# When VLAD met Hilbert

**Mehrtash Harandi** MEHRTASH.HARANDI@NICTA.COM.AU
*NICTA and Australian National University*
*Canberra, Australia*

**Mathieu Salzmann** MATHIEU.SALZMANN@NICTA.COM.AU
*NICTA and Australian National University*
*Canberra, Australia*

**Fatih Porikli** FATIH.PORIKLI@ANU.EDU.AU
*NICTA and Australian National University*
*Canberra, Australia*

## Abstract

Vectors of Locally Aggregated Descriptors (VLAD) have emerged as powerful image/video representations that compete with or even outperform state-of-the-art approaches on many challenging visual recognition tasks. 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.*, tensors) in the VLAD framework. Furthermore, we propose three approximate formulations that allow 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 several nonlinear formulations of *Vectors of Locally Aggregated Descriptors* (VLAD) that generalize 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 intensities, provide robustness to varying imaging conditions. From a historical point of view, this trend was gained momentum by the *Bag-of-Words* (BoW) model Sivic et al. (2005); Grauman and Darrell (2005); Lazebnik et al. (2006), which had a significant impact on recognition performance. Since then, the notable recent developments include dictionary-based solutions Winn et al. (2005); Yang et al. (2009), Fisher Vectors (FV) Perronnin and Dance (2007); Perronnin et al. (2010b), VLAD Jégou et al. (2010); Arandjelovic and Zisserman (2013) and Convolutional Neural Networks (CNN) Krizhevsky et al. (2012).

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.

- 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 properties of FVs. The most eminent one is its theoretical connection to the Fisher kernel Jaakkola et al. (1999).
- 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. 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 Gong et al. (2014).

Despite its unique features, 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, graphs, orthogonal matrices) have the potential to provide more robust descriptors. Furthermore, since VLAD typically yields a high-dimensional image representation, it is mostly restricted to employing linear classifiers. Nonetheless, the effectiveness of kernel-based methods has been proven many a time in visual recognition Gehler and Nowozin (2009); Bo et al. (2010); Perronnin et al. (2010a); Vedaldi and Zisserman (2012a).

In this paper, we present kernel based formulations of VLAD to address the aforementioned shortcomings. In particular, we first introduce a kernelized version of VLAD that relies on mapping of each local descriptor to a Reproducing Kernel Hilbert Space (RKHS). Since several valid kernel functions have recently been defined for non-vector data Jayasumana et al. (2013); Harandi et al. (2014b), such a RKHS mapping can be applied to descriptors on different manifold topologies including SPD matrices and linear subspaces (Grassmannian). Having a RKHS mapping, we can aggregate VLAD over various geometries, thus, ultimately generalize the use of VLAD to local descriptors defined in non-vector spaces. Furthermore, the inherent nonlinearity of mappings to RKHS allows us to exploit more advanced classifiers with kernel VLAD.

In the spirit of computational efficiency, we also design three novel nonlinear approximations of our kernel VLAD; a Nyström method that obtains an explicit mapping to the Hilbert space, a local subspace-based representation of the data in Hilbert space, and a Fourier approximations based on the Bochner theorem. These approximations enjoy the similar properties of kernel VLAD, yet have the additional benefit of providing us with faster coding schemes. Table 1 provides a summary of the proposed methods and their attributes.

Our experimental evaluation demonstrate the effectiveness of our approach at handling manifold-valued data in a VLAD framework. Furthermore, we evidence the benefit 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 (*i.e.*, at patch level), which are then aggregated into a global image representation Lazebnik et al. (2006); Perronnin and Dance (2007); Jégou et al. (2010); Perronnin et al. (2010b); van Gemert et al. (2010); Krizhevsky et al. (2012); Arandjelovic and Zisserman (2013).

Table 1: Proposed methods and their properties. **Kernel** denotes the type of kernel function the algorithm can accept. For example, the *fVLAD* algorithm can only work with certain type of kernel functions while the *kVLAD* method can accept all type of kernel functions. **Coding** reflects the form of the output of the algorithm. For example, in the case of *kVLAD*, the output codes are only known implicitly. **Complexity** is the computational load.

| Method | Kernel | Coding | Complexity |
|---|---|---|---|
| **kVLAD** | general | Implicit | High |
| **nVLAD** | general | Explicit | Low |
| **fVLAD** | specific | Explicit | Low |
| **sVLAD** | general | Explicit | Low |

When large amounts of training data are available, CNNs have now emerged as the method of choice to learn local descriptors. With 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 for the past decade. For instance, Lazebnik et al. (2006) aggregates histograms computed over different spatial regions. More recent developments, such as FVs Perronnin and Dance (2007) and VLAD Jégou et al. (2010), suggest that high-order statistics should be encoded in the aggregation process.

