Scalable Image Annotation Using a Product Compressive Sampling Approach

Anastasios Maronidis, Elisavet Chatzilari, Spiros Nikolopoulos and Ioannis Kompatsiaris
Information Technologies Institute
Centre for Research and Technology Hellas
Email: {amaronidis, ehatzi, nikolopo, ikom}@iti.gr

***Abstract*—The rise of big data, which need computationally demanding manipulation has posed unprecedented challenges in the machine learning community. In this context, a variety of dimensionality reduction methods has been introduced in order to deal with the large-scale aspect of the data. However, their employment in very large scales often becomes impractical due to memory and computation limitations. In parallel, Compressive Sampling (CS) has recently emerged as a powerful mathematical framework providing a suite of conditions and methods that allow for an almost lossless and efficient compression of sparse data. Given that the majority of big data problems entail the existence of sparse datasets, our goal in this paper is to investigate the potential of CS as a dimensionality reduction method in very large scales. Towards this end, we propose a novel Product Compressive Sampling (PCS) method that is used for scalable image annotation. The new method displays robustness equal to the typical CS method, while decreases the computational complexity dramatically. Another novel characteristic of our work consists in establishing a connection between the sparsity level of the data and the effectiveness of PCS as a dimensionality reduction method for image annotation. For this purpose, a new metric for estimating the data sparsity is proposed. Finally, in comparison with the state-of-the-art, we show that PCS displays competitive classification performance, while at the same moment proves to be orders of magnitude superior in terms of computational efficiency.**

I. Introduction

Nowadays, as enormous volumes of digital images are accumulated, great interest has been placed on the theoretical and practical aspects of extracting knowledge from massive datasets [1]. In addition, there is a tendency to increase the dimensionality of the image descriptors, since it leads to better classification results [2], [3]. However, as the number of dimensions increases, it becomes more difficult for classification schemes to handle the data. It is therefore clear that the need to establish a fair compromise among classification accuracy, computational efficiency and storage capacity proves of utmost importance.

In order to deal with these new requirements, a set of technological paradigms have recently been brought into the forefront of interest, including: smart sampling, incremental updating, distributed programming, indexing, etc. Among these paradigms, of particular interest we consider the case of smart sampling, where massive datasets are sampled and the analysis algorithms are applied only on the sampled data. A couple of questions though naturally arises: How many samples do we need and how do we ensure that the selected samples are representative of the entire dataset. Towards this end, we believe that the mathematical theory of Compressive Sampling (CS) has the potential to answer some of these questions.

The theory of CS offers a reversible scheme for efficiently compressing and reconstructing large pieces of digital data [4], [5], [6], [7]. More specifically, the objective of CS is firstly to design a projection matrix that can be used to transform high-dimensional data to a new target space of a much smaller dimensionality [8] and secondly to ensure all those conditions that the projection matrix must obey so as for the compressed data to be perfectly reconstructable in the initial high-dimensional space [7].

Although, the use of CS has already proven beneficial for a multitude of applications [9], an imperative assumption that ensures its correct functionality is the sparsity of the data involved [4], where by sparsity we mean the extend to which the data contain zero values. It is worth mentioning, that in real world problems, rarely the data are strictly sparse. Instead, it is often the case that most of the data components are close to zero, while only a small portion of them have significantly larger values. Indeed, state-of-the-art methods for feature extraction like the Fisher vectors [2] provide us with this kind of data. In such cases, the data are referred to as approximately sparse or more often as compressible. Fortunately, there are theoretical guarantees that prove the robustness of CS even when dealing with compressible data [10] and it is this capacity that has triggered our interest about the potential of CS to alleviate some of the problems related to large-scale data.

Provided that the data-sparsity or the data-compressibility assumption is satisfied, the great leap of CS is proving the rather surprising finding that a random matrix, i.e. whose values at each position have been generated, for instance, by a Gaussian distribution, qualifies as an appropriate projection matrix [4]. The ability to use a random, rather than a complicated and difficult to generate, projection matrix for compressing the data, offers a huge advantage in the context of large-scale data and specifically with respect to velocity- and memory-related challenges, e.g. real-time processing, live-streaming, etc.

Although CS is renown in the literature for its power to compactly represent digital data [11], in this paper we investigate the extent to which CS-based methods can be used for discrimination purposes in a scalable manner. More specifically, our aim is to identify potential ways for exploiting

the theory of CS in the large-scale image annotation domain. The rationale of our work is based on the outcome of some important works like [12], which claim that sparse representations have an inherent discriminant nature. Indeed, using a small subset among a pool of features offers a compact way to represent an image and therefore the selected features strongly characterise this image. Based on the above claim, we envisage that combining sparse representation with discriminative methods may lead to performance competitive to that of the state-of-the-art in large-scale classification problems [13], [14].

In this paper, we rely on the principles of CS to propose a novel scalable Product Compressive Sampling (PCS) method for dimensionality reduction in the image annotation domain. PCS decomposes a high-dimensional vector into a number of smaller vector segments performing an equal number of CS projections and concatenating the results. Through both a theoretical analysis and an experimental comparison with typical CS, we show that PCS exponentially reduces the computational load of the dimensionality reduction process, while at the same moment displays performance equivalent to CS. We further establish a connection between the performance of PCS and the level of the data sparsity. Finally, a comparison with the state-of-the-art, shows that our method displays competitive performance in terms of classification performance, while at the same time outperforms the state-of-the-art methods in terms of computational efficiency.

The remainder of this paper is organized as follows. A number of related works on dimensionality reduction as well as a variety of methods employing the notion of sparsity are presented in Section II. The proposed methodology for efficiently reducing the dimensionality of big data in the large-scale image annotation problem is analytically presented in Section III. Experimental results proving the potential of the proposed approach are provided in Section IV. Finally, conclusions are drawn in Section V.

II. Related Work

In this section, a brief review of the bibliography related to our work is presented. The review is divided into two subsections. In the first subsection we present a set of indicative dimensionality reduction methods, while in the second subsection we review the role of sparsity and compressibility in the development of new or in the improvement of existing methods.

A. Dimensionality Reduction

A plethora of methodologies have recently been proposed for reducing the dimensionality of large-scale data. In [15], dimensionality reduction methods have been classified into three main categories, which are briefly described hereunder.

The first category consists of methods based on statistics and information theory and among others includes Vector Quantisation (VQ) [16] and Principal Component Analysis (PCA) [17]. In addition, due to the advent of large-scale data, new scalable techniques such as the Product Quantisation (PQ) [18] have also been developed. PQ decomposes a vector space into a Cartesian product of quantised subspaces for constructing short codes representing high-dimensional vectors. In this vein, in [3] using the so-called Vector of Locally Aggregated Descriptors (VLAD), the authors employ PQ for performing nearest neighbour search during the indexing process of very large databases for retrieving the most similar images to a query image. Moreover, within the vector quantization context, effort has recently been allocated on the optimization of the kernel K-Means. For instance, a clustering algorithm, which models multiple information sources as kernel matrices is proposed in [19].

