

# Generalized Rank Pooling for Activity Recognition

Anoop Cherian<sup>1,3</sup> Basura Fernando<sup>1,3</sup> Mehrtash Harandi<sup>2,3</sup> Stephen Gould<sup>1,3</sup>  
<sup>1</sup>Australian Centre for Robotic Vision, <sup>2</sup>Data61/CSIRO  
<sup>3</sup>The Australian National University, Canberra, Australia  
 firstname.lastname@{anu.edu.au, data61.csiro.au}

## Abstract

Most popular deep models for action recognition split video sequences into short sub-sequences consisting of a few frames; frame-based features are then pooled for recognizing the activity. Usually, this pooling step discards the temporal order of the frames, which could otherwise be used for better recognition. Towards this end, we propose a novel pooling method, generalized rank pooling (GRP), that takes as input, features from the intermediate layers of a CNN that is trained on tiny sub-sequences, and produces as output the parameters of a subspace which (i) provides a low-rank approximation to the features and (ii) preserves their temporal order. We propose to use these parameters as a compact representation for the video sequence, which is then used in a classification setup. We formulate an objective for computing this subspace as a Riemannian optimization problem on the Grassmann manifold, and propose an efficient conjugate gradient scheme for solving it. Experiments on several activity recognition datasets show that our scheme leads to state-of-the-art performance.

## 1. Introduction

Activity recognition from videos is challenging as real-world actions are often complex, confounded with background activities and vary significantly from one actor to another. Efficient solutions to this difficult problem can facilitate several useful applications such as human-robot cooperation, visual surveillance, augmented reality, and medical monitoring systems. The recent resurgence of deep learning algorithms has demonstrated significant advancements in several fundamental problems in computer vision, including activity recognition. However, such solutions are still far from being practically useful and thus activity recognition continues to be a challenging research topic [2, 12, 34, 38, 45, 47].

Deep learning algorithms on long video sequences demand huge computational resources, such as GPU, memory, etc. One popular approach to circumvent this practical chal-

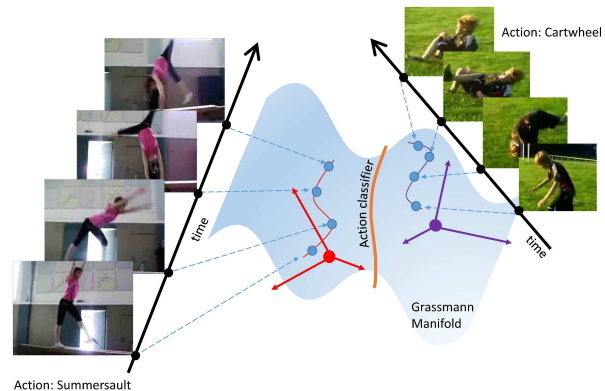


Figure 1. An illustration of our pooling scheme. For every video, our formulation learns the parameters of a low-dimensional subspace in which the projected video frames conform to their temporal order. We use the subspaces as respective representations of the sequences. Such subspaces belong to the Grassmann manifold, on which we learn non-linear action classifiers.

lenge is to train networks on sub-sequences consisting of one to a few tens of video frames. The activity predictions from such short temporal receptive fields are then aggregated via a pooling step [25, 7, 38], such as computing the average or maximum of the generated CNN features. Given that the features are from temporally-ordered input data, it is likely that they capture the temporal evolution of the actions in the sequence. Thus, a pooling scheme that can use this temporal structure is preferred for activity recognition.

In Fernando et al. [15, 13], the problem of pooling using the temporal structure is cast in a learning-to-rank setup, termed *Rank Pooling*, that computes a line in input space; the projection of input data onto this line preserving the temporal order. The parameters of this line are then used as a summarization of the video sequence. However, several issues remain unanswered, namely (i) while the line is assumed to belong to the input space, there is no guarantee that it captures other properties of the data (other than order), such as background, context, etc. which may be useful for recognition, (ii) the ranking constraints are linear, (iii)

each data channel (such as RGB) is assumed independent, and (iv) a single line for ordering is considered, while using multiple hyperplanes might lead to better characterization of the temporal action dynamics. In this paper, we propose a novel re-formulation of rank pooling that addresses all these drawbacks.

Instead of using a single line as a representation for the sequence, our main idea is to use a subspace parameterized by several orthonormal hyperplanes. We propose a novel learning-to-rank formulation to compute this subspace by minimizing an objective that jointly provides a low-rank approximation to the input data, while also preserves their temporal order in the subspace. The low-rank approximation helps capture the essential properties of the data that are useful for summarizing the action. Further, the temporal order is captured via a quadratic ranking function thereby capturing non-linear dependencies between the input data channels. Specifically, in our formulation, the temporal order is encoded as increasing lengths of the projections of the input data onto the subspace.

While our formulation provides several advantages for temporal pooling, it leads to a difficult non-convex optimization problem, due to the orthogonality constraints. Fortunately, we show that the subspaces in our formulation satisfy certain mathematical properties, and thus can be cast as a problem on the so called *Grassmann manifold*, for which there exists efficient Riemannian optimization algorithms. We propose to use a conjugate gradient descent algorithm for our problem which is often seen to converge fast.

We provide experiments on several popular action recognition datasets, preprocessed by extracting features from the fully-connected layers of VGG-net [39]. Following the standard practice, we use a two stream network [38] trained on single RGB frames and 20-channel optical flow images. Our experimental results show that the proposed scheme is significantly better at capturing the temporal structure of CNN features in action sequences compared to conventional pooling schemes or the basic form of rank-pooling [14], while also achieving state-of-the-art performances.