In a separate line of research, structured descriptors (*e.g.*, covariance descriptors or linear subspaces) have been shown to provide robust visual models Tuzel et al. (2008); Jayasumana et al. (2013); Harandi et al. (2013). 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 in visual recognition.

While a full review of kernel-based methods in computer vision is beyond the scope of this paper, the recent work of Mairal et al. (2014) is of particular relevance here. Mairal et al. (2014) 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, we derive several nonlinear formulations of VLAD. To this end, we first start by reviewing the conventional VLAD and then discuss our approach to kernelizing it, followed by three approximations of the resulting kernel VLAD.

### 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 Jégou et al. (2010), the input space $\mathbb{R}^d$ is partitioned into $m$ Voronoi cells by means of a codebook $\mathcal{C}$ with centers $\{\boldsymbol{c}_j\}_{j=1}^{m},\ \boldsymbol{c}_i \in \mathbb{R}^d$. To obtain the codebook, the k-means algorithm is typically employed. Nevertheless, the use of supervised algorithms has also recently been advocated to build more discriminative codebooks Peng et al. (2014). The VLAD code $\boldsymbol{v} \in \mathbb{R}^{md}$ for the query set $\mathcal{X}$ is obtained by concatenating $m$ Local Difference Vectors (LDV) $\delta_j$ storing, for each center, the sum of the differences between this center and each local descriptor assigned to this center. This can be written as

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

where

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

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 a vectorial 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 losses Pennec et al. (2006); Tuzel et al. (2006, 2008); Jayasumana et al. (2013). Here, we propose to 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 $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$. For example, $\mathbb{X}$ could be the space of SPD matrices, with the Gaussian kernel defined in Sra (2012); Jayasumana et al. (2013). According to the Moore-Aronszajn Theorem Aronszajn (1950), 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 $\phi : \mathbb{X} \to \mathcal{H}$, 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 $\phi$ 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 kmeans. To compute a VLAD code in $\mathcal{H}$, we need to provide solutions for 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

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

Therefore, for each local descriptor, the nearest codeword can can be determined using kernel values only, *i.e.*, without having to know the mapping $\phi$, which lets us directly define the assignments.

Unfortunately, expressing the LDVs in $\mathcal{H}$ is not this 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 $\phi$ 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

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

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

$$\begin{aligned}
\Big\langle \boldsymbol{v}_\mathcal{H}(\mathcal{X}), \boldsymbol{v}_\mathcal{H}(\mathcal{Y}) \Big\rangle_\mathcal{H} &= \sum_{s=1}^{m} \delta_s^T(\mathcal{X})\delta_s(\mathcal{Y}) \\
&= \sum_{s=1}^{m}\sum_{i=1}^{N_\mathcal{X}}\sum_{j=1}^{N_\mathcal{Y}} a_s^i a_s^j \Big(\phi(\boldsymbol{c}_s) - \phi(\boldsymbol{x}_i)\Big)^T \Big(\phi(\boldsymbol{c}_s) - \phi(\boldsymbol{y}_j)\Big) \\
&= \sum_{s=1}^{m}\sum_{i=1}^{N_\mathcal{X}}\sum_{j=1}^{N_\mathcal{Y}} a_s^i a_s^j \Big(k(\boldsymbol{x}_i, \boldsymbol{y}_j) + k(\boldsymbol{c}_s, \boldsymbol{c}_s) \\
&\quad - k(\boldsymbol{x}_i, \boldsymbol{c}_s) - k(\boldsymbol{y}_j, \boldsymbol{c}_s)\Big) \ , \qquad (4)
\end{aligned}$$

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

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

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 three approximations to kVLAD, that address this limitation while still benefiting from the nice properties of kVLAD.

### 2.3 Nyström Approximation (nVLAD)

As a first approximation to kVLAD, we propose to make use of the Nyström method. Following Perronnin et al. (2010a), this lets us obtain an explicit form for the mapping $\phi$ to the Hilbert space $\mathcal{H}$, and thus allows us to approximate a given kernel.