The second category includes methods based on Dictionaries, where a vector is represented as a linear combination of the dictionary atoms. For instance, sparse representations with over-complete dictionaries have been applied on image de-noising [20]. Utilizing the K-SVD technique, the authors train dictionaries based on a set of high-quality image patches or based on the patches of the noisy image itself. The image is iteratively de-noised through a sparse coding and a dictionary update stage. Based on a similar approach, K-SVD has been utilized for the restoration of images and video [21], where the sparse representations are obtained via a multi-scale dictionary learned using an example based approach. The above de-noising algorithm has also been extended for color image restoration, de-mosaicing and in-painting [22]. In addition, an over-complete dictionary design method that combines both the representation and the class discrimination power has been proposed in [23] for face recognition. The presented method is an extension of the above K-SVD algorithm [8] and is referred to as the D-KSVD, since a discriminative term is added into the main objective function of the regular K-SVD.

The third category consists of methods that seek for "interesting" projections leading to learning projection matrices. For instance, Linear Discriminant Projections (LDP) has been proposed as a useful tool for large-scale image recognition and retrieval [24]. Moreover, in this category, Hashing has also been proven a computationally attractive technique, which allows one to efficiently approximate kernels for very high-dimensional settings by means of a sparse projection into a lower dimensional space. For instance, in [25] hashing has been implemented for handling thousands of classes on large amounts of data and features. In the same fashion, specialized hash functions with unbiased inner-products that are directly applicable to a large variety of kernel methods have also been introduced. Exponential tail bounds that help explain why hash feature vectors have repeatedly led to strong empirical results are provided in [26]. The authors demonstrate that the interference between independently hashed subspaces is negligible with high probability, which allows large-scale multi-task learning in a very compressed space. Finally, advances have been made in manifold learning through the development of adaptive techniques that address the selection of the neighborhood size as well as the local geometric structure of the manifold [27].

Our method along with the core CS actually belong to the third category. The main differentiation of PCS and generally of CS-like methods from the rest methods presented above is that the projection matrices involved are randomly learned via e.g. a Gaussian distribution, while the other methods require intensive calculations in the training phase. For instance, PCA requires eigen-analysis of large covariance matrices, and PQ performs multiple times k-means clustering. In addition, although both PQ and PCS follow a similar vector-

decomposition scheme, in PCS the classification is performed in the reduced space, while on the contrary, in PQ the reduced data need to be firstly decompressed [28] adding an extra computational burden. The advantage of PCS in terms of computational complexity is absolutely suited for handling large-scale data, as it provides tremendous speed in real-time processes.

B. Sparsity

As we have already seen, a necessary pre-condition that guarantees the correct operation of our method is the sparsity or compressibility of the data involved. Apart from our method, the notion of sparsity and compressibility seems to play a key role in recent advances in the computer science community. For instance, in [29], the authors learn an over-complete dictionary using the K-SVD algorithm along with a set of predefined image patches. Using this dictionary they compress facial images by representing them as sparse linear combinations of the dictionary atoms. On the other hand, the potential of fusing index compression with binary bag of features representation has been investigated in [30]. The proposed method is based on an inverted file structure, which is merely concerned with the distances of the non-zero positions of sparse vectors. A "sparsified" version of PCA has also been proposed in [31]. Using elastic net methods, the principal components are represented as sparse linear combinations of the original variables, as opposed to the typical PCA algorithm.

In the same vein, recently, considerable effort has been allocated in assigning semantic information to sparse representations, in the fields of computer vision and pattern recognition [32]. It has been shown through a variety of experimental settings that sparse representations convey important semantic information. This assertion has been corroborated within diverse challenging application domains such as face recognition, image reconstruction, image classification, etc. In this context, semantic issues have been tackled using a sparse image reconstruction based approach in the challenging object recognition problem [33]. The images are encoded as super-vectors consisting of patch-specific Mixture Gaussian Models and reconstructed as linear combinations of a number of training images containing the objects in question. The reconstruction coefficients are inherited by the corresponding labels providing annotation to the query image.

Towards the same direction, the problem of accelerating sparse coding based scalable image annotation has been addressed in [34]. The pursuit of an accurate solution is based on an iterative bilevel method, which achieves reducing the large-scale sparse coding problem to a series of smaller sub-problems. Finally, the discriminative nature of sparse representations to perform classification has been experimentally proven in [12]. Solving a dedicated linear programming sparsity-encoding problem, the authors propose a face recognition algorithm robust to expressions, illumination and occlusions.

C. Measures for sparsity

From the above discussion, sparsity seems to be well-suited for large-scale problems. Similarly to the works presented above, our method proves also strongly dependent to the sparsity of the data. In the core CS theory, the sparsity is calculated using the l_0-norm, which counts the non-zero values of a vector [35]. However, l_0-norm is sensitive to small distortions of the vector values, since even a small perturbation of the zero coefficients may lead to considerably different estimation of sparsity. Furthermore, changing the non-zero values does not affect sparsity. For the above reasons, the l_0-norm may prove unsuitable for estimating the compressibility of vectors without zero values. Instead, the l_p-norm, also denoted as $\|\cdot\|_p$, which uses the p-th power of the vector coefficients, is often used for this purpose [35]. However, the latter totally defies the relativity among the vector values, which may lead to peculiar results. For instance, let $p = 1$ and consider the following vectors: $\mathbf{x}_1 = [4, 4, 4, 4, 4]^T$ and $\mathbf{x}_2 = [0, 0, 0, 0, 20]^T$. Then apparently $\|\mathbf{x}_1\|_1 = \|\mathbf{x}_2\|_1 = 20$, though we would expect that $\mathbf{x}_2$ is pretty much sparser than $\mathbf{x}_1$. Interestingly, notice that in this particular example, the l_0-norm returns more reasonable results.

Motivated by the above analysis, in our work, in order to overcome the previous disadvantages, we envisage that the relative difference among the values is more crucial than the values per se in estimating the compressibility of a vector. Based on this claim, we propose a new approach for estimating data sparsity, which makes use of the first order differences of a vector. As we will see in the following sections, the experimental results obtained are quite interesting advocating the potential application of our methodology to other domains as well.

III. Methodology

Based on the outcomes of the CS theory, we propose the use of random projections for reducing the dimensionality of large-scale datasets in order to perform image classification. In this case study, a number of images annotated with a predefined set of p high-level concepts c_j, for $j = 1, 2, \ldots, p$ is used as the training set. Towards this end, for each image a set of visual features are extracted providing a corresponding feature vector. The resulting feature vectors are subsequently subject to dimensionality reduction yielding their compressed versions, which are finally used for training a Support Vector Machine (SVM) model in an one-vs-all fashion. The steps of the above pipeline approach are described in more detail in the following subsections. A notation table is also provided (see Table I) in order to help the reader follow the presentation more easily.