Before moving on, we summarize the main contributions of our work:

- We propose a novel learning-to-rank formulation for capturing the temporal evolution of actions in video sequences by learning subspaces.
- We propose an efficient Riemannian optimization algorithm for solving our objective.
- We show that subspace representation on CNN features is highly beneficial for action recognition.
- We provide experiments on standard benchmarks demonstrating state-of-the-art performance.

## 2. Related Work

Training of convolutional neural networks directly on long video sequences is often computationally prohibitive. Thus, various simplifications have been explored to make the problem amenable, such as using 3D spatio-temporal convolutions [42], recurrent models such as LSTMs or RNNs [7, 8], decoupling spatial and temporal action components via a two-stream model [38, 12], early or late fusion of predictions from a set of frames [25]. While 3D convolutions and recurrent models can potentially learn the dynamics of actions in long sequences, training them is difficult due to the need for very large datasets and volumetric nature of the search space in a structurally-complex domain. Thus, in this paper, we focus on late fusion techniques on the CNN features generated by a two-stream model, and refer to recent surveys for a review of alternative schemes [23].

Typically, the independent action predictions by a CNN along the video sequence is averaged or fused via a linear SVM [38] without considering the temporal evolution of the CNN features. Rank-pooling [15] demonstrates better performances by accounting for the temporal information. They cast the problem in a learning-to-rank framework and propose an efficient algorithm for solving it via support-vector regression. While, this scheme uses hand-crafted features, extensions are explored in Fernando et al. [13, 16, 41] in a CNN setting via end-to-end learning. However, training such a deep architecture is slow as it requires computing the gradients of a bi-level optimization loss [18]. This difficulty can be circumvented via early-fusion of the frames as described in Bilen et al. [3], Wang et al. [46] by pooling the input frames or optical flow images, however one needs to solve a very high-dimensional ranking problem (with dimensionality equal to the size of input images), which may be slow. Instead, in this paper, we propose a generalization of the original ranking formulation [15] using subspace representations and show that our formulation leads to significantly better representation of the dynamic evolution of actions, while being computationally cheap.

There have been approaches using subspaces for action representations in the past. These methods are developed mostly for hand-crafted features and thus their performances on CNN features are not thoroughly understood. For example, in the method of Li et al [29], it is assumed that trajectories from actions evolve in the same subspace, and thus computing the subspace angles may capture the similarity between activities. In contrast, we learn subspaces over more general CNN features and constrain them to capture dynamics. In Le et al. [28], the standard independent subspace analysis algorithm is extended to learn invariant spatio-temporal features from unlabeled video data. Principal components analysis and its variants have been suggested for action recognition in Karthikeyan et al. [26]

using multi-set partial least squares to capture the temporal dynamics. This method also uses probabilistic subspace similarity learning proposed by Moghaddam et al. [31] to learn intra-action and inter-action models. An adaptive locality preserving projection method is proposed in Tseng et al. [43] to obtain a low-dimensional spatial subspace in which the linear structure of the data (such as that arising out of human body shape) is preserved.

Similar to our proposed approach, O’Hara et al. [32] introduce a subspace forest representation for action recognition that considers each video segment as points on a Grassmann manifold and a random-forest based approximate nearest neighbor scheme is used to find similar videos. Subspaces-of-Features, formed from local spatio-temporal features, is presented in Raytchev et al. [35], and uses Grassmann SVM kernels [37] for classification. A framework using multiple orthogonal classifiers for domain adaptation is presented in Etai and Wolf [30]. Similar kernel based recognition schemes are also proposed in Harandi et al. [21] and Turaga et al. [44]. In contrast, we are the first to propose subspace representations on CNN features for action recognition in a joint framework that includes non-linear chronological ordering constraints to capture the temporal evolution of actions.

### 3. Proposed Method

Let  $X = \langle x_1, x_2, \dots, x_n \rangle$  be a sequence of  $n$  consecutive data features, each  $x_t \in \mathbb{R}^d$ , produced by some dynamic process at discrete time instances  $t$ . In case of action recognition in video sequences using a two-stream CNN model,  $X$  represents a sequence of features where each  $x_t$  is the output of some CNN layer (for example, fully-connected FC6 of a VGG-net as used in our experiments) from a single RGB video frame or a small stack of consecutive optical flow images (similar to [38]).

Our goal in this paper is to generate a compact representation for  $X$  that summarizes the human action category and could be used for recognition of human actions from video. Towards this end, we assume that the per-frame features  $x_t$  encapsulates the action properties of a frame (such as local dynamics or object appearance), and such features across the sequence captures the dynamics of the action as characterized by the temporal variations in  $x_t$ . That is, we assume the features are generated by a function  $g$  parameterized by time  $t$ :

$$x_t = g(t), \quad (1)$$

where  $g$  abstracts the action dynamics and produces the action feature for every time instance. However, in the case of real-world actions in arbitrary video sequences, finding such a generator  $g$  is not viable. Instead, using the unidirectional nature of time, we impose an order to the generated features as suggested in [15, 14], where it is assumed that

the projections of  $x_t$  onto some line preserves the order.

