When VLAD met Hilbert

Mehrtash Harandi[1], Mathieu Salzmann[2], and Fatih Porikli[1]

[1]NICTA* and Australian National University, Canberra, Australia
[2]CVLab, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Abstract

*In many challenging visual recognition tasks where training data is limited, Vectors of Locally Aggregated Descriptors (VLAD) have emerged as powerful image/video representations that compete with or outperform state-of-the-art approaches. In this paper, we address two fundamental limitations of VLAD: its requirement for the local descriptors to have vector form and its restriction to linear classifiers due to its high-dimensionality. To this end, we introduce a kernelized version of VLAD. This not only lets us inherently exploit more sophisticated classification schemes, but also enables us to efficiently aggregate non-vector descriptors (*e.g.*, manifold-valued data) in the VLAD framework. Furthermore, we propose an approximate formulation that allows us to accelerate the coding process while still benefiting from the properties of kernel VLAD. Our experiments demonstrate the effectiveness of our approach at handling manifold-valued data, such as covariance descriptors, on several classification tasks. Our results also evidence the benefits of our nonlinear VLAD descriptors against the linear ones in Euclidean space using several standard benchmark datasets.*

1. Introduction

This paper introduces a nonlinear formulation of *Vectors of Locally Aggregated Descriptors* (VLAD) that generalizes their use to manifold-valued local descriptors, such as symmetric positive definite (SPD) matrices, and allows them to inherently exploit more sophisticated classification algorithms. Modern visual recognition techniques typically represent images by aggregating local descriptors, which, compared to image intensity, provide robustness to varying imaging conditions. From a historical point of view, this trend gained momentum since the introduction of the *Bag-of-Words* (BoW) model [38, 14, 24], which had a significant impact on recognition performance. Notable recent developments include dictionary-based solutions [50, 51], Fisher Vectors (FV) [30, 32], VLAD [21, 1] and Convolutional Neural Networks (CNN) [23].

Among the aforementioned techniques, VLAD stands out for the following reasons:

- VLAD is computed via primitive operations. This makes VLAD extremely attractive when computational complexity is a concern, and requires virtually no parameter-tuning, except the size of the codebook.
- In contrast to CNNs, training a VLAD encoder is straightforward and not contingent on having a large training set.
- VLAD can be considered as a special case of FVs and hence inherits several of their properties. The most eminent one is its theoretical connection to the Fisher kernel [19].
- From an empirical point of view, VLAD has been shown to either deliver state-of-the-art accuracy, or compete with the state-of-the-art methods when training data is limited. For instance, for scene classification on the MIT Indoor dataset, multi-scale VLAD, with only 4096 features, comfortably outperforms the mixture of FV and bag-of-parts, which relies on 221550 features [13].

Despite its unique properties, VLAD comes with its own limitations. In particular, VLAD is designed to work with local descriptors in the form of vectors. Yet, several recent studies in computer vision suggest that structural data (*e.g.*, SPD matrices [42, 40, 15, 20], graphs [46], orthogonal matrices [11, 16]) have the potential to provide more robust descriptors. Furthermore, since VLAD typically yields a high-dimensional image representation, it is mostly restricted to be employed with linear classifiers. The effectiveness of kernel-based methods, however, has been proven many a time in visual recognition [12, 4, 31, 45].

*NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the ARC through the ICT Centre of Excellence program.

In this paper, we introduce a nonlinear formulation of VLAD that addresses the aforementioned shortcomings. In particular, we first derive a kernelized version of VLAD that relies on mapping each local descriptor to a Reproducing Kernel Hilbert Space (RKHS). We then show that aggregation can be performed in the RKHS, which only involves computing kernel values. Since several valid kernel functions have recently been defined for non-vector data [20, 18], our formulation ultimately generalizes the use of VLAD to non-vector spaces, such as the SPD manifold and the Grassmannian (the manifold of linear subspaces). Furthermore, the inherent nonlinearity of mappings to RKHS effectively translates to exploiting more advanced classifiers within the VLAD framework.

In the spirit of computational efficiency, we then introduce a novel nonlinear approximation to our kernel VLAD, which makes use of local subspaces in the Hilbert space. This approximation enjoys properties similar to those of kernel VLAD, yet has the additional benefit of providing us with faster coding schemes. Importantly, both kernel VLAD and its approximation essentially preserve the simplicity of VLAD, in the sense that the extra computations merely consist of kernel evaluations potentially followed by projections (*i.e*., matrix multiplications). To give a concrete example, for head-pose estimation with manifold-valued data (see Section 4), our nonlinear approximation of kernel VLAD not only outperforms the Riemannian version of VLAD [11], but also encodes images 10 times faster.

Our experimental evaluation demonstrates the effectiveness of our approach at handling manifold-valued data in a VLAD framework. Furthermore, we evidence the benefits of exploiting nonlinear classifiers for visual recognition by comparing the performance of our nonlinear VLAD with the standard one on several benchmark datasets, where the local descriptors have a vector form.

1.1. Related Work

Most of the popular image classification methods extract local descriptors at patch level, and aggregate these descriptors into a global image representation [24, 30, 21, 32, 43, 23, 1]. When large amounts of training data are available, CNNs have now emerged as the method of choice to learn local descriptors. With a limited number of training samples, existing methods typically opt for handcrafted features, such as SIFT.

To aggregate local features, in addition to operations such as average-pooling and max-pooling, histogram-based solutions (*e.g*., BoW) have proven successful. Going beyond simple histograms has been an active topic of research in the past decade. For instance, [24] aggregates histograms computed over different spatial regions. More recent developments, such as FVs [30] and VLAD [21, 9], suggest that first- and potentially second-order statistics should be encoded in the aggregation process.

In a separate line of research, structured descriptors (*e.g*., covariance descriptors, linear subspaces) have been shown to provide robust visual models [42, 20, 16]. Being of a non-vectorial form, aggregating such descriptors is hard to achieve beyond simple histograms. Nonetheless, one would like to benefit from the best of both worlds, that is, using robust non-vectorial descriptors in conjunction with state-of-the-art aggregation techniques, such as VLAD. This, in essence, is what we propose to achieve in this paper via kernelization. Furthermore, our approach has the additional advantage of allowing us to inherently exploit nonlinear classifiers that have proven powerful for visual recognition.

