

Linearized Kernel Dictionary Learning

Alona Golts and Michael Elad, *IEEE Fellow*

Abstract—In this paper we present a new approach of incorporating kernels into dictionary learning. The kernel K-SVD algorithm (KKSVD), which has been introduced recently, shows an improvement in classification performance, with relation to its linear counterpart K-SVD. However, this algorithm requires the storage and handling of a very large kernel matrix, which leads to high computational cost, while also limiting its use to setups with small number of training examples. We address these problems by combining two ideas: first we approximate the kernel matrix using a cleverly sampled subset of its columns using the Nyström method; secondly, as we wish to avoid using this matrix altogether, we decompose it by SVD to form new “virtual samples”, on which any linear dictionary learning can be employed. Our method, termed “Linearized Kernel Dictionary Learning” (LKDL) can be seamlessly applied as a pre-processing stage on top of any efficient off-the-shelf dictionary learning scheme, effectively “kernelizing” it. We demonstrate the effectiveness of our method on several tasks of both supervised and unsupervised classification and show the efficiency of the proposed scheme, its easy integration and performance boosting properties.

Index Terms—Dictionary Learning, Supervised Dictionary Learning, Kernel Dictionary Learning, Kernels, KSVD.

I. INTRODUCTION

THE field of sparse representations has witnessed great success in an array of applications in signal and image processing. The basic operation in sparse representations is called “sparse coding”, which involves the reconstruction of the signals of interest using a sparse set of building blocks, referred to as “atoms”. The atoms are gathered in a structure called the “dictionary”, which can be manually crafted to contain mathematical functions that are proven successful in representing signals and images, such as wavelets [1], curvelets [2] and contourlets [3]. Alternatively, it can be learned adaptively from input examples, a task referred to as “dictionary learning” (DL). The latter approach has provided state-of-the-art results in classic image processing applications, such as denoising [4], inpainting [5], demosaicing [6], compression [7], [8] and more. Popular algorithms for dictionary learning are the MOD [9] and the K-SVD [10], which generalizes K-means clustering and learns an overcomplete dictionary that best sparsifies the input data.

Although successful in signal processing applications, the K-SVD algorithm “as-is” may not be suited for machine learning tasks such as classification or regression, as its primary goal is to achieve the best reconstruction of the input data, ignoring any discriminative information such as labels or annotations. Many suggestions have been made to extend DL to deal with labeled data. The SRC method by Wright *et al.* [11] achieved impressive results in face recognition by

sparse coding each test sample over a dictionary containing the train samples from all classes, and choosing the class that presents the best reconstruction error. In [12], [13] Mairal *et al.* added a discriminative term to the DL model, and later incorporated the learning of the classifier parameters within the optimization of DL. The work reported in [14] by Zhang *et al.* was the first to incorporate the learning of the classifier parameters within the framework of the K-SVD algorithm. A similar extension has been made in [15], [16] by Jiang *et al.*, where in addition to the classifier parameters, another discriminative term for the sparse codes was added and optimized using the regular K-SVD. In [17] Yang *et al.* created an optimization function which forces both the learned dictionary and the resulting sparse coefficients to be discriminative. These algorithms and others that relate to them have been shown to be quite competitive with the best available learning algorithms, leading often times to state-of-the-art results.

In machine learning, kernels have provided a straightforward way of extending a given algorithm to deal with nonlinearities. Prominent examples of such algorithms include kernel-SVM [18], kernel-PCA (KPCA) [19] and Kernel Fisher Discriminant (KFD) [20]. Suppose the original data can be mapped to a higher dimensional “feature space”, where tasks such as classification and regression are far easier. Under the proper conditions, the “kernel trick” allows one to train a learning algorithm in the higher-dimensional feature space, without using explicitly the exact mapping. This can be done by posing the entire algorithm in terms of inner products between the input signals, and later replacing these inner-products with kernels. One fundamental problem when using the kernel trick is that one is forced to access only the inner products of signals in feature space, instead of the signals themselves. A direct consequence of this is the need to store and manipulate a large kernel matrix \mathbf{K} of dimension $N \times N$ (N being the size of the training set), which contains the modified inner products of all pairs of input examples.

In recent years, kernels have also been incorporated in the field of sparse representations, both in tasks of sparse coding [21]–[27] and dictionary learning [23], [28]–[32]. The starting point of this paper is the kernel DL method termed “Kernel K-SVD” (KKSVD) by Nguyen *et al.* The novelty in [28] is in the ability to fully pose the entire DL scheme in terms of kernels, using a unique-structured dictionary which is a multiplication of two parts. The first, a constant matrix called the “base-dictionary”, contains all of the mapped signals in feature space, and the second, called the “coefficient-dictionary”, which is actually updated during the learning process. The KKSVD suffers from the same issues arising when applying the kernel trick in general. Specifically, in large-scale datasets, where the number of input samples is of the order of thousands and

beyond, the KKSVD quickly becomes impractical, both due to runtime and in the required storage space.

While kernel sparse representation is becoming more common, the existing algorithms are still challenging as they suffer from problems mentioned above. The arena of linear DL on the other hand, has a vast selection of existing tools that are implemented efficiently, enabling learning a dictionary quite rapidly in various settings and even if the number of examples to train on goes to the Millions. Indeed, in such extreme cases, online learning becomes appealing [33], [34].

As we show hereafter, our proposed method, “Linearized Kernel Dictionary Learning” (LKDL), enjoys the benefits of both worlds. LKDL is composed of two stages: kernel matrix approximation, followed by a linearization of the training process by the creation of “virtual samples” [35]. In the first stage, we apply the Nyström method to approximate the kernel matrix \mathbf{K} , using a sub-sampled set of its columns. We explore and compare several such sub-sampling strategies, including core-sets, k-means, uniform, column-norm and diagonal sampling. Rather than using \mathbf{K} (or its approximation), we proceed with the assumption that it originates from a linear kernel, i.e. $\mathbf{K} = \mathbf{F}^T \mathbf{F}$, and thus, instead of referring to \mathbf{K} , we calculate the virtual samples \mathbf{F} , using standard eigen-decomposition. After obtaining these virtual training and test sets, we apply an efficient off-the-shelf version of linear dictionary learning and continue with a standard classification scheme. This process essentially “linearizes” the kernel matrix and combines the nonlinear kernel information within that of the virtual samples.

We evaluate the performance of LKDL in three aspects: (1) first, we assure that the added nonlinearity in the form of the virtual datasets indeed improves classification results (with relation to linear DL) and performs comparably well as the exact kernelization performed in KKSVD; (2) we demonstrate the differences in runtime between the two methods and (3) we show the easiness of integration of LKDL with *any* existing DL algorithm, including supervised DL.

We should note that a shorter version of this paper has been submitted to NIPS 2015. This paper extends over that submission in several ways: (i) it broadens the survey of past work on supervised and kernel DL; (ii) it adds the combination of the proposed scheme with supervised DL, applied to two leading algorithms; and (iii) it expands the experimental results section substantially.

This paper is organized as follows: section II provides background to classical reconstructive DL with emphasis on the K-SVD and two methods of supervised DL, all of which are used later in the experimental part as the linear foundations over which our scheme is employed. Section III discusses Nguyen’s KKSVD algorithm for kernel DL and discusses its complexity. Section IV presents the details of our proposed algorithm, LKDL, for kernel DL. This section also builds a wider picture of this field, by surveying the relevant literature of incorporating kernels into sparse coding and the dictionary-learning. Section V shows results corroborating the effectiveness of our method, and finally, section VI concludes this paper and proposes future research directions.

II. LINEAR DICTIONARY LEARNING

This section provides background on classic reconstructive DL, as well two examples of discriminative, supervised DL. The purpose of this section is to recall several key algorithms, the MOD and K-SVD, the FDDL, and the LC-KSVD, which we will kernelize in later sections.

A. Background

In sparse representations, given an input signal $\mathbf{x} \in \mathbb{R}^p$ and a “dictionary” $\mathbf{D} \in \mathbb{R}^{p \times m}$, one wishes to find a “sparse representation” vector, $\gamma \in \mathbb{R}^m$ such that $\mathbf{x} \approx \tilde{\mathbf{x}} = \mathbf{D}\gamma$. The dictionary $\mathbf{D} = [\mathbf{d}_1, \dots, \mathbf{d}_m]$ consists of “atoms” which faithfully represent the set of signals $\mathbf{x} \in \mathcal{X}$. The task of finding a signal’s sparse representation is termed “sparse coding”¹ or “atom decomposition” and can be solved using the following optimization problem:

$$\gamma = \underset{\gamma}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{D}\gamma\|_2^2 \quad \text{s.t.} \quad \|\gamma\|_0 \leq q, \quad (1)$$

where q is the number of nonzero coefficients in γ , often referred to as the “cardinality” of the representation, and the term $\|\gamma\|_0$ is the l_0 -norm which counts the number of non-zeros in γ . This problem is known to be NP-hard in general, implying that even for moderate m (number of atoms), the amount of required computations becomes prohibitive. The group of algorithms which attempt to find an approximated solution to this problem are termed “pursuit algorithms”, and they can be roughly divided into two main approaches. The first are relaxation-based methods, such as the “basis-pursuit” [36], which relaxes the norm to be l_1 instead of l_0 . The l_1 -norm still promotes sparsity while making the optimization problem solvable with polynomial-time methods. The second family of algorithms used to approximate the solution of (1) are the greedy methods, such as the “matching-pursuit” [37], which find an approximation one atom at a time. In this paper we shall mostly address the latter group of pursuit algorithms, and more specifically, the Orthogonal Matching Pursuit (OMP) [38] algorithm, which is known to be efficient and easy to implement.

B. Classic Dictionary Learning

In “dictionary learning” (DL), one attempts to compute the dictionary $\mathbf{D} \in \mathbb{R}^{p \times m}$ that best sparsifies a set of examples, serving as the input data $\mathbf{X} \in \mathbb{R}^{p \times N}$. A commonly used formulation for DL is the following optimization problem:

$$\underset{\mathbf{D}, \Gamma}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{D}\Gamma\|_F^2 \quad \text{s.t.} \quad 1 \leq i \leq N \quad \|\gamma_i\|_0 \leq q, \quad (2)$$

where $\|\cdot\|_F$ is the Frobenius norm and $\Gamma = [\gamma_1, \dots, \gamma_N] \in \mathbb{R}^{m \times N}$ is a matrix containing the sparse coefficient vectors of all the input signals. The problem of DL can be solved iteratively using a Block Coordinate Descent (BCR) approach,

¹The term “Sparse Coding” might be confusing because it is used in machine learning and brain research for describing the process we refer to as “Dictionary Learning”. In this paper we follow the terminology of signal and image processing, and thus “sparse-coding” implies the quest for the sparse solution for an approximate linear system.

of alternating between the sparse coding and dictionary update stages. Two such popular methods for DL are the MOD [9] and K-SVD [10].