Given that the features  $x_t$  are often high-dimensional (as the ones we use, which are from the intermediate layers of a CNN), it is highly likely that the information regarding actions inhabits a low-dimensional feature subspace (instead of a line). Thus, we could write such a temporal order as:

$$\|U^T x_t\|^2 \leq \|U^T x_{t+1}\|^2 - \eta, \quad (2)$$

where  $U \in \mathcal{S}(p, d)$  denotes the parameters of a  $p$ -dimensional subspace, usually called a *frame* ( $p \ll d$ ) and  $\eta$  is a positive constant controlling the degree to which the temporal order is enforced. Such frames have orthonormal columns and belongs to the Stiefel manifold  $\mathcal{S}(p, d)$  [9]. Our main idea is to use  $U$  to represent the sequence  $X$ . To this end, we propose the following formulation for obtaining the low-rank subspace  $U$  from  $X$  given a rank  $p$  as follows:

$$\begin{aligned} \min_{U \in \mathcal{S}(p, d)} L(U) &\triangleq \frac{1}{2} \sum_{i=1}^n \|x_i - UU^T x_i\|^2 \\ \text{subject to} \quad &\|U^T x_i\|^2 \leq \|U^T x_j\|^2 - \eta, \quad \forall i < j. \end{aligned} \quad (3)$$

In the above formulation, the objective seeks a  $p$ -rank approximation to  $X$ . Note that the Stiefel manifold enforces the property that  $U$  has orthogonal columns, i.e.,  $U^T U = \mathbf{I}_p$ , the  $p \times p$  identity matrix.

#### 3.1. Properties of Formulation

In this subsection, we explore some properties of this formulation that allows for its efficient optimization.

##### Invariance to Right-Action by Orthogonal Group:

Note that our formulation in (3) can be written as  $L(U) = H(UU^T)$  for some function  $H$ . This means that for any matrix  $R$  in the orthogonal group  $\mathcal{O}(p)$ ,  $L(UR) = H(URR^T U) = H(UU^T)$ . This implies that all points of the form  $UR$  are minimizers of  $L(U)$ . Such a set forms an equivalence class of all linear subspaces that can be generated from a given  $U$  and is a point in the so called *Grassmann* manifold  $\mathcal{G}(p, d)$ . Thus, instead of minimizing over the Stiefel manifold, we could optimize the problem on the more general Grassmann manifold.

**Idempotence:** While, the objective in (3) appears to be a quadratic function in  $U$ , it can be shown to be reduced to a convex quadratic objective as follows. Observe that the matrix  $P = (I - UU^T)$  is symmetric idempotent, i.e.,  $P^T P = P P = P^2 = P$ . This implies that we can simplify the objective as follows:

$$\|x_i - UU^T x_i\|^2 = \text{Tr}(x_i x_i^T (\mathbf{I}_d - UU^T)). \quad (4)$$

Unfortunately, the overall formulation remains non-convex due to the orthogonality constraints on the subspace. However, this property comes handy in our convex reformulation of the objective in Section 3.3.

Using the above simplifications, introducing slack variables, and rewriting the constraints as hinge-loss, we can reformulate (3) and present our *generalized rank pooling* (GRP) objective as follows:

$$\begin{aligned} \min_{U: U \in \mathcal{G}(p,d)} F(U) \triangleq & \frac{1}{2} \sum_{i=1}^n \text{Tr}(x_i x_i^T (\mathbf{I}_d - U U^T)) + C \sum_{i < j} \xi_{ij} \\ & + \frac{\lambda}{2} \sum_{i < j} \max(0, \|U^T x_i\|^2 - \|U^T x_j\|^2 + 1 - \xi_{ij}), \end{aligned} \quad (5)$$

where  $\lambda \rightarrow \infty$  is a regularization parameter and  $\xi$  are non-negative slack variables.

### 3.2. Efficient Optimization

The optimization problem  $F(U)$  can be solved via Riemannian conjugate gradient on the Grassmann manifold. The gradient of the objective at the  $k$ -th conjugate gradient step has the following form:

$$\nabla_U F(U_k) = \left[ \begin{array}{c} \sum_{\substack{(\forall i < j) \wedge \\ \|U_k^T x_j\|^2 - \|U_k^T x_i\|^2 \leq 1}} \lambda (x_i x_i^T - x_j x_j^T) - X X^T \\ U_k \end{array} \right] \quad (6)$$

where the summation is over all constraint violations at a given iteration. Note that the complexity of this gradient computation is  $O(d^2)$  where  $d$  is the dimensionality of  $x_i$ , which may be expensive. Instead, below we propose a cheaper expression that leads to the same gradient.

Suppose  $V \in \{0, 1\}^{n \times n}$  be a binary upper-triangular matrix whose  $ij$ -th ( $j > i$ ) entry describes if the points  $x_i$  and  $x_j$  violate the ordering constraints given  $U_k$ . Then, we can rewrite the above gradient as:

$$\nabla_U F(U_k) = \left[ \sum_i \eta_i x_i (x_i^T U_k) \right] - X (X^T U_k), \quad (7)$$

$$\text{where } \eta_i = \left( \sum V_{(i,:)} - \sum V_{(:,i)} \right), \quad (8)$$

where  $V_{(i,:)}$  and  $V_{(:,j)}$  stand for the  $i$ -th row and  $j$ -th column of  $V$ , respectively. The complexity of computing  $\eta_i$  is  $O(n)$ , and the cost of computing the gradient is reduced to  $O(n + np)$ .

