

Non-Linear Temporal Subspace Representations for Activity Recognition

Anoop Cherian^{1,3} Suvrit Sra² Stephen Gould³ Richard Hartley³

¹MERL, Cambridge MA, ²MIT, Cambridge MA, ³ACRV, ANU Canberra

cherian@merl.com suvrit@mit.edu {stephen.gould, richard.hartley}@anu.edu.au

Abstract

Representations that can compactly and effectively capture the temporal evolution of semantic content are important to computer vision and machine learning algorithms that operate on multi-variate time-series data. We investigate such representations motivated by the task of human action recognition. Here each data instance is encoded by a multivariate feature (such as via a deep CNN) where action dynamics are characterized by their variations in time. As these features are often non-linear, we propose a novel pooling method, kernelized rank pooling, that represents a given sequence compactly as the pre-image of the parameters of a hyper-plane in a reproducing kernel Hilbert space, projections of data onto which captures their temporal order. We develop this idea further and show that such a pooling scheme can be cast as an order-constrained kernelized PCA objective. We then propose to use the parameters of a kernelized low-rank feature subspace as the representation of the sequences. We cast our formulation as an optimization problem on generalized Grassmann manifolds and then solve it efficiently using Riemannian optimization techniques. We present experiments on several action recognition datasets using diverse feature modalities and demonstrate state-of-the-art results.

1. Introduction

In this paper, we propose compact representations for non-linear multivariate data arising in computer vision applications, by casting them in the concrete setup of action recognition in video sequences. The concrete setting we pursue is quite challenging. Although, rapid advancement of deep convolutional neural networks has led to significant breakthroughs in several computer vision tasks (e.g., object recognition [22], face recognition [38], pose estimation [56]), action recognition continues to be significantly far from human-level performance. This gap is perhaps due to the spatio-temporal nature of the data and its size, which quickly outgrows processing capabilities of even the best hardware platforms. To tackle this, deep learning algorithms for video processing usually consider subsequences (a few

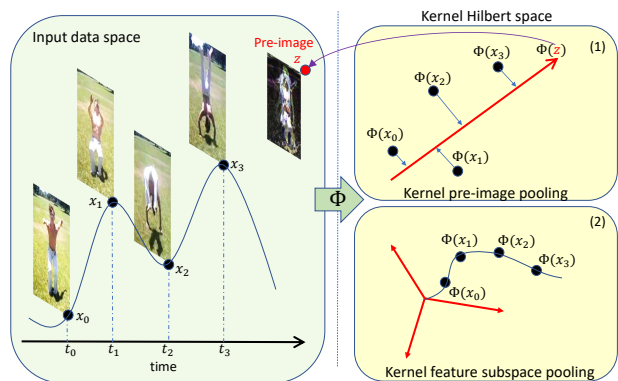


Figure 1. An illustration of our two kernelized rank pooling schemes, namely (1) Pre-image pooling, that uses the pre-image z (in the input space) as the pooled descriptor; this pre-image is computed such that the projections of kernel embeddings $\Phi(x)$ of input points x preserve the temporal frame order when projected onto $\Phi(z)$, and (2) kernel subspace pooling, that uses the parameters of the kernel subspace for pooling, such that the projections of $\Phi(x)$ onto this subspace captures the temporal order (as increasing distances from the subspace origin). Both schemes assume that the input data is non-linear, while their kernelized embeddings (in an infinite dimensional RKHS) may allow compact linear order-preserving projections – which can be used for pooling.

frames) as input, extract features from such clips, and then aggregate these features into compact representations, which are then used to train a classifier for recognition.

In the popular two-stream CNN architecture for action recognition [43, 14], the final classifier scores are fused using a linear SVM. A similar strategy is followed by other more recent approaches such as the 3D convolutional network [46, 4] and temporal segment networks [54]. Given that an action is comprised of ordered variations of spatio-temporal features, any pooling scheme that discards such temporal variation may lead to sub-optimal performance.

Consequently, various temporal pooling schemes have been devised. One recent promising scheme is *rank pooling* [17, 18], in which the temporal action dynamics are summarized as the parameters of a line in the input space that preserves the frame order via linear projections. To

estimate such a line, a rank-SVM [3] based formulation is proposed, where the ranking constraints enforce the temporal order (see Section. 3). However, this formulation is limited on several fronts. First, it assumes the data belongs to a Euclidean space (and thus cannot handle non-linear geometry, or sequences of structured objects such as positive definite matrices, strings, trees, etc.). Second, only linear ranking constraints are used, however non-linear projections may prove more fruitful. Third, data is assumed to evolve smoothly (or needs to be explicitly smoothed) as otherwise the pooled descriptor may fit to random noise.

In this paper, we introduce *kernelized rank pooling* (KRP) that aggregates data features after mapping them to a (potentially) infinite dimensional reproducing kernel Hilbert space (RKHS) via a feature map [42]. Our scheme learns hyperplanes in the feature space that encodes the temporal order of data via inner products; the pre-images of such hyperplanes in the input space are then used as action descriptors, which can then be used in a non-linear SVM for classification. This appeal to kernelization generalizes rank pooling to any form of data for which a Mercer kernel is available, and thus naturally takes care of the challenges described above. We explore variants of this basic KRP in Section 4.

A technical difficulty with KRP is its reliance on the computation of a pre-image of a point in feature space. However, given that the pre-images are finite-dimensional representatives of infinite-dimensional Hilbert space points, they may not be unique or may not even exist [32]. To this end, we propose an alternative kernelized pooling scheme based on feature subspaces (KRP-FS) where instead of estimating a single hyperplane in the feature space, we estimate a low-rank kernelized subspace subject to the constraint that projections of the kernelized data points onto this subspace should preserve temporal order. We propose to use the parameters of this low-rank kernelized subspace as the action descriptor. To estimate the descriptor, we propose a novel order-constrained low-rank kernel approximation, with orthogonality constraints on the estimated descriptor. While, our formulation looks computationally expensive at first glance, we show that it allows efficient solutions if resorting to Riemannian optimization schemes on a generalized Grassmann manifold (Section 4.3).

We present experiments on a variety of action recognition datasets, using different data modalities, such as CNN features from single RGB frames, optical flow sequences, trajectory features, pose features, etc. Our experiments clearly show the advantages of the proposed schemes achieving state-of-the-art results.

Before proceeding, we summarize below the main contributions of this paper.

