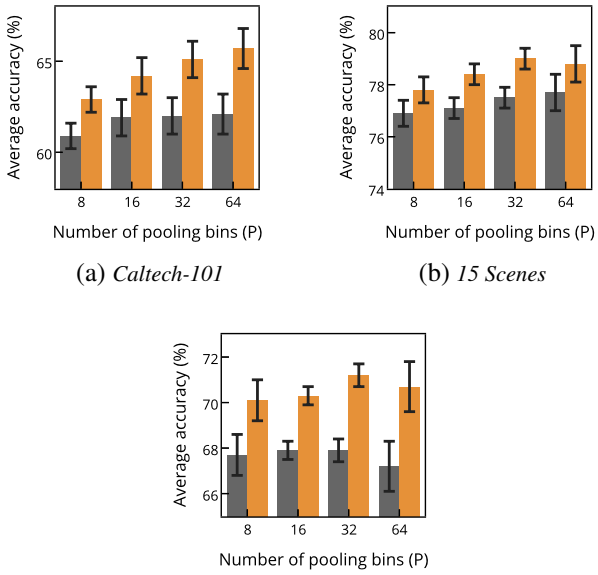
In this section, we compare our method to [8], which is, to the best of our knowledge, the most notable work on local pooling in the feature space within the adopted pipeline. This method relies on partitioning the feature space by clustering the input image's low-level descriptors over the codewords of a codebook obtained using k-means clustering and then jointly pooling only descriptors that belong to the same cluster, i.e., visually similar descriptors. Note that, in contrast to our method, this method partitions the feature space without any consideration of the semantics of the input image. Fig. 3 compares the classification performances of the two methods on *Caltech-101*, *15 Scenes*, and *17 Flowers*.

The obtained results clearly show that our method outperformed [8] for all datasets. In fact, using only 8 bins, our method achieved better results even when 32 or 64 bins (whichever performed better) were utilized by the comparative method. The obtained results emphasize that our features are pooled from a space partitioning of a better quality than that of the comparative method.

It would be interesting to empirically assess the quality of the space partitioning utilized in the two methods. To this end, we compared classification performance of features pooled from bins (space partitioning) obtained by three different methods: (1) Bregman co-clustering, (2) k-means, and (3) randomly selected from a k-means codebook of size 4096. The experiment was conducted on *Caltech-101*, *15 Scenes* and *17 Flowers* for $P \in \{8, 16, 32, 64\}$. The results obtained are shown in Fig. 4. As expected, our features always outperformed randomly pooled ones. However, a more interesting finding is that on the first two datasets, features of Boureau et al [8] almost always performed worse (or similar to) than those pooled from random bins. This result is an evidence that k-means is far from providing an optimal partitioning of the feature space.

5.3 Bregman pooling for image classification

In this section, the proposed method is compared to other works on three datasets: *Caltech-101*, *15 Scenes*, and *Caltech-256*. It should be noted that we decided to replace *17 Flowers* with *Caltech-256* and limit our comparison to two classification tasks due to (1) usually [12, 33] when reporting



(a) *Caltech-101* (b) *15 Scenes*

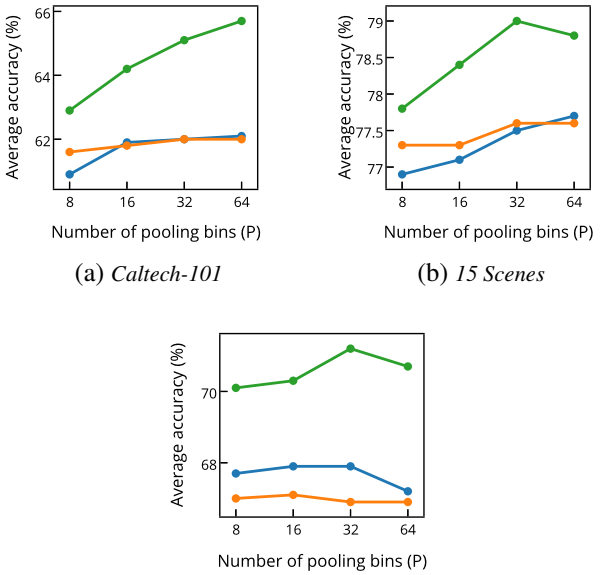
(c) *17 Flowers*

Fig. 3 Classification accuracy (%) comparison between the method in previous work [8] (gray) and our method (orange). Our method outperforms [8] on all datasets for less feature dimensionality.

results on *17 Flowers* color features (and sometimes kernel learning) are used. However, we only use SIFT features extracted from gray-scale images. Therefore we believe it is unfair to compare our method to other methods on this dataset. (2) The only two feature-space local pooling methods [8, 19] our method is compared to (Table 1), do not report performance on *17 Flowers* and use *Caltech-256* instead. We first compare Bregman pooling to other spatial pyramid (SP)-based methods in Sect. 5.3.1. Then in Sect. 5.3.2, the comparison is extended to state-of-the-art methods.