### 3.3. Convex Reformulation

The formulation introduced in (3) estimates all the subspaces together, however is non-convex. Instead, we show below that if we estimate the subspaces incrementally, that

is one at a time, then each sub-problem can be made convex and thus solved exactly. To this end, suppose we have obtained the subspace  $U_{q-1} \in \mathcal{S}(p, q-1)$  and we are solving for the  $q$ -th basis vector  $u_q$ . The objective for finding  $u_q$  can be recursively written as:

$$\begin{aligned} \min_{\|u_q\|=1, \xi \geq 0} & \frac{1}{2} \sum_{i=1}^n \|\hat{x}_i - u_q u_q^T \hat{x}_i\|^2 + C \sum_{i < j} \xi_{ij} \\ \text{subject to} & \|u_q^T \hat{x}_i\|^2 \leq \|u_q^T \hat{x}_j\|^2 - 1 + \xi_{ij}, \quad \forall i < j, \\ & \hat{x}_i = x_i - U_{q-1} \langle U_{q-1}, x_i \rangle, \quad \forall i \\ & u_q U_{q-1} = \mathbf{0}. \end{aligned} \quad (9)$$

In the above formulation, the main idea is to estimate each 1-dimensional subspace, and then subtract off the energy in  $X$  associated with this subspace, thus generating  $\hat{x}_i$  from  $x_i$ . Such unit subspaces are incrementally estimated (greedily) by following this procedure. As is clear, each solution of this objective is convex as it involves a quadratic objective, quadratic constraints, and a linear equality. Note that  $U_{q-1}$  is a constant matrix when estimating  $u_q$ . However, the greedy strategy may lead to sub-optimal results, as is empirically also observed in Table 4.

### 3.4. Conjugate Gradient on the Grassmannian

As described in the last section, we cast the generalized rank pooling objective as an optimization problem with an orthogonality constraint, which can generally be written as

$$\begin{aligned} \text{minimize}_U & F(U) \\ \text{subject to} & U^T U = \mathbf{I}_p, \end{aligned} \quad (10)$$

where  $F(U)$  is the desired cost function and  $U \in \mathbb{R}^{d \times p}$ . In the Euclidean space, problems of the form of (10) are typically cast as eigenvalue problems. However, the complexity of our cost function prohibits us from doing so. Instead, we propose to make use of manifold-based optimization techniques.

Recent advances in optimization methods formulate problems with unitary constraints as optimization problems on Stiefel or Grassmann manifolds [10, 1]. More specifically, the geometrically correct setting for the minimization problem in (10) is, in general, on a Stiefel manifold. However, if the cost function  $F(U)$  is independent from the choice of basis spanned by  $U$ , then the problem is on a Grassmann manifold. This is indeed what we showed in Section 3.1. We can therefore make use of Grassmannian optimization techniques, and, in particular, of Newton-type optimization, which we briefly review below.

Newton-type optimization, such as conjugate gradient (CG), over a Grassmannian is an iterative optimization routine that relies on the notion of Riemannian gradient. On

$\mathcal{G}(p, d)$ , the gradient is expressed as

$$\text{grad}_U F(U) = (\mathbf{I}_d - UU^T)\nabla_U(F), \quad (11)$$

where  $\nabla_U(F)$  is the  $d \times p$  matrix of partial derivatives of  $F(U)$  with respect to the elements of  $U$ . This is computed in Eq.(9) for our method. The descent direction expressed by  $\text{grad}_U F(U)$  identifies a curve  $\gamma(t)$  on the manifold, moving along it ensures a decrease in the cost function (at least locally). Points on  $\gamma(t)$  are obtained by the exponential map. In practice, the exponential map is approximated locally by a retraction (see Chapter 4 in [1] for definitions and detailed explanations). In the case of the Grassmannian, this can be understood as forcing the orthogonality constraint while making sure that the cost function decreases.

In our experiments, we make use of a conjugate gradient (CG) method on the Grassmannian. CG methods compute the new descent direction by combining the gradient at the current and the previous solutions. To this end, it requires transporting the previous gradient to the current point on the manifold which is achieved by the concept of Riemannian connections. On the Grassmann manifold, operations required for a CG method, have efficient numerical forms, which makes them well-suited to perform optimization on the manifold.

### 3.5. Classification on the Grassmanian

Once we obtain the subspace representation solving the GRP objective using manifold CG method, the next step is to train a classifier on these subspaces for action recognition. Since these subspaces are elements of the Grassmannian, we must use an SVM kernel defined on this manifold. To this end, there are several potential kernels [20], of which we use the exponential Projection metric kernel due to its empirical benefits on our problem as validated in Table 2. For two subspaces  $U_1$  and  $U_2$ , the exponential projection metric kernel  $K$  has the following form:

$$K(U_1, U_2) = \exp\left(\beta \|U_1^T U_2\|_F^2\right), \text{ for } \beta > 0. \quad (12)$$

## 4. Experiments

This section evaluates the proposed ranking method on four standard benchmark datasets on activity recognition, namely (i) the JHMDB dataset [24], (ii) the MPII Cooking activities dataset [36], (iii) the HMDB-51 dataset [27], and the UCF101 dataset [40]. In all our experiments, we use the standard 16-layer Imagenet pre-trained VGG-net deep learning network [39], which is then fine-tuned on the respective dataset and input modality, such as single RGB or a stack of 10 consecutive optical flow images. We provide the details and evaluation protocols for each of these datasets below.

**HMDB Dataset:** consists of 6766 videos from 51 different action categories. The videos are generally of low quality, with strong camera motion, and non-centered people.