A. Feature Extraction

A variety of techniques has been proposed for extracting a representative set of features from data, including SIFT [36], SURF [37] and MPEG7 [38]. Among these techniques, for the purpose of our study we choose SIFT, which has been proven quite robust in classification problems [36]. Given a raw image, 128-dimensional gray SIFT features are extracted at densely selected key-points at four scales, using the vl-feat library [39]. PCA is then applied on the SIFT features, decreasing their dimensionality from 128 to 80. The parameters of a Gaussian Mixture model with $K = 256$ components are learned by Expectation Maximization from a set of descriptors, which are randomly selected from the entire set of descriptors extracted by an independent set of images. The descriptors are encoded in a single feature vector using the Fisher vector encoding [2].

Moreover, each image is divided in 1 x 1, 3 x 1 and 2 x 2 regions, resulting in 8 total regions, also known as *spatial pyramids*. A feature vector is extracted for each pyramid by the Fisher vector encoding and the feature vector corresponding to the whole image (1 x 1) is calculated using sum pooling [40]. Finally, the feature vectors of all 8 spatial pyramids are concatenated to a single feature vector of 327680 components, which comprise the overall data dimensionality that is to be reduced in the experiments.

B. Dimensionality Reduction

As we have seen in the previous subsection, the feature extraction phase provides us with data lying in a very high-dimensional space. In an attempt to make these data more tractable we perform dimensionality reduction, while retaining the most important information by exploiting the theoretical findings of CS. Towards this end, we need to know the least number m of reduced dimensions required in order to guarantee the correct functioning of the compression-reconstruction scheme of CS. It has been theoretically proven that this number is strongly associated with the data sparsity [41]. More specifically, it has been shown that if k is the number of non-zero features of a vector, then m must be at least on the order of $O(k \cdot log(\frac{n}{k}))$, where n is the initial vector dimensionality. Moreover, based on [10], compressible data can be treated as approximately sparse and therefore, to a great extent, the aforementioned theoretical bounds are expected to apply on compressible data, as well.

Apparently, from the above discussion, in order to find the minimum allowed number of reduced dimensions, we need to calculate k. Clearly, for sparse data, this calculation is straightforward. However, since often the data are not strictly sparse, but rather compressible, we actually need to estimate an approximation of k. Towards this direction, in the following subsection we propose a differential approach to estimate k for compressible data.

1) An approach to estimate data compressibility: In this subsection, given a data matrix:

$$\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_N] = \begin{pmatrix} x_{11} & \cdots & x_{1N} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nN} \end{pmatrix}, \quad (1)$$

consisting of N n-dimensional feature vectors, we propose a methodology to estimate on average an approximation of the number k of comparably large non-zero values of these vectors. For each feature vector $\mathbf{x}_i$, we sort its absolute values in ascending order obtaining a new set of sorted vectors. For the sake of simplicity, let us assume that $\mathbf{X}$ contains the above sorted vectors. From $\mathbf{X}$, we calculate the mean vector $\overline{\mathbf{x}} = [\overline{x}_1, \ldots, \overline{x}_n]^T$ consisting of the average values

$$\overline{x}_j = \frac{1}{n} \sum_{i=1}^{N} x_{ji} \quad (2)$$

per each feature component $j \in \{1, 2, \ldots, n\}$ along the N rows of the data matrix. Since there is no guarantee that $\overline{\mathbf{x}}$ is sorted as well, we sort it once again in ascending order obtaining $\overline{\mathbf{x}}_s = [\overline{x}_{s_1}, \ldots, \overline{x}_{s_n}]^T$ with:

$$|\overline{x}_{s_1}| \leq |\overline{x}_{s_2}| \leq \cdots \leq |\overline{x}_{s_n}|, \quad (3)$$

where s_j are the indices of the sorted values. We finally calculate the first order differences δ_{s_j}, for $j \in \{2, \ldots, n\}$ and we find the index s'_j with the maximum first order difference:

$$s'_j = \underset{j}{\operatorname{argmax}}\{\delta_{s_j}\}. \quad (4)$$

This index, essentially conveys information about the data compressibility, since it indicates a large step of the "average" features from smaller to larger absolute values. Hence, the features with indices $s_j > s'_j$ have significantly larger absolute values than the remaining features. Therefore, they could be considered as non-zero's providing a reasonable estimation of k:

$$k = n - s'_j. \quad (5)$$

2) Dimensionality Reduction using Compressive Sampling: Given a set of data lying in a very high-dimensional space, the objective of CS is to design a projection matrix that can be used to transform these data to a new target space of a much smaller dimensionality. In this target space, the new data representations will eventually be used for classification. Using more strict formulation, let $\mathbf{x} \in \mathbb{R}^n$ be an n-dimensional vector, as this has been formed by the approach presented in Section III-A. Based on the capability of using random projections, an $m \times n$ ($m \ll n$) projection matrix $\mathbf{D}$, whose entries contain random values generated by a Gaussian distribution with mean zero and variance $1/n$ is built. The vector $\mathbf{x}$ is then transformed into $\mathbf{y} \in \mathbb{R}^m$, through $\mathbf{y} = \mathbf{D}\mathbf{x}$.

In practice, the rows of the above projection matrix are orthonormalized through a Singular Value Decomposition (SVD) step, so that $\mathbf{D}\mathbf{D}^T = \mathbf{I}$. Working with orthonormal bases provide practical benefits to the whole process. Indeed, there have been presented some interesting works in the literature, e.g., [42], [43], proving that the use of SVD offers robustness to the recovery of a compressed signal under the presence of noise. We anticipate that this property of SVD is also important for the classification task that we address in this paper.

Despite its power, in some practical cases, when the number of dimensions is too high, CS proves inefficient in handling big data, since due to insufficient memory resources it fails to perform SVD and big matrix multiplications. In this paper, in order to alleviate this shortcoming, we propose a novel method, which uses an implementation of CS based on a vector decomposition scheme as described in detail in the following subsection.

3) Product Compressive Sampling: An innovative scalable method to apply CS with computational efficiency, called Product Compressive Sampling (PCS) is proposed. PCS decomposes a high-dimensional vector into a number of smaller segments and applies CS reducing the dimensionality of each segment. The resulting reduced segments are then concatenated providing the final low-dimensional vector. More specifically, let

$$\mathbf{x} = [x_1, \ldots, x_n]^T, \quad (6)$$

be an n-dimensional vector and m the desired reduced dimensionality. At the first stage, the components x_j, $j \in \{1, 2, \ldots, n\}$ are sorted in ascending mode. To keep the notation as simple as possible, let us consider the above $\mathbf{x}$

as the sorted vector. Subsequently, $\mathbf{x}$ is subdivided into a set $\{\mathbf{x}_i\}_{i=1}^{b}$ of b vector-blocks of size $q = \frac{n}{b}$ each, with

$$\mathbf{x}_i = [x_i, x_{b+i}, x_{2b+i}, \ldots, x_{(q-1)b+i}]^T, \tag{7}$$

for $i = 1, 2, \ldots, b$. Having obtained the blocks, PCS reduces the dimensionality q of each block to $\frac{m}{b}$ and subsequently concatenates the resulting vectors to form a $b \times \frac{m}{b} = m$-dimensional final vector.