- We introduce a novel *order-constrained kernel PCA* objective that learns action representations in a kernelized feature space. We believe our formulation may be of

independent interest in other applications.

- We introduce a new pooling scheme, *kernelized rank pooling* based on kernel pre-images that captures temporal action dynamics in an infinite-dimensional RKHS.
- We propose efficient Riemannian optimization schemes on the generalized Grassmann manifold for solving our formulations.
- We show experiments on several datasets demonstrating state-of-the-art results.

2. Related Work

Recent methods for video-based action recognition use features from the intermediate layers of a CNN; such features are then pooled into compact representations. Along these lines, the popular two-stream CNN model [43] for action recognition has been extended using more powerful CNN architectures incorporating intermediate feature fusion in [14, 12, 46, 53], however typically the features are pooled independently of their temporal order during the final sequence classification. Wang et al. [55] enforces a grammar on the two-stream model via temporal segmentation, however this grammar is designed manually. Another popular approach for action recognition has been to use recurrent networks [58, 9]. However, training such models is often difficult [35] and need enormous datasets. Yet another popular approach is to use 3D convolutional filters, such as C3D [46] and the recent I3D [4]; however they also demand large (and clean) datasets to achieve their best performances.

Among recently proposed temporal pooling schemes, rank pooling [17] has witnessed a lot of attention due to its simplicity and effectiveness. There have been extensions of this scheme in a discriminative setting [15, 1, 18, 52], however all these variants use the basic rank-SVM formulation and is limited in their representational capacity as alluded to in the last section. Recently, in Cherian et al. [6], the basic rank pooling is extended to use the parameters of a feature subspace, however their formulation also assumes data embedded in the Euclidean space. In contrast, we generalize rank pooling to the non-linear setting and extend our formulation to an order-constrained kernel PCA objective to learn kernelized feature subspaces as data representations. To the best of our knowledge, both these ideas have not been proposed previously.

We note that kernels have been used to describe actions earlier. For example, Cavazza et al. [5] and Quang et al. [37] propose kernels capturing spatio-temporal variations for action recognition, where the geometry of the SPD manifold is used for classification, and Harandi et al. [29, 19] uses geometry in learning frameworks. Koniusz et al. [27, 26, 7] uses features embedded in an RKHS, however the resultant kernel is linearized and embedded in the Euclidean space. Vemulapalli et al. [50] uses SE(3) geometry to classify pose

sequences. Tseng [47] proposes to learn a low-rank subspace where the action dynamics are linear. Subspace representations have also been investigated [48, 21, 34], and the final representations are classified using Grassmannian kernels. However, we differ from all these schemes in that our subspace is learned using temporal order constraints, and our final descriptor is an element of the RKHS, offering greater flexibility and representational power in capturing non-linear action dynamics. We also note that there have been extensions of kernel PCA for computing pre-images, such as for denoising [32, 31], voice recognition [30], etc., but are different from ours in methodology and application.

3. Preliminaries

In this section, we setup the notation for the rest of the paper and review some prior formulations for pooling multivariate time series for action recognition. Let $X = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]$ be features from n consecutive frames of a video sequence, where we assume each $\mathbf{x}_i \in \mathbb{R}^d$.

Rank pooling [17] is a scheme to compactly represent a sequence of frames into a single feature that summarizes the sequence dynamics. Typically, rank pooling solves the following objective:

$$\min_{\mathbf{z} \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{z}\|^2 + \lambda \sum_{i < j} \max(0, \eta + \mathbf{z}^T \mathbf{x}_i - \mathbf{z}^T \mathbf{x}_j), \quad (1)$$

where $\eta > 0$ is a threshold enforcing the temporal order and λ is a regularization constant. Note that, the formulation in (1) is the standard Ranking-SVM formulation [3] and hence the name. The minimizing vector \mathbf{z} (which captures the parameters of a line in the input space) is then used as the pooled action descriptor for X in a subsequent classifier. The rank pooling formulation in [17] encodes the temporal order as increasing intercept of input features when projected onto this line.

The objective in (1) only considers preservation of the temporal order; as a result, the minimizing \mathbf{z} may not be related to input data at a semantic level (as there are no constraints enforcing this). It may be beneficial for \mathbf{z} to capture some discriminative properties of the data (such as human pose, objects in the scene, etc.), that may help a subsequent classifier. To account for these shortcomings, Cherian et al. [6], extended rank pooling to use the parameters of a subspace as the representation for the input features with better empirical performance. Specifically, [6] solves the following problem.

$$\min_{U \in \mathcal{G}(p, d)} \frac{1}{2} \sum_{i=1}^n \|\mathbf{x}_i - UU^T \mathbf{x}_i\|^2 + \sum_{i < j} \max(0, \eta + \|U^T \mathbf{x}_i\|^2 - \|U^T \mathbf{x}_j\|^2), \quad (2)$$

where instead of a single \mathbf{z} as in (1), they learn a subspace U (belonging to a p -dimensional Grassmann manifold embedded in \mathbb{R}^d), such that this U provides a low-rank approximation to the data, as well as, projection of the data points onto this subspace will preserve their temporal order in terms of their distance from the origin.

However, both the above schemes have limitations; they assume input data is Euclidean, which may be severely limiting when working with features from an inherently non-linear space. To circumvent these issues, in this paper, we explore kernelized rank pooling schemes that generalize rank pooling to data objects that may belong to any arbitrary geometry, for which a valid Mercer kernel can be computed. As will be clear shortly, our schemes generalize both [17] and [6] as special cases when the kernel used is a linear one. In the sequel, we assume an RBF kernel for the feature map, defined for $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d$ as: $\mathbf{k}(\mathbf{x}, \mathbf{z}) = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2}\right\}$, for a bandwidth σ . We use \mathbf{K} to denote the $n \times n$ RBF kernel matrix constructed on all frames in X , i.e., the ij -th element $\mathbf{K}_{ij} = \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)$, where $\mathbf{x}_i, \mathbf{x}_j \in X$. Further, for the kernel \mathbf{k} , let there exist a corresponding feature map $\Phi : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathcal{H}$, where \mathcal{H} is a Hilbert space for which $\langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle_{\mathcal{H}} = \mathbf{k}(\mathbf{x}, \mathbf{z})$, for all $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d$.