The recent work of Faraki *et al*. [11] also aims at extending the VLAD framework to non-vector data. Specifically, [11] makes use of the tangent bundle of a Riemannian manifold to aggregate manifold-valued data in a similar manner as VLAD. By contrast, our work is not limited to Riemannian manifolds. That is, while in [11] the data must lie on a Riemannian manifold, we only require the existence of a positive definite kernel defined over the data. For instance, our framework therefore also applies to local descriptors represented as graphs, thanks to the available kernel of, *e.g*., [46]. Furthermore, several studies suggest that embedding Riemannian manifolds into RKHS boosts recognition performance [18, 20]. As a matter of fact, this is also demonstrated by our experiments, where our approach outperforms the method of [11].

While a full review of kernel-based methods in computer vision is beyond the scope of this paper, the recent work of [26] is of particular relevance here. [26] introduces an approach to employing kernels within a CNN framework. Here, we perform a similar analysis within the VLAD framework, with the additional benefit of obtaining a representation that lets us work with manifold-valued data.

2. Nonlinear VLAD

In this section, after briefly reviewing the conventional VLAD, we derive our two nonlinear VLAD formulations: kernel VLAD and its local subspace-based approximation.

2.1. Conventional VLAD

Let $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}, \boldsymbol{x}_i \in \mathbb{R}^d$ be a set of local descriptors extracted from a query image or a video. In VLAD [21], the input space $\mathbb{R}^d$ is partitioned into m Voronoi cells by means of a codebook $\mathcal{C}$ with centers $\{\boldsymbol{c}_j\}_{j=1}^{m},\ \boldsymbol{c}_j \in \mathbb{R}^d$, obtained from training data. Typically, this codebook is computed using the k-means algorithm. Note that supervised algorithms have also recently been employed to build more discriminative codebooks [28]. In any event, given the codebook, the VLAD code $\boldsymbol{v} \in \mathbb{R}^{md}$ for the query set $\mathcal{X}$ is obtained by concatenating m Local Difference Vectors (LDV) δ_j storing, for each center, the sum of the differences between this

center and each local descriptor assigned to it. This can be written as

$$\boldsymbol{v}(\mathcal{X}) = \left[\delta_1^T(\mathcal{X}), \delta_2^T(\mathcal{X}), \cdots, \delta_m^T(\mathcal{X})\right]^T , \quad (1)$$

where

$$\delta_j(\mathcal{X}) = \sum_{i=1}^{N} a_j^i \big(\boldsymbol{c}_j - \boldsymbol{x}_i\big) , \quad (2)$$

with a_j^i a binary weight encoding whether the local descriptor $\boldsymbol{x}_i$ belongs to the Voronoi cell with center $\boldsymbol{c}_j$ or not, *i.e.*, $a_j^i = 1$ if and only if the closest codeword to $\boldsymbol{x}_i$ is $\boldsymbol{c}_j$.

2.2. Kernel VLAD (kVLAD)

As mentioned earlier, the conventional VLAD is designed to work with local descriptors of vector form. As such, it cannot handle structured data representations, such as SPD matrices, or subspaces. While such representations could in principle be vectorized, this would (i) yield impractically high-dimensional VLAD vectors; and (ii) ignore the geometry of these structured representations, which has been demonstrated to result in accuracy loss [29, 41, 42, 20]. Here, we address this problem by kernelizing VLAD.

To this end, let us redefine the query set of local descriptors as $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^N, \boldsymbol{x}_i \in \mathbb{X}$, where each descriptor lies in the space $\mathbb{X}$, which, in contrast to VLAD, is not restricted to be $\mathbb{R}^d$. In fact, the only constraint we impose is that $\mathbb{X}$ comes with a valid positive definite (*pd*) kernel $k : \mathbb{X}\times\mathbb{X} \rightarrow \mathbb{R}$. For example, $\mathbb{X}$ could be the space of SPD matrices, with the Gaussian kernel defined in [39, 20]. According to the Moore-Aronszajn Theorem [2], a *pd* kernel $k(\cdot,\cdot)$ induces a unique Hilbert space on $\mathbb{X}$, denoted hereafter by $\mathcal{H}$, with the property that there exists a mapping $\phi : \mathbb{X} \rightarrow \mathcal{H}$, such that $k(\boldsymbol{x},\boldsymbol{y}) = \langle\phi(\boldsymbol{x}),\phi(\boldsymbol{y})\rangle_{\mathcal{H}} = \phi(\boldsymbol{x})^T\phi(\boldsymbol{y})$. Here, we propose to make use of this property to map the local descriptors to $\mathcal{H}$, which is a vector space, and perform a VLAD-like aggregation in Hilbert space. The main difficulty arises from the fact that $\mathcal{H}$ may be infinite-dimensional, and, more importantly, that the mapping ϕ corresponding to a given kernel k is typically unknown.

Let us suppose that we are given a codebook $\mathcal{C} = \{\phi(\boldsymbol{c}_i)\}_{i=1}^m$ in $\mathcal{H}$. For instance, this codebook can be computed using kernel k-means. To compute a VLAD code in $\mathcal{H}$, we need means to perform the following operations:

1. Determine the assignments $\{a_j^i\}$ in $\mathcal{H}$.
2. Express the LDVs in $\mathcal{H}$.

To determine the assignments, we note that

$$\|\phi(\boldsymbol{x}) - \phi(\boldsymbol{y})\|^2 = k(\boldsymbol{x},\boldsymbol{x}) - 2k(\boldsymbol{x},\boldsymbol{y}) + k(\boldsymbol{y},\boldsymbol{y}) . \quad (3)$$

Therefore, for each local descriptor, the nearest codeword can be found using kernel values only, *i.e.*, without having to know the mapping ϕ, which lets us directly determine the assignments.

Unfortunately, expressing the LDVs in $\mathcal{H}$ is not as straightforward. Clearly, the form of the LDVs, given by

$$\delta_j(\mathcal{X}) = \sum a_j^i \Big(\phi(\boldsymbol{c}_j) - \phi(\boldsymbol{x}_i)\Big) ,$$

with a_j^i obtained using Eq. 3, cannot be computed explicitly if the mapping ϕ is unknown, which is typically the case for popular kernels, such as RBF kernels. However, in most practical applications, the VLAD vector is not important by itself; What really matters for visual recognition is a notion of distance between two VLAD vectors. We therefore turn to the problem of computing the distance of two VLAD vectors in Hilbert space.