5.3.1 Comparison with SP-based methods

For a fair comparison, we implemented Bregman pooling within an SP following the details shown in Sect. 5.1.2. It should be noted that only on *caltech-256* we changed the adopted baseline and used the one described in [51]. This baseline consists of multi-scale SIFT features, LLC coding and max pooling. This baseline showed impressive results on *Caltech-256* [8, 51] therefore we adopt it. To replicate the result reported in [51], we used their publicly available implementation in our experiment. The results obtained are shown in Table 1 for $P \in \{1, 8, 16\}$. We experimented with $P \in \{32, 64\}$ (not shown) and found that over-binning ($P > 16$) degrades the performance on all three datasets. This observation has been reported in [8]. Table 1 also quotes results reported for other SP-based methods. We refrained from graphing the results in Table 1 and limited our presentation to a tabular form due to: (1) graphing the results in



(a) *Caltech-101* (b) *15 Scenes*

(c) *17 Flowers*

Fig. 4 Classification accuracy (%) comparison among the method in previous work (blue), our method (green) and random (orange). On (a) and (b), features pooled from an appearance-based bins (previous work) almost always perform worse than those pooled from random bins.

Table 1 might confuse the reader into thinking that we have replicated the results in Table 1 (which is not the case). (2) This form (tabular) of comparing obtained results with those of previous methods is a common practice in the image classification community (See [8, 19, 32, 52]).

However, since all quoted works are extensions to the original SP model of Lazebnik et al [35], simply listing the obtained results does not give a clear insight into how each improves the model. Thus, in order to avoid comparing apples to oranges, we break the listed works into four main groups based on what component, of the SP model, each improves. Thus, works are grouped into (1) those that improve the coding step, including works by Yang et al [53], Wang et al [51] and Wang et al [52], (2) those that improve the pooling operator, including works by Yang et al [53] and Koniusz et al [33], (3) those that enrich the spatial information captured by the model, the works by Khan et al [31, 32], and finally (4) those that locally pool in the feature space, including works by Boureau et al [8], Fanello et al [19], and ours. Table 1 also includes studies by Boureau et al [7] and Chatfield et al [10], which are two widely cited benchmarking studies that extensively evaluated the model using different combinations of components and parameters. In the following, we discuss our obtained results within the context of each group individually.

Within the first group, Yang et al [53] and Wang et al [51] are highly successful extensions to the the SP model that adopt (aside from max pooling) two improved coding methods: SC and LLC coding, respectively. Our method was implemented within the former on the first two datasets and within the latter on the third dataset.

We achieved 2.0% and 1.8% performance boosts over Yang et al [53] on the first two datasets, and 0.7% performance boost over Wang et al [51] on the third dataset. These results indicate the importance of our local pooling over these two SP extensions. Our method also outperformed the recent Collaborative Linear Coding (CLC) [52] on *15 Scenes* by 0.2% (but with $+0.1$ in standard deviation). However, due to the differences in experimental settings used (We used single-scale SIFTs and a 4096-dim codebook, while Wang et al [52] used multi-scale SIFTs and a 2048-dim codebook.), it is difficult to compare the two precisely.

Within the second group, the AxMin@n pooling operator of Koniusz et al [33] outperformed all other methods on *Caltech-101*. In fact, our best performance fell 2.5% behind their reported performance. However, it should be noted that Koniusz et al [33] used dense SIFTs extracted at four different scales and thus each image is represented with an average number of 5200 descriptors. In any case, the results indicate the important role an adaptive pooling operator plays over the classification performance on this dataset. It is worth mentioning that within the same group, we obtained the best results on both *15 Scenes* and *Caltech-256*.

Our method also outperformed Khan et al [31, 32] on all three datasets. However, it is worth mentioning that even with a relatively small feature dimension (smaller codebook) and less dense low-level features, [32] achieved a highly competitive result on *15 Scenes*.