In MOD [9], once the sparse coefficients in iteration t , Γ_t , are calculated using a standard pursuit algorithm, the optimization problem becomes:

$$\mathbf{D}_t = \underset{\mathbf{D}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{D}\Gamma_t\|_F^2. \quad (3)$$

This convex sub-problem leads to the analytical batch update of the dictionary using Least-Squares:

$$\mathbf{D}_t = \mathbf{X}\Gamma_t^T(\Gamma_t\Gamma_t^T)^{-1} = \mathbf{X}\Gamma_t^\dagger. \quad (4)$$

The problem with MOD is the need to compute the pseudo-inverse of the often very-large Γ . The K-SVD algorithm by Aharon *et al.* [10] proposed alleviating this and speeding up the overall convergence by updating the dictionary one atom at a time. This amounts to the use of the standard SVD decomposition of rank-1 for the update of each atom.

C. Fisher Discriminant Dictionary Learning (FDDL)

The work reported in [17] proposes an elegant way of performing discriminative DL for the purpose of classification between L classes by modifying and extending the objective function posed in (2). A fundamental feature of this method is the assumption that the dictionary is divided into L disjoint parts, each serving a different class.

Let $\mathbf{X} = [\mathbf{X}_1, \dots, \mathbf{X}_L] \in \mathbb{R}^{p \times N}$ be the input examples of the L classes, where $\mathbf{X}_i \in \mathbb{R}^{p \times n_i}$ are the examples of class i . Denote $\mathbf{D} = [\mathbf{D}_1, \dots, \mathbf{D}_L] \in \mathbb{R}^{p \times M}$ and $\Gamma = [\Gamma_1, \dots, \Gamma_L] \in \mathbb{R}^{M \times N}$ the dictionary and the corresponding sparse representations. The part $\Gamma_i \in \mathbb{R}^{M \times n_i}$ can be further decomposed as follows: $\Gamma_i = [(\Gamma_i^1)^T, \dots, (\Gamma_i^j)^T, \dots, (\Gamma_i^L)^T]^T$, where $\Gamma_i^j \in \mathbb{R}^{m_j \times n_i}$ are the coefficients of the samples $\mathbf{X}_i \in \mathbb{R}^{p \times n_i}$ over the dictionary $\mathbf{D}_j \in \mathbb{R}^{p \times m_j}$. Armed with the above notations, we now turn to describe the objective function proposed in [14] for the discriminative DL task. This objective is composed of two parts. The first is based on the following expression:

$$r(\mathbf{X}_i, \mathbf{D}, \Gamma_i) = \|\mathbf{X}_i - \mathbf{D}\Gamma_i\|_F^2 + \|\mathbf{X}_i - \mathbf{D}_i\Gamma_i^i\|_F^2 + \sum_{\substack{j=1 \\ j \neq i}}^L \|\mathbf{D}_j\Gamma_i^j\|_F^2 \quad (5)$$

The first term demands a good representation of the i -th class samples using the whole dictionary, and the second term further demands a good representation for these examples using their own class' sub-dictionary. The third term is of different nature, forcing the i -th class examples to minimize their reliance on the other sub-dictionaries. Naturally, the overall penalty function will sum the expression in (5) for all the classes i .

We now turn to describe the second term in the objective function, which relies on the Fisher Discriminant Criterion [39]. We define two scatter expressions, both applied to the representations. The first, $S_W(\Gamma)$ computes the within class

spread, while the second, $S_B(\Gamma)$ computes the scatter between the classes:

$$S_W(\Gamma) = \sum_{i=1}^L \sum_{\gamma_k \in \Gamma_i} (\gamma_k - \mu_i)(\gamma_k - \mu_i)^T \quad (6)$$

$$S_B(\Gamma) = \sum_{i=1}^L n_i(\mu_i - \mu)(\mu_i - \mu)^T,$$

and μ, μ_i are the mean vectors of the learned sparse coefficient vectors, Γ and Γ_i correspondingly. Naturally, we aim to minimize the first while maximizing the second.

The final FDDL model is defined by the following optimization expression:

$$J_{(\mathbf{D}, \Gamma)} = \underset{(\mathbf{D}, \Gamma)}{\operatorname{argmin}} \left\{ \sum_{i=1}^L r(\mathbf{X}_i, \mathbf{D}, \Gamma_i) + \lambda_1 \|\Gamma\|_1 + \lambda_2 [\operatorname{tr}(S_W(\Gamma) - S_B(\Gamma)) + \eta \|\Gamma\|_F^2] \right\}. \quad (7)$$

The term $\|\Gamma\|_F^2$ serves as a regularization that ensures the convexity of (6).

The detailed optimization scheme of this rather complex expression is described in [17], along with two classification schemes, a global and a local one, depending on the size of the input dataset.

D. Label Consistent KSVD (LC-KSVD)

In [15], [16], an alternative discriminative DL approach is introduced, in which the learning of the dictionary, along with the parameters of the classifier itself, is performed simultaneously, leading to the scheme termed "Label-Consistent K-SVD" (LC-KSVD). These elements are combined in one optimization objective, which is handled using the standard K-SVD algorithm.

In order to improve the performance of a linear classifier, an extra term is added to the reconstructive DL optimization function:

$$\underset{\mathbf{D}, \mathbf{T}, \Gamma}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{D}\Gamma\|_F^2 + \alpha \|\mathbf{Q} - \mathbf{T}\Gamma\|_F^2 \quad s.t. \forall i, \|\gamma_i\|_0 \leq q. \quad (8)$$

The second term encourages the sparse coefficients to be discriminative. More specifically, the matrix $\mathbf{Q} = [\mathbf{q}_1, \dots, \mathbf{q}_N] \in \mathbb{R}^{m \times N}$ stands for the "ideal" sparse-coefficient matrix for discrimination, where \mathbf{q}_i is a binary vector encoding the assignment of each example to its destination atoms. The matrix $\mathbf{T} \in \mathbb{R}^{m \times m}$ transforms the sparse codes Γ to their idealized versions in \mathbf{Q} . This term thus promotes identical sparse codes for input signals from the same class and orthogonal sparse codes for signals from different classes.

In addition to the discriminative term added above, the authors in [15] propose learning the linear classifier within the framework of the DL. A linear predictive classifier is used of the form: $f(\gamma, \Theta) = \Theta\gamma$, where $\Theta \in \mathbb{R}^{L \times m}$. The overall objective function suggested is:

$$\underset{\mathbf{D}, \Theta, \mathbf{T}, \Gamma}{\operatorname{argmin}} \left\{ \|\mathbf{X} - \mathbf{D}\Gamma\|_F^2 + \alpha \|\mathbf{Q} - \mathbf{T}\Gamma\|_F^2 + \beta \|\mathbf{H} - \Theta\Gamma\|_F^2 \right\}, \quad s.t. \quad \forall i, \|\gamma_i\|_0 \leq q, \quad (9)$$

where the classification error is represented by the term $\|\mathbf{H} - \Theta\Gamma\|_F^2$, Θ contains the classifier parameters, $\mathbf{H} =$

$[\mathbf{h}_1, \dots, \mathbf{h}_N] \in \mathbb{R}^{L \times N}$ is the label matrix of all input examples, in which the vector $\mathbf{h}_i = [0, 0, \dots, 0, 1, 0, \dots, 0]^T$ contains only zeros apart from the index corresponding to the class of the example. The optimization function in (9) can also be written as follows:

$$\operatorname{argmin}_{\mathbf{D}_{new}, \mathbf{\Gamma}} \|\mathbf{X}_{new} - \mathbf{D}_{new} \mathbf{\Gamma}\|_F^2, \quad s.t. \quad \forall i, \quad \|\gamma_i\|_0 \leq q, \quad (10)$$

where $\mathbf{X}_{new} = (\mathbf{X}^T, \sqrt{\alpha} \mathbf{Q}^T, \sqrt{\beta} \mathbf{H}^T)^T \in \mathbb{R}^{(p+m+L) \times N}$ and $\mathbf{D}_{new} = (\mathbf{D}^T, \sqrt{\alpha} \mathbf{T}^T, \sqrt{\beta} \mathbf{\Theta}^T)^T \in \mathbb{R}^{(p+m+L) \times m}$. The unified columns in \mathbf{D}_{new} are all normalized to unit l_2 norm. The optimization objective in (10) can be solved using standard DL algorithms, such as K-SVD.

The authors propose two cases of LC-KSVD: LC-KSVD2, in which the parameters of the classifier are learned along with the dictionary, as shown in (9) and the second, LC-KSVD1, in which they are calculated separately by: $\mathbf{\Theta} = (\mathbf{\Gamma} \mathbf{\Gamma}^T + \tau_2 \mathbf{I})^{-1} \mathbf{\Gamma} \mathbf{H}^T$. More details on these expressions and the numerical scheme for minimizing the objective function can be found in [15], [16]. A new sample \mathbf{x} is classified by first sparse coding over the dictionary \mathbf{D} , and then, applying the classifier $\mathbf{\Theta}$ to estimate the label j .

III. KERNEL DICTIONARY LEARNING

This section focuses on kernel sparse representations, with emphasis of the kernel-KSVD method by Nguyen *et al.*, which we will compare with later on this paper.

A. Kernels - The Basics

In machine learning, it is well-known that a non-linear mapping of the signal of interest to higher dimension may improve its discriminability in tasks such as classification. Let $\mathbf{x} \in \mathcal{X}$ be a signal in input space, which is embedded to a higher dimensional space \mathcal{F} using the mapping $\Phi, \mathbf{x} \in \mathbb{R}^p \rightarrow \Phi(\mathbf{x}) \in \mathbb{R}^P$ ($P \gg p$ and it might even be infinite). The space in which this new signal $\Phi(\mathbf{x})$ lies is called the “feature space”. The next step in machine learning algorithms, in particular in classification, would be learning a classifier based on the mapped input signals and labels. This task can be prohibitive if tackled directly. A way around this hurdle is the “kernel trick” [40], [41], which allows computing inner products between pairs of signals in the feature space, using a simple nonlinear function operating on the two signals in input space:

$$\kappa(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle = \Phi(\mathbf{x})^T \Phi(\mathbf{x}'), \quad (11)$$

where κ is the “kernel”. This relation holds true for positive-semi-definite (p.s.d) and Mercer kernels [18]. Thus, suppose that the learning algorithm can be fully posed in terms of inner products. In such a case, one can achieve a “kernelized” version by swapping the inner products with the kernel function, without ever operating in the feature space.