To this end, let $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{N_\mathcal{X}}, \boldsymbol{x}_i \in \mathbb{X}$ and $\mathcal{Y} = \{\boldsymbol{y}_i\}_{i=1}^{N_\mathcal{Y}}, \boldsymbol{y}_i \in \mathbb{X}$ be two sets of local descriptors. The implicit VLAD code of $\mathcal{X}$ in $\mathcal{H}$ can be expressed as

$$\boldsymbol{v}_\mathcal{H}(\mathcal{X}) = \left[\delta_1^T(\mathcal{X}), \delta_2^T(\mathcal{X}), \cdots, \delta_m^T(\mathcal{X})\right]^T ,$$

and similarly for $\boldsymbol{v}_\mathcal{H}(\mathcal{Y})$. Now, we have

$$\Big\langle \boldsymbol{v}_\mathcal{H}(\mathcal{X}), \boldsymbol{v}_\mathcal{H}(\mathcal{Y})\Big\rangle_\mathcal{H} = \sum_{s=1}^{m} \delta_s^T(\mathcal{X})\delta_s(\mathcal{Y}) \quad (4)$$

$$= \sum_{s=1}^{m}\sum_{i=1}^{N_\mathcal{X}}\sum_{j=1}^{N_\mathcal{Y}} a_s^i a_s^j \Big(\phi(\boldsymbol{c}_s) - \phi(\boldsymbol{x}_i)\Big)^T \Big(\phi(\boldsymbol{c}_s) - \phi(\boldsymbol{y}_j)\Big)$$

$$= \sum_{s=1}^{m}\sum_{i=1}^{N_\mathcal{X}}\sum_{j=1}^{N_\mathcal{Y}} a_s^i a_s^j \Big(k(\boldsymbol{x}_i,\boldsymbol{y}_j) + k(\boldsymbol{c}_s,\boldsymbol{c}_s) - k(\boldsymbol{x}_i,\boldsymbol{c}_s) - k(\boldsymbol{y}_j,\boldsymbol{c}_s)\Big),$$

which again only depends on kernel values.

With this inner product, a linear SVM, in its dual form, can directly be used for classification[1]. In our experiments, we rely on this approach, which we refer to as kernel VLAD or **kVLAD** for short.

This inner product, however, also allows us to employ an RBF-based kernel SVM, since

$$\|\boldsymbol{v}_\mathcal{H}(\mathcal{X}) - \boldsymbol{v}_\mathcal{H}(\mathcal{Y})\|^2 = \langle \boldsymbol{v}_\mathcal{H}(\mathcal{X}), \boldsymbol{v}_\mathcal{H}(\mathcal{X})\rangle_\mathcal{H} - 2\langle \boldsymbol{v}_\mathcal{H}(\mathcal{X}), \boldsymbol{v}_\mathcal{H}(\mathcal{Y})\rangle_\mathcal{H} + \langle \boldsymbol{v}_\mathcal{H}(\mathcal{Y}), \boldsymbol{v}_\mathcal{H}(\mathcal{Y})\rangle_\mathcal{H} .$$

Note that this essentially yields two layers of kernels, *i.e.*, the RBF kernel of the SVM makes use of the distance, which itself is expressed in terms of kernel values.

While effective in practice, our kVLAD algorithm, as any kernel method, becomes computationally expensive when dealing with large datasets. In the remainder of this section, we therefore introduce an approximation to kVLAD that addresses this limitation while still benefiting from the nice properties of kVLAD.

[1] Note that this yields a slightly different optimization problem than the standard kernel SVM formulation, since in our case the inner product itself depends on several kernel values.

2.3. Nonlinear VLAD via Local Subspaces (sVLAD)

Here, we introduce a nonlinear formulation of VLAD that approximates our kVLAD algorithm. To this end, we propose to make use of local subspaces to derive a novel approximation of a Hilbert space $\mathcal{H}$. This approximation is motivated by the following observation: By looking at Eq. 2, we can see that the contribution of each codeword in the VLAD vector is independent of the other codewords, particularly since each local descriptor is assigned to a single codeword. As such, there is no reason for the approximation of $\mathcal{H}$ to be shared across all the codewords and descriptors. We therefore propose to define approximate Hilbert spaces for each codeword individually.

To this end, let $\{\boldsymbol{t}_{s,j}\}_{j=1}^{N_s}$ be the set of training samples that generate the codeword $\boldsymbol{c}_s$. In other words, as in the conventional VLAD where $\boldsymbol{c}_s = \frac{1}{N_s}\sum_j \boldsymbol{t}_{s,j}$, we have $\phi(\boldsymbol{c}_s) = \frac{1}{N_s}\sum_i \phi(\boldsymbol{t}_{s,j})$. While, due to the unknown nature of ϕ, such a codeword cannot be explicitly computed, we can still evaluate the kernel function at this codeword, since

$$k(\boldsymbol{x}, \boldsymbol{c}_s) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{c}_s) = \frac{1}{N_s}\sum_j k(\boldsymbol{x}, \boldsymbol{t}_{s,j}) \ .$$

Here, we therefore propose to exploit the subspaces spanned by the training samples associated to each individual codeword to obtain an approximate representation of $\mathcal{H}$. More specifically, let $\mathcal{S}_s = span(\{\phi(\boldsymbol{t}_{s,j})\}_{j=1}^{N_s})$. We then define

$$\overline{\delta}_s(\mathcal{X}) = \sum_{i=1}^{N} a_s^i \Big(\pi_s\big(\phi(\boldsymbol{c}_s)\big) - \pi_s\big(\phi(\boldsymbol{x}_i)\big) \Big) \ , \qquad (5)$$

with $\pi_s : \mathcal{H} \rightarrow \mathcal{S}_s$ the projection onto $\mathcal{S}_s$. This projection can be obtained as follows. Let $\boldsymbol{K}_s$ be the kernel matrix estimated from the training samples generating $\boldsymbol{c}_s$, *i.e.*, $[\boldsymbol{K}_s]_{i,j} = k(\boldsymbol{t}_{s,i}, \boldsymbol{t}_{s,j})$. By eigendecomposition, we can write $\boldsymbol{K}_s = \boldsymbol{U}_s \Lambda_s \boldsymbol{U}_s^T$. Then, $\Phi_s \boldsymbol{U}_s \Lambda_s^{-1/2}$, with $\Phi_s = [\phi(\boldsymbol{t}_{s,1}), \cdots, \phi(\boldsymbol{t}_{s,N_s})]$, forms a basis for $\mathcal{S}_s$. As such, we can write

$$\pi_s(\boldsymbol{x}) = \Lambda_s^{-1/2} \boldsymbol{U}_s \Big[k(\boldsymbol{x}, \boldsymbol{t}_{s,1}), \cdots, k(\boldsymbol{x}, \boldsymbol{t}_{s,N_s}) \Big] . \qquad (6)$$

The LDVs $\overline{\delta}_s(\mathcal{X})$ can then be obtained for all codewords, and concatenated to form the final **sVLAD** representation.