In the above approach, it is clear that $\frac{n}{b}$ and $\frac{m}{b}$ must be integers, since the former expresses the length of each block constructed from the initial vector, while the latter is the length of each block after the reduction. Provided that the above limitation is satisfied, a random projection matrix $\mathbf{D}$ of size $\frac{m}{b} \times \frac{n}{b}$ is firstly orthonormalized through an SVD step and then used for reducing the data of each block $\mathbf{x}_i$ to $\frac{m}{b}$ dimensions, producing $\mathbf{y}_i$, for $i = 1, 2, \ldots, b$, through

$$\mathbf{y}_i = \mathbf{D}\mathbf{x}_i. \tag{8}$$

The finally reduced vector is built by concatenating the obtained blocks:

$$\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_b], \tag{9}$$

which consists of m dimensions. At this point, it is worth noting that in the extreme case where $b = 1$, PCS collapses to the typical CS method, since in this case the whole vector is considered as one block.

The functionality of PCS relies upon the assumption of the data sparsity. In fact, when subdividing a sparse vector we expect that, on average, the vector segments are also supposed to commensurately inherit the sparsity of the main vector, therefore permitting the use of random projections per each segment. As a matter of fact, let us consider a feature vector containing k non-zero values. Then, as already mentioned, the lower bound m_L of the reduced dimensionality for the initial high-dimensional vector must be on the order of $O(k \cdot log(\frac{n}{k}))$. Assume now that the initial vector is decomposed into b equal length blocks employing the proposed approach. From the way the several blocks are constructed (see eq. 7), the sparsity of each block is expected to be on average $\frac{k}{b}$, since actually the above block decomposition procedure comprises a uniform subsampling of the initial sorted vector. Consequently, the corresponding lower bound m'_L for each block becomes:

$$m'_L = O(\frac{k}{b} log(\frac{\frac{n}{b}}{\frac{k}{b}})) = \frac{1}{b} O(k \cdot log(\frac{n}{k})) = \frac{1}{b} m_L, \tag{10}$$

which after the concatenation phase is aggregated to $b \cdot \frac{1}{b} m_L = m_L$. This result in a few words means that PCS maintains on average the dimensionality bounds of CS in the reduction process.

Concluding this subsection, it is worth noting that, in a classification scheme, the same block indices must be used across all training and test samples. In this case, in order to keep the dimensionality bounds as loose as possible, the block indices can be determined and fixed through the application of eq. 7 to the mean vector among the training samples.

In the following subsection, we analytically show that the computational complexity of PCS is orders of magnitude smaller than the one of CS offering PCS credibility as a scalable dimensionality reduction method.

TABLE I. NOTATION TABLE

$\mathbf{x}$	A feature vector
n	Dimensionality of $\mathbf{x}$
x_i	i-th component of $\mathbf{x}$
x_{s_i}	s_i-th component of $\mathbf{x}$ after sorting in ascending order
k	Number of non-zeros in a vector
δ_i	First order difference of $\mathbf{x}$ at i-th position
$\mathbf{y}$	Projected vector through $\mathbf{y} = \mathbf{D}\mathbf{x}$
m	Dimensionality of $\mathbf{y}$
$\mathbf{D}$	$m \times n$ projection matrix
b	Number of blocks $\mathbf{x}$ is decomposed into
q	Block length
m_L	Lower bound of reduced dimensions using CS
c_k	k-th concept used for annotation

4) Computational Analysis: From the above discussion, the computational load using CS consists of performing SVD on a random matrix along with a number of matrix multiplications during the projection phase. The number of calculations involved in the above process is exponentially associated with the number of dimensions. Based on this, PCS earns its advantage by splitting the initial CS problem to a number of smaller CS sub-problems by decomposing a vector into an equal number of blocks. In this way, PCS achieves to alleviate the computational load of CS in the way explicitly described in the following analysis.

The computational complexity of performing SVD on an $m \times n$ matrix is on the order of $4m^2n + 8mn^2 + 9n^3$. Similarly, the computational complexity of multiplying an $m \times n$ matrix by an $n \times 1$ vector is on the order of $\mathcal{O}(mn)$. In order to compare PCS with CS, for the sake of simplicity, let us investigate the case where the number of dimensions in the reduced space is $\frac{n}{2}$. Under these circumstances, using CS requires SVD on a $n \times \frac{n}{2}$ random matrix with computational complexity $\mathcal{O}(5n^3)$, and a multiplication between the transpose of the resulting matrix by the initial $n \times 1$ vector, which is $\mathcal{O}(\frac{n^2}{2})$. On the other hand, using PCS with b blocks, requires SVD calculation of a matrix of size $\frac{n}{b} \times \frac{n}{2b}$, which is $\mathcal{O}(\frac{1}{b^3} 5n^3)$, that is $\mathcal{O}(b^3)$ times less than using CS. Subsequently, the resulting projection matrix is multiplied by the b vector-blocks, which is $\mathcal{O}(\frac{1}{b^2} \frac{n^2}{2})$, that is $\mathcal{O}(b^2)$ times less than using CS.

The previous analysis, clearly shows that the computational benefit using PCS is exponentially associated with the number of blocks used. However, this might come at the cost of a loss of information, since by significantly shrinking the length of the blocks might cause the loss of vector-structure. This uncertainty led us to investigate the performance of PCS as a function of the number of blocks used, through an experiment presented in Section IV. As we will see, the results show that the performance of PCS proves to be independent of the number of blocks used.

C. Classification

The vectors obtained after the dimensionality reduction methodology described in Subsection III-B are used in the classification phase. For classifying the data, we use Support Vector Machines (SVM) [44]. For each concept c_k, $k = 1, 2, \ldots, p$, a binary linear SVM classifier $(\mathbf{w}_k, \mathbf{b}_k)$, where $\mathbf{w}_k$ is the normal vector to the hyperplane and $\mathbf{b}_k$ the bias term, is

trained using the labelled training set. The images labelled with c_k are chosen as positive examples, while all the rest are used as negative examples in an one-versus-all fashion. For each test image $\mathbf{x}_i$, the distance from the hyperplane $\mathcal{V}(\mathbf{x}_i, c_k)$ is extracted by applying the SVM classifier (see Eq. 11).

$$\mathcal{V}(\mathbf{x}_i, c_k) = < \mathbf{w}_k, \mathbf{x}_i > + \mathbf{b}_k \,. \tag{11}$$

The values $\mathcal{V}(\mathbf{x}_i, c_k)$ obtained from Eq. 11 are used as prediction scores indicating the likelihood that a sample $\mathbf{x}_i$ depicts the concept c_k.