**JHMDB Dataset:** is a subset of HMDB dataset consisting of 968 clips and 21 different action classes. The dataset was mainly created for evaluating the impact of human pose estimation for action recognition, and thus all videos contain humans whose body-parts are clearly visible.

**MPII Cooking Activities Dataset:** consists of high-resolution videos of activities in a kitchen related to cooking several dishes. In comparison to the other two datasets, the videos are captured by a static camera. However, the activities could be very subtle such as *slicing* or *cutting vegetables*, *washing* or *wiping plates*, etc. that needs to be recognized. There are 5609 video clips and 65 annotated actions.

**UCF101 Dataset:** contains 13320 videos distributed in 101 action categories. This dataset is different from the above ones in that it contains mostly coarse sports activities with strong camera motion and low resolution videos.

### 4.1. Evaluation

The HMDB, UCF101, and JHMDB datasets use mean average accuracy over 3-splits as their evaluation criteria. The MPII dataset uses 7-fold cross-validation and reports results on mean average precision (mAP). For the latter, we use the evaluation code published with the dataset.

### 4.2. Preprocessing

The JHMDB, HMDB, and UCF101 datasets are relatively low resolution and thus we resize the images to input sizes that are required by the standard VGG-net model (that is, 224x224). We use the TVL1 optical flow implementation in Opencv to generate the 10-channel stack of flow images, where each flow image is rescaled in the range 0-255, and then saved as a JPEG image, which is the standard practice.

For the MPII dataset, as the videos are originally in very high-resolution, we use a set of morphological operations to crop regions of interest before resizing them to the CNN input size. To be specific, we first resize the images into half their resolution, followed by computing the absolute difference between the frames, and summing up the differences across the sequence. Next, we apply median filtering, dilation, and connected component analysis to generate binary activity masks, and crop the sequences to the smallest rectangle that includes all the valid components. Once the sequences are cropped to these regions of interest, we use them as inputs to the CNNs, and also use them to compute stacked flow images.

### 4.3. Training CNNs

As alluded to earlier, we use the two-stream model of [38], but uses the VGG-net architecture as it has demonstrated significant benefits [22, 12]. However, our methods are not restricted to any specific architecture and could be used for deeper models such as the ResNet [11]. The two network streams are trained independently against the softmax cross-entropy loss. The RGB stream is fine-tuned from the ImageNet model, while the flow stream is fine-tuned from the UCF101 model publicly available as part of [12]. We used a fixed learning rate of  $10^{-4}$  and an input batch size of 50 frames. The CNN training was stopped as soon as the loss on the validation set started increasing, which happened in about 6K iterations for the RGB stream and 40K iterations for the optical flow. For the HMDB and JHMDB dataset, we use the 95% of the respective training set in each split for fine-tuning the models, and rest is used as the validation set. The MPII cooking activities dataset comes with a training, validation, and a test set. For the UCF101 dataset, we used the models from [12] directly.

### 4.4. Results

This section provides a systematic evaluation of the influence of various hyper-parameters in our model, namely (i) influence of the number of subspaces used in our model, (ii) influence of the threshold used in enforcing the temporal order, (iii) comparison of the performance difference FC6 and FC7 CNN layer outputs in GRP, and (iv) an analysis of various Grassmannian kernels. We use the JHMDB and MPII datasets as common test beds for this analysis. In the following, we use the notations FLOW to denote a 10-channel stack of flow images, and RGB to denote the single RGB images.

We use the rectified output of the fully-connected layer fc6 of the VGG-net for all our evaluations which are 4096 dimensional vectors. All the features are unit-normalized before applying the pooling. We use the MANOPT [4] software package for implementing the Grassmannian conjugate gradient. We run 100 iterations of this algorithm in all our experiments. Unless otherwise specified, we use the projection metric kernel [19] for classifying the subspaces on the Grassmannian. As for FLOW + RGB, which combines the GRP results from both FLOW and RGB streams, we use sum of two separate projection metric kernels one from each modality for classification.

#### 4.4.1 Number of Subspaces and Ranking Threshold

In Figure 2(a), we compare the accuracy against an increasing number of subspaces used in the GRP formulation on the split-1 of the JHMDB dataset. We also compare the performance when not using the ranking constraints in the formulation. As is clear, temporal order constraints is ben-

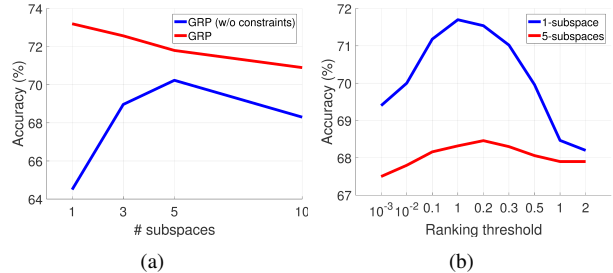


Figure 2. Left: Evaluation of accuracy against increasing subspace dimensionality (keeping fixed ranking threshold at 0.1). Right: Evaluation of accuracy against increasing ranking threshold  $\eta$  keeping subspace dimensions fixed. Both results are on split 1 of JHMDB dataset and uses FLOW + RGB and does not use slack variables in the optimization.

eficial, leading to about 9% improvement in accuracy when using 1-2 subspaces, and about 3-5% when using a larger number. This difference suggests that larger number of subspaces, which more closely approximates the input data, may be perhaps be capturing the background non-dynamics related features, which are not useful for classification.