Remark 1. *Note that one can also use only the top r eigenvectors of $\boldsymbol{K}_s$ to construct an r-dimensional local subspace in $\mathcal{H}$. This would not only yield the same dimensionality for all local subspaces, but could also potentially help discarding the noise associated to the $\{\boldsymbol{t}_{s,i}\}_{i=1}^{N_s}$.*

Remark 2. *Recall that in FV the coding scheme takes into account the Gaussian distribution centered at each codeword. In VLAD, coding is simplified by assuming that the Gaussian distributions are isotropic and all have the same variance. Interestingly, our sVLAD formulation relaxes this assumption by explicitly considering the eigenvalues of the kernel (covariance) computed from the data associated to each codeword.*

2.4. Normalization

Recent studies have shown that the discriminative power of VLAD can be boosted by additional post-processing steps, such as ℓ_2 power normalization and signed square rooting normalization [1, 13]. The ℓ_2 power normalization, where each block in VLAD is normalized individually, can easily be performed in kVLAD, since

$$\|\delta_s(\mathbb{X})\|_{\mathcal{H}}^2 = \sum_{i,j=1}^{N_{\mathbb{X}}} a_s^i a_s^j \Big(k(\boldsymbol{x}_i, \boldsymbol{x}_j) + k(\boldsymbol{c}_s, \boldsymbol{c}_s) - k(\boldsymbol{x}_i, \boldsymbol{c}_s) - k(\boldsymbol{x}_j, \boldsymbol{c}_s) \Big)$$

only depends on kernel values. As a result, the inner product of Eq. 4 after normalizing each VLAD block independently, *i.e.*,

$$\Big\langle \bar{\boldsymbol{v}}_{\mathcal{H}}(\mathbb{X}), \bar{\boldsymbol{v}}_{\mathcal{H}}(\mathbb{Y}) \Big\rangle_{\mathcal{H}} = \sum_{s=1}^{k} \frac{\Big\langle \delta_s(\mathbb{X}), \delta_s(\mathbb{Y}) \Big\rangle}{\|\delta_s(\mathbb{X})\|_{\mathcal{H}} \|\delta_s(\mathbb{Y})\|_{\mathcal{H}}} \ ,$$

will also only depend on kernel values. By contrast, however, the signed square rooting normalization can only be achieved when explicit forms of the descriptors are available, *i.e.*, in sVLAD.

3. Further Discussions

Our sVLAD formulation makes use of several local approximations of a Hilbert space. Other approaches have been proposed in the past to speed up kernel methods via Hilbert space approximations. Note, however, that these techniques yield one global approximation of the Hilbert space. In particular, the Nyström approximation [49, 31] makes use of data to approximate the kernel values. Furthermore, other methods approximate the inner product of specific kernel functions [33, 45]. For the sake of completeness, below, we derive approximations to kVLAD based on the aforementioned techniques. Note that our experiments demonstrate the benefit of our sVLAD formulation over these other approximations.

3.1. Nyström Approximation (nVLAD)

We start by deriving an approximation to kVLAD via the Nyström method [49, 31]. Such an approximation yields an explicit form for the mapping ϕ to the Hilbert space $\mathcal{H}$, and thus allows us to approximate a given kernel.

More specifically, let $\mathcal{T} = \{\boldsymbol{t}_i\}_{i=1}^{M}$, $\boldsymbol{t}_i \in \mathbb{X}$ be a collection of M training examples, and let $\boldsymbol{K}$ be the corresponding kernel matrix, *i.e.*, $[\boldsymbol{K}]_{i,j} = k(\boldsymbol{t}_i, \boldsymbol{t}_j)$. We seek to approximate the elements of $\boldsymbol{K}$ as inner products between r-dimensional vectors. In other words, we aim to

find a matrix $\boldsymbol{Z} \in \mathbb{R}^{r \times M}$, such that $\boldsymbol{K} \simeq \boldsymbol{Z}^T\boldsymbol{Z}$. The best such approximation in the least-squares sense is given by $\boldsymbol{Z} = \boldsymbol{\Sigma}^{1/2}\boldsymbol{V}$, with $\boldsymbol{\Sigma}$ and $\boldsymbol{V}$ the top r eigenvalues and corresponding eigenvectors of $\boldsymbol{K}$. From the Nyström method, for a new sample $\boldsymbol{x} \in \mathbb{X}$, the r-dimensional vector representation of the space induced by $k(\boldsymbol{x}, \cdot)$ can be written as

$$z_N(\boldsymbol{x}) = \boldsymbol{\Sigma}^{-1/2}\boldsymbol{V}\Big[k(\boldsymbol{x}, \boldsymbol{t}_1), \cdots, k(\boldsymbol{x}, \boldsymbol{t}_M)\Big]^T. \quad (7)$$

Given a set of local descriptors $\mathbb{X} = \{\boldsymbol{x}_i\}$, our **nVLAD** algorithm then consists of computing the corresponding $\{z_N(\boldsymbol{x}_i)\}$, and making use of Eq. 1 and Eq. 2 with this new representation.

3.2. Fourier Approximation (fVLAD)

The previous approximation applies to general kernels defined on both Euclidean and non-Euclidean data. In the Euclidean case, however, other approximations have been proposed for specific kernels [33, 45]. Since our experiments on Euclidean data all rely on RBF kernels, here, we discuss an approximation of this type of kernels based on the Bochner Theorem [34].

According to the Bochner Theorem [34], a shift-invariant kernel[2], such as the Euclidean RBF kernel, can be expressed with the Fourier integral. As shown in [33], for real-valued kernels, this can be written as

$$k(\boldsymbol{x}_i - \boldsymbol{x}_j) = \int_{\mathbb{R}^d} p(\omega) z_F(\boldsymbol{x}_i) z_F(\boldsymbol{x}_j) d\omega, \quad (8)$$

where $z_F(\boldsymbol{x}) = \sqrt{2}\cos(\omega^T\boldsymbol{x} + b)$, with b a random variable drawn from $[0, 2\pi]$. In other words, $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = k(\boldsymbol{x}_i - \boldsymbol{x}_j)$ is the expected value of $z_F(\boldsymbol{x}_i)z_F(\boldsymbol{x}_j)$ under the distribution $p(\omega)$. For the RBF kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|(\boldsymbol{x}_i - \boldsymbol{x}_j)\|^2/2\sigma^2)$, we have $p(\omega) = \mathcal{N}(0, \sigma^{-2}\mathbf{I}_d)$.