1. Note that this will yield a slightly different optimization problem than the standard kernel-based formulation, since in our case the inner product itself depends on several kernel values.

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 $\boldsymbol{Z} \diagdown \times \mathbb{M}$, such that $\boldsymbol{K} \simeq \boldsymbol{Z}^T\boldsymbol{Z}$. The best such approximation in the east-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

$$z_N(\boldsymbol{x}) = \boldsymbol{\Sigma}^{-1/2}\boldsymbol{V}\Big[k(\boldsymbol{x}, \boldsymbol{t}_1), \cdots, k(\boldsymbol{x}, \boldsymbol{t}_M)\Big]^T. \tag{5}$$

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.

## 2.4 Local Subspace Approximation (sVLAD)

Here, we introduce a novel approximation of the Hilbert space $\mathcal{H}$ based on the idea of local subspaces. To this end, we first note that the Nyström approximation yields one single global estimate of $\mathcal{H}$, used across all the codewords and all the descriptors. However, 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. Therefore, there is no reason for the approximation of $\mathcal{H}$ to be shared across all the codewords and descriptors. This motivates us to define approximate 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 $\phi$, such a codeword cannot be explicitly computed, we can still evaluate the kernel function at this codeword, since

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

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

$$\bar{\delta}_s(\mathcal{X}) = \sum_{i=1}^{N} a_s^i\Big(\pi_s\big(\phi(\boldsymbol{c}_s)\big) - \pi_s\big(\phi(\boldsymbol{x}_i)\big)\Big) \ , \tag{6}$$

with $\pi_s : \mathcal{H} \to \mathcal{S}_s$ the projection onto $\mathcal{S}_s$. These projections can be obtained following a similar intuition as for nVLAD. More precisely, 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

$$\pi_s(\boldsymbol{x}) = \Lambda_s^{-1/2}\boldsymbol{U}_s\Big[k(\boldsymbol{x}, \boldsymbol{t}_{s,1}), \cdots, k(\boldsymbol{x}, \boldsymbol{t}_{s,N_s})\Big]. \tag{7}$$

The LDVs $\overline{\delta}_s(\mathcal{X})$ can then be obtained for all codeword $\boldsymbol{c}_s$, and concatenated to for 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 $\{\mathbf{t}_{s,i}\}_{i=1}^{N_s}$.*

### 2.5 Fourier Approximation (fVLAD)

The previous two approximations apply to general kernels and both Euclidean and non-Euclidean data. In the Euclidean case, however, other approximations have been proposed for specific kernels Rahimi and Recht (2007); Vedaldi and Zisserman (2012b). 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 Rudin (2011).

According to the Bochner Theorem Rudin (2011), a shift-invariant kernel[2], such as Euclidean RBF kernel, can be obtained by the Fourier integral. As shown in Rahimi and Recht (2007), for real-valued kernels, this can be expressed as

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

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 form 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

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

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**.

### 2.6 Further Considerations

**Normalization:**
Recent developments have suggested that the discriminatory power of VLAD could be boosted by additional post-processing steps, such as $\ell_2$ power normalization and signed square rooting normalization Arandjelovic and Zisserman (2013); Gong et al. (2014). The $\ell_2$ power normalization, where each block in VLAD is normalized individually, can easily be performed in kVLAD, since

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

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

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

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

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 nVLAD, sVLAD and fVLAD.
**Kernelizing Fisher Vectors:**
Due to the connection between VLAD and FVs, it seems natural to rely on the ideas discussed above to kernerlize FVs. One difficulty in kernelizing FV, however, arises from the fact that Gaussian distributions, which are required to model the probability distributions in FV, 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.*, Zhou and Chellappa (2006), 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 FV.

## 3. Experiments

We now evaluate our different algorithms, i.e., kVLAD, nVLAD, sVLAD and fVLAD, 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 becom 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.

### 3.1 SPD Manifold

In computer vision, SPD matrices have been shown to provide powerful representations for images and videos via region covariances Tuzel et al. (2006). Such representations have been successfully employed to categorize, e.g., textures Tuzel et al. (2006); Harandi et al. (2014a), pedestrians Tuzel et al. (2008) and faces Harandi et al. (2014a).