4. Our Approach

Given a sequence of temporally-ordered features X , our main idea is to use the kernel trick to map the features to a (plausibly) infinite-dimensional RKHS [42], in which the data is linear. We propose to learn a hyperplane in the RKHS, projections of the data to which will preserve the temporal order. We formalize this idea below and explore variants.

4.1. Kernelized Rank Pooling

For data points $\mathbf{x} \in X$ and their RKHS embeddings $\Phi(\mathbf{x})$, a straightforward way to extend (1) is to use a direction $\Phi(\mathbf{z})$ in the feature space, projections of $\Phi(\mathbf{x})$ onto this line will preserve the temporal order. However, given that we need to retrieve \mathbf{z} in the input space, to be used as the pooled descriptor, one way (which we propose) is to compute the pre-image \mathbf{z} of $\Phi(\mathbf{z})$, which can then be used as the action descriptor in a subsequent non-linear action classifier. Mathematically, this *basic kernelized rank pooling* (BKRP) formulation is:

$$\begin{aligned} \arg \min_{\mathbf{z} \in \mathbb{R}^d} \text{BKRP}(\mathbf{z}) &:= \frac{1}{2} \|\mathbf{z}\|^2 + \\ &\lambda \sum_{i < j} \max(0, \eta + \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{z}) \rangle - \langle \Phi(\mathbf{x}_j), \Phi(\mathbf{z}) \rangle) \quad (3) \\ &= \frac{1}{2} \|\mathbf{z}\|^2 + \lambda \sum_{i < j} \max(0, \eta + \mathbf{k}(\mathbf{x}_i, \mathbf{z}) - \mathbf{k}(\mathbf{x}_j, \mathbf{z})). \quad (4) \end{aligned}$$

As alluded to earlier, a technical issue with (4) (and also with (1)) is that the optimal direction \mathbf{z} may ignore any useful

properties of the original data X (instead could be some line that preserves the temporal order alone). To make sure \mathbf{z} is similar to $\mathbf{x} \in X$, we write an improved (4) as:

$$\begin{aligned} \arg \min_{\mathbf{z} \in \mathbb{R}^d, \xi \geq 0} \text{IBKRP}(\mathbf{z}) &:= \frac{1}{2} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{z}\|^2 + C \sum_{i,j=1}^n \xi_{ij} \\ &+ \lambda \sum_{i < j} \max(0, \eta - \xi_{ij} + \mathbf{k}(\mathbf{x}_i, \mathbf{z}) - \mathbf{k}(\mathbf{x}_j, \mathbf{z})), \end{aligned} \quad (5)$$

where the first component says that the computed pre-image is not far from the input data.¹ The variables ξ represent non-negative slacks and C is a positive constant.

The above formulation assumes a pre-image \mathbf{z} always exists, which may not be the case in general, or may not be unique even if it exists [32]. We could avoid this problem by simply not computing the pre-image, instead keeping the representation in the RKHS itself. To this end, in the next section, we propose an alternative formulation in which we assume useful data maps to a p -dimensional subspace of the RKHS, in which the temporal order is preserved. We propose to use the parameters of this subspace as our sequence representation. Compared to a single \mathbf{z} to capture action dynamics (as described above), a subspace offers a richer representation (as is also considered in [6])

4.2. Kernelized Subspace Pooling

Reusing our notation, let $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ be the n points in \mathbb{R}^d from a sequence. Since it is difficult to work in the complete Hilbert space \mathcal{H} , we restrict ourselves to the subspace of \mathcal{H} spanned by the $\Phi(\mathbf{x}_i), i = 1, 2, \dots, n$. For convenience, let this space be called \mathcal{H} . Assuming that they are all linearly independent, the Representer theorem [51, 40] says that the points $\{\Phi(\mathbf{x}_i)\}$ can be chosen as a basis for \mathcal{H} (not in general an orthonormal basis, however). In this case, \mathcal{H} is a space of dimension n .

As alluded to above, we are concerned with the case where all the $\Phi(\mathbf{x}_i)$ may lie close to some p -dimensional subspace of \mathcal{H} , denoted by V . This space will initially be unknown, and is to be determined in some manner. Denote by $\Omega_p : \mathcal{H} \rightarrow V$, the orthogonal projection from \mathcal{H} to V . Assume that V has an orthonormal basis, $\{\mathbf{e}_i \mid i = 1, \dots, p\}$. In terms of the basis $\{\Phi(\mathbf{x}_j)\}$ for \mathcal{H} , we can write

$$\mathbf{e}_i = \sum_{j=1}^n a_{ij} \Phi(\mathbf{x}_j), \quad (6)$$

for appropriate scalars a . Let $\Omega_p(\Phi(\mathbf{x}))$ denote the embedding of the input data point $\mathbf{x} \in \mathbb{R}^d$ in this kernelized subspace. Then,

$$\Omega_p(\Phi(\mathbf{x})) = \sum_{i=1}^p \langle \Phi(\mathbf{x}), \mathbf{e}_i \rangle \mathbf{e}_i. \quad (7)$$

¹When \mathbf{x} is not an object in the Euclidean space, we assume $\|\cdot\|$ to define some suitable distance on the data.

Substituting (6) into (7), we obtain

$$\begin{aligned} \Omega_p(\Phi(\mathbf{x})) &= \sum_{i=1}^p \sum_{j=1}^n a_{ij} \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}_j) \rangle \left(\sum_{\ell=1}^n a_{i\ell} \Phi(\mathbf{x}_\ell) \right) \\ &= \sum_{i=1}^p \sum_{j=1}^n \sum_{\ell=1}^n k(\mathbf{x}, \mathbf{x}_j) a_{ji}^T a_{i\ell} \Phi(\mathbf{x}_\ell), \end{aligned} \quad (8)$$

which can be written using matrix notation as:

$$\Omega_p(\Phi(\mathbf{x})) = \Phi(X) A A^T \mathbf{k}(X, \mathbf{x}), \quad (9)$$

where $\mathbf{k}(X, \mathbf{x})$ denotes the $n \times 1$ vector whose j -th dimension is given by $\mathbf{k}(\mathbf{x}_j, \mathbf{x})$ and A is an $n \times p$ matrix. Note that there is a certain abuse of notation here in that $\Phi(X)$ is a matrix with n columns, each column of which is an element in \mathcal{H} , where as the other matrices are those of real numbers.