Let $\{\omega_i\}_{i=1}^r$, $\omega_i \in \mathbb{R}^d$, be i.i.d. samples drawn from the normal distribution $\mathcal{N}(0, \sigma^{-2}\mathbf{I}_d)$, and $\{b_i\}_{i=1}^r$ be samples uniformly drawn from $[0, 2\pi]$. Then, the r-dimensional estimate of $\phi(\boldsymbol{x}) \in \mathcal{H}$ is given by

$$z_F(\boldsymbol{x}) = \sqrt{\frac{2}{r}}\Big[\cos(\omega_1^T\boldsymbol{x} + b_1), \cdots, \cos(\omega_r^T\boldsymbol{x} + b_r)\Big]. \quad (9)$$

Similarly to nVLAD, we can then compute $z_F(\boldsymbol{x}_i)$ for each local descriptor $\boldsymbol{x}_i$, and use Eq. 1 and Eq. 2 to obtain a code. In our experiments, we refer to this approach, which only applies to Euclidean data, as **fVLAD**.

3.3. Kernelizing Fisher Vectors

Due to the connection between VLAD and FVs, it seems natural to rely on the ideas discussed above to kernelize FVs. One difficulty in kernelizing FVs, however, arises from the fact that Gaussian distributions, which are required to model the probability distributions in FVs, are not well-defined in RKHS. More specifically, to fit a Gaussian distribution in a d-dimensional space, at least d independent observations (training samples) are required, to ensure that the covariance matrix of the distribution is not rank deficient. Obviously, for an infinite dimensional RKHS, this requirement cannot be met. While, in principle, it is possible to regularize the distributions, *e.g.*, [52], we believe that an in-depth analysis of this approach to kernelize FVs goes beyond the scope of this paper. Note, however, that our approximations of $\mathcal{H}$ can be applied verbatim to derive approximate formulations of kernel FVs.

[2] A kernel function is shift invariant if $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = k(\boldsymbol{x}_i - \boldsymbol{x}_j)$.

4. Experiments

We now evaluate our algorithms, *i.e.*, kVLAD and sVLAD, on several recognition tasks. As mentioned before, our main motivation for this work was to be able to exploit the power of the VLAD aggregation scheme to tackle problems where the input data is not in vectorial form. Therefore, we focus on two such types of data, which have become increasingly popular in computer vision, namely Covariance Descriptors (CovDs), which lie on SPD manifolds, and linear subspaces, which form Grassmann manifolds. Nevertheless, in addition to this manifold-valued data, we also evaluate our algorithms in Euclidean space.

4.1. SPD Manifold

In computer vision, SPD matrices have been shown to provide powerful representations for images and videos via region covariances [41]. Such representations have been successfully employed to categorize, *e.g.*, textures [41, 17], pedestrians [42] and faces [17].

SPD matrices can be thought of as an extension of positive numbers and form the interior of the positive semidefinite cone. It is possible to directly employ the Frobenius norm as a similarity measure between SPD matrices, hence analyzing problems involving such matrices via Euclidean geometry. However, as several studies have shown, undesirable phenomena may occur when Euclidean geometry is utilized to manipulate SPD matrices [29, 42, 20]. Here, instead, we make use of the Stein divergence defined as

$$\delta_S^2(\boldsymbol{A}, \boldsymbol{B}) = \ln\det\Big(\frac{\boldsymbol{A} + \boldsymbol{B}}{2}\Big) - \frac{1}{2}\ln\det\big(\boldsymbol{A}\boldsymbol{B}\big)\ . \quad (10)$$

This divergence was shown to yield a positive definite Gaussian kernel [39], named the Stein kernel given by $k_{\mathcal{S}} : \mathcal{S}_{++}^n \times \mathcal{S}_{++}^n \rightarrow \mathbb{R}$ such that $k_{\mathcal{S}}(\boldsymbol{A}, \boldsymbol{B}) = \exp(-\sigma\delta_S^2(\boldsymbol{A}, \boldsymbol{B}))$. In all our experiments on SPD manifolds, the bandwidth of this kernel was determined by cross-validation on the training data.

A standard approach when dealing with an SPD manifold consists of flattening the manifold using the diffeo-

morphism $\log : \mathcal{S}^n_{++} \rightarrow \mathrm{Sym(n)}$, where $\log$ and $\mathrm{Sym(n)}$ denote the principal matrix logarithm and the space of symmetric matrices of size n, respectively. Given that $\mathrm{Sym(n)}$ is a vector space, one can then directly employ tools from Euclidean geometry, here the VLAD algorithm, to analyze SPD matrices mapped to that space. We refer to this baseline as log-Euclidean VLAD or *lE-VLAD* following the terminology used in [3]. Note that this strategy has been successfully employed in several recent studies (*e.g.*, for semantic segmentation [5]). We also report the results of the Nyström approximation (nVLAD) of Section 3.1.[3]

Furthermore, we also compare our algorithms against the state-of-the-art Weighted ARray of COvariances (WARCO) [40], Riemannian-VLAD (R-VLAD) [11], Covariance Discriminative Learning (CDL) [47], and Riemannian Sparse Representation using the Stein divergence (RSR-S) [15]. In WARCO, an image is decomposed into a number of overlapping patches, each of which is represented with a CovD. Classification is then performed by combining the output of a set of kernel classifiers trained on local patches. R-VLAD makes use of the tangent bundle of a Riemannian manifold to aggregate manifold-valued data in a similar manner to VLAD. In essence, WARCO and R-VLAD pursue the same goal as us, *i.e.*, to aggregate local non-vectorial descriptors, which makes them probably the most relevant baselines, here. By contrast, following [47, 15], we have used both CDL and RSR-S holistically, *i.e.*, every image was described by one SPD matrix.

In the following experiments on the SPD manifold, we used a codebook of size 32 for all variants of the VLAD algorithm. Empirically, we observed that, for any algorithm, larger codebooks did not significantly improve the performance. To provide a fair comparison against WARCO, we used the same set of features as [40]. Specifically, from a local patch, a 13×13 CovD was extracted using the features

$$f(x,y) = [h_1(Y), \cdots, h_8(Y), Y, C_b, C_r, \|g(Y)\|, \angle(g(Y))]^T,$$

where $f(x,y)$ denotes the feature vector at location (x,y) and Y, C_b and C_r are the three color channels from the CIELab color space at (x,y). $h_i(\cdot)$ is the scaled symmetric Difference of Offset Gaussian filter bank, and $\|g(Y)\|$ and $\angle(g(Y))$ are the gradient magnitude and orientation calculated on the Y channel (see [40] for details). The same set of features was used for all the algorithms.