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 Pennec et al. (2006); Tuzel et al. (2008); Jayasumana et al. (2013). Here, instead, we make use of the Stein divergence defined as

$$\delta_S^2(\boldsymbol{A}, \boldsymbol{B}) = \ln\det\left(\frac{\boldsymbol{A}+\boldsymbol{B}}{2}\right) - \frac{1}{2}\ln\det\left(\boldsymbol{A}\boldsymbol{B}\right) . \tag{10}$$

This divergence was shown to yield a positive definite Gaussian kernel Sra (2012), named the Stein kernel given by $k_{\mathcal{S}} : \mathcal{S}_{++}^{n} \times \mathcal{S}_{++}^{n} \to \mathbb{R}$ 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 diffeomorphism $\log : \mathcal{S}_{++}^{n} \to \mathrm{Sym(n)}$, 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. In our experiments, we refer to this baseline as log-Euclidean VLAD or *le-VLAD* following the terminology used in Arsigny et al. (2006). Note that this strategy has been successfully employed in several recent studies (*e.g.*, for image semantic segmentation Carreira et al. (2012)).

Furthermore, we also compare our algorithms against the state-of-the-art Weighted ARray of COvariances (WARCO) algorithm Tosato et al. (2013). In WARCO, an image is decomposed into a number of overlapped 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. In essence, WARCO pursues the same goal as us, *i.e.*, to aggregate local non-vectorial descriptors, which makes it probably the most relevant baseline, here.

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 use the same set of features as Tosato et al. (2013). More specifically, from a local patch, a $13 \times 13$ CovD is extracted using the features

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

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 Tosato et al. (2013) for details).

**Head Orientation Classification.** As a first experiment, we consider the problem of classifying head orientation using the *QMUL* and *HOCoffee* datasets Tosato et al. (2013). The QMUL head dataset contains 19292 images of size $50 \times 50$, captured in an airport terminal. The HOCoffee dataset (see Fig. 2 for examples) contains 18117 head images of size $50 \times 50$. The images typically include a margin of 10 pixels on average, so that the actual average dimension of the heads is $30 \times 30$ pixels. Both datasets come with predefined training and test samples.

In Table. 2, we report the performance of kVLAD, sVLAD and nVLAD, as well as of WARCO and le-VLAD, on the QMUL and HOCoffee datasets. Note that kVLAD and sVLAD both yield higher accuracies than the state-of-the-art WARCO algorithm. For example, on HOCoffee, the accuracy of kVLAD surpasses that of WARCO by more than 5%. Note also that kVLAD and sVLAD yield very similar accuracies, which evidences the good quality of our local subspace approximation. Interestingly, 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,

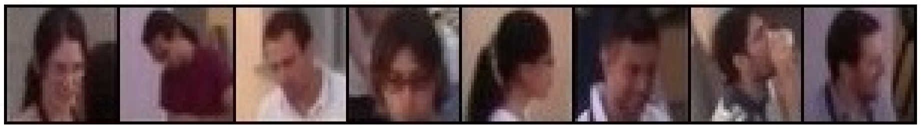
Figure 1: Samples from the HOCoffee dataset.

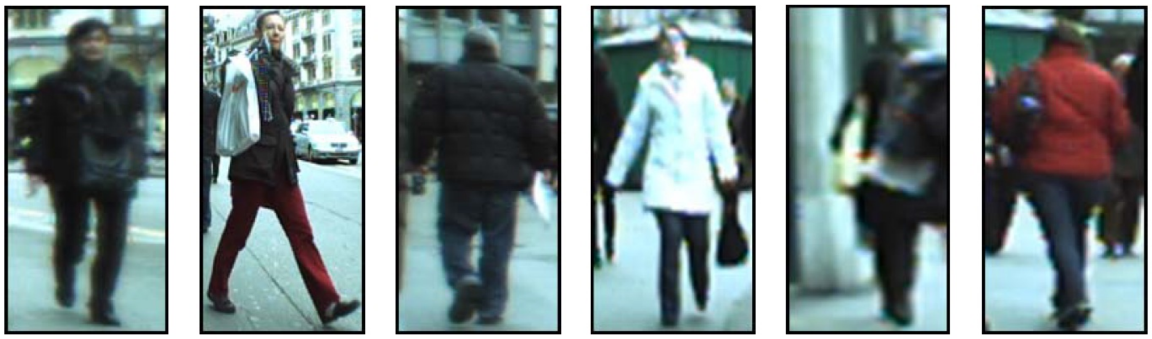
Figure 2: Samples from the HOC dataset.

since nVLAD relies on a single subspace for all its codewords, whereas sVLAD exploits more local representations.