In case there are N input signals $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N] \in \mathbb{R}^{p \times N}$, the “kernel matrix” $\mathbf{K} \in \mathbb{R}^{N \times N}$ holds the kernel values of all pairs of input signals:

$$\mathbf{K}_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle, \quad \forall i, j = 1..N. \quad (12)$$

An inherent constraint in kernel algorithms is the fact that the solution vectors, for example the principal components in KPCA, are expansions of the mapped signals in feature space:

$$\mathbf{v} = \sum_{i=1}^N \alpha_i \Phi(\mathbf{x}_i). \quad (13)$$

The subspace in which the possible solutions lie, can be viewed as an N dimensional surface residing in \mathcal{F} [42]. Motivated by the inability to directly approach the mapped signals in feature space, researchers have suggested embedding the N dimensional surface to a finite Euclidean subspace, where all geometrical properties, such as distances and angles between pairs of $\Phi(\mathbf{x}_i)$'s, are preserved [43]. The embedding is called the “kernel empirical map” and the resulting subspace is referred to as the “empirical feature subspace”. One way to embed a given signal \mathbf{x} to the empirical feature space is by calculating kernel values originating from inner products with all input training examples:

$$\mathbf{x} \rightarrow [\kappa(\mathbf{x}, \mathbf{x}_1), \dots, \kappa(\mathbf{x}, \mathbf{x}_N)]^T. \quad (14)$$

B. Kernel Dictionary Learning

A straightforward way to kernelize dictionary learning would be exchanging all the signals (and dictionary atoms) with their respective representations in feature space: $\mathbf{x} \rightarrow \Phi(\mathbf{x})$, $\mathbf{d} \rightarrow \Phi(\mathbf{d})$ and rephrasing the algorithm such that it contains solely inner products between pairs of these ingredients. A difficulty with this approach is that during the learning process, the dictionary atoms are in feature space. As there is no exact reverse mapping from the updated inner products to their corresponding signals in input space, there is no direct way of accessing the updated dictionary atoms, as practiced in linear DL.

In order to solve this problem, the authors in [28] suggest decomposing the dictionary in feature space into: $\Phi(\mathbf{D}) = \Phi(\mathbf{X}) \mathbf{A}$, where $\Phi(\mathbf{X})$ is the constant part, called the “base-dictionary”, which consists of all mapped input signals, and \mathbf{A} is the only part updated during the learning, called the “coefficient-dictionary”. Just like in the case of the KPCA [19], the obtained dictionary is limited to an N -dimensional manifold in the feature space.

The kernel dictionary learning can now be formulated as the following optimization problem:

$$\operatorname{argmin}_{\mathbf{A}, \mathbf{\Gamma}} \|\Phi(\mathbf{X}) - \Phi(\mathbf{X}) \mathbf{A} \mathbf{\Gamma}\|_F^2 \quad s.t. \quad \forall i = 1..N \quad \|\gamma_i\|_0 \leq q. \quad (15)$$

Similarly to linear DL, this optimization problem can be solved iteratively by first performing sparse coding with a fixed dictionary \mathbf{A} , then updating the dictionary according to the computed sparse representations $\mathbf{\Gamma}$, and so on, until convergence is reached. The kernelized equivalent of sparse coding is given by:

$$\operatorname{argmin}_{\gamma} \|\Phi(\mathbf{z}) - \Phi(\mathbf{X}) \mathbf{A} \gamma\|_2^2 \quad s.t. \quad \|\gamma\|_0 \leq q, \quad (16)$$

where \mathbf{z} is the input signal. As mentioned earlier, the sparse coding algorithm we focus on in this paper, as well as in Nguyen's KKSVD [28], is the OMP [38] and its kernel

version, KOMP [28]. Table I presents two of the main stages in the OMP algorithm, which are the Atom-Selection (AS) and Least-Squares (LS) stages, and their kernelized version. As can be seen, these stages can be completely represented using the coefficient dictionary \mathbf{A} , the sparse representation vector γ and the kernel functions $\mathbf{K}(\mathbf{X}, \mathbf{X}) \in \mathbb{R}^{N \times N}$ and $\mathbf{K}(\mathbf{z}, \mathbf{X}) = [\kappa(\mathbf{z}, \mathbf{x}_1), \dots, \kappa(\mathbf{z}, \mathbf{x}_N)] \in \mathbb{R}^{1 \times N}$.

The dictionary update stage, can also be kernelized. In the MOD algorithm [9], the update of \mathbf{A} in iteration $t+1$ is given by: $\mathbf{A}_{t+1} = \mathbf{\Gamma}_t^T (\mathbf{\Gamma}_t \mathbf{\Gamma}_t^T)^{-1} = \mathbf{\Gamma}_t^+$, being the solution to: $\arg \min_{\mathbf{A}} \|\Phi(\mathbf{X}) - \Phi(\mathbf{X}) \mathbf{A} \mathbf{\Gamma}_t^T\|_F^2$. A similar update can be derived for the K-SVD algorithm, as described in depth in [28], [30].

C. Difficulties in KDL

There are a few difficulties that arise when dealing with kernels, and specifically in kernel dictionary learning. In the input space, a signal $\mathbf{x} \in \mathbb{R}^p$ can be described using its own p features, while in feature space it is described by its relationship with all of the other N input signals. The runtime and memory complexity of a kernel learning algorithm changes accordingly and depends on the number of input signals, instead of on the dimension of the signals. This observation is also true for Nguyen's KDL where the kernel matrix \mathbf{K} is used during the sparse coding and dictionary update stages, and must be stored in full. In applications where the number of input samples is large, this dependency on the kernel matrix becomes prohibitive. In table I, one can see the complexity of the main stages in the KOMP algorithm and compare it to the linear OMP version. It is clear that both the atom-selection and the least-squares stages are governed quadratically on the size of the input dataset.

Another inherent difficulty in kernel methods is the need to tailor each algorithm such that it is formulated solely through inner products. This constraint creates complex and cumbersome expressions and is not always possible, as some steps in the algorithm may contain a mixture of the signals and their mapped version.

IV. THE PROPOSED ALGORITHM

Section II and III gave some background to the task we address in this paper. We saw that kernelization of the DL task can be beneficial, but unfortunately, we also identified key difficulties this process is accompanied by. In this work we aim to propose a systematic and simple path for kernelizing existing dictionary learning algorithms, in a way that will avoid the problems mentioned above. More specifically, we desire to be able to kernelize any existing DL algorithm, be it unsupervised or supervised, and do so while being able to work on massive training sets without the need to compute, store, or manipulate the kernel matrix \mathbf{K} . In this section we outline such a solution, by carefully describing its key ingredients.

A. Kernel matrix approximation

Let $\mathbf{X} \in \mathbb{R}^{p \times N}$ be the input signals and $\mathbf{K} \in \mathbb{R}^{N \times N}$ their corresponding kernel matrix. We shall further assume that \mathbf{K}

is of rank $r \leq N$. As long as the kernel satisfies Mercer's conditions of positive-semi-definiteness it can be written as an inner product between mapped signals in feature space: $\mathbf{K}_{i,j} = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$. Assume, for the sake of the discussion here, that the kernel function applies a simple inner product, i.e.: $\mathbf{K}_{i,j} = \langle \mathbf{f}_i, \mathbf{f}_j \rangle = \mathbf{f}_i^T \mathbf{f}_j$, where $\mathbf{f}_i, \mathbf{f}_j$ are the feature vectors corresponding to \mathbf{x}_i and \mathbf{x}_j , respectively. Thus, the kernel matrix would have the form: $\mathbf{K} = \mathbf{F}^T \mathbf{F} = \Phi(\mathbf{X})^T \Phi(\mathbf{X})$, where \mathbf{F} is a matrix of size $r \times N$ (r is the feature-space dimension, and we have assumed that it is smaller than N). One can refer to the vectors $\{\mathbf{f}_i\}_{i=1}^N$ in \mathbf{F} as "Virtual Samples" [35]. This way, instead of learning using the kernel matrix \mathbf{K} , one could work on these virtual samples directly using a linear learning algorithm, leading to the same outcome. In the following, we will leverage on this insight.

The kernel matrix is generally symmetric and positive-semi-definite, and as such can be decomposed using eigen-decomposition as follows: $\mathbf{K} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T$, where $\mathbf{\Lambda} \in \mathbb{R}^{r \times r}$ is a diagonal matrix containing all of the nonzero eigenvalues of \mathbf{K} in descending order and $\mathbf{U} \in \mathbb{R}^{N \times r}$ contains the matching orthonormal eigenvectors. An approximation of the virtual samples can be achieved by:

$$\mathbf{F} = \mathbf{\Lambda}^{1/2} \mathbf{U}^T = \mathbf{\Lambda}^{-1/2} \mathbf{U}^T \mathbf{K}. \quad (17)$$

The virtual samples can be viewed as a mapping of the original input signals to an r -dimensional empirical feature space.

$$\mathbf{x} \rightarrow \mathbf{\Lambda}^{-1/2} \mathbf{U}^T (\kappa(\mathbf{x}, \mathbf{x}_1), \kappa(\mathbf{x}, \mathbf{x}_2), \dots, \kappa(\mathbf{x}, \mathbf{x}_N))^T. \quad (18)$$

An approximated kernel empirical map of dimension $k \leq r$ can also be obtained by considering only the top k eigenvalues and corresponding eigenvectors $\rightarrow \mathbf{F}_k = (\mathbf{\Lambda}_k)^{1/2} (\mathbf{U}_k)^T$.

This "linearization" is the mediator between kernel DL which is obligated to store and manipulate the kernel matrix \mathbf{K} , and linear DL that can deal with very large datasets. The decomposition of the matrix \mathbf{K} to its eigenvalues and eigenvectors is a demanding task in itself, both in time $O(N^2 k)$ and in space $O(N^2)$. Next we will show how a good approximation of the matrix \mathbf{K} can be constructed with only a subset of its columns, using the popular Nyström method.

B. Nyström method

A common necessity in many algorithms in signal processing and machine learning is deriving a relatively accurate and efficient approximation of a large matrix. An attractive method that has gained popularity in recent years is the Nyström method [44]–[46], which generates a low-rank approximation using a subset of the input data. The original Nyström method, first introduced by Williams and Seeger [44], proposed using uniform sampling without replacement.