Head Orientation Classification. As a first experiment, we considered the problem of classifying head orientation using the *QMUL* and *HOCoffee* datasets [40]. The QMUL head dataset contains 19292 images of size 50×50, captured in an airport terminal. The HOCoffee dataset contains 18117 head images of size 50×50. The images typically include a margin of 10 pixels on average, so that the actual average dimension of the heads is 30×30 pixels. Both datasets come with predefined training and test samples.

[3]The Fourrier approximation fVLAD only applies to Euclidean data.

Method	QMUL	HOCoffee	HOC
R-VLAD [11]	91.8%	85.0%	81.3%
WARCO [40]	91%	80%	78%
CDL [47]	81.6%	71.2%	77.8%
RSR-S [15]	82.7%	65.9%	78.9%
lE-VLAD	87.6%	82.4%	79.7%
nVLAD	88.9%	83.4%	81.4%
kVLAD	92.2%	**85.3 %**	83.1%
sVLAD	**92.7%**	84.0%	**84.1 %**

Table 1. Recognition accuracies for QMUL, HOCoffe and HOC.

The results of all the algorithms on both datasets are reported in Table 1. Note that kVLAD outperforms the state-of-the-art on both datasets, and that sVLAD even outperforms kVLAD on QMUL. This can be attributed to the square root normalization, which is not possible for kVLAD. Without this normalization, the performance of sVLAD drops by roughly 1%, and thus remains close to, but slightly lower than that of kVLAD. Among the approximations, sVLAD is superior to nVLAD. This is not really surprising, since nVLAD uses a single subspace for all its codewords, whereas sVLAD exploits local representations.

Body Orientation Classification. As a second task on the SPD manifold, we considered the problem of determining body orientation from images using the Human Orientation Classification (HOC) dataset [40]. The HOC dataset contains 11881 images of size 64×32 and comprises 4 orientation classes (Front, Back, Left, and Right). In Table 1, we compare the performance of our algorithms with the baselines. First, we note that all VLAD variants, including R-VLAD, lE-VLAD and nVLAD, are superior to WARCO. This demonstrates the effectiveness of the VLAD aggregation scheme. Moreover, we note that our algorithms outperform R-VLAD, lE-VLAD and nVLAD. The highest accuracy is obtained by sVLAD, which, again, in comparison to kVLAD, benefits from the square root normalization.

Altogether, our experiments on SPD manifolds demonstrate that our approach offers an attractive solution to exploiting the information from local patches. Note that, except for a handful of studies (*e.g.*, WARCO, R-VLAD), CovDs are usually extracted from entire images, hence making them questionable for challenging classification tasks. This is typically due to the fact that aggregating non-vectorial data is an open problem, to which we provide a solution in this paper.

4.2. Grassmann Manifold

The space of p dimensional subspaces in $\mathbb{R}^d$ for $0 < p \leq d$ is not a Euclidean space, but a Riemannian manifold known as the Grassmann manifold $\mathcal{G}(d,p)$. A point $\mathcal{U} \in \mathcal{G}(d,p)$ is typically represented by a $d \times p$ matrix $\boldsymbol{U}$

with orthonormal columns, such that $\mathcal{U} = \mathrm{Span}(\boldsymbol{U})$. The choice of the basis to represent $\mathcal{U}$ is arbitrary and metrics on $\mathcal{G}(d,p)$ are defined so as to be invariant to this choice. The projection distance is a typical choice of such metric. It was recently shown to induce a valid positive definite kernel on $\mathcal{G}(d,p)$ [18], *i.e.*, the projection RBF kernel defined as

$$k_p(\boldsymbol{A},\boldsymbol{B}) = \exp(\sigma\|\boldsymbol{A}^T\boldsymbol{B}\|_F^2),\ \sigma > 0\ . \qquad (11)$$

As for the SPD manifold, the bandwidth of this kernel was obtained by cross-validation on the training data.

Several state-of-the-art image-set matching methods model sets of images as subspaces [16, 18]. However, to the best of our knowledge, all these methods rely on a holistic subspace representation. The only exception is again R-VLAD [11], which, as our approach, can aggregate local subspaces obtained by breaking an image-set into smaller blocks to form a complete image-set descriptor.

In our experiments, in addition to R-VLAD and nVLAD, we compare the results of our algorithms with four baselines: First, similarly to the log-Euclidean approach on SPD manifolds, we propose to flatten $\mathcal{G}(d,p)$ at $\mathbf{I}_{d\times p}$[4] and perform conventional VLAD in the resulting Euclidean space. We refer to this method as lE-VLAD. As a second baseline, we make use of the state-of-the-art Grassmannian Sparse Coding (gSC) algorithm of [16], which describes each image-set with a single linear subspace. We also employ the kernel version of the Affine Hull Method (kAHM) introduced in [6] and the CDL algorithm [47] as other state-of-the-art baselines for image-set matching. Below, we evaluate the performance of our algorithms and of the baselines on three different classification problems, *i.e.*, action classification, object recognition and pose categorization from image-sets.

Action Recognition. As a first experiment on the Grassmannian, we made use of the Ballet dataset [48]. The Ballet dataset consists of 8 complex motion patterns performed by 3 subjects. We extracted 1200 image-sets by grouping 5 frames depicting the same action into one image-set. The local descriptors for each image-set were obtained by splitting the set into small blocks of size $32\times32\times3$ and computing a Histogram of Oriented Gradient (HOG) [8] for each block. We then created subspaces of size 31×3, hence points on $\mathcal{G}(31,3)$. We randomly chose 50% of the image-sets for training and used the remaining sets as test samples. We report the average accuracy over 10 such random splits.

We report the accuracy of all the algorithms in Table 2. Note that all the local approaches outperform the holistic gSC method. The maximum accuracy is obtained by sVLAD, thus showing the power of our approximation.

Given the simplicity of lE-VLAD, it is interesting to verify if it can measure up to our algorithms by enlarging its dictionary. To this end, we increased the size of the dictionary in lE-VLAD up to the point where the performance started to decrease (256 atoms). While this indeed improved the accuracy of lE-VLAD up to, at best, 91.7%, it remains significantly below the performance of sVLAD.