Using these notation, we propose a novel kernelized feature subspace learning (KRP-FS) formulation below, with ordering constraints:

$$\arg \min_{A \in \mathbb{R}^{n \times p}} F(A) := \frac{1}{2} \sum_{i=1}^n \|\Phi(\mathbf{x}_i) - \Omega_p(\Phi(\mathbf{x}_i))\|_{\mathcal{H}}^2 \quad (10)$$

$$\text{s.t. } \|\Omega_p(\Phi(\mathbf{x}_i))\|_{\mathcal{H}}^2 + \eta \leq \|\Omega_p(\Phi(\mathbf{x}_j))\|_{\mathcal{H}}^2, \quad \forall i < j, \quad (11)$$

where (10) learns a p -dimensional feature subspace of the RKHS in which most of the data variability is contained, while (11) enforces the temporal order of data in this subspace, as measured by the length (using a Hilbertian norm $\|\cdot\|_{\mathcal{H}}$) of the projection of the data points \mathbf{x} onto this subspace. Our main idea is to use the parameters of this kernelized feature subspace projection Ω_p as the pooled descriptor for our input sequence, for subsequent sequence classification. To ensure that such descriptors from different sequences are comparable², we need to ensure that Ω_p is normalized, i.e., has orthogonal columns in the feature space, viz., $\Omega_p^T \Omega_p = \mathbf{I}_p$. In the view of (6), this implies the basis $\{\mathbf{e}_i\}$ for V satisfies $\langle \mathbf{e}_i, \mathbf{e}_j \rangle = \delta_{ij}$ (the delta function) and boils down to:

$$\Omega_p^T \Omega_p = A^T \mathbf{K} A = \mathbf{I}_p, \quad (12)$$

where \mathbf{K} is the kernel constructed on X and is symmetric positive definite (SPD). Incorporating these conditions and including slack variables (to accommodate any outliers) using a regularization constant C , we rewrite (10), (11) as:

$$\arg \min_{\substack{A \in \mathbb{R}^{n \times p}, \\ A^T \mathbf{K} A = \mathbf{I}_p, \xi \geq 0}} F(A) := \frac{1}{2} \sum_{i=1}^n \|\Phi(\mathbf{x}_i) - \Omega_p(\Phi(\mathbf{x}_i))\|_{\mathcal{H}}^2 + C \sum_{i < j} \xi_{ij} \quad (13)$$

$$\text{s.t. } \|\Omega_p(\Phi(\mathbf{x}_i))\|_{\mathcal{H}}^2 + \eta - \xi_{ij} \leq \|\Omega_p(\Phi(\mathbf{x}_j))\|_{\mathcal{H}}^2, \quad \forall i < j. \quad (14)$$

²Recall that such normalization is common when using vectorial representations, in which case they are unit normalized.

It may be noted that our objective (13) essentially depicts kernel principal components analysis (KPCA) [41], albeit the constraints make estimation of the low-rank subspace different, demanding sophisticated optimization techniques for efficient solution. We address this concern below, by making some key observations regarding our objective.

4.3. Efficient Optimization

Substituting for the definitions of \mathbf{K} and Ω_p , the formulation in (13) can be written using hinge-loss as:

$$\begin{aligned} \arg \min_{\substack{A \in \mathbb{R}^{n \times p} | A^T \mathbf{K} A = \mathbf{I}_p \\ \xi \geq 0}} F(A) := & \frac{1}{2} \sum_{i=1}^n -2\mathbf{k}(X, \mathbf{x}_i)^T A A^T \mathbf{k}(X, \mathbf{x}_i) \\ & + \mathbf{k}(X, \mathbf{x}_i)^T A A^T \mathbf{K} A A^T \mathbf{k}(X, \mathbf{x}_i) + C \sum_{i < j} \xi_{ij} \\ & + \lambda \sum_{i < j} \max \left(0, \mathbf{k}(X, \mathbf{x}_i)^T A A^T \mathbf{K} A A^T \mathbf{k}(X, \mathbf{x}_j) + \right. \\ & \left. - \mathbf{k}(X, \mathbf{x}_j)^T A A^T \mathbf{K} A A^T \mathbf{k}(X, \mathbf{x}_i) + \eta - \xi_{ij} \right). \quad (15) \end{aligned}$$

As is clear, the variable A appears as $A A^T$ through out and thus our objective is invariant to any right rotations by an element of the p -dimensional orthogonal group $\mathcal{O}(p)$, i.e., for $R \in \mathcal{O}(p)$, $F(A) = F(AR)$. This, together with the condition in (12) suggests that the optimization on A as defined in (15) can be solved over the so called *generalized Grassmann manifold* [11][Section 4.5] using Riemannian optimization techniques.

We use a Riemannian conjugate gradient (RCG) algorithm [44] on this manifold for solving our objective. A key component for this algorithm to proceed is the expression for the Riemannian gradient of the objective $\text{grad}_A F(A)$, which for a generalized Grassmannian can be obtained from the Euclidean gradient $\nabla_A F(A)$ as:

$$\text{grad}_A F(A) = \mathbf{K}^{-1} \nabla_A F(A) - A \text{sym}(A^T \nabla_A F(A)), \quad (16)$$

where $\text{sym}(L) = \frac{L+L^T}{2}$ is the symmetrization operator [33][Section 4]. The Euclidean gradient for (15) is as follows: let $S_1 = \mathbf{K} \mathbf{K} A$, $S_2 = \mathbf{K} A A^T$, and $S_3 = A^T \mathbf{K} A$, then

$$\nabla_A F(A) = S_1 (S_3 - 2\mathbf{I}_p) + S_2 S_1 + \lambda (\mathbf{K}_{12} A S_3 + S_2 \mathbf{K}_{12} A), \quad (17)$$

where $\mathbf{K}_{12} = \mathbf{K}_1 \mathbf{K}_1^T - \mathbf{K}_2 \mathbf{K}_2^T$, $\mathbf{K}_1, \mathbf{K}_2$ are the kernels capturing the order violations in (15) for which the hinge-loss is non-zero; \mathbf{K}_1 collecting the sum of all violations for \mathbf{x}_i and \mathbf{K}_2 the same for \mathbf{x}_j . If further scalability is desired, one can also invoke stochastic Riemannian solvers such as Riemannian-SVRG [59, 24] instead of RCG. These methods extend the variance reduced stochastic gradient methods to Riemannian manifolds, and may help scale the optimization to larger problems.