Let $\mathbf{K} \in \mathbb{R}^{N \times N}$ be a symmetric positive semi-definite matrix, and in particular for the discussion here, a kernel matrix. Suppose $c \leq N$ columns from the matrix \mathbf{K} are sampled uniformly without replacement to form the reduced matrix $\mathbf{C} \in \mathbb{R}^{N \times c}$. Without loss of generality, the matrices \mathbf{C} and \mathbf{K} can be permuted as follows:

$$\mathbf{C} = \begin{bmatrix} \mathbf{W} \\ \mathbf{S} \end{bmatrix} \quad \text{and} \quad \mathbf{K} = \begin{bmatrix} \mathbf{W} & \mathbf{S}^T \\ \mathbf{S} & \mathbf{B} \end{bmatrix}, \quad (19)$$

TABLE I

COMPLEXITY OF THE ATOM SELECTION (AS) AND THE LEAST SQUARE (LS) STAGES IN LINEAR AND KERNEL-OMP. \mathbf{I}_S IS THE CURRENT SUPPORT VECTOR AND $|\mathbf{I}_S|$ ITS LENGTH, \mathbf{D}_S , \mathbf{A}_S AND γ_S ARE SUB-MATRICES OF \mathbf{D} , \mathbf{A} , AND γ , RESPECTIVELY, CORRESPONDING TO \mathbf{I}_S . \mathbf{r}_t IS THE RESIDUAL.

	Term	Complexity
OMP-AS	$\langle \mathbf{r}_t, \mathbf{d}_j \rangle = \langle \mathbf{z} - \mathbf{D}_S \gamma_S, \mathbf{d}_j \rangle = \mathbf{z}^T \mathbf{d}_j - \gamma_S^T \mathbf{D}_S^T \mathbf{d}_j$	$O(p \mathbf{I}_S + p)$
KOMP-AS [28]	$\mathbf{K}(\mathbf{z}, \mathbf{X}) \mathbf{a}_j - \gamma_S^T \mathbf{A}_S^T \mathbf{K}(\mathbf{X}, \mathbf{X}) \mathbf{a}_j$	$O(N^2 + \mathbf{I}_S N + N)$
OMP-LS	$\gamma_S = (\mathbf{D}_S^T \mathbf{D}_S)^{-1} \mathbf{D}_S^T \mathbf{z}$	$O(p \mathbf{I}_S ^2 + p \mathbf{I}_S + \mathbf{I}_S ^3)$
KOMP-LS [28]	$\gamma_S = [\mathbf{A}_S^T \mathbf{K}(\mathbf{X}, \mathbf{X}) \mathbf{A}_S]^{-1} (\mathbf{K}(\mathbf{z}, \mathbf{X}) \mathbf{A}_S)^T$	$O(N^2 \mathbf{I}_S + N \mathbf{I}_S + \mathbf{I}_S ^3)$

where $\mathbf{W} \in \mathbb{R}^{c \times c}$ is the kernel matrix of the intersection of the chosen c columns with c rows, $\mathbf{B} \in \mathbb{R}^{(N-c) \times (N-c)}$ is the kernel matrix composed of the $N - c$ remaining rows and columns, and $\mathbf{S} \in \mathbb{R}^{(N-c) \times c}$, is a mixture of both. The Nyström method uses both \mathbf{C} and \mathbf{W} to construct an approximation of the matrix \mathbf{K} as follows:

$$\mathbf{K} \approx \mathbf{C} \mathbf{W}^\dagger \mathbf{C}^T, \quad (20)$$

where $(\cdot)^\dagger$ denotes the pseudo-inverse. The symmetric matrix \mathbf{W} can also be posed in terms of eigenvalues and eigenvectors: $\mathbf{W} = \mathbf{V} \Sigma \mathbf{V}^T$, where Σ is a diagonal matrix containing the eigenvalues of \mathbf{W} in descending order and \mathbf{V} contains the matching orthonormal eigenvectors. The pseudo-inverse of \mathbf{W} is given by $\mathbf{W}^\dagger = \mathbf{V} \Sigma^\dagger \mathbf{V}^T$. The expression of $(\mathbf{W}^\dagger)^{1/2}$ can be similarly derived: $(\mathbf{W}^\dagger)^{1/2} = (\Sigma^\dagger)^{1/2} \mathbf{V}^T$.

We can represent \mathbf{K} as before, using linear inner-products of the virtual samples, and plug in Nyström's approximation:

$$\mathbf{K} = \mathbf{F}^T \mathbf{F} = \mathbf{C} \mathbf{W}^\dagger \mathbf{C}^T = \mathbf{C} \mathbf{V} \Sigma^\dagger \mathbf{V}^T \mathbf{C}^T, \quad (21)$$

and derive the final expression of the virtual samples by:

$$\mathbf{F} = (\Sigma^\dagger)^{1/2} \mathbf{V}^T \mathbf{C}^T. \quad (22)$$

The rank- k ($k \leq c$) approximation can similarly be derived:

$$\mathbf{F}_k = \left(\Sigma_k^\dagger \right)^{1/2} \mathbf{V}_k^T \mathbf{C}_k^T, \quad (23)$$

where $\Sigma_k = \text{diag}(\sigma_1, \dots, \sigma_k) \in \mathbb{R}^{k \times k}$ contains the k largest eigenvalues of \mathbf{W} and $\mathbf{V}_k \in \mathbb{R}^{c \times k}$, the corresponding orthonormal eigenvectors.

After performing the Nyström approximation, the space complexity of kernel DL reduces from $O(N^2)$ to $O(Nc)$, the size of the matrix \mathbf{C} , which is used during the computation of the virtual samples. The time complexity of the Nyström method is $O(Nck + c^2k)$, where $O(Nck)$ represents the multiplication of $\mathbf{V}_k^T \mathbf{C}_k^T$ and $O(c^2k)$ stands for the eigenvalue decomposition (and inversion) of the reduced matrix \mathbf{W}_k .

Note that the process of computing the virtual samples may seem inefficient, but it is performed only once, after which the complexity of the DL is dictated by the chosen algorithm, and not by the "kernelization". In addition, in scenarios where the number of input examples is very large, the ratio c/N in Nyström's method can be reduced greatly, i.e. $c \ll N$, making the approximation even less dominant in terms of runtime and memory, while retaining almost the same accuracy.

C. Sampling Techniques

Since the Nyström method creates an approximation of a large symmetric matrix based on a subset of its columns, the chosen sampling scheme plays an important part. The basic method proposed originally by Williams and Seeger was uniform sampling without replacement [44]. The columns of the Gram matrix can be alternatively sampled from a nonuniform distribution. Two such examples of nonuniform sampling include "column-norm sampling" [47], where the weight of the i th column \mathbf{k}^i is its l_2 norm: $p_i = \|\mathbf{k}^i\|^2 / \|\mathbf{K}\|_F^2$, and "diagonal sampling" [48] where the weight is proportional to the corresponding diagonal element: $p_i = \mathbf{K}_{ii}^2 / \sum_{i=1}^N \mathbf{K}_{ii}^2$. These methods can be made more sophisticated but require additional complexity: $O(N)$ in time and space for diagonal sampling and $O(N^2)$ for column-norm sampling. A comprehensive theoretical and empirical comparison of these three methods is provided in [49].

In [50], Zhang *et al.* suggested an alternative approach of selecting a few "representative" columns in \mathbf{K} by first performing K-means clustering, then computing the reduced matrix \mathbf{C} based on these so-called "cluster centers". Denote by \mathbf{X}_R the resulting c cluster centers, created from the original data \mathbf{X} . The computation of the kernel matrices \mathbf{C} and \mathbf{W} would be: $\mathbf{C} = \mathbf{K}(\mathbf{X}, \mathbf{X}_R)$ and $\mathbf{W} = \mathbf{K}(\mathbf{X}_R, \mathbf{X}_R)$. Zhang *et al.* also show that the combination of k-means clustering with the Nyström method minimizes the approximation error.

Another appealing sampling technique has been suggested in the context of coresets [51]. The idea is to sample the given data by emphasizing unique samples that are ill-represented by the others. In the context of our problem, we sample c signals from \mathbf{X} according to the following distribution: $p_i = \text{err}(\mathbf{x}_i, \boldsymbol{\mu}) / \sum_{\mathbf{x}_i \in \mathbf{X}} \text{err}(\mathbf{x}_i, \boldsymbol{\mu})$, where $\text{err}(\mathbf{x}_i, \boldsymbol{\mu}) = \|\mathbf{x}_i - \boldsymbol{\mu}\|_2^2$ is the representation error of the signal \mathbf{x}_i , corresponding to the mean of all training signals $\boldsymbol{\mu} = (1/N) \sum_{i=1}^N \mathbf{x}_i$.

D. Linearized Kernel Dictionary Learning (LKDL)

Let $\{\mathbf{x}_i, y_i\}_{i=1}^N$ be a labeled² training set, arranged as a structure in L categories: $\mathbf{X}_{\text{train}} = [\mathbf{X}_1, \dots, \mathbf{X}_L] \in \mathbb{R}^{p \times N}$, where \mathbf{X}_i contains the training samples that belong to the i th class and $N = \sum_{i=1}^L n_i$. Our process of kernel dictionary learning is divided in two parts: the first, a pre-processing stage that creates new virtual training and test samples, followed by a second stage of applying a standard DL. This whole process is termed "Linearized Kernel Dictionary Learning" (LKDL).

²We consider here the case of labeled data, but the labels can be omitted, thus reducing to the simple representative DL format.