IV. Experiments

We conducted a series of experiments in order to investigate the potential of PCS as a dimensionality reduction technique in the problem of image annotation in large-scale datasets. For training the SVM classification models, the implementation of LibSVM was used [45]. The mean Average Precision (mAP) served as the evaluation metric. All the experiments were run on a 12 core Intel ® Xeon (R) CPU ES-2620 v2 @ 2.10 GHz with 128 GB memory.

For the experiments we have used the benchmarking dataset of the PASCAL VOC 2012 competition [46]. The dataset consists of 5717 training and 5823 test images collected from flickr. The images are annotated with 20 concepts (person, dog, airplane, car, chair, etc.) in a multi-label manner. Applying the feature extraction procedure (cf. Section III-A) on the above training and test datasets, we came up with a set of 11540 feature vectors each of 327680 dimensions, which require approximately 14 GB of memory using single floating format. Such excessive memory requirements are clearly intractable for many practical reasons.

A. Compressibility estimation and minimum dimensionality investigation

For calculating the compressibility of the data, we applied the methodology described in Section III-B1 on the dataset. In this way, we estimated the average number k of non-zero features for the samples. The sorted absolute values of the mean vector $\overline{\mathbf{x}}$ as this has been produced using the approach proposed in Section III-B1 are plotted in Fig. 1. From this figure, it is clear that the majority of the values are close to zero, while only a small portion of them considerably differs. The first order differences of these distances are illustrated in Fig. 2. The maximum value (peak) of the latter figure is realized at 324480. According to the proposed methodology, the number k of non-zero features of this particular sample can be approximated by counting the number of feature indices lying on the right side of this peak, which in this case is 3200. As a consequence, on average, only around 1% of the vector features have values, which are significantly larger than zero, hence the data could be considered compressible and eligible for the application of PCS. In addition, recalling from Section III-B that the number m of reduced dimensions must be on the order of $k \cdot log(\frac{n}{k})$, in our case this value is approximately 6464, which means that theoretically, at least approximately 6464 dimensions are required in order to guarantee the almost perfect reconstruction of the compressed data.

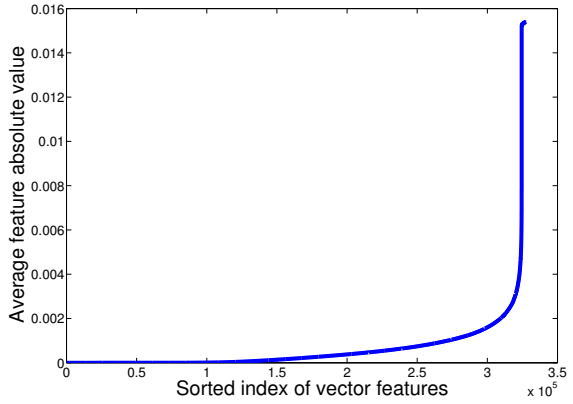

Fig. 1. Feature distances of mean vector from zero sorted in ascending order.

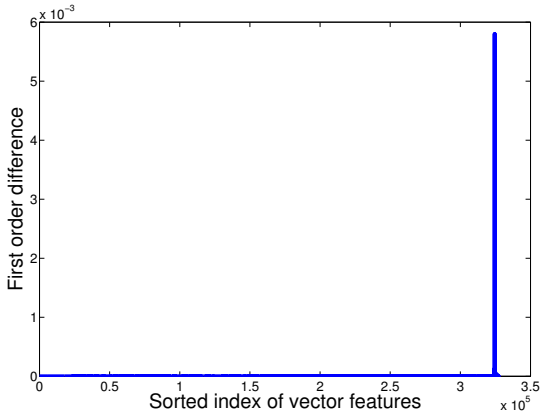

Fig. 2. First order differences of feature distances from zero.

B. Investigating the robustness of PCS as a function of the number of blocks

From the presentation of Subsection III-B3, although it has been theoretically shown that increasing b benefits the computational complexity of PCS process, the question how can b affect the classification performance using PCS is still open. In an attempt to answer this question, we set the number of reduced dimensions to 163840, i.e., half the initial dimensionality of the data and we varied the number of blocks b in the range $1, 2, 2^2, \ldots, 2^9$. For each setting, we counted the time elapsed during the dimensionality reduction process and we calculated the classification performance using PCS. The results are collectively illustrated in Fig. 3. The horizontal axis depicts the number of blocks used. The left vertical axis depicts the mAP, while the right vertical axis depicts the computational time required for reducing the dimensionality of the training data using PCS. The latter includes the time required for both the SVD calculation and the matrix multiplication processes.

Observing the "PCS computational time" curve in Fig. 3, it is clear that as the number of blocks increases, the computational complexity decreases exponentially. This result can be associated with the theoretical findings, verifying the computational analysis presented in Section III-B4. Furthermore, interestingly, the robustness of PCS using different numbers of

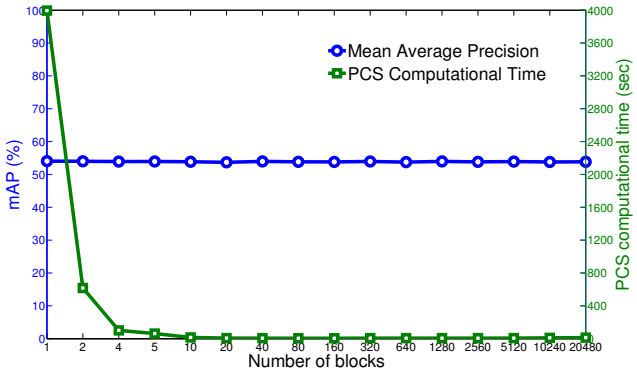

Fig. 3. Mean average precision and training computational time versus number of blocks used in PCS.

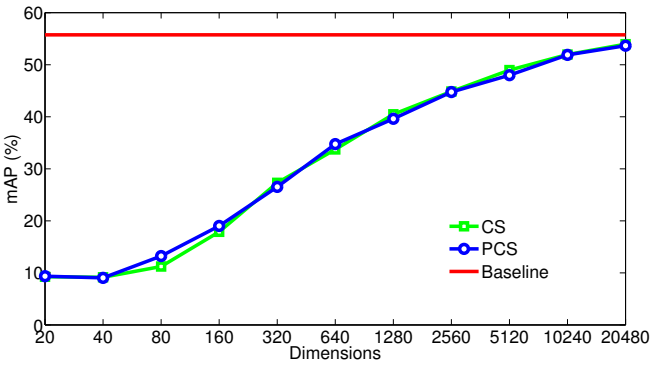

Fig. 4. Mean Average Precision using PCS in a range of different dimensions.

blocks is evident (see Fig. 3, "Mean average precision" curve). This finding allows for using the maximum possible number of blocks, since it offers the minimum computational load, while it does not deteriorate the classification performance. Closing this subsection, it is worth recalling that PCS for $b = 1$, i.e., using one block, collapses to CS (cf. Section III-B3).