4.4. Action Classification

Once we find A per video sequence by solving (15), we use $\Omega = \Phi(X)A$ (note that we omit the subscript p from Ω_p as it is by now understood) as the action descriptor. However, as Ω is semi-infinite, it cannot be directly computed, and thus we need to resort to the kernel trick again for measuring the similarity between two encoded sequences. Given that $\Omega \in \mathcal{G}(p)$ belong to a generalized Grassmann manifold, we can use any Grassmannian kernel for computing this similarity. Among several such kernels reviewed in [20], we found the exponential projection metric kernel to be empirically beneficial. For two sequences $X_1 \in \mathbb{R}^{d \times n_1}$, $X_2 \in \mathbb{R}^{d \times n_2}$, their subspace parameters A_1, A_2 and their respective KRP-FS descriptors $\Omega_1, \Omega_2 \in \mathcal{G}(p)$, the exponential projection metric kernel is defined as:

$$\mathbb{K}_{\mathcal{G}}(\Omega_1, \Omega_2) = \exp \left(\nu \|\Omega_1^T \Omega_2\|_F^2 \right), \text{ for } \nu > 0. \quad (18)$$

Substituting for Ω 's, we have the following kernel for action classification, whose ij -th entry is given by:

$$\mathbb{K}_{\mathcal{G}}^{ij}(\Omega_1, \Omega_2) = \exp \left(\nu \|A_i \mathbf{K} A_j\|_F^2 \right), \quad (19)$$

where $\mathbf{K} \in \mathbb{R}^{n_1 \times n_2}$ is an (RBF) kernel capturing the similarity between sequences.

4.5. Nyström Kernel Approximation

A challenge when using our scheme on large datasets is the need to compute the kernel matrix (such as in a non-linear SVM); this computation can be expensive. To this end, we resort to the well-known Nyström-based low-rank kernel approximations [10]. Technically, in this approximation, only a few columns of the kernel matrix are computed, and the full kernel is approximated by a low-rank outer product. In our context, let $\mathbb{K}_{\mathcal{G}} \in \mathbb{R}^{n \times m}$ (for $m \ll n$) represents a matrix with m randomly chosen columns of $\mathbb{K}_{\mathcal{G}}$, then the Nyström approximation $\widehat{\mathbb{K}}_{\mathcal{G}}$ of $\mathbb{K}_{\mathcal{G}}$ is given by:

$$\widehat{\mathbb{K}}_{\mathcal{G}} = \widetilde{\mathbb{K}}_{\mathcal{G}} M \widetilde{\mathbb{K}}_{\mathcal{G}}^T, \quad (20)$$

where M is the (pseudo) inverse of the first $m \times m$ sub-matrix of $\widetilde{\mathbb{K}}_{\mathcal{G}}$. Typically, only a small fraction (1/8-th of $\mathbb{K}_{\mathcal{G}}$ in our experiments, Table 4) of columns are needed to approximate the kernel without any significant loss in performance.

5. Computational Complexity

Evaluating the objective in (15) takes $\mathcal{O}(n^3 p + m^3 + mnd)$ operations and computing the Euclidean gradient in (17) needs $\mathcal{O}(n^3 + n^2 p)$ computations for each iteration of the solver. While, this may look more expensive than the basic ranking pooling formulation, note that here we use kernel matrices, which for action recognition datasets, are much smaller in comparison to very high dimensional (CNN) features used for frame encoding. See Table 3 for empirical run-time analysis.

6. Experiments

In this section, we provide experiments on several action recognition datasets where action features are represented in diverse ways. Towards this end, we use (i) the JHMDB and MPII Cooking activities datasets, where frames are encoded using a VGG two-stream CNN model, (ii) HMDB dataset, for which we use features from a ResNet-152 CNN model, (iii) UTKinect actions dataset, using non-linear features from 3D skeleton sequences corresponding to human actions, and (iv) hand-crafted bag-of-words features from the MPII dataset. For all these datasets, we compare our methods to prior pooling schemes such as average pooling (in which features per sequence are first averaged and then classified using a linear SVM), rank pooling (RP) [17], and generalized rank pooling (GRP) [6]. We use the publicly available implementations of these algorithms without any modifications. Our implementations are in Matlab and we use the ManOpt package [2] for Riemannian optimization. Below, we detail our datasets and their pre-processing, following which we furnish our results.

6.1. Datasets and Feature Representations

HMDB Dataset [28]: is a standard benchmark for action recognition, consisting of 6766 videos and 51 classes. The standard evaluation protocol is three-split cross-validation using classification accuracy. To generate features, we use a standard two-stream ResNet-152 network pre-trained on the UCF101 dataset (available as part of [12]). We use the 2048D pool5 layer output to represent per-frame features for both streams.

JHMDB Dataset [23]: consists of 960 video sequences and 21 actions. The standard evaluation protocol is average classification accuracy over three-fold cross-validation. For feature extraction, we use a two-stream model using a VGG-16 network. To this end, we fine-tune a network, pre-trained on the UCF101 dataset (provided as part of [14]). We extract 4096D fc6 layer features as our feature representation.

MPII Cooking Activities Dataset [39]: consists of cooking actions of 14 individuals. The dataset has 5609 video sequences and 64 different actions. For feature extraction, we use the fc6 layer outputs from a two-stream VGG-16 model. We also present experiments with dense trajectories (encoded as bag-of-words using 4K words). We report the mean average precision over 7 splits.

UTKinect Actions [57]: is a dataset for action recognition from 3D skeleton sequences; each sequence has 74 frames. There are 10 actions in the dataset performed by 2 subjects. We use this dataset to demonstrate the effectiveness of our schemes to explicit non-linear features. We encode each 3D pose using a Lie algebra based scheme that maps the skeletons into rotation and translation vectors which are objects in the SE(3) geometry as described in [50]. We report the average classification accuracy over 2 splits.

6.2. Analysis of Model Parameters

Ranking Threshold η : An important property of our schemes to be verified is whether it reliably detects temporal order in the extracted features and if so how does the ordering parameter η influence the generated descriptor for action classification. In Figures 2(a), 2(b), we plot the accuracy against increasing values of the ranking threshold η for features from the RGB and flow streams on JHMDB split-1. The threshold η was increased from 10^{-4} to 1 at multiples of 10. We see from the Figure 2(a) that increasing η does influence the nature of the respective algorithms, however each algorithm appears to have its own setting that gives the best performance; e.g., IBKRP achieves best at $\eta = 0.1$, while KRP-FS takes the best value at $\eta = 0.0001$. We also plot the same for GRP [6] and a linear kernel; the latter takes a dip in accuracy as η increases, because higher values of η are difficult to satisfy for our unit norm features when only linear hyperplanes are used for representation. In Figure 2(b), we see a similar trend for the RGB stream.