**Body Orientation Classification.** As a second task on the SPD manifold, we consider the problem of determining body orientation from images using the Human Orientation Classification (HOC) dataset Tosato et al. (2013). The HOC dataset contains 11881 images of size $64 \times 32$ (see Fig. 1 for examples) and comprises 4 orientation classes (Front, Back, Left, and Right). In Table. 2, we compare the performance of kVLAD, sVLAD and nVLAD against that of WARCO and le-VLAD. First, we note that all VLAD variants, including le-VLAD, are superior to the WARCO algorithm. This demonstrates the effectiveness of the VLAD aggregation scheme. Moreover, we note that all our algorithms outperform le-VLAD. 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), 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 is an open problem, to which we provide a solution in this paper.

Table 2: Recognition accuracies for QMUL, HOCoffe and HOC.

| **Method** | **QMUL** | **HOCoffee** | **HOC** |
|---|---|---|---|
| **lE-VLAD** | 87.6% | 77.2% | 79.7% |
| **WARCO** Tosato et al. (2013) | 91% | 80% | 78% |
| **nVLAD** | 88.9% | 83.4% | 81.4% |
| **sVLAD** | 92.7% | 84.0% | 84.1% |
| **kVLAD** | 92.2% | 85.3% | 83.1% |

## 3.2 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 $\mathcal{G}(p, d)$. A point $\mathcal{U} \in \mathcal{G}(p, d)$ 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 $\mathcal{G}(p, d)$ 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 $\mathcal{G}(p, d)$ Harandi et al. (2014b), *i.e.*, the projection RBF kernel defined as

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

As for the SPD manifold, in our experiments, 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 Harandi et al. (2013, 2014b). However, to the best of our knowledge, all these methods rely on a holistic subspace representation. This again is probably due to the fact that, before this paper, no aggregation schemes on Grassmann manifolds have ever been proposed. Our approach, by contrast, enables us to break an image-set into smaller blocks, represent each block by a linear subspace, and aggregating these subspace to form a complete image-set descriptor.

In our experiments, we compare the results of our algorithms against two baselines: First, similarly to the log-Euclidean approach on SPD manifolds, we propose to flatten $\mathcal{G}(p, d)$ at $\mathbf{I}_{d \times p}$[3] and perform conventional VLAD in the resulting Euclidean space. We will 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 Harandi et al. (2013), which describes each image-set with a single linear subspace.

Below, we evaluate the performance of our algorithms and of the baselines on three different classification problems, *i.e.*, object recognition, action classification and pose categorization from image-sets.

**Action Recognition.** As a first experiment on the Grassmannian, we make use of the Ballet dataset Wang and Mori (2009). The Ballet dataset consists of 8 complex motion patterns performed by 3 subjects (see Fig. 3 for examples). 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 $32 \times 32 \times 3$ and utilizing Histogram of Oriented Gradient (HOG) Dalal and Triggs (2005). We then created subspaces of size $31 \times 3$, hence points on $\mathcal{G}(3, 31)$. We randomly chose 50% of imagesets for training and used the remaining sets as test samples. The process of random splitting was repeated ten times and the average classification accuracy is reported.

In Table 3, we report the accuracy of algorithms and of the gSC and le-VLAD baselines. First, note that all the local approaches outperform the holistic gSC method. Furthermore, similarly to the two experiments on SPD manifolds, the maximum accuracy is obtained by sVLAD, closely followed by kVLAD.

**Object Recognition.** For the task of object recognition from image-sets, we used the CIFAR dataset Krizhevsky and Hinton (2009). The CIFAR dataset contains 60000 images ($32 \times 32$ pixels) from 10 different object categories. From this dataset, we generated 6000 image-sets, each one

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

Figure 3: Samples from the Ballet dataset.

Figure 4: Samples from CMU-PIE.