The proposed method also outperformed [8, 19] on all datasets. Our better performance over [8] can be understood in light of the results presented in Sect. 5.2. However, a comparison with [19] is difficult due to the lack of (1) a public implementation of their method and (2) reported results over different datasets. However, we achieved 0.4% ($P = 8$) and 0.5% ($P = 16$) boosts in performance over their reported results on *Caltech-256*. Analyzing the significance of this boost is impossible as Fanello et al [19] did not report their standard deviation. One major drawback common to all methods within this group is the inflated feature dimension. This is inevitable as the feature space is partitioned within every cell of the pyramid. Although we report better performance than previous works for smaller feature dimensions, still our features have much larger dimensions than those of other SP-based methods.

Finally, it is worth mentioning that both AxMin@n pooling [33] and CLC coding [52] can be easily implemented within our method. Moreover, it would be interesting, in the future, to test how adopting either (or both) of them affects the classification performance of the proposed method.

Table 1 Average classification accuracy (%) comparison on *Caltech-101*, *15 Scenes*, and *Caltech-256* datasets.

Method	Caltech-101	15 Scenes	Caltech-256	Feature dimension
Lazebnik et al [35]	64.6 ± 0.8	81.4 ± 0.5	-	4200, 8400
Yang et al [53]	73.2 ± 0.5	80.2 ± 0.9	40.1 ± 0.9	21504
Wang et al [51]	73.4	-	47.7	43008, 86016
Boureau et al [7][a]	71.8 ± 1.0	84.1 ± 0.5	-	21504
Boureau et al [8]	77.3 ± 0.6	83.3 ± 1.0	41.6 ± 0.6[b]	1397760, 365568, 344064
Chatfield et al [10]	76.1 ± 0.6	-	-	84000
Khan et al [31]	67.1	82.5	-	5000
Koniusz et al [33]	$\mathbf{81.3 \pm 0.6}$	-	-	86016
Wang et al [52]	-	84.3 ± 0.2	-	43008
Fanello et al [19]	-	-	47.9	1134592
Khan et al [32]	68.4	83.7	39.3[b]	13200, 13200, [c]
Proposed				
$P = 1$	76.8 ± 0.8	82.7 ± 0.3	47.7 ± 0.4	86016
$P = 8$	78.4 ± 0.8	84.3 ± 0.3	48.3 ± 0.3	774144
$P = 16$	78.8 ± 0.8	$\mathbf{84.5 \pm 0.3}$	$\mathbf{48.4 \pm 0.3}$	1462272

Works are listed in a chronological order.
Bold values indicate the best performance.
Some works do not report standard deviation.
A '-' means that the result is not reported in the corresponding work.
The feature dimension column lists dimension(s) of the image representation(s) used on the three datasets respectively.
[a] Intersection kernels are used rather than linear SVM.
[b] 30 training images per class are used.
[c] Feature dimension on *Caltech-256* is larger than 13200 but not clearly reported.

Table 2 State-of-the-art methods on *Caltech-101*, *15 Scenes* and *Caltech-256*.

Dataset	Method	Result
Caltech-101	He et al [27]	93.4 ± 0.5
15 Scenes	Zhou et al [54]	90.2 ± 0.3
Caltech-256	Chatfield et al [11]	77.6 ± 0.1

5.3.2 Comparison with state-of-the-art methods

To complete the picture, Table 2 shows the best classification results obtained on *Caltech-101*, *15 Scenes* and *Caltech-256* of which we are aware. From Table 2, we can see that the three best performing methods [11, 27, 54] are all based on convolutional neural networks [34, 36].

By comparing Tables 1 and 2, we can see a huge gap separating SP-based methods from those based on convolutional neural networks. In fact, convolutional neural networks have shown outstanding classification results on the majority of datasets recently. However, training convolutional neural networks requires huge amounts of data, time and processing power. For instance, Zhou et al [54] trained their network with more than 2.4 M images, and training took 6 days using a single GPU. On the other hand, He et al [27] and Chatfield et al [11] used 1.2 M images of ImageNet [46] as training data, and training the two networks took two weeks and three weeks, respectively.

It should be noted that methods based on convolutional neural networks are of a deep architecture which (mainly) differs from the "flat" architecture (of SP-based methods, for example) in that image features are *learned* not *handcrafted*. Therefore, the distinction between the two families of methods should always be kept in mind. Thus, we believe that the proposed method cannot benefit the recent deep-learning methods.