Object Recognition. For the task of object recognition from image-sets, we used the CIFAR dataset [22]. The CIFAR dataset contains 60000 images (32×32 pixels) from 10 different object categories. From this dataset, we generated 6000 image-sets, each one containing 10 random images of the same object. In our experiments, we used 1500 image-sets for training and the remaining 4500 image-sets as test data. We report accuracies averaged over 10 random image-set generation processes. To generate local descriptors, we decomposed each image-set into small blocks of size $8 \times 8 \times 5$. Each block was then represented by a point on $\mathcal{G}(64,5)$ using SVD.

In Table 2, we compare the results of our algorithms with those of the baselines. Here, kVLAD yields the best accuracy, closely followed by sVLAD.

Pose Classification. As a last experiment on the Grassmannian, we evaluated the performance of our algorithms on the task of pose categorization using the CMU-PIE face dataset [37]. The CMU-PIE face dataset contains images of 67 subjects under 13 different poses and 21 different illuminations. The images were closely cropped to enclose the face region and resized to 64×64. We extracted 1700 image-sets by grouping 6 images with the same pose, but different illuminations, into one image-set. The local descriptors for each image-set were obtained by splitting the set into small blocks of size $32\times32\times3$ from which we computed Histogram of LBP [27]. We then created subspaces of size 58×3, hence points on $\mathcal{G}(58,3)$. Table 2 compares the results of our algorithms with those of the baselines. The highest accuracy is obtained by kVLAD, this time by a larger margin over sVLAD, which nonetheless remains the second best. Note that, here, flattening the manifold through its tangent space at $\mathbf{I}_{58\times3}$ seems to incur strong distortions, as indicated by the low performance of lE-VLAD.

4.3. Euclidean Space

Our final experiments are devoted to Euclidean spaces. To this end, we made use of the Pascal VOC 2007 dataset [10] and of the Flicker Material Database (FMD) [36]. The Pascal VOC 2007 dataset [10] contains 9963 images from 20 object categories. The FMD dataset contains 1000 images from 10 different material categories [36]. Both datasets have been extensively used to benchmark coding techniques. For these datasets, the computational cost of kVLAD becomes overwhelming because of the large amount of local descriptors they involve. Therefore, we only report the results of sVLAD. We compare

[4] We use $\mathbf{I}_{d\times p}$ to denote the truncated identity matrix.

Method	Ballet	CIFAR	CMU-PIE
R-VLAD [11]	93.9%	50.5%	69.5%
gSC [16]	79.7%	59.9%	75.5%
kAHM [6]	85.8%	36.1%	55.3%
CDL [47]	73.1%	54.7%	64.6%
lE-VLAD	91.1%	46.2%	59.6%
nVLAD	88.9%	62.2%	79.5%
kVLAD	92.2%	**67.9 %**	**86.3 %**
sVLAD	**94.4%**	65.2%	80.1%

Table 2. Accuracies for Ballet, CIFAR and CMU-PIE.

Method	mAP
SPM [24]	54.3%
OCP [35]	57.2%
Sup-VLAD [28]	**60.9%**
VLAD	54.7%
nVLAD	56.2%
fVLAD	55.8%
sVLAD	60.3%

Table 3. Mean Average Precision (mAP) for the VOC 2007 dataset.

Method	CCR
aLDA [36]	44.6%
MS4C [25]	50.0%
$\mathbf{DTD_{RBF}}$ [7]	53.1%
VLAD	49.4%
nVLAD	52.3%
fVLAD	50.3%
sVLAD	**55.2%**

Table 4. Correct Classification Rate (CCR) for the FMD dataset.

these results with those of the two approximations fVLAD and nVLAD, as well as with those of conventional VLAD (implementation provided in [44]). Note that lE-VLAD and R-VLAD both boil down to conventional VLAD in the Euclidean case.

For these experiments, we set the size of the codebooks to 256 and used SIFT descriptors as local features, with dimensionality reduced to 80 using PCA. For fVLAD and nVLAD, the size of the RKHS was chosen to be 256 (almost 3 times larger than the original space). While increasing the dimensionality of the RKHS could potentially improve the results, it would come at the expense of increasing the computational burden of coding.

For Pascal VOC 2007, Table 3 compares the recognition accuracies of the above-mentioned techniques with the following additional baselines: Spatial Pyramid Matching (SPM) [24], Object-Centric spatial Pooling (OCP) [35] and supervised dictionary learning for VLAD (Sup-VLAD) [28]. Similarly to our experiments on manifolds, sVLAD outperforms the fixed approximation techniques (*i.e.*, fVLAD and nVLAD). Importantly, we observe that the three approximations outperform traditional methods such as SPM and VLAD. Furthermore, sVLAD also outperforms the state-of-the-art pooling method OCP [35], and performs roughly on par with the supervised Sup-VLAD. This latter comparison motivates an interesting future research direction to learn a supervised dictionary in RKHS.

For FMD, Table 4 compares the recognition accuracies of our algorithms with nVLAD, fVLAD, VLAD and the state-of-the-art methods augmented Latent Dirichlet Allocation (aLDA) [36], Multi-Scale Spike-and-Slab Sparse Coding (MS4C) [25], and Describable attributes ($\mathrm{DTD_{RBF}}$) [7]. In essence, we can see that sVLAD outperforms (i) VLAD and the other approximations; and (ii) the state-of-the-art aLDA, MS4C and DTD methods.

4.4. Coding Times

Below, we report the coding times of our two algorithms. For sVLAD, this means the time to build one image descriptor, which is virtually the time to classify one sample. For kVLAD, however, since no descriptor is explicitly built, we report the time to compute Eq. 4. Note that this timing cannot truly be compared to that of sVLAD, but still gives an indication of the speed of kVLAD. All these coding times were obtained on a quad-core machine using Matlab.

When measuring these timings, we used the following parameters (corresponding to our previous experiments). We used a codebook of size 32 for the SPD and Grassmann manifolds, and a codebook of size 256 in Euclidean space. Note that, for the Euclidean case, we assumed that 1000 local descriptors were computed on each image, while, for the SPD and Grassmann manifolds, this number was set to 100. All the coding times are reported in Table 5. For kVLAD, the time to classify one image can be roughly obtained by multiplying the values in the table by the number of training samples. As mentioned before, this makes kVLAD ill-suited for large datasets. By contrast, these timings show that sVLAD takes roughly one second to classify each image, which is quite competitive.

Method	SPD	Grassmann	Euclidean
kVLAD	80ms	155ms	45ms
sVLAD	750ms	1700ms	950ms

Table 5. Coding times for kVLAD and sVLAD (see text for details).