The pre-processing stage is shown in algorithm 1. First, the initial training set \mathbf{X}_{train} is sampled using one of the techniques mentioned in section IV-C, creating the reduced set $\mathbf{X}_R = [\mathbf{x}_{R_1}, \dots, \mathbf{x}_{R_c}] \in \mathbb{R}^{p \times c}$. Then the matrix $\mathbf{C} \in \mathbb{R}^{N \times c}$ in Nystrom's method is calculated by simply applying the chosen kernel on each and every pair of columns in \mathbf{X}_{train} and \mathbf{X}_R . Next, the reduced matrix $\mathbf{W} \in \mathbb{R}^{c \times c}$ is both calculated and later on inverted using rank- k eigen-decomposition. Finally the virtual training samples $\mathbf{F}_{train} \in \mathbb{R}^{k \times N}$ are calculated using equation (23). The Nystrom method permits approximating a new test vector \mathbf{f}_{test} using equation (18), by using the mapping already calculated based on the training set, and multiplying by the joint kernel vector of the sampled set \mathbf{X}_R and the current test sample: $\mathbf{K}(\mathbf{X}_R, \mathbf{x}_{test})$:

$$\mathbf{f}_{test} = \left(\Sigma_k^\dagger \right)^{1/2} \mathbf{V}_k^T [\kappa(\mathbf{x}_{R_1}, \mathbf{x}_{test}), \dots, \kappa(\mathbf{x}_{R_c}, \mathbf{x}_{test})]^T. \quad (24)$$

Once the training and test sets are represented as virtual samples: \mathbf{F}_{train} and \mathbf{F}_{test} , any linear DL-based classification method can be implemented. In the context of classification we follow Nguyen's "distributive" approach [30] of learning L separate dictionaries $[\mathbf{D}_1, \dots, \mathbf{D}_L]$ per each class, then classifying each test sample by first computing its sparse coefficient vector over each of the dictionaries $\{\mathbf{D}_i\}_{i=1}^L$, and finally choosing the class corresponding to the smallest reconstruction error:

$$r_i = \|\mathbf{f}_{test} - \mathbf{D}_i \gamma_i\|^2, \quad \forall i = 1..L. \quad (25)$$

Algorithm 1 LKDL Pre-Processing

- 1: **Input:** $\mathbf{X}_{train} = [\mathbf{X}_1, \dots, \mathbf{X}_L]$, \mathbf{X}_{test} , the kernel κ , $smpl_method$, c , k
 - 2: $\mathbf{X}_R = sub_sample(\mathbf{X}_{train}, smpl_method, c)$
 - 3: Compute $\mathbf{C}_{train} = \mathbf{K}(\mathbf{X}_{train}, \mathbf{X}_R)$
 - 4: Compute $\mathbf{W} = \mathbf{K}(\mathbf{X}_R, \mathbf{X}_R)$
 - 5: Approximate \mathbf{W}_k using k largest eigenvalues and eigenvectors $\mathbf{W}_k = \mathbf{V}_k \Sigma_k \mathbf{V}_k^T$
 - 6: Compute virtual train set $\mathbf{F}_{train} = \left(\Sigma_k^\dagger \right)^{1/2} \mathbf{V}_k^T \mathbf{C}_{train}^T$
 - 7: Compute $\mathbf{C}_{test} = \mathbf{K}(\mathbf{X}_{test}, \mathbf{X}_R)$
 - 8: Compute virtual test set $\mathbf{F}_{test} = \left(\Sigma_k^\dagger \right)^{1/2} \mathbf{V}_k^T \mathbf{C}_{test}^T$
 - 9: **Output:** $\mathbf{F}_{train} = [\mathbf{F}_1, \dots, \mathbf{F}_L]$, \mathbf{F}_{test}
-

E. Relation to Past Work

The existing works on kernel sparse representations can be roughly divided to two categories. The first corresponds to 'analytical' methods that operate solely in the feature domain and use the kernel trick to find an analytical solution, be it sparse coding or dictionary update [23], [24], [28], [31]. The other category refers to 'empirical' or 'approximal' methods that operate in the input space, while making some approximation or assumption regarding the mapped signals in feature space, in order to alleviate some of the constraints when working with kernels [21], [22], [25]. Naturally, our work belongs to the second group of contributions.

In 2002, Bengio *et al.* [21] kernelized the matching pursuit algorithm by using the kernel empirical map of the input training examples as dictionary atoms. By referring to the kernel empirical map Φ_e instead of the actual mapped signals in \mathcal{F} , the authors could perform standard linear matching pursuit without having to rewrite the algorithm in terms of inner products. In this case, the constraint of a p.s.d kernel was no longer mandatory. In 2005 [22], a similar concept of embedding the signals to a kernel empirical map was used to kernelize the basis pursuit algorithm. This approach of working in the input domain with an approximation of the kernel feature space is very similar to ours and can be described by the following embedding, evaluated over the entire training dataset $\{\mathbf{x}_i\}_{i=1}^N$:

$$\mathbf{x} \rightarrow \Phi_e(\mathbf{x}) = [\kappa(\mathbf{x}_1, \mathbf{x}), \dots, \kappa(\mathbf{x}_N, \mathbf{x})]^T. \quad (26)$$

The case in our algorithm, where all the training signals are involved in the approximation of the kernel matrix ($c = N$, $\mathbf{C} = \mathbf{W} = \mathbf{K}$), results in a similar expression for the virtual samples:

$$\mathbf{F} = (\Sigma^{1/2})^\dagger \mathbf{V}^T \mathbf{C}^T = (\Sigma^{1/2})^\dagger \mathbf{V}^T \mathbf{K}^T, \quad (27)$$

where Σ and \mathbf{V} are the eigenvalues and eigenvectors of the matrix \mathbf{K} . The embedding in this case is thus

$$\Phi_e(\mathbf{x}) = (\Sigma^{1/2})^\dagger \mathbf{V}^T [\kappa(\mathbf{x}_1, \mathbf{x}), \dots, \kappa(\mathbf{x}_N, \mathbf{x})]^T. \quad (28)$$

Contrary to [21], [22], our embedding preserves the similarities in the high-dimensional feature space, represented by the inner products, i.e.,

$$\Phi_e(\mathbf{x})^T \Phi_e(\mathbf{x}') \approx \kappa(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^T \Phi(\mathbf{x}'), \quad (29)$$

where we have used the expression $\mathbf{K}^\dagger = \mathbf{V} \Sigma^\dagger \mathbf{V}^T$. In addition, both [21] and [22] focus on sparse coding only and do not address the accuracy of the kernel empirical map, nor its dimension, which can be highly restrictive in large-scale datasets.

Both Gao *et al.* in 2010 [23] and Li *et al.* in 2011 [24], proposed an analytical approach of kernelizing the basis pursuit and orthogonal matching pursuit algorithms. Contrary to [21] and [22], the authors replaced all the inner products by kernels and worked entirely in the feature domain. Classification of faces and objects were achieved in [23] using a similar approach as in the SRC algorithm [11]. Aside from kernelizing the SRC algorithm, [23] also suggested updating the dictionary one atom at a time. By zeroing the derivative of the optimization function with respect to each atom, the authors acquired in the same term, a mixture of both the atom itself and its kernel with the input examples. As the resulting equation could not be solved analytically, an iterative fixed point update was implemented.

In 2012 Zhang *et al.* [25] provided an alternate approach of kernelizing the SRC algorithm. Instead of working with the implicit mapped signals in the feature space $\Phi(\mathbf{y})$, the authors performed dimensionality reduction first, using the KPCA algorithm, then fed the resulting nonlinear features to a linear l_1 basis pursuit solver. It can be shown that kernel PCA eventually entails the eigendecomposition of the

kernel matrix (more accurately, the centered kernel matrix), as does our algorithm. The difference is that our method, apart from providing an accurate kernel mapping which preserves similarities in feature space, also avoids dealing with the kernel matrix altogether in the training stage, making it possible to work with large datasets.

V. EXPERIMENTAL RESULTS

In the following section we highlight the three main benefits of incorporating LKDL with existing DL: (1) improvement in discriminability, which results in better classification (2) a small added computational effort by LKDL in comparison with typical kernel methods and (3) the ability to incorporate the LKDL seamlessly in virtually any existing linear DL algorithm, contributing to more compact dictionaries and sparse representations.

A. Unsupervised Dictionary Learning

In this part we demonstrate the performance of our algorithm in digit classification on the USPS and MNIST databases. Our method of classification consists of first pre-processing the training and test data using LKDL, then performing regular, standard dictionary learning, using existing tools and finally deploying the classification scheme in section IV-D. For sparse coding and dictionary learning, we use the batch-OMP and efficient-KSVD implementations from the latest OMP-Box (v10) and KSVD-Box (v13) libraries³ [52]. During all experiments we use the KKSVD algorithm explained in section III-B [28], [30] as our reference, in addition to regular linear KSVD. We use the original code of Nguyen's KKSVD⁴. A fair comparison in accuracy and runtime, between LKDL and KKSVD can be made, as KKSVD uses the same functions from the OMP and KSVD libraries mentioned earlier. The k-means⁵ and coresets⁶ sampling techniques were also adopted from existing code. All of the tests were performed on a 64-Bit Windows7 Intel(R) Core(TM) i7-4790K CPU with 16GB memory. The initial dictionary is a random subset of m columns from the training set in each class.

1) *USPS dataset*: The USPS dataset consists of 7,291 training and 2,007 test images of digits of size 16×16 . All images are stacked as vectors of dimension $p = 256$ and normalized to unit l_2 norm. Following the experiment in [30], we choose the following parameters: 300 dictionary atoms per class, cardinality of 5 and 5 iterations of DL. The chosen kernel is polynomial of order 4, i.e. $\kappa(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^4$. The approximation parameters were chosen empirically using coarse-to-fine search and were set to: $c = 20\%$ of N training samples and $k = 256$, the original dimension of the digits. The displayed results are an average of 10 repeated iterations with different initialization of the sub-dictionaries and different sampled columns \mathbf{X}_R in Nyström's method.

First we evaluate the quality of the representation of the kernel matrix using Nyström's method. We randomly choose

2,000 samples from USPS and approximate the resulting kernel matrix. In order to isolate the effect of column sub-sampling, we do not perform additional dimensionality reduction using eigen-decomposition and thus choose $k = 256$. Five sampling techniques were examined: uniform [44], diagonal [48], column-norm [47], k-means [50] and coresets [51]. We also added the ideal reconstruction using rank- c SVD decomposition, which is optimal with respect to minimizing the approximation error, but takes much longer time to compute. We perform the comparison using the normalized approximation error:

$$err = \frac{\|\mathbf{K} - \tilde{\mathbf{K}}\|_F}{\|\mathbf{K}\|_F}, \quad (30)$$

where \mathbf{K} is the original kernel matrix and $\tilde{\mathbf{K}}$ its Nyström approximation. Fig. 1a shows the quality of the approximation versus the c/N ratio, the percent of samples chosen for the Nyström approximation. As expected, SVD performs the best, as it is meant exactly for the purpose of providing the ideal rank- c approximation of \mathbf{K} . The second best approximation is obtained by k-means, which provides 98.5% accuracy in terms of the normalized approximation error, with only 10% of the samples. All other methods perform roughly the same. The differences in approximation quality reduce as the percent of chosen samples grows to half of the input dataset.

Next we examine the effect of sub-sampling on the classification accuracy of the entire database of USPS. Fig. 1b shows the classification accuracy as a function of c/N , along with the constant results of linear KSVD and KKSVD (which do not depend on c). There is a gap of 1% between the results of linear KSVD and its kernel variants, which suggests that kernelization improves the discriminability of the input signals. It can be seen that k-means sampling again performs best and reaches classification accuracy of KKSVD, with only a fraction of the samples. In general, the percent of samples in Nyström approximation does not have much impact on the final classification accuracy (apart from small fluctuations that arise from the randomness of each run). This can be explained by the simplicity of the digit images and the relatively large number of training examples.