C. Investigating the classification performance of PCS compared to CS

Our next concern was to investigate the classification performance of PCS, as a function of the number m of the reduced dimensions, compared to that of CS. For this purpose, a series of consecutive experiments was carried out, where at each iteration we set $m = 10 \cdot 2^p$ and we varied the exponent p in the range from 1 ($m = 20$) to 11 ($m = 20480$), while maintaining the remaining settings unchanged. In this context, using different integers for m, we employed both CS and PCS and we projected the initial high-dimensional data into the corresponding m-dimensional space. The above projected data where subsequently fed into SVM for classification. The classification performance results are illustrated in Fig. 4 for both CS (green rectangles) and PCS (blue circles). The reduced number of dimensions is depicted in the horizontal axis, while the mAP is depicted in the vertical axis. For comparison reasons, the baseline mAP using SVM on the initial data, with no dimensionality reduction is also depicted with the solid red line.

A couple of important remarks could be drawn from Fig. 4. First, it is strongly evident that the two curves corresponding to CS and PCS are almost identical, proving the equal classification performance of PCS compared to CS. This finding combined with the computational complexity advantage of PCS (cf. Section III-B4) proves its superiority over CS. In the same vein, notice that although the initial dimensionality of the data is 327680, the maximum reduced dimensionality was set to 20480. The reason is that for larger dimensions CS was infeasible due to the orthonormalisation of a very big matrix, which reinforces the advantage of PCS versus CS. Second, it is clear that the mAP for both CS and PCS is ever increasing as a function of the number of the reduced dimensions. Moreover, a closer inspection of Fig. 4, interestingly shows that the mAP starts to converge satisfactorily to the baseline in the vicinity of the theoretically required number of dimensions, i.e., 6464, which can be associated with the lower boundary analysis presented in Section III-B.

D. PCS vs random feature selection and dependance of PCS on data sparsity

A reasonable question that might arise so far is why not use a random feature selection (RFS) approach instead of calculating random linear combinations of the initial features and how does the data sparsity – and consequently compressibility – affect the performance of random projections. In an attempt to answer these questions, we propose a methodology for comparing the performance of PCS with RFS as a function of the sparsity level of the data. The methodology is based on "sparsifying" the original dataset using six different threshold values and investigating the robustness of the two above methods in the resulting artificial datasets. From a practical point of view, sparsifying the data is supposed to deteriorate the classification performance of both methods, since it leads to considerable loss of information. However, from this artificial experiment, we expect that a number of important findings regarding the effect of the data sparsity on random projections can be derived.

For each sparsity level and dimensionality we calculated the difference between the mAP's obtained by using PCS and RFS and we estimated the percentage gain in classification performance obtained by PCS over RFS. The results are illustrated in Fig. 5. The x-axis depicts the number of reduced dimensions. The y-axis depicts the percentage of mAP gain using PCS versus RFS. Each curve corresponds to a different sparsity level, as this is expressed by the percentage of zeros contained in the sparsified data. From Fig. 5, it is evident that the gain in performance using PCS instead of RFS increases as the sparsity level increases too and as the number of dimensions decreases. This improvement of performance reaches the percentage of $\simeq 80\%$ in 1280 dimensions with 97.72% sparsity, highlighting the robustness of PCS versus RFS in low dimensions on sparse data.

Intuitively, at a first glance, there is seemingly nothing special about random projections (e.g., PCS) against RFS, due to the random nature of both. However, random projections clearly take into account all the initial data features, while in contrast, selecting a number of specific features inevitably avoids the rest leading to considerable loss of information. This

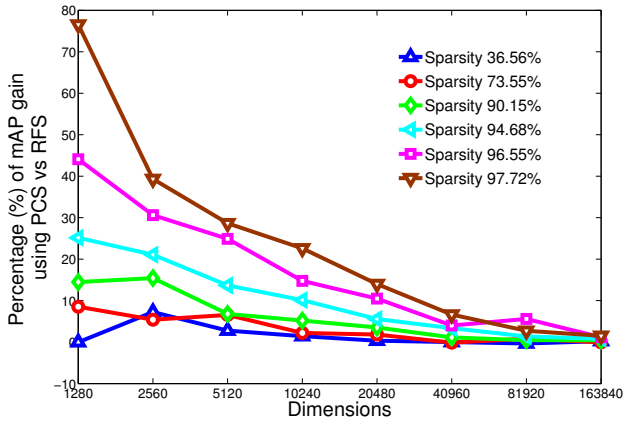

Fig. 5. Percentage (%) of mAP gain using PCS vs RFS.

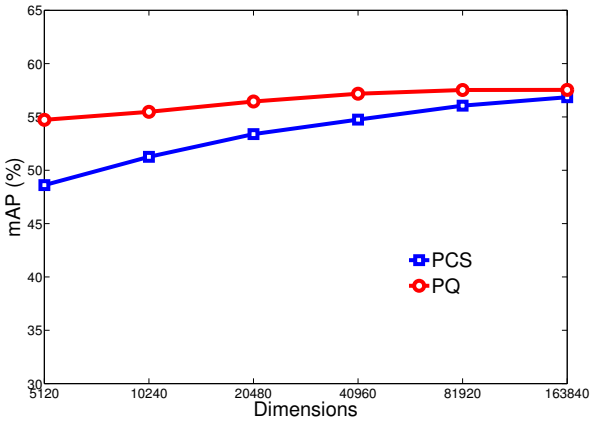

Fig. 6. Comparison between PCS and PQ.

advantage provides credibility to PCS as a smart dimensionality reduction method over other naive random feature selection schemes under the data sparsity assumption.

E. Comparison with the state-of-the-art

In this section, we compare the performance of PCS with PQ [18] and PCA [17]. Regarding PQ, the experiments were carried out using the initial 327680 dimensional data. However, since the application of PCA on the 327680 dimensional data was infeasible (cf. Section IV-E2), the experiments involving PCA were conducted using only one out of the eight spatial pyramids of the data. That is, the dimensionality of the data for this particular set of experiments was 40960. For this purpose, the results are separately presented for PQ and PCA.

1) PCS vs PQ: In this subsection, PCS is compared to PQ. As for the latter, we have used the k-means implementation presented in [18]. The theoretical complexity of this implementation is $O(IkNn)$, where k is the number of the prototype centroids trained by k-means, I is the number of iterations, n is the initial dimensionality (327680 in our case) and N is the number of training samples (5717 in our case). The dependance of the above complexity on I, k and N does not allow a direct comparison with the corresponding of PCS. However, since in the literature it has been shown that a considerably large number of all these three parameters is needed [18], it is clear that PQ might face severe computational difficulties during its application. In our experiment, we set $I = 100$ and $k = 256$, as proposed in [18]. Using these settings, in the following we provide a comparison between PCS and PQ from both a classification performance and a computational complexity point of view, where the latter has been based on the cpu times during the experiment.