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 $\mathcal{G}(5, 64)$ using SVD. In Table. 3, we compare the results of kVLAD, sVLAD and nVLAD against those of le-VLAD and gSC. Here, kVLAD yields the best accuracy 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 Sim et al. (2003). The CMU-PIE face dataset contains images of 67 subjects under 13 different poses and 21 different illuminations (see Fig. 4 for examples). The images were closely cropped to enclose the face region and resized to $64 \times 64$. 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 $32 \times 32 \times 3$ from which we computed Histogram of LBP Ojala et al. (2002). We then created subspaces of size $58 \times 3$, hence points on $\mathcal{G}(3, 58)$. Table 3 compares the results of nVLAD, sVLAD and kVLAD against those of gSC and le-VLAD. The highest accuracy is obtained by kVLAD, this time by a large margin over the second best, sVLAD. Note that, with this dataset, flattening the manifold through its tangent space at $\mathbf{I}_{58 \times 3}$ seems to incur strong distortions, as indicated by low performance of le-VLAD.

### 3.3 Euclidean Space

Our final experiments are devoted to Euclidean spaces. To this end, we compare the performance of sVLAD, fVLAD and nVLAD against the conventional VLAD (implementation provided in Vedaldi and Fulkerson (2008)) on Pascal VOC 2007 Everingham et al. (2010) and on the Flicker Material Database (FMD) Sharan et al. (2013). Pascal VOC 2007 Everingham et al. (2010) contains 9963 images from 20 object categories. The FMD contains 1000 images from 10 different material

Table 3: Accuracies for Ballet, CIFAR and CMU-PIE.

| Method | Ballet | CIFAR | CMU-PIE |
|---|---|---|---|
| **gSC** | 79.7% | 59.9% | 75.5% |
| **le-VLAD** | 91.1% | 46.2% | 59.6% |
| **nVLAD** | 88.9% | 62.2% | 79.5% |
| **sVLAD** | 94.4% | 65.2% | 80.1% |
| **kVLAD** | 92.2% | 67.9% | 86.3% |

Figure 5: Examples of the FMD texture dataset.

categories Sharan et al. (2013). Both datasets have been extensively used to benchmark coding techniques.

In our experiments, we set the size of the codebook to 256 and extracted SIFT descriptors (with whitening) as local features. 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.

Table. 4 compares the recognition accuracies of the proposed coding techniques against conventional VLAD. Similarly to our experiments on manifolds, sVLAD outperforms the fixed approximation techniques (*i.e.*, fVLAD and nVLAD). Among these fixed approximation techniques, nVLAD is superior to fVLAD on these two datasets. Nevertheless, all three algorithms achieve higher accuracies than the conventional VLAD. For example, the difference between sVLAD and conventional VLAD on Pascal VOC07 exceeds 5%. We acknowledge that the computational load of kVLAD becomes overwhelming on Pascal VOC07 and FMD as a result of large amount of local descriptors.

Table 4: Recognition accuracies for VOC07 (mAP) and FMD datasets.

| Method | VOC07 | FMD |
|---|---|---|
| **VLAD** | 54.7% | 49.4% |
| **nVLAD** | 56.2% | 52.3% |
| **fVLAD** | 55.8% | 50.3% |
| **sVLAD** | 60.3% | 55.2% |

Table 5: Running times for fVLAD, nVLAD, sVLAD and kVLAD on three different geometries. Note that the running times for fVLAD, nVLAD and sVLAD show the coding time for an image/video, while, in the case of kVLAD where not explicit encoding is performed, it shows the time needed to evaluate Eq. 4

| Method | SPD | Grassmann | Euclidean |
|---|---|---|---|
| **nVLAD** | 650ms | 1600ms | 35ms |
| **fVLAD** | N/A | N/A | 100ms |
| **sVLAD** | 750ms | 1700ms | 950ms |
| **kVLAD** | 80ms | 155ms | 45ms |

### 3.4 Encoding times

Before concluding, we provide the coding times for the proposed methods on the three different geometries studied in this work. In particular, we measured the encoding times of sVLAD, fVLAD and nVLAD on a Quad-core machine using Matlab. We also measured the running time to compute Eq. 4, which shows the computational load of kVLAD.

The parameters values of the algorithms when measuring these timings were those used in our experiments. More specifically, for the SPD and Grassmann manifolds, the size of codebook was chosen to be 32, while, in the case of Euclidean space, it was set to 256. 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. Table 5 reports all the running times.