To further validate the usefulness of our ranking strategy, we fixed the number of subspaces and increased the ranking threshold  $\eta$  in (3) from  $10^{-3}$  to 2 in steps. Our plot in Figure 2(b) shows that the accuracy of recognition increases significantly when the temporal order is enforced in the subspace reconstruction. However, when the number of subspaces is larger, this constraint does not help much, as is observed in the previous experiment as well. These two plots clearly demonstrate the correctness of our scheme and the usefulness of the ranking constraints in subspace representation. In the sequel, we use 2 subspaces in all our experiments, as it was seen most often to provide good results on the validation datasets.

In Table 1, we compare the influence of the ranking constraints on the FLOW and the RGB channels separately. We note that these constraints have a bigger influence on the FLOW stream than on the RGB stream, implying that the dynamics are mostly being captured in the FLOW, as is obvious, while the RGB stream CNN is perhaps learning mostly the background context. A similar observation is also made in [46]. Nevertheless, it is noteworthy that these constraints does improve the performance even on the RGB stream.

#### 4.4.2 Choice of Grassmannian Kernel

Another choice in our setup is the Grassmannian kernel to use. In Harandi et al. [20], a list of several useful kernels on this manifold is presented, each one behaving differently with respect to the application. To this end, we decided to evaluate the performance of these kernels on the subspaces

| Method/Dataset        | FLOW        | RGB         |
|-----------------------|-------------|-------------|
| MPII                  | mAP (%)     | mAP (%)     |
| GRP (w/o constraints) | 51          | 48.9        |
| GRP-Grassmann         | 52.1        | 50.3        |
| JHMDB                 | Avg.Acc.(%) | Avg.Acc.(%) |
| GRP (w/o constraints) | 59.4        | 41.8        |
| GRP-Grassmann         | 64.2        | 42.5        |

Table 1. Comparison between the influence of GRP on FLOW and RGB separately on JHMDB and MPII datasets. These experiments use the split-1 of the respective datasets. Results of FLOW+RGB are in Table 4.

generated from GRP. In Table 2, we compare these kernels on the split-1 of MPII and the JHMDB datasets. We use the polynomial and RBF variants of the standard Projection Metric and the Binet-Cauchy distances. As depicted in the table, the linear kernel and the Binet-Cauchy kernels did not seem to perform well, but both the projection metric kernels seems to showcase significant benefits.

| Method/Dataset     | MPII (mAP%) | JHMDB(Avg. Acc%) |
|--------------------|-------------|------------------|
| Linear             | 24.2        | 46.6             |
| Poly. Proj. Metric | 50.4        | 65.3             |
| RBF Proj. Metric   | 52.1        | 66.8             |
| Poly. Binet-Cauchy | 33.6        | 40.0             |
| RBF Binet-Cauchy   | 33.5        | 38.0             |

Table 2. Comparison between the choice of different kernels for classification on the Grassmannian. We use the CNN features from the FLOW stream alone for this evaluation, using 2 subspaces.

#### 4.5. Comparison of CNN Features

Next, we evaluate the usefulness of CNN features from the FC6 and FC7 layers. In Table 4.5, we provide this comparison on the split-1 of the JHMDB dataset, separately for the FLOW, RGB, and the combined streams. We see that consistently, the GRP on the FC6 layer performs better, perhaps it encodes more temporal information than the layers upper in the hierarchy. While, this posits that perhaps even lower intermediate layer features such as from Pool5 might be better. However, the dimensionality of these features is significantly higher and makes the GRP optimization harder in its current form.

| Features | FLOW         | RGB          | FLOW + RGB   |
|----------|--------------|--------------|--------------|
| JHMDB    | Avg. Acc (%) | Avg. Acc (%) | Avg. Acc (%) |
| FC6      | 64.2         | 42.5         | 73.8         |
| FC7      | 63.4         | 40.3         | 72.0         |
| MPII     | mAP (%)      | mAP (%)      | mAP (%)      |
| FC6      | 52.1         | 50.3         | 53.8         |
| FC7      | 45.6         | 46.5         | 50.7         |

Table 3. Accuracy comparison using FC6 and FC7 features.

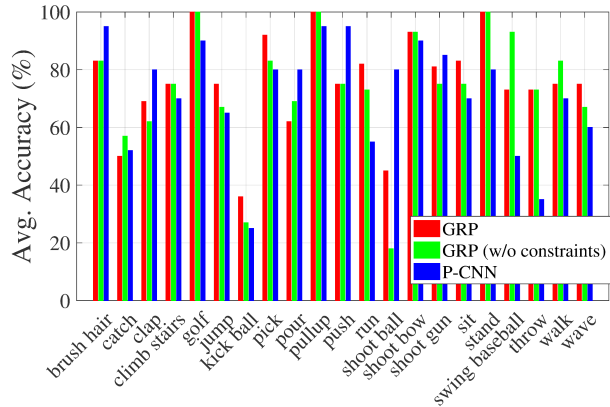


Figure 3. Detailed comparison of the improvements afforded by GRP against the variant without ranking constraints and a recent state-of-the-art method [6] on the JHMDB dataset (3-splits).