The classification performance comparison between PCS and PQ is illustrated in Fig. 6. The mAP is depicted in the y-axis, while the number m of reduced dimensions is depicted in the x-axis. Based on the result of Section IV-A on the least required number of reduced dimensions, we allowed m to take values in the range from 5120 to 163840 dimensions. From Fig. 6, it is clear that although PQ outperforms PCS, the difference between the two becomes negligible at high dimensions. Fig. 7 jointly illustrates the percentage loss in performance (left vertical axis) and the corresponding speed-up (right vertical axis) using PCS versus PQ. Interestingly, from Fig. 7, the loss in performance at 163840 dimensions is only 1%, which is a difference of 0.69 units of mAP, while PCS provides a 362 times speed-up over PQ. The above loss in performance becomes approximately 11% at 5120 dimensions, while PCS is 23 times faster than PQ at the same dimensions. Such differences in computational efficiency in conjunction with the corresponding small compromise in terms of classification performance, provide a huge potential of PCS as an effective dimensionality reduction method in large-scale classification problems.

2) PCS vs PCA: The mAP results from the comparison between PCS and PCA are plotted in Fig. 8. Since in this case the initial dimensionality of the data was 40960, the number of reduced dimensions was allowed to take smaller values down to 1280. Similarly to the comparison with PQ, from Fig. 8, we observe that although in low dimensions, the superiority of PCA over PCS is evident, by increasing the number of dimensions, PCS exhibits classification performance competitive to PCA.

The robustness of PCA in low dimensions, is well justified

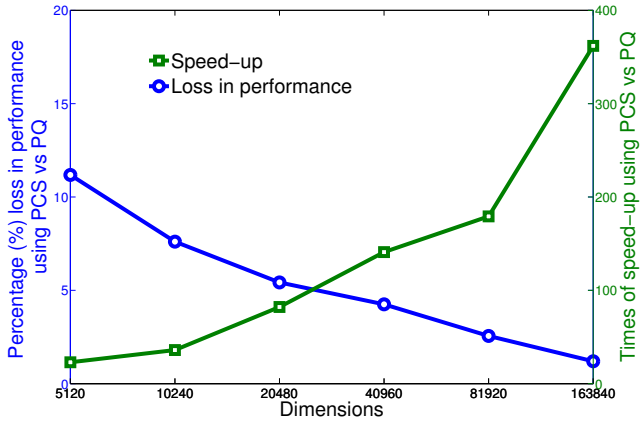

Fig. 7. Comparison between PCS and PQ.

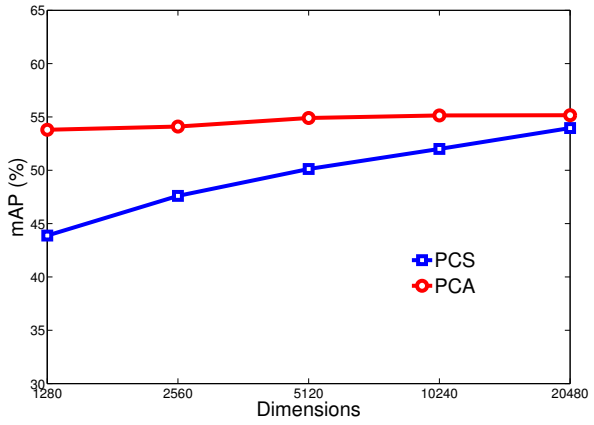

Fig. 8. Comparison between PCS and PCA.

by the fact that PCA by definition attempts to encode the data information into the least possible eigenvectors. However, this advantage comes at the cost of excessive computational and memory requirements. More specifically, regardless of the number of reduced dimensions, PCA requires the computation and eigen-analysis of the data covariance matrix, which is on the order of $n^2N + N^3$, where n is the dimensionality and N is the number of the samples. The above computational complexity in conjunction with the excessive memory requirements may render the application of PCA prohibitive. As a matter of fact, it is worth mentioning that in this particular experiment, it was infeasible to apply PCA on the data consisting of all eight spatial pyramids (327680 dimensions), since 400 GB of memory was required only for storing the covariance matrix. This is the reason why, as stated previously, we used only one out of the eight spatial pyramids.

On the other hand, CS requires the construction of an $m \times n$ random matrix, where m is much smaller than n, and the SVD of this matrix during the basis orthonormalization step. Furthermore, using PCS, the SVD is decomposed into a number of smaller SVD steps, leading to a computational load orders of magnitude less than the corresponding of PCA. As a

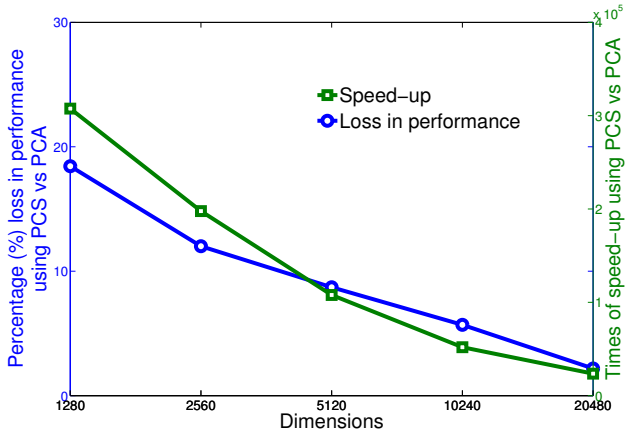

Fig. 9. Comparison between PCS and PCA.

matter of fact, a comparison analogous to Fig. 7 is illustrated in Fig. 9. Although the loss in performance using PCS instead of PCA reaches 18% at 1280 dimensions, PCS is more than 300000 times faster. Moreover, the above loss in performance reduces to only around 2% at 20480 dimensions, while the corresponding speed-up using PCS is around 24000.

Summarising the results of the comparison between PCS, PQ and PCA, the deterioration in performance of PCS when reducing the number of dimensions can be attributed to the democratic nature of the reduced dimensions, which postulates a representative number of dimensions in order to maintain the important information. It must be emphasised though that the profit in terms of computational complexity using PCS compensates for the corresponding loss in performance, which for many practical reasons can be proven negligible.

V. Conclusions

The main contribution of this paper is the novel Product Compressive Sampling method for large-scale image classification. The proposed method dramatically decreases the computational complexity of CS, while displays robustness equivalent to the typical CS. Through a comparison with baseline as well as state-of-the-art methods, it has been shown that PCS displays performance competitive to PCA and PQ provided that an appropriate number of dimensions is maintained. This fact in conjunction with the superiority of PCS in terms of computational complexity, provide credibility to PCS as an effective dimensionality reduction method in the domain of large-scale image classification.

This paper comprises a preliminary experimental study in smart sampling of big data using CS-like methods. The early results obtained are very encouraging proving the potential of the proposed method in the image annotation domain. Towards corroborating this claim, in the near future, we intend to extend this study in more datasets reinforcing our experimental results. Moreover, since very high-dimensional feature representations have recently achieved excellent results in other domains, like for example in unconstrained face recognition, in the future we intend to demonstrate our technical approach in those domains as well.