5. Conclusions and Future Work

In this paper, we have introduced a kernel extension of the VLAD encoding scheme. We have also proposed a novel approximation to this kernel formulation in the interest of speeding up the coding process. Not only do the resulting algorithms let us exploit more sophisticated classification schemes in the VLAD framework, but they also allow us to aggregate local descriptors that do not lie in Euclidean space. Our experiments have evidenced that our algorithms outperform state-of-the-art methods, such as WARCO [40] and R-VLAD [11], on several manifold-based recognition tasks. Furthermore, they have also shown that our new coding schemes yield superior results compared to the conventional VLAD algorithm. In the future, we plan to explore possible ways of kernelizing the Fisher vector method [30]. Since our local approximation of a Hilbert space has empirically proven superior to other approximation, we also intend to study its use in other kernel-based algorithms.

References

[1] R. Arandjelovic and A. Zisserman. All about VLAD. In *CVPR*, 2013.

[2] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 1950.

[3] V. Arsigny, O. Commowick, X. Pennec, and N. Ayache. A log-Euclidean framework for statistics on diffeomorphisms. In *MICCAI*, pages 924–931. Springer, 2006.

[4] L. Bo, X. Ren, and D. Fox. Kernel descriptors for visual recognition. In *NIPS*, 2010.

[5] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*. Springer, 2012.

[6] H. Cevikalp and B. Triggs. Face recognition based on image sets. In *CVPR*, pages 2567–2573. IEEE, 2010.

[7] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *CVPR*, pages 3606–3613, 2014.

[8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[9] J. Delhumeau, P.-H. Gosselin, H. Jégou, and P. Pérez. Revisiting the VLAD image representation. In *Proceedings of the ACM international conference on Multimedia*, pages 653–656. ACM, 2013.

[10] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2), 2010.

[11] M. Faraki, M. T. Harandi, and F. Porikli. More about VLAD: A leap from Euclidean to Riemannian manifolds. In *CVPR*, June 2015.

[12] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *CVPR*, 2009.

[13] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, pages 392–407. Springer, 2014.

[14] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, 2005.

[15] M. Harandi, R. Hartley, B. Lovell, and C. Sanderson. Sparse coding on symmetric positive definite manifolds using Bregman divergences. *TNNLS*, PP(99):1–1, 2015.

[16] M. Harandi, R. Hartley, C. Shen, B. Lovell, and C. Sanderson. Extrinsic methods for coding and dictionary learning on Grassmann manifolds. *IJCV*, 114(2):113–136, 2015.

[17] M. Harandi, M. Salzmann, and R. Hartley. From manifold to manifold: Geometry-aware dimensionality reduction for SPD matrices. In *ECCV*. Springer, 2014.

[18] M. Harandi, M. Salzmann, S. Jayasumana, R. Hartley, and H. Li. Expanding the family of Grassmannian kernels: An embedding perspective. In *ECCV*. Springer, 2014.

[19] T. Jaakkola, D. Haussler, et al. Exploiting generative models in discriminative classifiers. In *NIPS*, 1999.

[20] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, and M. Harandi. Kernel methods on Riemannian manifolds with gaussian rbf kernels. *TPAMI*, 2015.

[21] H. Jégou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid. Aggregating local image descriptors into compact codes. *TPAMI*, 34(9):1704–1716, 2012.

[22] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Tech. Rep*, 2009.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[24] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[25] W. Li. Learning multi-scale representations for material classification. In X. Jiang, J. Hornegger, and R. Koch, editors, *Pattern Recognition*, volume 8753, pages 757–764. Springer International Publishing, 2014.

[26] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. In *NIPS*, pages 2627–2635, 2014.

[27] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *TPAMI*, 24(7), 2002.

[28] X. Peng, L. Wang, Y. Qiao, and Q. Peng. Boosting VLAD with supervised dictionary learning and high-order statistics. In *ECCV*. Springer, 2014.

[29] X. Pennec, P. Fillard, and N. Ayache. A Riemannian framework for tensor computing. *IJCV*, 66(1), 2006.

[30] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007.

[31] F. Perronnin, J. Sánchez, and Y. Liu. Large-scale image categorization with explicit data embedding. In *CVPR*, 2010.

[32] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *ECCV*. Springer, 2010.

[33] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, 2007.

[34] W. Rudin. *Fourier analysis on groups*. John Wiley & Sons, 2011.

[35] O. Russakovsky, Y. Lin, K. Yu, and L. Fei-Fei. Object-centric spatial pooling for image classification. In *ECCV*, pages 1–15. Springer, 2012.

[36] L. Sharan, C. Liu, R. Rosenholtz, and E. H. Adelson. Recognizing materials using perceptually inspired features. *IJCV*, 103(3), 2013.

[37] T. Sim, S. Baker, and M. Bsat. The CMU pose, illumination, and expression database. *TPAMI*, 25(12), 2003.

[38] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering objects and their location in images. In *ICCV*, 2005.

[39] S. Sra. A new metric on the manifold of kernel matrices with application to matrix geometric means. In *NIPS*, pages 144–152, 2012.

[40] D. Tosato, M. Spera, M. Cristani, and V. Murino. Characterizing humans on Riemannian manifolds. *TPAMI*, 35(8), 2013.

[41] O. Tuzel, F. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. In *ECCV*. Springer, 2006.

[42] O. Tuzel, F. Porikli, and P. Meer. Pedestrian detection via classification on Riemannian manifolds. *TPAMI*, 30(10), 2008.

[43] J. C. van Gemert, C. J. Veenman, A. W. Smeulders, and J.-M. Geusebroek. Visual word ambiguity. *TPAMI*, 32(7), 2010.

[44] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms, 2008.

[45] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *TPAMI*, 34(3), 2012.

[46] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. Graph kernels. *JMLR*, 11:1201–1242, 2010.

[47] R. Wang, H. Guo, L. Davis, and Q. Dai. Covariance discriminative learning: A natural and efficient approach to image set classification. In *CVPR*, pages 2496–2503, June 2012.

[48] Y. Wang and G. Mori. Human action recognition by semilatent topic models. *TPAMI*, 31(10), 2009.

[49] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In T. Leen, T. Dietterich, and V. Tresp, editors, *NIPS*, pages 682–688. MIT Press, 2001.

[50] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, 2005.

[51] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[52] S. K. Zhou and R. Chellappa. From sample similarity to ensemble similarity: Probabilistic distance measures in reproducing kernel Hilbert space. *TPAMI*, 28(6):917–929, 2006.