#### 4.6. Comparison between Pooling Techniques

Now that we have a clear understanding of the behavior of GRP under disparate scenarios, we compare it against other popular pooling methods. To this end, we compare to (i) standard average pooling, (ii) Rank pooling [15], which uses only a line for enforcing the temporal order, (iii) our GRP scheme but without ordering constraints, (iv) GRP-Grassmannian, which is our proposed scheme, and (v) our convex reformulation of GRP, as described in Section 3.3. For Rank pooling, we use the publicly available code from the authors without any modifications. In Table 4, we provide these comparisons on the split-1 of all the four datasets. The results show that GRP is significantly better than average or Rank pooling consistently on all the four datasets. Further, surprisingly, we note that a low-rank reconstruction of the CNN features by itself provides a very good summarization of the actions useful for recognition. While, using subspaces for action recognition has been done several times in the past [21, 44], we are not aware of any work that shows these benefits on CNN features. However, using ranking constraints on low-rank subspaces leads to even better results. Specifically, there is about 7% improvement on the JHMDB dataset, and about 4% on the MPII dataset, 3% on the HMDB datasets. We also note from these results that GRP-incremental works similar to GRP-Grassmannian, but shows slightly lower performance on an average. This is not surprising, given that it is a greedy method. Computationally it is seen to be significantly slower than GRP-Grassmannian, that computes all the subspaces together.

#### 4.7. Comparison to the State of the Art

In Tables 5, 6, 7, and 8, we compare GRP against state-of-the-art pooling and action recognition methods using CNNs and hand-crafted features. For all comparisons, we use the published results and follow the exact evaluation

| Method/Dataset        | MPII-mAP (%) | JHMDB-Avg.Acc.(%) | HMDB-Avg. Acc.(%) | UCF101-Avg. Acc. (%) |
|-----------------------|--------------|-------------------|-------------------|----------------------|
| Avg. Pooling [38]     | 38.1         | 55.9              | 53.6              | 88.5                 |
| Rank Pooling [15]     | 47.2         | 55.2              | 51.4              | 63.8                 |
| GRP (w/o constraints) | 50.1         | 67.5              | 62.2              | 90.4                 |
| GRP-Grossman          | 53.8         | 73.8              | 65.2              | 91.2                 |
| GRP-incremental       | 51.2         | 74.3              | 64.6              | 89.9                 |

Table 4. Comparison of various pooling techniques on the four datasets. We use the RGB+FLOW together for this evaluation on split-1.

protocols. From the tables, it is clear that GRP outperforms the best methods on MPII and JHMDB datasets, while demonstrates promising results on HMDB and UCF101 datasets. For example, against rank pooling [15], our scheme leads to significant benefits, by about 10-20% on MPII and JHMDB datasets (Table 4), while against dynamic images [3] without hand-crafted features it is better by 2-3% on HMDB and UCF101 datasets. This shows that using subspaces leads to better characterization of the actions. Our results on these datasets are lower than the recent method in [12] that uses sophisticated residual deep models with intermediate stream fusion. However, in comparison, our pooling scheme is very general and could be used in any problem where ordering constraints are natural. In Figure 3, we analyze the results of GRP, GRP-without constraints, and the recent P-CNN scheme [6]. Out of 21 actions in this dataset, GRP outperforms P-CNN on 13. On 19 actions either GRP performs better or equal than the variant without constraints, thus substantiating its benefits.

| Algorithm                           | mAP(%)      |
|-------------------------------------|-------------|
| P-CNN + IDT-FV [6]                  | 71.4        |
| Interaction Part Mining [49]        | 72.4        |
| Holistic + Pose [36]                | 57.9        |
| Video Darwin [15]                   | 72.0        |
| Semantic Features [50]              | 70.5        |
| Hierarchical Mid-Level Actions [41] | 66.8        |
| Higher-order Pooling [5]            | 73.1        |
| GRP (w/o constraints)               | 66.1        |
| GRP                                 | 68.4        |
| GRP + IDT-FV                        | <b>75.5</b> |

Table 5. MPII Cooking Activities (7 splits)

| Algorithm                   | Avg. Acc. (%) |
|-----------------------------|---------------|
| P-CNN [6]                   | 61.1          |
| P-CNN + IDT-FV [6]          | 72.2          |
| Action Tubes [17]           | 62.5          |
| Stacked Fisher Vectors [33] | 69.03         |
| IDT + FV [45]               | 62.8          |
| Higher-order Pooling [5]    | 73.3          |
| GRP (w/o constraints)       | 64.1          |
| GRP                         | 70.6          |
| GRP + IDT-FV                | <b>73.7</b>   |

Table 6. JHMDB Dataset (3 splits)

| Algorithm                        | Avg. Acc. (%) |
|----------------------------------|---------------|
| Two stream [38]                  | 59.4          |
| Spatio-Temporal ResNet [11]      | <b>70.3</b>   |
| Temporal Segment Networks [48]   | 69.4          |
| TDD + IDT-FV [47]                | 65.9          |
| Dynamic Image + IDT-FV [3]       | 65.2          |
| Hier. Rank Pooling + IDT-FV [13] | 66.9          |
| Dynamic Flow + IDT-FV [46]       | 67.4          |
| GRP (w/o constraints)            | 63.1          |
| GRP                              | 65.4          |
| GRP + IDT-FV                     | 67.0          |