Following Nguyen's setup in [28] and [30], we inspect the effect of corrupting the test images with white Gaussian noise and missing pixels. We use the same parameters as before and repeat the experiment 10 times with different random corruptions. The results of classification accuracy versus the standard deviation of the noise and the percent of missing pixels are given in Fig. 2a and 2b. It is evident that adding the kernel improves the robustness of the database to both noise and missing pixels. The performance of LKDL follows that of KKSVD with a only 20% of the training samples. The trend shown in our results is similar to that in [30], although the results are slightly lower. This can be explained by the fact that in [30], the authors did not use the traditional partitioning of training and test data of the USPS dataset. In this simulation, the coresets sampling technique was the best in dealing with signal corruptions, which is the reason it is the only method shown.

³Found in <http://www.cs.technion.ac.il/~ronrubin/software.html>

⁴Found in <http://www.umiacs.umd.edu/~hien/KKSVD.zip>

⁵K-means - <http://www.mathworks.com/matlabcentral/fileexchange/31274-fast-k-means-center-k-means.m>

⁶Coresets - <http://web.media.mit.edu/~michaf/index.html>

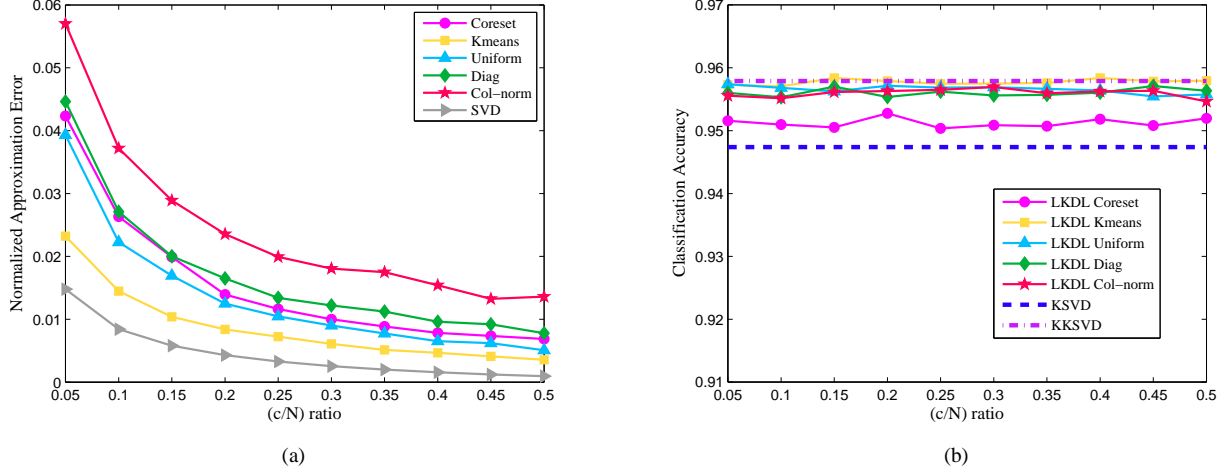


Fig. 1. Approximation error (a) and classification accuracy (b) as a function of c/N , percent of samples used in Nyström method.

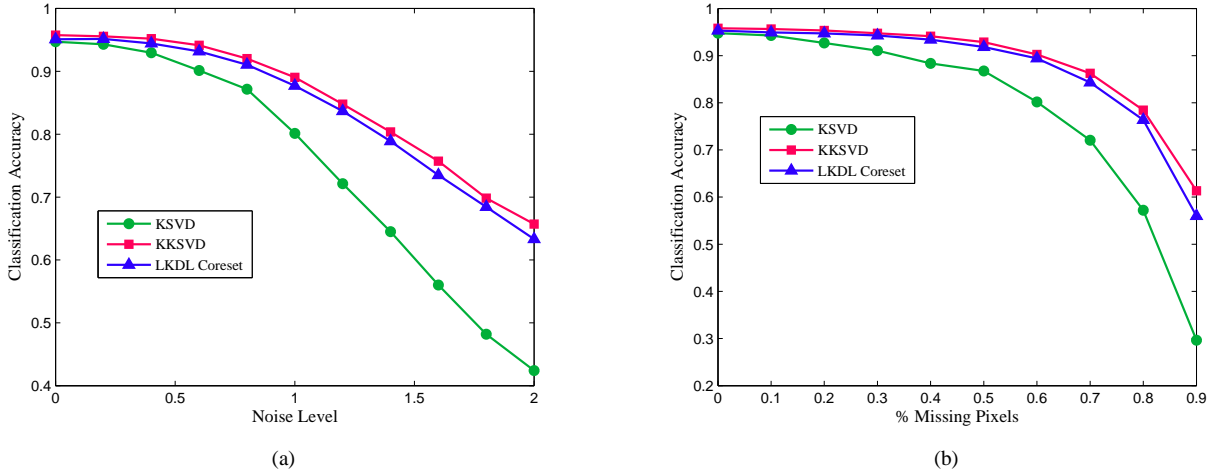


Fig. 2. Classification accuracy in the presence of Gaussian noise (a) and missing pixels (b).

2) *MNIST dataset*: Next we demonstrate the differences in runtime between our method and KKSVD using the larger-scale digit database of MNIST, which consists of 60,000 training and 10,000 test images of digits of size 28×28 . Same as before, the digits were stacked in vectors of dimension $p = 784$ and normalized to unit l_2 norm. We examine the influence of gradually increasing the training set on the classification accuracy and training time of the input data. In this simulation, the entire training set of 60,000 examples is reduced by randomly choosing a defined fraction of the samples, while maintaining the test set untouched. The runtime measured in LKDL includes the time needed to prepare both the training and test virtual samples, along with training the entire input dataset using linear KSVD. As for KKSVD, the runtime includes the preparation of the kernel sub-matrices for each class and the kernel DL using KKSVD. Parameters in the simulation were: 2 DL iterations, cardinality of 11,700 atoms per digit, polynomial kernel of order 2, $c = 15\%$ and $k = 784$. The results were averaged over 5 runs.

The results can be seen in Fig. 3a and 3b. Again, the coreset

sampling method was chosen, as it provided the best results. The accuracy of LKDL versus KKSVD is comparable, while slightly worse, due to the approximation, but still better than the linear version of KSVD. The runtime of LKDL follows the one of KSVD, along with a component of calculating the virtual datasets. This is expected since our method “piggy-backs” on KSVD’s performance and complexity. KKSVD’s performance however, is dependent quadratically on the number of input samples in each class. When the database is large, the calculation of the virtual datasets (which is performed only once), is negligible versus the alternative of performing kernel sparse coding thousands of times during the DL process.

Note that we chose a relatively small number of DL iterations in order to reduce the already-long computation time of KKSVD. A larger number of DL iterations will lead to an even greater difference in runtime between KKSVD and LKDL. For training the entire database of MNIST, LKDL is 19-times faster than KKSVD.

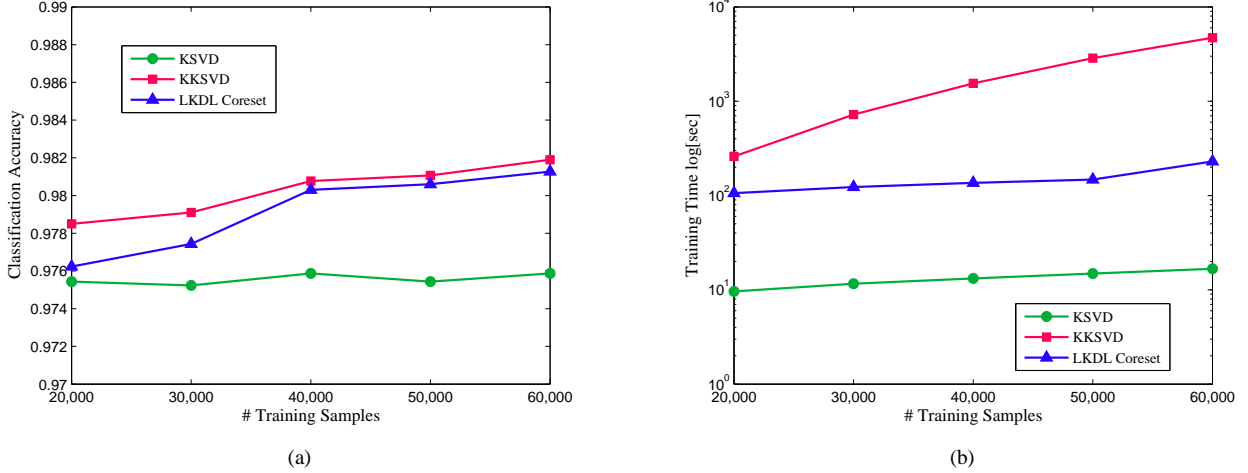


Fig. 3. Accuracy (a) and total training time (b) versus the number of input training examples in MNIST database. Runtime is shown in logarithmic scale.

B. Supervised Dictionary Learning

In the following set of experiments we demonstrate the easiness of combining our pre-processing stage with any DL algorithm, in particular the LC-KSVD [15] and FDDL [17], both of which are supervised dictionary learning techniques that were mentioned earlier. We do so using the original code of LC-KSVD⁷ and FDDL⁸. Throughout all tests, the training and test sets were pre-processed using LKDL to produce virtual training and test sets, which were later on fed as input to the DL and classification stages of each method. In all experiments, no code has been modified, except for exterior parameters which can be tuned to provide better results. The point in this setup is using an existing technique of supervised DL and showing the improvement that our method can provide.

1) *Evaluation on the USPS Database:* We start with comparing the classification accuracy of the same database from before, the USPS. First we perform regular FDDL with the following parameters: 5 DL iterations, 300 dictionary atoms per class, where the dictionary is first initialized using K-means clustering of the training examples. The scalars controlling the tradeoff in the DL and optimization expressions remained the same as in the demo provided by the authors: $\lambda_1 = 0.1, \lambda_2 = 0.001$ and $g_1 = 0.1, g_2 = 0.001$ (in [17], these are referred to as γ_1, γ_2). As for LKDL pre-processing, the chosen parameters were: Polynomial kernel of degree 3, K-means based sub-sampling of 20% of the training samples ($c/N = 0.2$) and $k = 256$. All results were averaged over 10 iterations with different initializations.

Table II shows the classification results with and without LKDL. There is a clear improvement in the results when adding LKDL as pre-processing. However the obtained results in this experiment are lower than those reported in [17]. This can be explained by the fact that we used the original database of USPS, while the provided code had a demo intended for an extended translation-invariant version of USPS. In addition,

TABLE II
CLASSIFICATION ACCURACY OF FDDL ON THE USPS DIGIT DATABASE, WITH AND WITHOUT LKDL PRE-PROCESSING

Algorithm	Accuracy
FDDL	95.79
FDDL + LKDL	96.03

the exterior parameters $\lambda_1, \lambda_2, g_1, g_2$ were tweaked especially for the extended USPS, thus may have provided worse results in our case.