It is worth mentioning that prior to the dominance of deep-learning methods (Table 2), state-of-the-art was defined by methods, such as BossaNova [1] and Fisher Vectors (FV) [43]. Both of these methods enrich the BoW representation via using extra knowledge learned from the low-level features. The combined use of these two methods achieves a classification accuracy of 88.9% on *15 Scenes* [32].

6 Conclusions

In this paper, we have proposed a novel feature-space local pooling method for the commonly adopted architecture of image classification. In contrast to methods in previous works, our method produces pooling bins that are aware of the semantic context of the input image within the dataset. This is achieved by partitioning the feature space over clusters of visual prototypes common to images belonging to the same category (i.e., images of similar semantics). The clusters are obtained by Bregman co-clustering applied offline on a subset of training data.

The proposed method was experimentally validated on four different datasets belonging to three different classification tasks. The results obtained demonstrate that (1) our method outperforms methods in previous works on local pooling in the feature space for less feature dimensionality and (2) when implemented within a spatial pyramid (SP), our method achieves comparable results on three of the datasets used.

A possible extension to our method is to use multiple kernel learning to combine the locally pooled features into a final image representation.

Acknowledgements This work was partly supported by Grant-in-Aid for Scientific Research (B) 25280036, Japan Society for the Promotion of Science (JSPS).

References

1. Avila S, Thome N, Cord M, Valle E, De A AraúJo A (2013) Pooling in image representation: The visual codeword point of view. Computer Vision and Image Understanding (CVIU) 117(5):453–465
2. Banerjee A, Dhillon I, Ghosh J, Merugu S, Modha DS (2007) A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. Journal of Machine Learning Research (JMLR) 8:1919–1986
3. Bay H, Ess A, Tuytelaars T, Van Gool L (2008) Speeded-up robust features (SURF). Computer Vision and Image Understanding (CVIU) 110(3):346–359
4. Bengio Y, Courville A, Vincent P (2013) Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 35(8):1798–1828
5. Boiman O, Shechtman E, Irani M (2008) In defense of nearest-neighbor based image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1–8
6. Boureau YL (2012) Learning hierarchical feature extractors for image recognition. PhD thesis, New York University
7. Boureau YL, Bach F, LeCun Y, Ponce J (2010) Learning mid-level features for recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 2559–2566
8. Boureau YL, Le Roux N, Bach F, Ponce J, LeCun Y (2011) Ask the locals: multi-way local pooling for image recognition. In: Proceedings of the International Conference on Computer Vision (ICCV), pp 2651–2658
9. Bregman L (1967) The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. USSR Computational Mathematics and Mathematical Physics 7(3):200 – 217
10. Chatfield K, Lempitsky V, Vedaldi A, Zisserman A (2011) The devil is in the details: an evaluation of recent feature encoding methods. In: Proceedings of the British Machine Vision Conference (BMVC), pp 76.1–76.12
11. Chatfield K, Simonyan K, Vedaldi A, Zisserman A (2014) Return of the devil in the details: Delving deep into convolutional nets. arXiv:1405.3531
12. Chen Q, Song Z, Hua Y, Huang Z, Yan S (2012) Hierarchical matching with side information for image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 3426–3433
13. Cheng Y, Church GM (2000) Biclustering of expression data. In: Proceedings of the International Society for Computational Biology, pp 93–103
14. Csurka G, Dance CR, Fan L, Willamowski J, Bray C (2004) Visual categorization with bags of keypoints. In: Workshop on Statistical Learning in Computer Vision (ECCV), pp 1–22
15. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 886–893
16. Dhillon IS, Mallela S, Modha DS (2003) Information-theoretic co-clustering. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp 89–98
17. Everingham M, Gool L, Williams CK, Winn J, Zisserman A (2010) The pascal visual object classes (VOC) challenge. International Journal of Computer Vision (IJCV) 88(2):303–338
18. Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ (2008) Liblinear: A library for large linear classification. Journal of Machine Learning Research (JMLR) 9:1871–1874
19. Fanello S, Noceti N, Ciliberto C, Metta G, Odone F (2014) Ask the image: Supervised pooling to preserve feature locality. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 851–858
20. Fei-Fei L, Perona P (2005) A bayesian hierarchical model for learning natural scene categories. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 524–531
21. Fei-Fei L, Fergus R, Perona P (2004) Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 178–178
22. Fukushima K (1980) Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics 36:193–202
23. Grauman K, Darrell T (2005) The pyramid match kernel: discriminative classification with sets of image features. In: Proceedings of the International Conference on Computer Vision (ICCV), pp 1458–1465
24. Griffin G, Holub A, Perona P (2007) The Caltech 256. Tech. rep., California institute of technology
25. Gupta A, Bowden R (2012) Unity in diversity: Discovering topics from words: Information theoretic co-clustering for visual categorization. In: Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP), pp 628–633
26. Hartigan JA (1972) Direct clustering of a data matrix. Journal of the American Statistical Association 67(337):123–129
27. He K, Zhang X, Ren S, Sun J (2014) Spatial pyramid pooling in deep convolutional networks for visual recognition. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 346–361
28. Hubel DH, Wiesel TN (1962) Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. Journal of Physiology 160:106–154
29. Jegou H, Perronnin F, Douze M, Sanchez J, Perez P, Schmid C (2012) Aggregating local image descriptors into compact codes. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 34(9):1704–1716
30. Jia Y, Huang C, Darrell T (2012) Beyond spatial pyramids: Receptive field learning for pooled image features. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 3370–3377
31. Khan R, Barat C, Muselet D, Ducottet C, Saint-Etienne F, Etienne F (2012) Spatial orientations of visual word pairs to improve bag-of-visual-words model. In: Proceedings of the British Machine Vision Conference (BMVC), pp 102–112