Number of Subspaces p : In Figure 2(c), we plot the classification accuracy against increasing number of subspaces on the HMDB dataset split-1. We show the results for both optical flow and image streams separately and also plot the same for GRP (which is a linear variant of KRP-FS). The plot shows that using more subspaces is useful, however beyond say 10-15, the performance saturates suggesting perhaps temporally ordered data inhabits a low-rank subspace, as was the original motivation to propose this idea.

Dimensionality Reduction: To explore the possibility of dimensionality reduction (using PCA) of the CNN features and understand how well the kernelization results be (does the lower-dimensional features still capture the temporal cues?), in Figure 2(d), we plot accuracy against increasing dimensionality in PCA on 2048D ResNet-152 features from HMDB-51 split1. As the plot shows, for almost all the variants of our methods, using a reduced dimension does result in useful representations – perhaps because we remove noisy dimensions in this process. We also witness that KRP-FS performs the best in generating representations from very low-dimensional features.

Nyström Approximation Quality: In Table 4, we plot the kernel approximation quality using Nyström on HMDB split-1 features for the KRP-FS variant. We uniformly sampled data in the order of $1/2^k$ -th original data size, where k is varied from 5 to 2. We see from the table that the accuracy decreases only marginally ($< 1\%$) with the approximation. In the sequel, we use a factor of 1/8-th data size.

Runtime Analysis: For this experiment, we use a single core Intel 2.6 GHz CPU. To be compatible with others, we re-implemented [17] as a linear kernel. Table 3 presents the results. As expected, our methods are slightly costlier

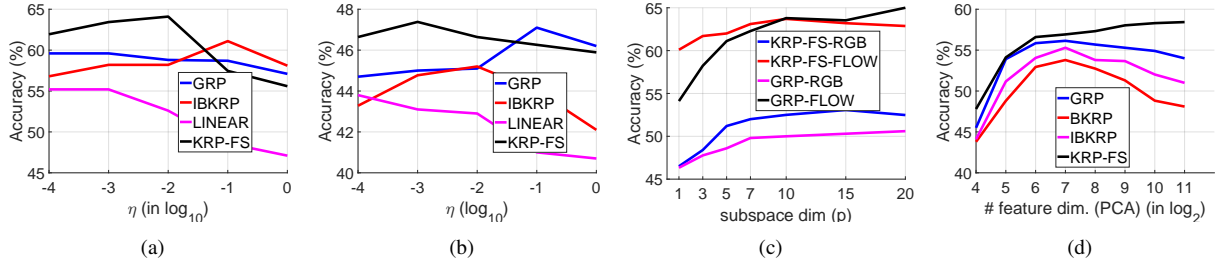


Figure 2. Analysis of parameters in our schemes: (a,b) accuracy against increasing ordering threshold η for RGB and FLOW streams respectively on JHMDB dataset split-1, (c) classification accuracy against increasing subspace dimensionality on HMDB-51 split1, and (d) Effect of applying PCA to input features before using our schemes (on HMDB split1).

than others, mainly due to the need to compute the kernel. However, they are still fast, and even with our unoptimized Matlab implementation, could run at real-time rates (27fps). Further, increasing the number of subspaces in KRP-FS does not make a significant impact on the running time; e.g., (increasing from 3 to 20 increased by 1.2ms).

Feature Pre-processing: As described in [17] and [15], taking a signed-square root (SSR) and temporal moving-average (MA) of the features improve accuracy. In Table 2, we revisit these steps in the kernelized setting (KRP-FS) using 3 subspaces. It is clear these steps bring only very marginal improvements. This is unsurprising; as is known RBF kernel already acts as a low-pass filter.

Homogeneous Kernels: A variant of rank pooling [17] uses homogeneous kernels [49] to map data into a linearizable Hilbert space onto which linear rank pooling is applied. In Table 1, we use a Chi-squared kernel for rank pooling and compare the performance to BKRP (using RBF kernel). While we observe a 5% improvement over [17] when using homogeneous kernels, still BKRP (which is more general) significantly outperforms it.

HMDB Dataset	FLOW	RGB	FLOW+RGB
RP [17]	56.7	38.3	63.1
Hom. Chi-Sq. RP [17]	61.5	43.8	66.5
BKRP (ours)	54.9	45.9	69.5

Table 1. Comparisons to rank pooling using a homogeneous kernel linearization of CNN features via a Chi-Squared kernel as in [17].

Method	FLOW	RGB
with MA + SSR	61.5	51.6
w/o MA + SSR	61.4	51.4
w/o MA + w/o SSR	60.8	51.3

Table 2. Effect of Moving Average (MA) and signed-square root (SSR) of CNN features before KRP-FS on HMDB split-1.

6.3. Comparisons between Pooling Schemes

In Tables 5, 6, and 7, we compare our schemes between themselves and similar methods. As the tables show, both

IBKRP and KRP-FS demonstrate good performances against their linear counterparts. We also find that linear rank pooling (RP), as well as BKRP are often out-performed by average pooling (AP) – which is unsurprising given that the CNN features are non-linear (for RP) and the pre-image computed (as in BKRP) might not be a useful representation without the reconstructive term as in IBKRP or KRP-FS. We also find that KRP-FS is about 3-5% better than its linear variant GRP on most of the datasets.

RP	GRP	BKRP	IBKRP	KRP-FS
1.1	3.8	6.7	8.8	9.5

Table 3. Avg. run time (time taken / frame) – in milli-seconds – on the HMDB dataset. CNN forward pass time is not included.

Kernel Sampling Factor	Accuracy
1/32	60.56
1/8	61.43
1/2	61.7

Table 4. Influence of Nyström approximation to the KRP-FS kernel, using 3 subspaces on HMDB split-1.