## 4. Conclusions and Future Work

In this paper, we have introduced a kernel extension of the VLAD encoding scheme. We have also proposed several approximations to this kernel formulation in the interest of speeding up the encoding process. Not only do the resulting algorithm 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 Tosato et al. (2013) and gSC Harandi et al. (2013), on several manifold-based recognition tasks. Furthermore, they have also shown that our new encoding schemes yield superior results compared to the conventional VLAD algorithm. In the future, we plan to explore possible ways of kernelizing the Fisher vector Perronnin and Dance (2007) method. We also intend to study the concept of coresets Har-Peled and Mazumdar (2004) to reduce the computational complexity of coding.

## References

Relja Arandjelovic and Andrew Zisserman. All about vlad. In *CVPR*, 2013.

Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 1950.

Vincent Arsigny, Olivier Commowick, Xavier Pennec, and Nicholas Ayache. A log-euclidean framework for statistics on diffeomorphisms. In *MICCAI*. 2006.

Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Kernel descriptors for visual recognition. In *NIPS*, 2010.

Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*. 2012.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2), 2010.

Peter Gehler and Sebastian Nowozin. On feature combination for multiclass object classification. In *CVPR*, 2009.

Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*. 2014.

Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, 2005.

Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *ACM symposium on Theory of computing*, 2004.

Mehrtash Harandi, Conrad Sanderson, Chunhua Shen, and Brian Lovell. Dictionary learning and sparse coding on grassmann manifolds: An extrinsic solution. In *ICCV*, 2013.

MehrtashT. Harandi, Mathieu Salzmann, and Richard Hartley. From manifold to manifold: Geometry-aware dimensionality reduction for spd matrices. In *ECCV*. 2014a.

MehrtashT. Harandi, Mathieu Salzmann, Sadeep Jayasumana, Richard Hartley, and Hongdong Li. Expanding the family of grassmannian kernels: An embedding perspective. In *ECCV*. 2014b.

Tommi Jaakkola, David Haussler, et al. Exploiting generative models in discriminative classifiers. In *NIPS*, 1999.

Sadeep Jayasumana, Richard Hartley, Mathieu Salzmann, Hongdong Li, and Mehrtash Harandi. Kernel methods on the riemannian manifold of symmetric positive definite matrices. In *CVPR*, 2013.

Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Tech. Rep*, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. In *NIPS*. 2014.

Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *TPAMI*, 24(7), 2002.

Xiaojiang Peng, Limin Wang, Yu Qiao, and Qiang Peng. Boosting vlad with supervised dictionary learning and high-order statistics. In *ECCV*. 2014.

Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A riemannian framework for tensor computing. *IJCV*, 66(1), 2006.

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

Florent Perronnin, Jorge Sánchez, and Yan Liu. Large-scale image categorization with explicit data embedding. In *CVPR*, 2010a.

Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*. 2010b.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NIPS*, 2007.

Walter Rudin. *Fourier analysis on groups*. John Wiley & Sons, 2011.

Lavanya Sharan, Ce Liu, Ruth Rosenholtz, and Edward H Adelson. Recognizing materials using perceptually inspired features. *IJCV*, 103(3), 2013.

Terence Sim, Simon Baker, and Maan Bsat. The cmu pose, illumination, and expression database. *TPAMI*, 25(12), 2003.

Josef Sivic, Bryan C Russell, Alexei A Efros, Andrew Zisserman, and William T Freeman. Discovering objects and their location in images. In *ICCV*, 2005.

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

Diego Tosato, Mauro Spera, Marco Cristani, and Vittorio Murino. Characterizing humans on riemannian manifolds. *TPAMI*, 35(8), 2013.

Oncel Tuzel, Fatih Porikli, and Peter Meer. Region covariance: A fast descriptor for detection and classification. In *ECCV*. 2006.

Oncel Tuzel, Fatih Porikli, and Peter Meer. Pedestrian detection via classification on Riemannian manifolds. *TPAMI*, 30(10), 2008.

Jan C van Gemert, Cor J Veenman, Arnold WM Smeulders, and J-M Geusebroek. Visual word ambiguity. *TPAMI*, 32(7), 2010.

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

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

Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *TPAMI*, 34(3), 2012b.

Yang Wang and Greg Mori. Human action recognition by semilatent topic models. *TPAMI*, 31(10), 2009.

John Winn, Antonio Criminisi, and Thomas Minka. Object categorization by learned universal visual dictionary. In *ICCV*, 2005.

Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

Shaohua Kevin Zhou and Rama Chellappa. From sample similarity to ensemble similarity: Probabilistic distance measures in reproducing kernel hilbert space. *TPAMI*, 28(6), 2006.