32. Khan R, Barat C, Muselet D, Ducottet C (2015) Spatial histograms of soft pairwise similar patches to improve the bag-of-visual-words model. Computer Vision and Image Understanding (CVIU) 132(0):102–112
33. Koniusz P, Yan F, Mikolajczyk K (2013) Comparison of mid-level feature coding approaches and pooling strategies in visual concept detection. Computer Vision and Image Understanding (CVIU) 117(5):479–492
34. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NIPS), pp 1097–1105
35. Lazebnik S, Schmid C, Ponce J (2006) Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 2169–2178
36. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11):2278–2324
37. Liu J, Shah M (2007) Scene modeling using co-clustering. In: Proceedings of the International Conference on Computer Vision (ICCV), pp 1–7
38. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision (IJCV) 60(2):91–110
39. MacQueen JB (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability, pp 281–297
40. Mikolajczyk K, Tuytelaars T, Schmid C, Zisserman A, Matas J, Schaffalitzky F, Kadir T, Gool LV (2005) A comparison of affine region detectors. International Journal of Computer Vision (IJCV) 65(1-2):43–72
41. Nilsback ME, Zisserman A (2006) A visual vocabulary for flower classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1447–1454
42. Oliva A, Torralba A (2001) Modeling the shape of the scene: A holistic representation of the spatial envelope. International Journal of Computer Vision (IJCV) 42(3):145–175
43. Perronnin F, Sánchez J, Mensink T (2010) Improving the fisher kernel for large-scale image classification. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 143–156
44. Rematas K, Fritz M, Tuytelaars T (2013) The pooled NBNN kernel: Beyond image-to-class and image-to-image. In: Proceedings of the Asian Conference on Computer Vision (ACCV), pp 176–189
45. Russakovsky O, Lin Y, Yu K, Fei-Fei L (2012) Object-centric spatial pooling for image classification. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 1–15
46. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2014) ImageNet Large Scale Visual Recognition Challenge. arXiv:1409.0575
47. Sivic J, Zisserman A (2003) Video Google: a text retrieval approach to object matching in videos. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1470–1477
48. Vapnik VN (1998) Statistical learning theory, 1st edn. John Wiley and Sons, Inc
49. Vedaldi A, Fulkerson B (2008) VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/
50. Wang C, Huang K (2014) How to use bag-of-words model better for image classification. Image and Vision Computing DOI 10.1016/j.imavis.2014.10.013
51. Wang J, Yang J, Yu K, Lv F, Huang T, Gong Y (2010) Locality-constrained linear coding for image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 3360–3367
52. Wang Z, Feng J, Yan S (2014) Collaborative linear coding for robust image classification. International Journal of Computer Vision (IJCV) pp 1–12
53. Yang J, Yu K, Gong Y, Huang T (2009) Linear spatial pyramid matching using sparse coding for image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1794–1801
54. Zhou B, Lapedriza A, Xiao J, Torralba A, Oliva A (2014) Learning deep features for scene recognition using places database. In: Advances in Neural Information Processing Systems (NIPS), pp 487–495
55. Zhou X, Yu K, Zhang T, Huang TS (2010) Image classification using super-vector coding of local image descriptors. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 141–154