6.4. Comparisons to the State of the Art

In Tables 9, 10, and 11, we showcase comparisons to state-of-the-art approaches. Notably, on the challenging HMDB dataset, we find that our method KRP-FS achieves 69.8% on 3-split accuracy, which is better to GRP by about 4%. Further, by combining with Fisher vectors– IDT-FV – (using dense trajectory features), which is a common practice, we outperform other recent state-of-the-art methods. We note that recently Carreira and Zissermann [4] reports about 80.9% accuracy on HMDB-51 by training deep models on the larger Kinetics dataset [25]. However, as seen from Table 9, our method performs better than theirs (by about 3%) when not using extra data. We outperform other methods on MPII and JHMDB datasets as well – specifically, KRP-FS when combined with IDT-FV, outperforms GRP+IDT-FV by about 1–2% showing that learning representations in the kernel space is indeed useful.

6.5. Comparisons to Hand-crafted Features

In Table 8, we evaluate our method on the bag-of-features dense trajectories on MPII and non-linear features for encoding human 3D skeletons on the UTKinect actions. As is clear from the tables, all our pooling schemes significantly improve the performance of linear rank pooling and GRP schemes. As expected, IBKRP is better than BKRP by nearly 8% on UT Kinect actions. We also find that KRP-FS performs the best most often, with about 7% better accuracy on the MPII cooking activities dataset against GRP and UT Kinect actions. These experiments demonstrate the representation effectiveness of our method with regard to the diversity of the data features.

JHMDB Dataset	FLOW	RGB	FLOW+RGB
Avg. [43]	63.8	47.8	71.2
RP [16]	41.1	47.3	56.0
GRP [6]	64.2	42.5	70.8
BKRP (ours)	65.8	49.3	73.4
IBKRP (ours)	68.2	49.0	76.2
KRP-FS (ours)	67.5	46.2	74.6

Table 5. Classification accuracy on the JHMDB dataset split-1.

HMDB Dataset	FLOW	RGB	FLOW+RGB
Avg. Pool [43]	57.2	45.2	65.6
RP [16]	56.7	38.3	63.1
GRP [6]	65.3	47.8	68.3
BKRP (ours)	54.9	45.9	69.5
IBKRP (ours)	58.2	46.8	69.6
KRP-FS (ours)	66.1	54.1	71.9

Table 6. Classification accuracy on the HMDB dataset split-1.

MPII Dataset	FLOW	RGB	FLOW+RGB
Avg. [43]	48.1	41.7	51.1
RP [16]	49.0	40.0	50.6
GRP [6]	52.1	50.3	53.8
BKRP (ours)	40.5	35.5	42.9
IBKRP (ours)	52.1	43.2	55.9
KRP-FS (ours)	48.2	44.7	57.2

Table 7. Classification accuracy (mAP%) on the MPII dataset split1.

7. Conclusions

In this paper, we looked at the problem of compactly representing temporal data for the problem of action recognition in video sequences. To this end, we proposed kernelized subspace representations obtained via solving a kernelized PCA objective. The effectiveness of our schemes were substantiated exhaustively via experiments on several benchmark datasets and diverse data types. Given the generality of our approach, we believe it will be useful in several domains that use sequential data.

Algorithm	Acc.(%)	Algorithm	Acc.(%)
Avg. Pool	42.1	SE(3) [50]	97.1
RP [17]	45.3	Tensors [27]	98.2
GRP [6]	46.1	RP [17]	75.5
BKRP	46.5	BKRP	84.8
IBKRP	49.5	IBKRP	92.1
KRP FS	53.0	KRP FS	99.0

Table 8. Performances of our schemes on: dense trajectories from the MPII dataset (left) and UT-Kinect actions (right). For KRP-FS on UTKinect actions, we use 15 subspaces.

Algorithm	Avg. Acc. (%)
ST Multiplier Network[13]	68.9%
ST Multiplier Network + IDT[13]	72.2%
Two-stream I3D[4]	66.4%
Temporal Segment Networks [55]	69.4
Hier. Rank Pooling + IDT-FV [15]	66.9
GRP	65.4
GRP + IDT-FV	67.0
BRKP	64.1
IBKRP	66.3
IBKRP + IDT-FV	67.6
KRP-FS	69.8
KRP-FS + IDT-FV	72.7

Table 9. HMDB Dataset (3 splits)

Algorithm	mAP(%)
Interaction Part Mining [60]	72.4
Video Darwin [17]	72.0
Hier. Mid-Level Actions [45]	66.8
PCNN + IDT-FV [8]	71.4
GRP [6]	68.4
GRP + IDT-FV [6]	75.5
BRKP	66.3
IBKRP	68.7
IBKRP + IDT-FV	71.8
KRP-FS	70.0
KRP-FS + IDT-FV	76.1

Table 10. MPII Cooking Activities (7 splits)

Algorithm	Avg. Acc. (%)
Stacked Fisher Vectors [36]	69.03
Higher-order Pooling [7]	73.3
P-CNN + IDT-FV [8]	72.2
GRP [6]	70.6
GRP + IDT-FV [6]	73.7
BRKP	71.5
IBKRP	73.3
IBKRP + IDT-FV	73.5
KRP-FS	73.8
KRP-FS + IDT-FV	74.2

Table 11. JHMDB Dataset (3 splits)