2) *Evaluation on the Extended YaleB Database:* Next, we show the benefit of combining our method with LC-KSVD on the “Extended YaleB” face recognition database, which consists of 2,414 frontal images that were taken under varying lighting conditions. There are 38 classes in YaleB and each class roughly contains 64 images, which are split in half to training and test sets, following the experiment described in [15]. The original 192×168 images are projected to 504-dimensional vectors using a randomly generated constant matrix from a zero-mean normal distribution. We use a dictionary size of 570 (in average 15 images per class) and sparsity factor of 30, same as in [15]. The kernel chosen for LKDL was Gaussian of the form: $\kappa(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|_2^2 / 2\sigma^2)$, where $\sigma = 1$. Due to the small size of the dataset, no sub-sampling was performed and c was set to be the entire size of the training set. The value of the parameter k (the dimension of the signal after eigen-decomposition) was set to 400, as it appeared that further dimensionality reduction of the already reduced 504-dimensional vector improved the results. In order to use the Gaussian kernel, the samples in the training and test sets were l_2 normalized, thus the original parameters of $\sqrt{\alpha}$ and $\sqrt{\beta}$ in expression (9) had to be changed from 4 and 2 to $1/30$ and $1/91$ correspondingly. These parameters were chosen using a coarse-to-fine search and provided the best classification results. We use the original classification scheme in [15], [16].

Table III shows the classification results of LC-KSVD1 and LC-KSVD2, with and without LKDL pre-processing. It

⁷Found in <http://www.umiacs.umd.edu/~zhuolin/LCKSVD/>

⁸Found in http://www.vision.ee.ethz.ch/~yangme/database_mat/FDDL.zip

TABLE III
CLASSIFICATION ACCURACY OF LC-KSVD1 AND LC-KSVD2 ON THE
EXTENDED YALEB DATABASE, WITH AND WITHOUT LKDL
PRE-PROCESSING

Algorithm	Accuracy
LC-KSVD1	94.49
LC-KSVD1 + LKDL	96.33
LC-KSVD2	94.99
LC-KSVD2 + LKDL	96.33

is clear that the addition of the nonlinear kernel function increases the discriminability of the input samples and improves classification results by up to 1.8% and 1.3% in the case of LC-KSVD1 and LC-KSVD2, correspondingly. In fact, it appears that our LKDL blurs the differences between these two methods, meaning, there is no preference as to whether the classifier will be learned separately or jointly along with the dictionary.

The improved discriminability of LKDL combined with LC-KSVD, versus LC-KSVD alone, can be demonstrated by inspecting the resulting sparse coefficients of the test set. In Fig. 4 one can see the obtained sum of absolute values of the sparse coefficient vectors of all 32 test samples from class ‘10’. The ideal distribution of atoms chosen during sparse-coding should be concentrated around atoms: $[139, \dots, 150]$, which belong to class ‘10’. One can see that in the case of LC-KSVD, there are a few “successful” atoms which largely contribute to the reconstruction of the test samples, while in LC-KSVD combined with LKDL, the contribution is distributed more evenly between all of the atoms in that class. In addition, LC-KSVD alone will often choose atoms not corresponding with the given class, while in LKDL, the contribution of these atoms is fairly small.

Next we explore the impact of LKDL on the size of the learned dictionary. Fig. 5a shows the results of LC-KSVD1 and LC-KSVD2, with and without LKDL, versus the average number of dictionary atoms for each class. It is clear that LKDL improves the results of both LC-KSVD1 and LC-KSVD2. With the addition of LKDL, a smaller dictionary with 7 atoms per person achieves the same results of LC-KSVD alone with 15 atoms per person. This gap in performance grows as the size of the dictionary becomes smaller and reaches a 20% difference for 1 atom per person. The conclusion is that a more compact dictionary can be learnt using the combination of LC-KSVD and LKDL, without compromising accuracy.

Fig. 5b shows a similar experiment of the dependency of classification on the sparsity factor, i.e. the number of atoms used in the sparse reconstruction of a given signal. The combination of LKDL and LC-KSVD with a sparsity of 15 achieves a better accuracy than that of LC-KSVD alone with a sparsity of 30. From both these figures it can be seen that the addition of LKDL can be helpful in reducing the complexity of the DL problem, without compromising the accuracy.

3) *Evaluation on the AR Face Database:* The AR Face database consists of 4,000 color images of faces belonging to 126 classes. Each class consists of images taken over two sessions, containing different lighting conditions, facial variations and facial disguises (sunglasses and scarves). Following

TABLE IV
CLASSIFICATION ACCURACY OF LC-KSVD1 AND LC-KSVD2 ON THE
AR FACE DATABASE, WITH AND WITHOUT LKDL PRE-PROCESSING

Algorithm	Accuracy
LC-KSVD1	92.5
LC-KSVD1 + LKDL	94
LC-KSVD2	93.7
LC-KSVD2 + LKDL	94.7

the experiment in [16], 2,600 images were chosen, first 50 classes of males and first 50 classes of females. Out of 26 images in each class, 20 were chosen for training and the rest for evaluation. We use the already-processed dataset⁹ in [16], where the original images of size 165×120 pixels were reduced to 540-dimensional vectors using random projection as in Extended YaleB. The cardinality is same as before set to 30 and the number of atoms in DL is set to 500 (5 atoms per class). As before, we normalized all the signals to unit l_2 -norm. The parameters $\sqrt{\alpha}$ and $\sqrt{\beta}$ were determined using coarse-to-fine 5-fold cross validation. We have noticed that an optimal parameter of $\sqrt{\alpha}$ for LC-KSVD1 is not necessarily as good for LC-KSVD2, thus we chose two sets of parameters: $\sqrt{\alpha} = \sqrt{\beta} = 1/150$ for the optimal result of LC-KSVD1 (the value of β is not really used in LC-KSVD1), and $\sqrt{\alpha} = \sqrt{\beta} = 1/120$ for LC-KSVD2.

In table IV we compare the classification results of LC-KSVD1 and LC-KSVD2, with and without LKDL pre-processing. As can be seen our method improves the performance of LC-KSVD1 by 1.5% and LC-KSVD2 by 1%.

VI. CONCLUSION

In this paper we have discussed some of the problems arising when trying to incorporate kernels in DL, and payed special attention to the **kernel-KSVD algorithm by Nguyen *et al.* [28], [30]. We proposed a novel kernel DL scheme, called “LKDL”, which acts as a kernelizing pre-processing stage, before performing standard DL.** We used the concept of virtual training and test sets and described the different aspects of calculating these signals. We demonstrated in several experiments on different datasets the benefits of combining our LKDL pre-processing stage, both in accuracy of classification and in runtime. Lastly, we have shown the easiness of integrating our method with existing supervised and unsupervised DL algorithms. It is our hope that the proposed methodology will encourage users to consider kernel DL for their tasks, knowing that the extra-effort involved in incorporating the kernel layer is near-trivial. We intend to freely share the code that reproduces all the results shown in this paper.

Our future research directions include combining LKDL with online DL. We would also like to examine the benefit of applying LKDL to the sparse coefficients instead of the input signals and maybe combining both options. Lastly, our goal is improving the sampling ratio, i.e. the size of the matrix C , using more advanced sampling techniques.

⁹Found in <http://www.umiacs.umd.edu/~zhuolin/LCKSVD/>

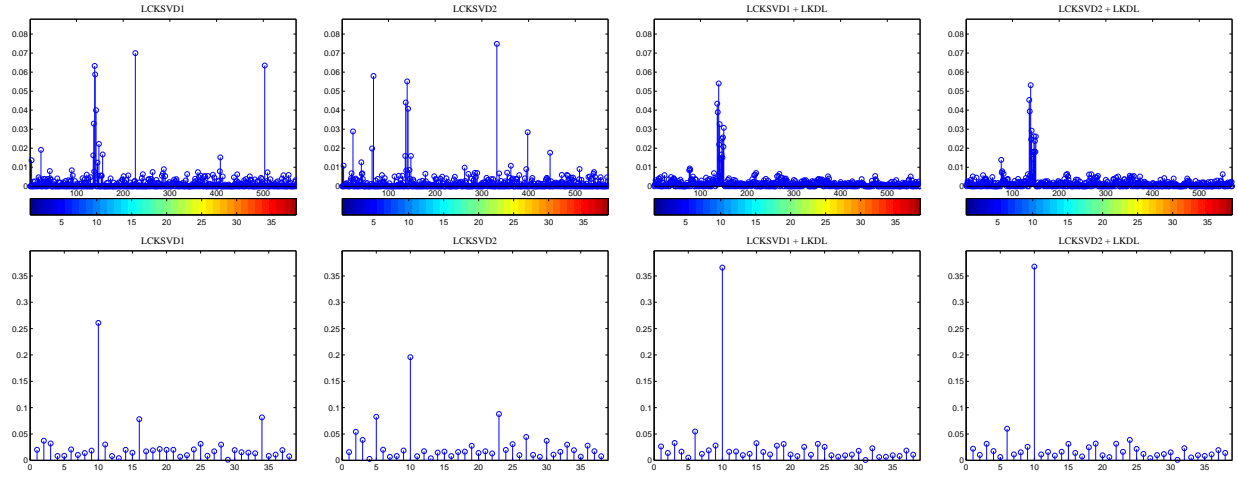


Fig. 4. Upper row: sum of absolute values of sparse coefficient vectors (of size 570, the size of the dictionary) corresponding to test examples from class ‘10’ in Extended YaleB database. The columns from left to right represent LC-KSVD1 and LC-KSVD2 with and without the addition of LKDL pre-processing. The additional colorbar features 38 bars which correspond to 38 classes in Extended YaleB. Bottom row: additional summation of the absolute values of sparse coefficients in every class. As expected, the majority of nonzero values in all sparse coefficient vectors originate from class ‘10’.