Table 7. HMDB Dataset (3 splits)

| Algorithm                        | Avg. Acc. (%) |
|----------------------------------|---------------|
| Two stream [38]                  | 88.0          |
| Spatio-Temporal ResNet [11]      | <b>94.6</b>   |
| Temporal Segment Networks [48]   | 94.2          |
| TDD + IDT-FV [47]                | 91.5          |
| C3D + IDT-FV [42]                | 90.4          |
| Dynamic Image + IDT-FV [3]       | 89.1          |
| Hier. Rank Pooling + IDT-FV [13] | 91.4          |
| Dynamic Flow + IDT-FV [46]       | 91.3          |
| GRP(w/o constraints)             | 90.1          |
| GRP                              | 91.9          |
| GRP + IDT-FV                     | 92.3          |

Table 8. UCF101 Dataset (3 splits)

## 5. Conclusions

We presented a novel algorithm, generalized rank pooling, to summarize the action dynamics in video sequences. Our main proposition was to use the parameters of a low-rank subspace as the pooled representation, where the deep learned features from each frame of the sequence is assumed to preserve their temporal order in this subspace. As such subspaces belong to the Grassmannian, we proposed an efficient conjugate gradient optimization scheme for pooling. Extensive experiments on four action recognition datasets demonstrated the advantages of our scheme<sup>1</sup>.

**Acknowledgements:** This research was supported by the Australian Research Council (ARC) through the Centre of Excellence for Robotic Vision (CE140100016). AC thanks the National Computational Infrastructure (NCI) for the support in experiments.

<sup>1</sup>Code will be available at <http://users.cecs.anu.edu.au/~cherian/>.



## References

- [1] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, USA, 2008. 4, 5
- [2] J. K. Aggarwal and M. S. Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3):16, 2011. 1
- [3] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *CVPR*, 2016. 2, 8
- [4] 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
- [5] A. Cherian, P. Koniusz, and S. Gould. Higher-order pooling of CNN features via kernel linearization for action recognition. In *WACV*, 2017. 8
- [6] G. Chéron, I. Laptev, and C. Schmid. P-CNN: Pose-based cnn features for action recognition. *ICCV*, 2015. 7, 8
- [7] 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. 1, 2
- [8] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*, 2015. 2
- [9] 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. 3
- [10] 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. 4
- [11] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, 2016. 6, 8
- [12] 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, 8
- [13] B. Fernando, P. Anderson, M. Hutter, and S. Gould. Discriminative hierarchical rank pooling for activity recognition. In *CVPR*, 2016. 1, 2, 8
- [14] B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, and T. Tuytelaars. Rank pooling for action recognition. *TPAMI*, 39(4):773–787, 2017. 2, 3
- [15] B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *CVPR*, 2015. 1, 2, 3, 7, 8
- [16] B. Fernando and S. Gould. Learning end-to-end video classification with rank-pooling. In *ICML*, 2016. 2
- [17] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015. 8
- [18] S. Gould, B. Fernando, A. Cherian, P. Anderson, R. S. Cruz, and E. Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 2016. 2
- [19] J. Hamm and D. D. Lee. Grassmann discriminant analysis: a unifying view on subspace-based learning. In *ICML*, 2008. 6
- [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, 6
- [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, 7
- [22] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, 2016. 6
- [23] S. Herath, M. Harandi, and F. Porikli. Going deeper into action recognition: A survey. *Image and Vision Computing*, 60:4 – 21, 2017. 2
- [24] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *ICCV*, 2013. 5
- [25] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 1, 2
- [26] S. Karthikeyan, U. Gaur, B. S. Manjunath, and S. Grafton. Probabilistic subspace-based learning of shape dynamics modes for multi-view action recognition. In *ICCV Workshops*, 2011. 2
- [27] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In *ICCV*, 2011. 5
- [28] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 2011. 2
- [29] B. Li, M. Ayazoglu, T. Mao, O. I. Camps, and M. Szaier. Activity recognition using dynamic subspace angles. In *CVPR*, 2011. 2
- [30] E. Littwin and L. Wolf. The multiverse loss for robust transfer learning. In *CVPR*, 2016. 3
- [31] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *TPAMI*, 19(7):696–710, 1997. 3
- [32] S. O’Hara and B. A. Draper. Scalable action recognition with a subspace forest. In *CVPR*, 2012. 3
- [33] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *ECCV*. 2014. 8
- [34] R. Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010. 1
- [35] B. Raychev, R. Shigenaka, T. Tamaki, and K. Kaneda. Action recognition by orthogonalized subspaces of local spatio-temporal features. In *ICIP*, 2013. 3
- [36] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele. A database for fine grained activity detection of cooking activities. In *CVPR*, 2012. 5, 8
- [37] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004. 3
- [38] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 1, 2, 3, 6, 8
- [39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2, 5

- [40] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 5
- [41] B. Su, J. Zhou, X. Ding, H. Wang, and Y. Wu. Hierarchical dynamic parsing and encoding for action recognition. In *ECCV*, 2016. 2, 8
- [42] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *ICCV*, 2015. 2, 8
- [43] C.-C. Tseng, J.-C. Chen, C.-H. Fang, and J.-J. J. Lien. Human action recognition based on graph-embedded spatiotemporal subspace. *Pattern Recognition*, 45(10):3611 – 3624, 2012. 3
- [44] 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, 7
- [45] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 1, 8
- [46] J. Wang, A. Cherian, and F. Porikli. Ordered pooling of optical flow sequences for action recognition. In *WACV*, 2017. 2, 6, 8
- [47] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015. 1, 8
- [48] 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. 8
- [49] 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
- [50] Y. Zhou, B. Ni, S. Yan, P. Moulin, and Q. Tian. Pipelining localized semantic features for fine-grained action recognition. In *ECCV*. 2014. 8