References

- [1] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *CVPR*, 2016. [2](#)
- [2] N. Boumal, B. Mishra, P.-A. Absil, R. Sepulchre, et al. Manopt, a matlab toolbox for optimization on manifolds. *JMLR*, 15(1):1455–1459, 2014. [6](#)
- [3] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *ICML*, 2007. [2](#), [3](#)
- [4] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, July 2017. [1](#), [2](#), [7](#), [8](#)
- [5] J. Cavazza, A. Zunino, M. San Biagio, and V. Murino. Kernelized covariance for action recognition. In *ICPR*, 2016. [2](#)
- [6] A. Cherian, B. Fernando, M. Harandi, and S. Gould. Generalized rank pooling for action recognition. In *CVPR*, 2017. [2](#), [3](#), [4](#), [6](#), [8](#)
- [7] A. Cherian, P. Koniusz, and S. Gould. Higher-order pooling of CNN features via kernel linearization for action recognition. In *WACV*, 2017. [2](#), [8](#)
- [8] G. Chéron, I. Laptev, and C. Schmid. P-cnn: Pose-based cnn features for action recognition. *arXiv preprint arXiv:1506.03607*, 2015. [8](#)
- [9] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *arXiv preprint arXiv:1411.4389*, 2014. [2](#)
- [10] P. Drineas and M. W. Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *JMLR*, 6(Dec):2153–2175, 2005. [5](#)
- [11] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998. [5](#)
- [12] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, 2016. [2](#), [6](#)
- [13] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal multiplier networks for video action recognition. In *CVPR*, 2017. [8](#)
- [14] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. *arXiv preprint arXiv:1604.06573*, 2016. [1](#), [2](#), [6](#)
- [15] B. Fernando, P. Anderson, M. Hutter, and S. Gould. Discriminative hierarchical rank pooling for activity recognition. In *CVPR*, 2016. [2](#), [7](#), [8](#)
- [16] B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, and T. Tuytelaars. Rank pooling for action recognition. *PAMI*, (99), 2016. [8](#)
- [17] B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *CVPR*, 2015. [1](#), [2](#), [3](#), [6](#), [7](#), [8](#)
- [18] B. Fernando and S. Gould. Learning end-to-end video classification with rank-pooling. In *ICML*, 2016. [1](#), [2](#)
- [19] M. Harandi, M. Salzmann, and R. Hartley. Joint dimensional-reduction and metric learning: A geometric take. In *ICML*, 2017. [2](#)
- [20] M. T. Harandi, M. Salzmann, S. Jayasumana, R. Hartley, and H. Li. Expanding the family of Grassmannian kernels: An embedding perspective. In *ECCV*, 2014. [5](#)
- [21] M. T. Harandi, C. Sanderson, S. Shirazi, and B. C. Lovell. Kernel analysis on Grassmann manifolds for action recognition. *Pattern Recognition Letters*, 34(15):1906–1915, 2013. [3](#)
- [22] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017. [1](#)
- [23] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *ICCV*, 2013. [6](#)
- [24] H. Kasai, H. Sato, and B. Mishra. Riemannian stochastic variance reduced gradient on Grassmann manifold. *arXiv preprint arXiv:1605.07367*, 2016. [5](#)
- [25] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. [7](#)
- [26] P. Koniusz and A. Cherian. Sparse coding for third-order super-symmetric tensors with application to texture recognition. In *CVPR*, 2016. [2](#)
- [27] P. Koniusz, A. Cherian, and F. Porikli. Tensor representations via kernel linearization for action recognition from 3D skeletons. In *ECCV*, 2016. [2](#), [8](#)
- [28] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *2011 International Conference on Computer Vision*, pages 2556–2563. IEEE, 2011. [6](#)
- [29] S. Kumar Roy, Z. Mhammedi, and M. Harandi. Geometry aware constrained optimization techniques for deep learning. In *CVPR*, 2018. [2](#)
- [30] J. T. Kwok, B. Mak, and S. Ho. Eigenvoice speaker adaptation via composite kernel principal component analysis. In *NIPS*, 2004. [3](#)
- [31] J.-Y. Kwok and I.-H. Tsang. The pre-image problem in kernel methods. *IEEE Transactions on Neural Networks*, 15(6):1517–1525, 2004. [3](#)
- [32] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de-noising in feature spaces. In *NIPS*, 1998. [2](#), [3](#), [4](#)
- [33] B. Mishra and R. Sepulchre. Riemannian preconditioning. *SIAM Journal on Optimization*, 26(1):635–660, 2016. [5](#)
- [34] S. O’Hara and B. A. Draper. Scalable action recognition with a subspace forest. In *CVPR*, 2012. [3](#)
- [35] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *ICML*, 2013. [2](#)
- [36] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *ECCV*. 2014. [8](#)
- [37] H. Quang Minh, M. San Biagio, L. Bazzani, and V. Murino. Approximate log-Hilbert-Schmidt distances between covariance operators for image classification. In *CVPR*, 2016. [2](#)
- [38] R. Ranjan, V. M. Patel, and R. Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark

- localization, pose estimation, and gender recognition. *TPAMI*, 2017. 1
- [39] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele. A database for fine grained activity detection of cooking activities. In *CVPR*, 2012. 6
- [40] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *Intl. Conf. on Computational Learning Theory*, 2001. 4
- [41] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *ICANN*. Springer, 1997. 5
- [42] B. Scholkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001. 2, 3
- [43] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 1, 2, 8
- [44] S. T. Smith. Optimization techniques on riemannian manifolds. *Fields institute communications*, 3(3):113–135, 1994. 5
- [45] B. Su, J. Zhou, X. Ding, H. Wang, and Y. Wu. Hierarchical dynamic parsing and encoding for action recognition. In *ECCV*, 2016. 8
- [46] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *ICCV*, 2015. 1, 2
- [47] C.-C. Tseng, J.-C. Chen, C.-H. Fang, and J.-J. J. Lien. Human action recognition based on graph-embedded spatio-temporal subspace. *Pattern Recognition*, 45(10):3611 – 3624, 2012. 3
- [48] P. Turaga, A. Veeraraghavan, A. Srivastava, and R. Chellappa. Statistical computations on Grassmann and Stiefel manifolds for image and video-based recognition. *PAMI*, 33(11):2273–2286, 2011. 3
- [49] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *PAMI*, 34(3):480–492, 2012. 7
- [50] R. Vemulapalli, F. Arrate, and R. Chellappa. Human action recognition by representing 3D skeletons as points in a Lie group. In *CVPR*, 2014. 2, 6, 8
- [51] G. Wahba. *Spline models for observational data*. SIAM, 1990. 4
- [52] J. Wang, A. Cherian, and A. Porikli. Ordered pooling of optical flow sequences for action recognition. In *WACV*, 2016. 2
- [53] J. Wang, A. Cherian, F. Porikli, and S. Gould. Video representation learning using discriminative pooling. In *CVPR*, 2018. 2
- [54] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015. 1
- [55] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 2, 8
- [56] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *CVPR*, 2016. 1
- [57] L. Xia, C.-C. Chen, and J. Aggarwal. View invariant human action recognition using histograms of 3d joints. In *CVPRW*, 2012. 6
- [58] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 2
- [59] H. Zhang, S. J. Reddi, and S. Sra. Riemannian SVRG: Fast stochastic optimization on riemannian manifolds. In *NIPS*, 2016. 5
- [60] Y. Zhou, B. Ni, R. Hong, M. Wang, and Q. Tian. Interaction part mining: A mid-level approach for fine-grained action recognition. In *CVPR*, 2015. 8