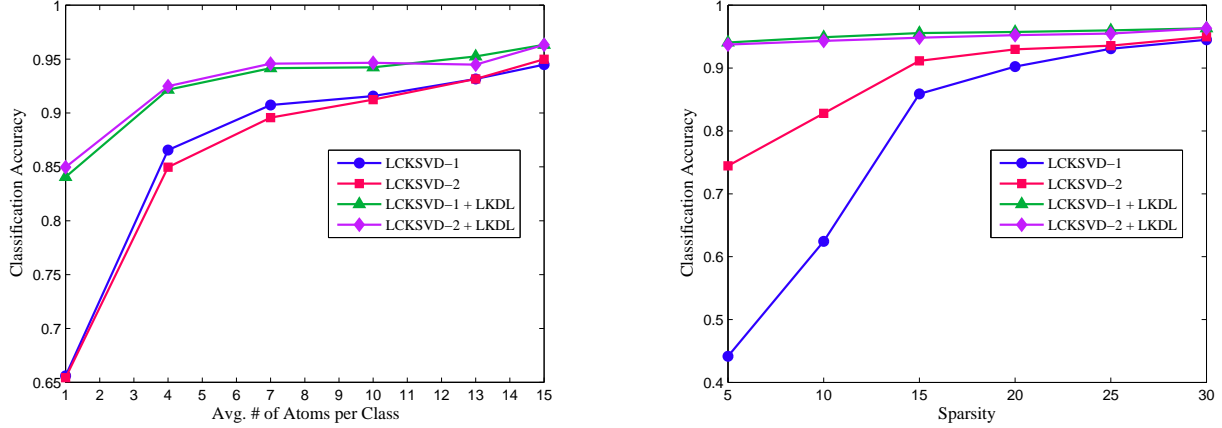


Fig. 5. Dependence of accuracy in the average number of atoms per class (a) and the sparsity factor (b).

REFERENCES

- [1] S. Mallat, “A wavelet tour of signal processing,” *Academic Press*, 1999.
- [2] E. J. Candes and D. L. Donoho, “Recovering edges in ill-posed inverse problems: Optimality of curvelet frames,” *Ann. Statist.*, vol. 30, no. 3, pp. 784–842, Jun. 2002.
- [3] M. N. Do and M. Vetterli, “Contourlets: a directional multiresolution image representation,” *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2002.
- [4] M. Elad and M. Aharon, “Image denoising via sparse and redundant representations over learned dictionaries,” *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.
- [5] J. M. Fadili, J. L. Starck, and F. Murtagh, “Inpainting and zooming using sparse representations,” *The Computer Journal, Oxford Journals*, vol. 52, no. 1, pp. 64–79, 2007.
- [6] J. Mairal, M. Elad, and G. Sapiro, “Sparse representation for color image restoration,” *IEEE Trans. Image Process.*, vol. 17, no. 1, pp. 53–69, Jan. 2008.
- [7] 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, May 2008.
- [8] J. Zepeda, C. Guillemot, and E. Kijak, “Image compression using sparse representations and the iteration-tuned and aligned dictionary,” *IEEE J. Sel. Top. Signal Process.*, vol. 5, no. 5, pp. 1061–1073, Sep. 2011.
- [9] K. Engan, S. Aase, O. Hakon, and J. Husoy, “Method of optimal directions for frame design,” *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 5, pp. 2443–2446, 1999.
- [10] M. Aharon, M. Elad, and A. Bruckstein, “K-SVD an algorithm for designing overcomplete dictionaries for sparse representations,” *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.
- [11] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, “Robust face recognition via sparse representation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.
- [12] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach, “Supervised dictionary learning,” *Advances in Neural Information Processing Systems (NIPS)*, pp. 1033–1040, 2009.
- [13] J. Mairal, F. R. Bach, and J. Ponce, “Task driven dictionary learning,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 791–804, Apr. 2012.
- [14] Q. Zhang and B. Li, “Discriminative K-SVD for dictionary learning in face recognition,” *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2691–2698, 2010.
- [15] Z. Jiang, Z. Lin, and L. S. Davis, “Learning a discriminative dictionary for sparse coding via label consistent label consistent K-SVD,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1697–1704, 2011.
- [16] —, “Label consistent K-SVD: Learning a discriminative dictionary for recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2651–2664, 2013.

- [17] M. Yang, L. Zhang, X. Feng, and D. Zhang, "Fisher discrimination dictionary learning for sparse representation," *IEEE International Conference on Computer Vision (ICCV)*, pp. 543–550, 2011.
- [18] V. Vapnik, *The nature of statistical learning theory*. Springer New York, 2000.
- [19] B. Scholköpf, A. Smola, and K. R. Müller, "Kernel principal component analysis," *Artificial Neural Networks ICANN*, pp. 583–588, 1997.
- [20] B. Scholköpf and K. R. Müller, "Fisher discriminant analysis with kernels," *Proceedings of the 1999 IEEE Signal Processing Society Workshop Neural Networks for Signal Processing*, pp. 23–25, 1999.
- [21] P. Vincent and Y. Bengio, "Kernel matching pursuit," *Mach. Learn.*, vol. 48, no. 1–3, pp. 165–187, 2002.
- [22] V. Guigue, A. Rakotomamonjy, and S. Canu, "Kernel basis pursuit," *Machine Learning (ECML) 2005*, pp. 146–157, 2005.
- [23] S. Gao, I. W. H. Tsang, and L. T. Chia, "Kernel sparse representation for image classification and face recognition," *Computer Vision – ECCV*, pp. 1–14, 2010.
- [24] H. Li, Y. Gao, and J. Sun, "Fast kernel sparse representation," *IEEE International Conference on Digital Image Computing Techniques and Applications (DICTA)*, pp. 72–77, 2011.
- [25] L. Zhang, W. D. Zhou, P. C. Chang, J. Liu, T. Wang, and F. Z. Li, "Kernel sparse representation-based classifier," *IEEE Trans. Signal Process.*, vol. 60, no. 4, pp. 1684–1695, Apr. 2012.
- [26] M. Jian and C. Jung, "Class-discriminative kernel sparse representation-based classification using multi-objective optimization," *IEEE Trans. Signal Process.*, vol. 61, no. 18, pp. 4416–4427, Sep. 2013.
- [27] Y. Zhou, K. Liu, R. E. Carrillo, K. E. Barner, and F. Kiamilev, "Kernel-based sparse representation for gesture recognition," *Pattern Recognition*, vol. 46, no. 12, pp. 3208–3222, Dec. 2013.
- [28] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Kernel dictionary learning," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 2021–2024, 2012.
- [29] M. T. Harandi, C. Sanderson, R. Hartley, and B. C. Lovell, "Sparse coding and dictionary learning for symmetric positive definite matrices: A kernel approach," *Computer Vision ECCV 2012*, pp. 216–229, 2012.
- [30] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Design of non-linear kernel dictionaries for object recognition," *IEEE Trans. Image Process.*, vol. 22, no. 12, pp. 5123–5135, Dec. 2013.
- [31] M. J. Gangeh, A. Ghodsi, and M. S. Kamel, "Kernelized supervised dictionary learning," *IEEE Trans. Signal Process.*, vol. 61, no. 19, pp. 4753–4767, Oct. 2013.
- [32] A. Shrivastava, H. V. Nguyen, V. M. Patel, and R. Chellappa, "Design of non-linear discriminative dictionaries for image classification," *Computer Vision-ACCV 2012*, pp. 660–674, 2012.
- [33] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pp. 689–696, 2009.
- [34] K. Skretting and K. Engan, "Recursive least squares dictionary learning algorithm," *IEEE Trans. Signal Process.*, vol. 58, no. 4, pp. 2121–2130, Apr. 2010.
- [35] K. Zhang, L. Lan, Z. Wang, and F. Moerchen, "Scaling up kernel SVM on limited resources: A low-rank linearization approach," *International Conference on Artificial Intelligence and Statistics*, pp. 1425–1434, 2012.
- [36] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1998.
- [37] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, 1993.
- [38] Y. C. Pati, R. Rezaifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," *IEEE Comput. Soc. Press*, pp. 40–44, 1993.
- [39] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley-Interscience, 2000.
- [40] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc.*, vol. 68, pp. 337–404, 1950.
- [41] J. Mercer, "Functions of positive and negative type and their connection with the theory of integral equations," *Philos. Trans. Roy. Soc. London*, pp. 415–446, 1909.
- [42] B. Scholköpf, S. Mika, C. J. Knirsch, K. R. Müller, G. Ratsch, and A. J. Smola, "Input space versus feature space in kernel-based methods," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1000–1017, 1999.
- [43] H. Xiong, M. N. S. Swamy, and M. O. Ahmad, "Optimizing the kernel in the empirical feature space," *IEEE Trans. Neural Networks*, vol. 16, no. 2, pp. 460–474, Mar. 2005.
- [44] C. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," *Proceedings of the 14th Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 682–688, 2002.
- [45] C. Fowlkes, S. F. Belongie, Chung, and J. Malik, "Spectral grouping using the Nyström method," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 26, no. 2, pp. 214–225, 2004.
- [46] C. Fowlkes, S. Belongie, and J. Malik, "Efficient spatiotemporal grouping using the Nyström method," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.
- [47] P. Drineas, R. Kannan, and M. W. Mahoney, "Fast monte carlo algorithms for matrices i: Approximating matrix multiplication," *SIAM Journal on Computing*, vol. 36, no. 1, pp. 132–157, Jan. 2006.
- [48] P. Drineas and M. W. Mahoney, "On the Nyström method for approximating a gram matrix for improved kernel-based learning," *The Journal of Machine Learning Research*, no. 6, pp. 2153–2175, 2005.
- [49] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling techniques for the Nyström method," *International Conference on Artificial Intelligence and Statistics*, pp. 304–311, 2009.
- [50] K. Zhang, I. W. Tsang, and J. T. Kwok, "Improved Nyström low-rank approximation and error analysis," *Proceedings of the 25th International Conference on Machine Learning (ACM)*, pp. 1232–1239, 2008.
- [51] D. Feldman, M. Feigin, and N. Sochen, "Learning big (image) data via coresets for dictionaries," *Journal of Mathematical Imaging and Vision*, vol. 46, no. 3, pp. 276–291, Jul. 2013.
- [52] R. Rubinstein, M. Zibulevsky, and M. Elad, "Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit," *CS TECHNION*, vol. 40, 2008.



Alona Golts received her B.Sc. (2009) in Electrical Engineering and Physics, from the department of Electrical Engineering at the Technion, Israel, where she is currently pursuing her M.Sc degree. Alona has served in the Israeli Air Force from 2009 to 2015, under the reserve excellence program "Psgot".



Michael Elad received his B.Sc. (1986), M.Sc. (1988) and D.Sc. (1997) from the department of Electrical engineering at the Technion, Israel. Since 2003 he is a faculty member at the Computer Science department at the Technion, and since 2010 he holds a full-professorship position. Michael Elad works in the field of signal and image processing, specializing in particular on inverse problems, sparse representations and superresolution. Michael received the Technions best lecturer award six times, he is the recipient of the 2007 Solomon Simon Mani award for excellence in teaching, the 2008 Henri Taub Prize for academic excellence, and the 2010 Hershel-Rich prize for innovation. Michael is an IEEE Fellow since 2012. He is serving as an associate editor for SIAM SIIMS, and ACHA. Michael is also serving as a senior editor for IEEE SPL.