Apart from the quantitative estimation of the data sparsity, the investigation of the nature of sparsity and its impact on the effectiveness of PCS is also encompassed in our future plans. In this context, based on the interesting works presented in [9] and [47], we envisage that the sparsity structure also plays some crucial role and should be equally considered as a factor affecting PCS in classification problems. For instance, in the vector decomposition phase, there is plenty of space for investigating more sophisticated ways to divide a vector, based on sparsity patterns contained in the vector. Finally, there might be a huge potential that dictionary learning methods, which have been applied in CS could also be adapted to PCS and benefit from the computational advantages stemming from this framework.

Acknowledgment

This work was supported by the European Commission Seventh Framework Programme under Grant Agreement Number FP7-601138 PERICLES.

REFERENCES

[1] J. Bacardit and X. Llorà, "Large-scale data mining using genetics-based machine learning," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 3, no. 1, pp. 37–61, 2013.

[2] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Computer Vision–ECCV 2010*. Springer, 2010, pp. 143–156.

[3] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *CVPR, 2010 IEEE Conference on*. IEEE, 2010, pp. 3304–3311.

[4] E. J. Candès *et al.*, "Compressive sampling," in *Proceedings of the international congress of mathematicians*, vol. 3. Madrid, Spain, 2006, pp. 1433–1452.

[5] E. J. Candes and J. Romberg, "Quantitative robust uncertainty principles and optimally sparse decompositions," *Foundations of Computational Mathematics*, vol. 6, no. 2, pp. 227–254, 2006.

[6] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *Information Theory, IEEE Transactions on*, vol. 52, no. 2, pp. 489–509, 2006.

[7] D. L. Donoho, "Compressed sensing," *Information Theory, IEEE Transactions on*, vol. 52, no. 4, pp. 1289–1306, 2006.

[8] M. Aharon, M. Elad, and A. Bruckstein, "-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4311–4322, 2006.

[9] M. F. Duarte and Y. C. Eldar, "Structured compressed sensing: From theory to applications," *Signal Processing, IEEE Transactions on*, vol. 59, no. 9, pp. 4053–4085, 2011.

[10] R. Baraniuk, "Compressive sensing," *IEEE signal processing magazine*, vol. 24, no. 4, 2007.

[11] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 21–30, 2008.

[12] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 2, pp. 210–227, 2009.

[13] K. Huang and S. Aviyente, "Sparse representation for signal classification," in *NIPS*, 2006, pp. 609–616.

[14] J. Yang, A. Bouzerdoum, F. H. C. Tivive, and S. L. Phung, "Dimensionality reduction using compressed sensing and its application to a large-scale visual recognition task," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*. IEEE, 2010, pp. 1–8.

[15] C. O. S. Sorzano, J. Vargas, and A. P. Montano, "A survey of dimensionality reduction techniques," *arXiv preprint arXiv:1403.2877*, 2014.

[16] R. M. Gray and D. L. Neuhoff, "Quantization," *Information Theory, IEEE Transactions on*, vol. 44, no. 6, pp. 2325–2383, 1998.

[17] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2005.

[18] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *PAMI*, vol. 33, no. 1, pp. 117–128, 2011.

[19] S. Yu, L.-C. Tranchevent, X. Liu, W. Glanzel, J. A. Suykens, B. De Moor, and Y. Moreau, "Optimized data fusion for kernel k-means clustering," *PAMI*, vol. 34, no. 5, pp. 1031–1039, 2012.

[20] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *Image Processing, IEEE Transactions on*, vol. 15, no. 12, pp. 3736–3745, 2006.

[21] J. Mairal, G. Sapiro, and M. Elad, "Learning multiscale sparse representations for image and video restoration," DTIC Document, Tech. Rep., 2007.

[22] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *Image Processing, IEEE Transactions on*, vol. 17, no. 1, pp. 53–69, 2008.

[23] Q. Zhang and B. Li, "Discriminative k-svd for dictionary learning in face recognition," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2691–2698.

[24] H. Cai, K. Mikolajczyk, and J. Matas, "Learning linear discriminant projections for dimensionality reduction of image descriptors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 2, pp. 338–352, 2011.

[25] Q. Shi, J. Petterson, G. Dror, J. Langford, A. L. Strehl, A. J. Smola, and S. Vishwanathan, "Hash kernels," in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 496–503.

[26] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 1113–1120.

[27] Z. Zhang, J. Wang, and H. Zha, "Adaptive manifold learning," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 2, pp. 253–265, 2012.

[28] J. Sánchez and F. Perronnin, "High-dimensional signature compression for large-scale image classification," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 1665–1672.

[29] O. Bryt and M. Elad, "Compression of facial images using the k-svd algorithm," *Journal of Visual Communication and Image Representation*, vol. 19, no. 4, pp. 270–282, 2008.

[30] H. Jégou, M. Douze, and C. Schmid, "Packing bag-of-features," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 2357–2364.

[31] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," *Journal of computational and graphical statistics*, vol. 15, no. 2, pp. 265–286, 2006.

[32] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1031–1044, 2010.

[33] C. Wang, S. Yan, L. Zhang, and H.-J. Zhang, "Multi-label sparse coding for automatic image annotation," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 1643–1650.

[34] J. Huang, H. Liu, J. Shen, and S. Yan, "Towards efficient sparse coding for scalable image annotation," in *Proceedings of the 21st ACM international conference on Multimedia*. ACM, 2013, pp. 947–956.

[35] R. Baraniuk, M. A. Davenport, M. F. Duarte, and C. Hegde, "An introduction to compressive sensing," *Connexions e-textbook*, 2011.

[36] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2. Ieee, 1999, pp. 1150–1157.

[37] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[38] B. S. Manjunath, J.-R. Ohm, V. V. Vasudevan, and A. Yamada, "Color and texture descriptors," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 11, no. 6, pp. 703–715, 2001.

[39] A. Vedaldi and B. Fulkerson, "Vlfeat: An open and portable library of computer vision algorithms," in *Proceedings of the international conference on Multimedia*. ACM, 2010, pp. 1469–1472.

[40] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman, "The devil is in the details: an evaluation of recent feature encoding methods," 2011.

[41] M. A. Davenport, M. F. Duarte, Y. C. Eldar, and G. Kutyniok, "Introduction to compressed sensing," *Preprint*, vol. 93, 2011.

[42] L. Xu and Q. Liang, "Compressive sensing using singular value decomposition," in *Wireless Algorithms, Systems, and Applications*. Springer, 2010, pp. 338–342.

[43] M. Hong, Y. Yu, H. Wang, F. Liu, and S. Crozier, "Compressed sensing mri with singular value decomposition-based sparsity basis," *Physics in medicine and biology*, vol. 56, no. 19, p. 6311, 2011.

[44] V. Vapnik, *The nature of statistical learning theory*. springer, 2000.

[45] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[46] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge 2012," 2012.

[47] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde, "Model-based compressive sensing," *Information Theory, IEEE Transactions on*, vol. 56, no. 4, pp. 1982–2001, 2010.